RDBMS Lab

LAB CODE: 14ITL504

DEPARTMENT OF INFORMATION TECHNOLOGY

BAPATLA ENGINEERING COLLEGE

(AUTONOMOUS)

BAPATLA

LAB COURSE DESCRIPTION

1. **Course code: 14ITL504**
2. **Course Title: RDBMS Lab**
3. **Core** [√] **Elective** [ ]
4. **Pre-requisites (if any): C**
5. **Semester / year in which offered: 2017-2018: III Year I Sem**
6. **No. of weeks of Instruction: 15**
7. **No. of hours per week: 3**
8. **List of programs: separate sheet has been attached**
9. **Evaluation procedure:**

Index	Description Evaluation	Total Marks
A	**Marks allotted for day-to-day lab work**	**20 M**
B	**Marks allotted for record**	**5 M**
C	**Marks Awarded for Lab Exam**	**15 M**
D	**Continuous Internal Evaluation Marks (A+B+C)**	**40 M**
E	**Semester End Examination (Writeup:10M, Record:5M, Viva:15M, Experiment:30M)**	**60 M**
	Total Marks	**100 M**

Lab Course Objectives:

The major objective of this lab is to provide a strong formal foundation in database concepts, technology and practice to the participants to groom them into well-informed database application developers.

- **To present SQL and procedural interfaces to SQL comprehensively**
- **To give an introduction to systematic database design approaches covering conceptual design, logical design and an overview of physical design.**
- **To give a good formal foundation on the relational model of data.**

Lab Course Outcomes:

After undergoing this laboratory module, the participant should be able to:

- **Understand, appreciate and effectively explain the underlying concepts of database technologies.**
- **Design and implement a database schema for a given problem-domain.**
- **Normalize a database.**
- **Populate and query a database using SQL DML/DDL commands.**
- **Declare and enforce integrity constraints on a database using a state-of-the-art RDBMS.**
- **Programming PL/SQL including stored procedures, stored functions, cursors, packages.**

List of Programs:

I. Simple queries: selection, projection, sorting on a simple table

i. Small-large number of attributes
ii. Distinct output values
iii. Renaming attributes
iv. Computed attributes
v. Simple-complex conditions (AND, OR, NOT)
vi. Partial Matching operators (LIKE, %, _, *, ?)
vii. ASC-DESC ordering combinations
viii. Checking for Nulls

II. Multi-table queries(JOIN OPERATIONS)

i. Simple joins (no INNER JOIN)
ii. Aliasing tables – Full/Partial name qualification
iii. Inner-joins (two and more (different) tables)
iv. Inner-recursive-joins (joining to itself)
v. Outer-joins (restrictions as part of the WHERE and ON clauses)
vi. Using where & having clauses

III. Nested queries

i. In, Not In
ii. Exists, Not Exists
iii. Dynamic relations (as part of SELECT, FROM, and WHERE clauses)

IV. Set Oriented Operations

i. Union

ii. Difference
iii. Intersection
iv. Division

V. DDL & TCL Commands.

i. Creating objects: tables, views, users, sequences, Collections etc.
ii. Privilege management through the Grant/Revoke commands
iii. Transaction processing using Commit/Rollback
iv. Save points.

VI. PL/SQL Programming I

i. Programs using named and unnamed blocks
ii. Programs using Cursors, Cursor loops and records

VII. PL/SQL Programming II

i. Creating stored procedures, functions and packages
ii. Error handling and Exception
iii. Triggers and auditing triggers

VIII. User Defined Types

i. Creating Objects
ii. Creating User Defined Operators

EXP I. INTRODUCTION

RDBMS is one of the most widely used pluggable software components in most enterprise software applications. Though RDBMS applications have been there since the early 1970s, they have improved tremendously in terms of their features, the size of data they can hold, and the complexity over the last 20 years.

Oracle is one of the very widely used commercial RDBMS systems and at this point is the market leader as far as RDBMS is concerned. Oracle9i is RDBMS software which supports SQL – 1999. It also supports programming language extension to SQL called PL/SQL which is Oracle proprietary and is very popular to do program development at the database server level. Other popular DBMS software like Sybase and SQL Server support their own programming extensions to SQL – 1999.

Oracle has many tools such as SQL * PLUS, Oracle Forms, Oracle Report Writer, Oracle Graphics etc.

- **SQL * PLUS**: The SQL * PLUS tool is made up of two distinct parts. These are
  - **Interactive SQL:** Interactive SQL is designed for create, access and manipulate data structures like tables and indexes.
  - **PL/SQL:** PL/SQL can be used to developed programs for different applications.
- **Oracle Forms:** This tool allows you to create a data entry screen along with the suitable menu objects. Thus it is the oracle forms tool that handles data gathering and data validation in a commercial application.
- **Report Writer:** Report writer allows programmers to prepare innovative reports using data from the oracle structures like tables, views etc. It is the report writer tool that handles the reporting section of commercial application.
- **Oracle Graphics:** Some of the data can be better represented in the form of pictures. The oracle graphics tool allows programmers to prepare graphs using data from oracle structures like tables, views etc.

SQL (Structured Query Language):

Structured Query Language is a database computer language designed for managing data in relational database management systems(RDBMS), and originally based upon Relational Algebra. Its scope includes data query and update, schema creation and modification, and data access control. SQL was one of the first languages for Edgar F.

Codd's relational model in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks"[3] and became the most widely used language for relational databases.

- IBM developed SQL in mid of 1970's.
- Oracle incorporated in the year 1979.
- SQL used by IBM/DB2 and DS Database Systems.
- SQL adopted as standard language for RDBS by ASNI in 1989.

DATA TYPES:

CHAR (Size): This data type is used to store character strings values of fixed length. The size in brackets determines the number of characters the cell can hold. The maximum number of character is 255 characters.

1. **VARCHAR (Size) / VERCHAR2 (Size)**: This data type is used to store variable length alphanumeric data. The maximum character can hold is 2000 character.
2. **NUMBER (P, S):** The NUMBER data type is used to store number (fixed or floating point). Number of virtually any magnitude may be stored up to 38 digits of precision. Number as large as $9.99 * 10^{124}$. The precision (p) determines the number of places to the right of the decimal. If scale is omitted then the default is zero. If precision is omitted, values are stored with their original precision up to the maximum of 38 digits.
3. **DATE:** This data type is used to represent date and time. The standard format is DD-MM-YY as in 17-SEP-2009. To enter dates other than the standard format, use the appropriate functions. Date time stores date in the 24-Hours format. By default the time in a date field is 12:00:00 am, if no time portion is specified. The default date for a date field is the first day the current month.
4. **LONG:** This data type is used to store variable length character strings containing up to 2GB. Long data can be used to store arrays of binary data in ASCII format. LONG values cannot be indexed, and the normal character functions such as SUBSTR cannot be applied.
5. **RAW:** The RAW data type is used to store binary data, such as digitized picture or image. Data loaded into columns of these data types are stored without any further conversion. RAW data type can have a maximum length of 255 bytes. LONG RAW data type can contain up to 2GB.

INTERACTIVE SQL:

Syntax : VERB(Parameter_1,Parameter_2,Parameter_3,.....Parameter_n);

SQL language is sub-divided into several language elements, including:

- *Clauses*, which are in some cases optional, constituent components of statements and queries.
- *Expressions,* which can produce either scalar values or tables consisting of columns and rows of data.
- *Predicates* which specify conditions that can be evaluated to SQL three-valued logic (3VL) Boolean truth values and which are used to limit the effects of statements and queries, or to change program flow.
- *Queries* which retrieve data based on specific criteria.
- *Statements* which may have a persistent effect on schemas and data, or which may control transactions, program flow, connections, sessions, or diagnostics.
- SQL statements also include the semicolon (";") statement terminator. Though not required on every platform, it is defined as a standard part of the SQL grammar.
- *Insignificant white space* is generally ignored in SQL statements and queries, making it easier to format SQL code for readability.

There are five types of SQL statements. They are:

1. DATA DEFINITION LANGUAGE (DDL)
2. DATA MANIPULATION LANGUAGE (DML)
3. DATA RETRIEVAL LANGUAGE (DRL)
4. TRANSATIONAL CONTROL LANGUAGE (TCL)
5. DATA CONTROL LANGUAGE (DCL)

1. DATA DEFINITION LANGUAGE (DDL): The Data Definition Language (DDL) is used to create and destroy databases and database objects. These commands will primarily be used by database administrators during the setup and removal phases of a database project. Let's take a look at the structure and usage of four basic DDL commands:

1. CREATE 2. ALTER 3. DROP 4. RENAME

1. CREATE:

(a)CREATE TABLE: This is used to create a new relation and the corresponding

Syntax: CREATE TABLE relation_name

(field_1 data_type(Size),field_2 data_type(Size), .. .);

Example:

SQL>CREATE TABLE Student (sno NUMBER(3),sname CHAR(10),class CHAR(5));

(b)CREATE TABLE..AS SELECT....: This is used to create the structure of a new relation from the structure of an existing relation.

Syntax: CREATE TABLE (relation_name_1, field_1,field_2,.....field_n) AS SELECT field_1,field_2,.....field_n FROM relation_name_2;

Example: SQL>CREATE TABLE std(rno,sname) AS SELECT sno,sname FROM student;

2. ALTER:

(a)ALTER TABLE ...ADD...: This is used to add some extra fields into existing relation.

Syntax: ALTER TABLE relation_name ADD(new field_1 data_type(size), new field_2 data_type(size),..);

Example : SQL>ALTER TABLE std ADD(Address CHAR(10));

(b)ALTER TABLE...MODIFY...: This is used to change the width as well as data type of fields of existing relations.

Syntax: ALTER TABLE relation_name MODIFY (field_1 newdata_type(Size), field_2 newdata_type(Size),....field_newdata_type(Size));

Example:SQL>ALTER TABLE student MODIFY(sname VARCHAR(10),class VARCHAR(5));

3. DROP TABLE: This is used to delete the structure of a relation. It permanently deletes the records in the table.

Syntax: DROP TABLE relation_name;

Example: SQL>DROP TABLE std;

4. RENAME: It is used to modify the name of the existing database object.

Syntax: RENAME TABLE old_relation_name TO new_relation_name;

Example: SQL>RENAME TABLE std TO std1;

5. TRUNCATE: This command will remove the data permanently. But structure will not be removed.

Syntax: TRUNCATE TABLE <Table name>

Example TRUNCATE TABLE student;

<u>Difference between Truncate & Delete:-</u>

- ✓ By using truncate command data will be removed permanently & will not get back where as by using delete command data will be removed temporally & get back by using roll back command.

- ✓ By using delete command data will be removed based on the condition where as by using truncate command there is no condition.
- ✓ Truncate is a DDL command & delete is a DML command.

2. DATA MANIPULATION LANGUAGE (DML): The Data Manipulation Language (DML) is used to retrieve, insert and modify database information. These commands will be used by all database users during the routine operation of the database. Let's take a brief look at the basic DML commands:

1. INSERT 2. UPDATE 3. DELETE

1. INSERT INTO: This is used to add records into a relation. These are three type of INSERT INTO queries which are as

a) Inserting a single record

Syntax: INSERT INTO relationname(field_1,field_2,.field_n)VALUES (data_1,data_2,.....data_n);

Example: SQL>INSERT INTO student(sno,sname,class,address)VALUES (1,'Ravi','M.Tech','Palakol');

b) Inserting all records from another relation

Syntax: INSERT INTO relation_name_1 SELECT Field_1,field_2,field_n
FROM relation_name_2 WHERE field_x=data;

Example: SQL>INSERT INTO std SELECT sno,sname FROM student
WHERE name = 'Ramu';

c) Inserting multiple records

Syntax: INSERT INTO relation_name field_1,field_2,.....field_n) VALUES
(&data_1,&data_2,.....&data_n);

Example: SQL>INSERT INTO student(sno,sname,class,address)
VALUES(&sno,'&sname','&class','&address');

Enter value for sno: 101
Enter value for name: Ravi
Enter value for class: M.Tech
Enter value for name: Palakol

2. UPDATE-SET-WHERE: This is used to update the content of a record in a relation.

Syntax: SQL>UPDATE relation name SET Field_name1=data,field_name2=data,
WHERE field_name=data;

Example: SQL>UPDATE student SET sname = 'kumar' WHERE sno=1;

3. DELETE-FROM: This is used to delete all the records of a relation but it will retain the structure of that relation.

a) DELETE-FROM: This is used to delete all the records of relation.

Syntax: SQL>DELETE FROM relation_name;

Example: SQL>DELETE FROM std;

b) DELETE -FROM-WHERE: This is used to delete a selected record from a relation.

Syntax: SQL>DELETE FROM relation_name WHERE condition;

Example: SQL>DELETE FROM student WHERE sno = 2;

3. DRL(DATA RETRIEVAL LANGUAGE): Retrieves data from one or more tables.

1. SELECT FROM: To display all fields for all records.

Syntax : SELECT * FROM relation_name;

Example : SQL> select * from dept;

```
DEPTNO     DNAME        LOC
--------   ------------ ----------
  10       ACCOUNTING   NEW YORK
  20       RESEARCH     DALLAS
  30       SALES        CHICAGO
  40       OPERATIONS   BOSTON
```

2. SELECT FROM: To display a set of fields for all records of relation.

Syntax: SELECT a set of fields FROM relation_name;

Example: SQL> select deptno, dname from dept;

```
DEPTNO     DNAME
-------    ----------
10         ACCOUNTING
20         RESEARCH
30         SALES
```

3. SELECT - FROM -WHERE: This query is used to display a selected set of fields for a selected set of records of a relation.

Syntax: SELECT a set of fields FROM relation_name WHERE condition;

Example: SQL> select * FROM dept WHERE deptno<=20;

```
DEPTNO     DNAME        LOC
------     ------------ ------------
10         ACCOUNTING   NEW YORK
20         RESEARCH     DALLAS
```

4. SELECT - FROM -GROUP BY: This query is used to group to all the records in a relation together for each and every value of a specific key(s) and then display them for a selected set of fields the relation.

Syntax: SELECT a set of fields FROM relation_name GROUP BY field_name;

Example:

```
SQL> SELECT EMPNO, SUM (SALARY) FROM EMP GROUP BY EMPNO;
EMPNO      SUM (SALARY)
------     ----------
  1         3000
  2         4000
  3         5000
  4         6000
4 rows selected.
```

5. SELECT - FROM -ORDER BY: This query is used to display a selected set of fields from a relation in an ordered manner base on some field.

Syntax: SELECT a set of fields FROM relation_name
ORDER BY field_name;

Example:

```
SQL> SELECT empno,ename,job FROM emp ORDER BY job;
EMPNO      ENAME     JOB
------     ---------  --------
 4         RAVI       MANAGER
 2         aravind    Manager
 1         sagar      clerk
 3         Laki       clerk
4rows selected.
```

6. JOIN using SELECT - FROM - ORDER BY: This query is used to display a set of fields from two relations by matching a common field in them in an ordered manner based on some fields.

Syntax: SELECT a set of fields from both relations FROM relation_1, relation_2 WHERE relation_1.field_x = relation_2.field_y ORDER BY field_z;

Example:

```
SQL>SELECT empno,ename,job,dname FROM emp,dept
                                WHERE emp.deptno = 20 ORDER BY job;
EMPNO      ENAME     JOB          DNAME
------     ------    -------      ----------
7788       SCOTT     ANALYST      ACCOUNTING
```

```
7902        FORD        ANALYST     ACCOUNTING
------
7566        JONES               MANAGER   OPERATIONS
7566        JONES               MANAGER   SALES
 20 rows selected.
```

7. JOIN using SELECT - FROM - GROUP BY: This query is used to display a set of fields from two relations by matching a common field in them and also group the corresponding records for each and every value of a specified key(s) while displaying.

Syntax: SELECT a set of fields from both relations FROM relation_1,relation_2 WHERE relation_1.field-x=relation_2.field-y GROUP BY field-z;

Example:

```
SQL> SELECT empno,SUM(SALARY) FROM emp,dept
     WHERE emp.deptno =20 GROUP BY empno;
EMPNO       SUM (SALARY)
-------     --------
7369        3200
7566        11900
7788        12000
7876        4400
```

8. UNION: This query is used to display the combined rows of two different queries, which are having the same structure, without duplicate rows.

Syntax: SELECT field_1,field_2,..... FROM relation_1 WHERE (Condition) UNION SELECT field_1,field_2,..... FROM relation_2 WHERE (Condition);

Example:

```
SQL> SELECT * FROM STUDENT;
     SNO        SNAME
     -----      -------
     1          kumar
     2          ravi
     3          ramu

SQL> SELECT * FROM STD;
     SNO        SNAME
     -----      -------
     3          ramu
```

5	lalitha
9	devi
1	kumar

SQL> SELECT * FROM student UNION SELECT * FROM std;

SNO	SNAME
---	---
1	kumar
2	ravi
3	ramu
5	lalitha
9	devi

9. INTERSET: This query is used to display the common rows of two different queries, which are having the same structure, and to display a selected set of fields out of them.

Syntax: SELECT field_1,field_2,.. FROM relation_1 WHERE (Condition) INTERSECT SELECT field_1,field_2,.. FROM relation_2 WHERE(Condition);

Example : SQL> SELECT * FROM student INTERSECT SELECT * FROM std;

SNO	SNAME
---	---
1	Kumar

10. MINUS: This query is used to display all the rows in relation_1,which are not having in the relation_2.

Syntax: SELECT field_1,field_2,.....FROM relation_1 WHERE(Condition) MINUS SELECT field_1,field_2,..... FROM relation_2 WHERE(Conditon);

SQL> SELECT * FROM student MINUS SELECT * FROM std;

SNO	SNAME
---	---
2	RAVI
3	RAMU

3. TRANSATIONAL CONTROL LANGUAGE (T.C.L):

A transaction is a logical unit of work. All changes made to the database can be referred to as a transaction. Transaction changes can be mode permanent to the database

only if they are committed a transaction begins with an executable SQL statement & ends explicitly with either role back or commit statement.

1. COMMIT: This command is used to end a transaction only with the help of the commit command transaction changes can be made permanent to the database.

Syntax: SQL>COMMIT;

Example: SQL>COMMIT;

2. SAVE POINT: Save points are like marks to divide a very lengthy transaction to smaller once. They are used to identify a point in a transaction to which we can latter role back. Thus, save point is used in conjunction with role back.

Syntax: SQL>SAVE POINT ID;

Example: SQL>SAVE POINT xyz;

3. ROLE BACK: A role back command is used to undo the current transactions. We can role back the entire transaction so that all changes made by SQL statements are undo (or) role back a transaction to a save point so that the SQL statements after the save point are role back.

Syntax: ROLE BACK(current transaction can be role back)

ROLE BACK to save point ID;

Example: SQL>ROLE BACK;

SQL>ROLE BACK TO SAVE POINT xyz;

4. DATA CONTROL LANGUAGE (D.C.L):

DCL provides uses with privilege commands the owner of database objects (tables), has the soul authority ollas them. The owner (data base administrators) can allow other data base uses to access the objects as per their requirement

1. GRANT: The GRANT command allows granting various privileges to other users and allowing them to perform operations with in their privileges

For Example, if a uses is granted as 'SELECT' privilege then he/she can only view data but cannot perform any other DML operations on the data base object GRANTED privileges can also be withdrawn by the DBA at any time

Syntax: SQL>GRANT PRIVILEGES on object_name To user_name;

Example: SQL>GRANT SELECT, UPDATE on emp To hemanth;

2. REVOKE: To with draw the privileges that has been GRANTED to a uses, we use the REVOKE command

Syntax: SQL>REVOKE PRIVILEGES ON object-name FROM user_name;

Example: SQL>REVOKE SELECT, UPDATE ON emp FROM ravi;

1. Creation, altering and dropping of tables and inserting rows into a table (use constraints while creating tables) examples using SELECT command.

1. CREATE:

(a)CREATE TABLE: This is used to create a new relation

Syntax: CREATE TABLE relation_name
(field_1 data_type(Size),field_2 data_type(Size), .. .);

Example:

SQL>CREATE TABLE Student (sno NUMBER(3) **PRIMARY KEY** ,sname CHAR(10),class CHAR(5));

2. ALTER:

(a)ALTER TABLE ...ADD...: This is used to add some extra fields into existing relation.

Syntax: ALTER TABLE relation_name ADD(new field_1 data_type(size), new field_2 data_type(size),..);

Example : SQL>ALTER TABLE std ADD(Address CHAR(10));

(b)ALTER TABLE...MODIFY...: This is used to change the width as well as data type of fields of existing relations.

Syntax: ALTER TABLE relation_name MODIFY (field_1 newdata_type(Size), field_2 newdata_type(Size),....field_newdata_type(Size));

Example: SQL>ALTER TABLE student MODIFY(sname VARCHAR(10),class VARCHAR(5));

3. DROP TABLE: This is used to delete the structure of a relation. It permanently deletes the records in the table.

Syntax: DROP TABLE relation_name;

Example: SQL>DROP TABLE student;

4. INSERT:

Syntax: INSERT INTO relation_name field_1,field_2,.....field_n) VALUES (&data_1,&data_2,.....&data_n);

Example: SQL>INSERT INTO student(sno,sname,class,address) VALUES(&sno,'&sname','&class','&address');

Enter value for sno: 101

Enter value for name: SIRISHA

Enter value for class: CSE

Enter value for address: Palakol

5. SELECT FROM: To display all fields for all records.

Syntax : SELECT * FROM relation_name;

Example : SQL> select * from student;

SNO	SNAME	CLASS	ADDRESS
101	SIRISHA	CSE	PALAKOL
102	DEVAKI	CSE	NARSAPUR
103	KUMAR	CAD	BHIMAVARAM
104	RAVI	VLSI	PALAKOL

2. SELECT FROM: To display a set of fields for all records of relation.

Syntax: SELECT a set of fields FROM relation_name;

Example: SQL> select sno, sname from student;

SNO	SNAME
---	---
101	SIRISHA
102	DEVAKI
103	KUMAR
104	RAVI

3. SELECT - FROM -WHERE: This query is used to display a selected set of fields for a selected set of records of a relation.

Syntax: SELECT a set of fields FROM relation_name WHERE condition;

Example: SQL> select * FROM student WHERE class='CSE';

SNO	SNAME	CLASS	ADDRESS
---	---	---	---
101	SIRISHA	CSE	PALAKOL
102	DEVAKI	CSE	NARSAPUR

There are 5 constraints available in ORACLE:

1. NOT NULL: When a column is defined as NOTNULL, then that column becomes a mandatory column. It implies that a value must be entered into the column if the record is to be accepted for storage in the table.

Syntax:

CREATE TABLE Table_Name(column_name data_type(*size*) **NOT NULL,**);

Example:

CREATE TABLE student (sno **NUMBER(3)NOT NULL,** name **CHAR**(**10**));

2. UNIQUE: The purpose of a unique key is to ensure that information in the column(s) is unique i.e. a value entered in column(s) defined in the unique constraint must not be repeated across the column(s). A table may have many unique keys.

Syntax:

CREATE TABLE Table_Name(column_name data_type(*size*) **UNIQUE, ...**);

Example:

CREATE TABLE student (sno **NUMBER(3) UNIQUE,** name **CHAR**(**10**));

3. CHECK: Specifies a condition that each row in the table must satisfy. To satisfy the constraint, each row in the table must make the condition either TRUE or unknown (due to a null).

Syntax:

CREATE TABLE Table_Name(column_name data_type(*size*) **CHECK(*logical expression*), ...**);

Example: **CREATE TABLE** student (sno **NUMBER (3),** name **CHAR**(**10**),class **CHAR(5),CHECK**(class **IN**('CSE','CAD','VLSI'));

4. PRIMARY KEY: A field which is used to identify a record uniquely. A column or combination of columns can be created as primary key, which can be used as a reference from other tables. A table contains primary key is known as Master Table.

- ✓ It must uniquely identify each record in a table.
- ✓ It must contain unique values.
- ✓ It cannot be a null field.
- ✓ It cannot be multi port field.
- ✓ It should contain a minimum no. of fields necessary to be called unique.

Syntax:

CREATE TABLE Table_Name(column_name data_type(*size*) **PRIMARY KEY, ...**);

Example:

CREATE TABLE faculty (fcode **NUMBER(3) PRIMARY KEY,** fname **CHAR**(**10**));

5. FOREIGN KEY: It is a table level constraint. We cannot add this at column level. To reference any primary key column from other table this constraint can be used. The table in which the foreign key is defined is called a **detail table**. The table that defines the primary key and is referenced by the foreign key is called the **master table**.

D

Syntax: SUM (Column name)

Example: SELECT SUM (Sal) From emp;

3. AVG: AVG followed by a column name returns the average value of that column values.

Syntax: AVG (n1,n2..)

Example: Select AVG(10, 15, 30) FROM DUAL;

4. MAX: MAX followed by a column name returns the maximum value of that column.

Syntax: MAX (Column name)

Example: SELECT MAX (Sal) FROM emp;

```
SQL> select deptno,max(sal) from emp group by deptno;
        DEPTNO       MAX(SAL)
        ------   --------
        10       5000
        20       3000
        30       2850
SQL> select deptno,max(sal) from emp group by deptno having max(sal)<3000;
    DEPTNO     MAX(SAL)
    -----      --------
      30      2850
```

5. MIN: MIN followed by column name returns the minimum value of that column.

Syntax: MIN (Column name)

Example: SELECT MIN (Sal) FROM emp;

```
SQL>select deptno,min(sal) from emp group by deptno having min(sal)>1000;
   DEPTNO   MIN(SAL)
   -----    --------
     10     1300
```

VIEW: In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

A view is a virtual table, which consists of a set of columns from one or more tables. It is similar to a table but it doest not store in the database. View is a query stored as an object.

Syntax: CREATE VIEW view_name AS SELECT set of fields FROM relation_name WHERE (Condition)

1. Example:

```
SQL>CREATE VIEW employee AS SELECT empno,ename,job FROM EMP
        WHERE job = 'clerk';
        View created.
SQL> SELECT * FROM EMPLOYEE;
        EMPNO     ENAME     JOB
        ----      ------    -------
        7369      SMITH     CLERK
        7876      ADAMS     CLERK
        7900      JAMES     CLERK
        7934      MILLER    CLERK
```

2.Example:

```
CREATE VIEW [Current Product List] AS
SELECT ProductID,ProductName
FROM Products
WHERE Discontinued=No
```

DROP VIEW: This query is used to delete a view , which has been already created.

Syntax: DROP VIEW View_name;

Example : SQL> DROP VIEW EMPLOYEE;
View dropped

4. Queries using Conversion functions (to_char, to_number and to_date), string functions (Concatenation, lpad, rpad, ltrim, rtrim, lower, upper, initcap, length, substr and instr), date functions (Sysdate, next_day, add_months, last_day, months_between, least, greatest, trunc, round, to_char, to_date)

1. Conversion functions:

To_char: TO_CHAR (number) converts n to a value of VARCHAR2 data type, using the optional number format fmt. The value n can be of type NUMBER, BINARY_FLOAT, or BINARY_DOUBLE.

```
SQL>select to_char(65,'RN')from dual;
LXV
```

To_number : TO_NUMBER converts expr to a value of NUMBER data type.

SQL> Select to_number('1234.64') from Dual;

1234.64

To_date: TO_DATE converts char of CHAR, VARCHAR2, NCHAR, or NVARCHAR2 data type to a value of DATE data type.

SQL>SELECT TO_DATE('January 15, 1989, 11:00 A.M.') FROM DUAL;

TO_DATE('

15-JAN-89

2. String functions:

Concat: CONCAT returns char1 concatenated with char2. Both char1 and char2 can be any of the datatypes

SQL>SELECT CONCAT('ORACLE','CORPORATION')FROM DUAL;

ORACLECORPORATION

Lpad: LPAD returns expr1, left-padded to length n characters with the sequence of characters in expr2.

SQL>SELECT LPAD('ORACLE',15,'*')FROM DUAL;

*********ORACLE

Rpad: RPAD returns expr1, right-padded to length n characters with expr2, replicated as many times as necessary.

SQL>SELECT RPAD ('ORACLE',15,'*')FROM DUAL;

ORACLE*********

Ltrim: Returns a character expression after removing leading blanks.

SQL>SELECT LTRIM('SSMITHSS','S') FROM DUAL;

MITHSS

Rtrim: Returns a character string after truncating all trailing blanks

SQL>SELECT RTRIM('SSMITHSS','S') FROM DUAL;

SMITH

Lower: Returns a character expression after converting uppercase character data to lowercase.

SQL>SELECT LOWER('DBMS')FROM DUAL;

dbms

Upper: Returns a character expression with lowercase character data converted to uppercase

SQL>SELECT UPPER('dbms')FROM DUAL;

DBMS

Length: Returns the number of characters, rather than the number of bytes, of the given string expression, excluding trailing blanks.

SQL>SELECT LENGTH('DATABASE')FROM DUAL;

8

Substr: Returns part of a character, binary, text, or image expression.

SQL>SELECT SUBSTR('ABCDEFGHIJ'3,4)FROM DUAL;

CDEF

Instr: The INSTR functions search string for substring. The function returns an integer indicating the position of the character in string that is the first character of this occurrence.

SQL>SELECT INSTR('CORPORATE FLOOR','OR',3,2)FROM DUAL;

14

3. Date functions:

Sysdate:

SQL>SELECT SYSDATE FROM DUAL;

29-DEC-08

next_day:

SQL>SELECT NEXT_DAY(SYSDATE,'WED')FROM DUAL;

05-JAN-09

add_months:

SQL>SELECT ADD_MONTHS(SYSDATE,2)FROM DUAL;

28-FEB-09

last_day:

SQL>SELECT LAST_DAY(SYSDATE)FROM DUAL;

31-DEC-08

months_between:

SQL>SELECT MONTHS_BETWEEN(SYSDATE,HIREDATE)FROM EMP;

4

Least:

SQL>SELECT LEAST('10-JAN-07','12-OCT-07')FROM DUAL;

10-JAN-07

Greatest:

SQL>SELECT GREATEST('10-JAN-07','12-OCT-07')FROM DUAL;

10-JAN-07

Trunc:

SQL>SELECT TRUNC(SYSDATE,'DAY')FROM DUAL;

28-DEC-08

Round:

SQL>SELECT ROUND(SYSDATE,'DAY')FROM DUAL;

28-DEC-08

to_char:

SQL> select to_char(sysdate, "dd\mm\yy") from dual;

24-mar-05.

to_date:

SQL> select to_date(sysdate, "dd\mm\yy") from dual;

24-mar-o5.

EXP-II Simple queries: selection, projection, sorting on a simple table

A) Creating The Tables

1) Create table **EMPLOYEE** with the following attributes and then insert the following data in to **EMPLOYEE** table.

FNAME MINIT LNAME SSN BDATE ADDRESS SEX SAL SUPERSSN DNO

John B Smith 123456789 09-JAN-1965 731 Fondren, Houston, TX M 30000 333445555 5

Franklin T Wong 333445555 08-DEC-1955 638 Voss, Houston, TX M 40000 888665555 5

Alicia J Zelaya 999887777 19-JUL-1968 3321, Castle, Spring, TX F 25000 987654321 4

Jennifer S Wallace 987654321 20-JUN-1941 291, Berry, Bellaire, TX F 43000 888665555 4

Ramesh K Narayan 666884444 15-SEP-1962 975 Fire Oak, Humble, TX M 38000 333445555 5

Joyce A English 453453453 31-JUL-1972 5631 Rice, Houston, TX F 25000 333445555 5

Ahmad V Jabbar 987987987 29-MAR-1969 980 Dallas, Houston, TX M 25000 987654321 4

James E Borg 888665555 10-NOV-37 450 Stone, Houston, Tx M 55000 1

Ans: Create table employee (fname varchar(15) NOT NULL,**minit** char(5),**lname** varchar2(15),**ssn** varchar2(9) NOT NULL,**bdate** date,**address** varchar2(30),**sex** char(3),**salary** decimal(10,2), **superssn** varchar2(9),**dno** number(7)**);**

```
Select Run SQL Command Line

SQL>
SQL> DESC EMPLOYEE;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 FNAME                                              VARCHAR2(8)
 MINIT                                              VARCHAR2(1)
 LNAME                                              VARCHAR2(8)
 SSN                                                NUMBER(10)
 BDATE                                              DATE
 ADDRESS                                            VARCHAR2(25)
 SALARY                                             NUMBER(6)
 SUPERSSN                                           NUMBER(10)
 DNO                                                NUMBER(1)
```

```
Run SQL Command Line
SQL> SELECT * FROM EMPLOYEE;

FNAME    M LNAME           SSN BDATE     ADDRESS                        SALARY
-------- - -------- ---------- --------- ------------------------- ----------
  SUPERSSN        DNO
---------- ----------
JOHN     B SMITH     123456789 09-JAN-65 731 FONDREN,HOUSTON,TX         30000
 333445555          5

FRANKLIN T WONG      333445555 08-DEC-55 638 VOSS,HOUSTON,TX            30006
 333445555          5

ALICIA   J ZELAYA    999887777 19-JAN-68 3321 CASTLE,SPRING,TX          45000
 987654321          4

JENNIFER S WALLACE   987654321 02-JUN-41 291 BERRY,BELLAIRE,TX          32000
 987654321          4

JOYCE    A ENGLISH  4534534531 31-JUL-72 5631 RICE,HOUSTON,TX           12000
 333445555          5

AHMED    V JABBAR   9817987987 29-MAR-62 980 DALLAS,HOUSTON,TX          45000
 987654321          5


6 rows selected.

SQL> _
```

2. Create table **DEPARTMENT** with the following attributes and then insert the following data into **DEPARTMENT** table.

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
Research	5	333445555	22-MAY-1988
Administration	4	987654321	01-JAN-1995
Headquarters	1	888665555	19-JUN-1981

Ans: Create table department (dname varchar2(15) NOT NULL,**dnumber number(7), mgrssn varchar2(9),mgrstartdate date);**

```
Run SQL Command Line
SQL> DESC DEPARTMENT;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DNAME                                              VARCHAR2(15)
 DNUMBER                                            NUMBER(1)
 MGR_SSN                                            NUMBER(9)
SQL> _
```

```
Run SQL Command Line
SQL> SELECT * FROM DEPARTMENT;

DNAME              DNUMBER    MGR_SSN
---------------- ---------- ----------
RESEARCH                  5  333445555
ADMINISTRATION            4  987654321
HEAD QUARTERS             1  888665555

SQL>
```

3. Create table **DEPT_LOCATIONS** with the following attributes and insert the following data into **DEPT_LOCATIONS** table.

DNUMBER	DLOCATION
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Ans: Create table dept_locations(dnumber number(7),**dlocation** varchar2(15));

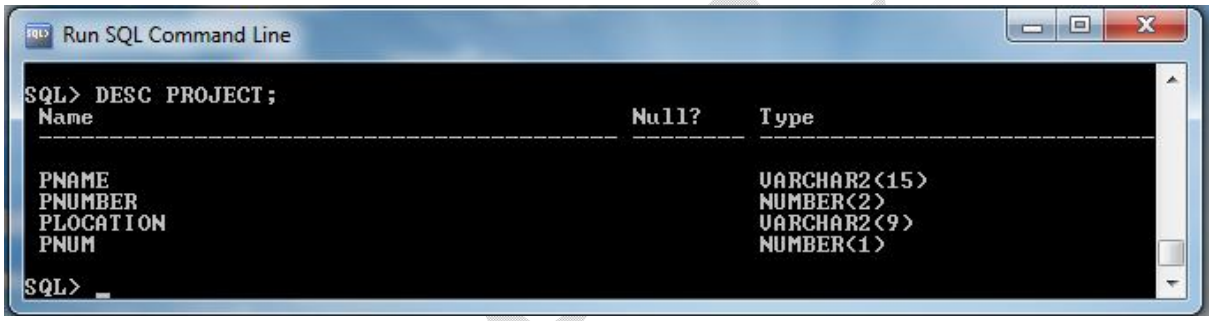
```
Run SQL Command Line
SQL> DESC DEPT_LOCATIONS;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DNUMBER                                            NUMBER(1)
 DLOCATION                                          VARCHAR2(9)
SQL>
```

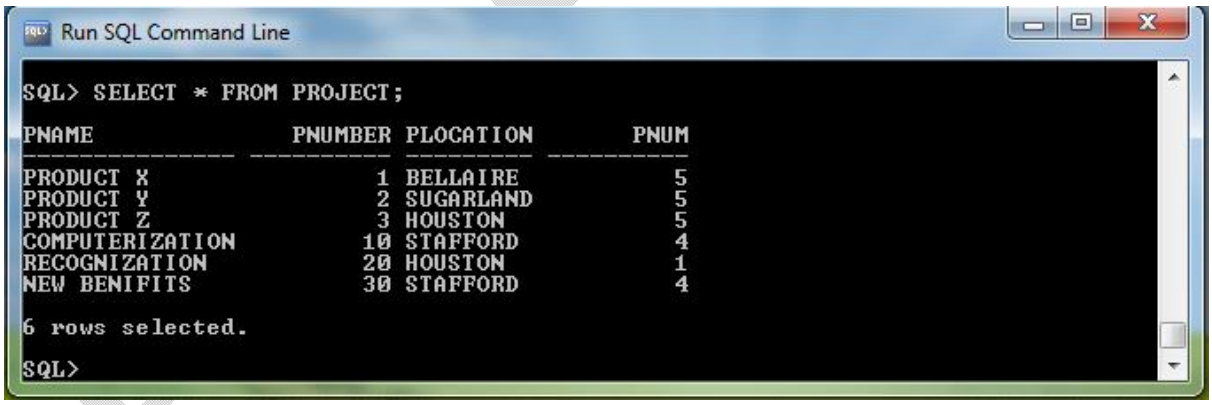
```
Run SQL Command Line
SQL> SELECT * FROM DEPT_LOCATIONS;

   DNUMBER DLOCATION
---------- ---------
         1 HOUSTON
         4 STAFFORD
         5 BELLAIRE
         5 SUGARLAND
         5 HOUSTON

SQL> _
```

4.Create table **PROJECT** with the following attributes and insert the following data into **PROJECT** table.

PNAME	PNUMBER	PLOCATION	DNUM
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

Ans: create table project(pname varchar2(15) NOT NULL,**pnumber** number(7), **plocation** varchar2(15),**dnum** number(4) **);**



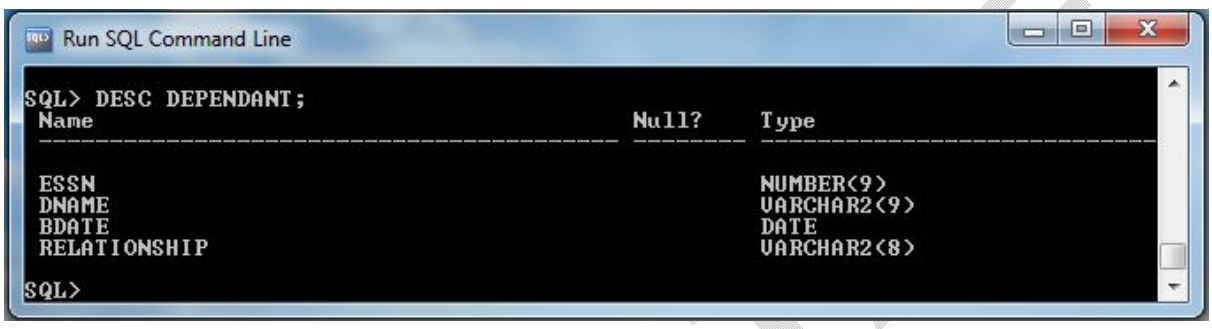


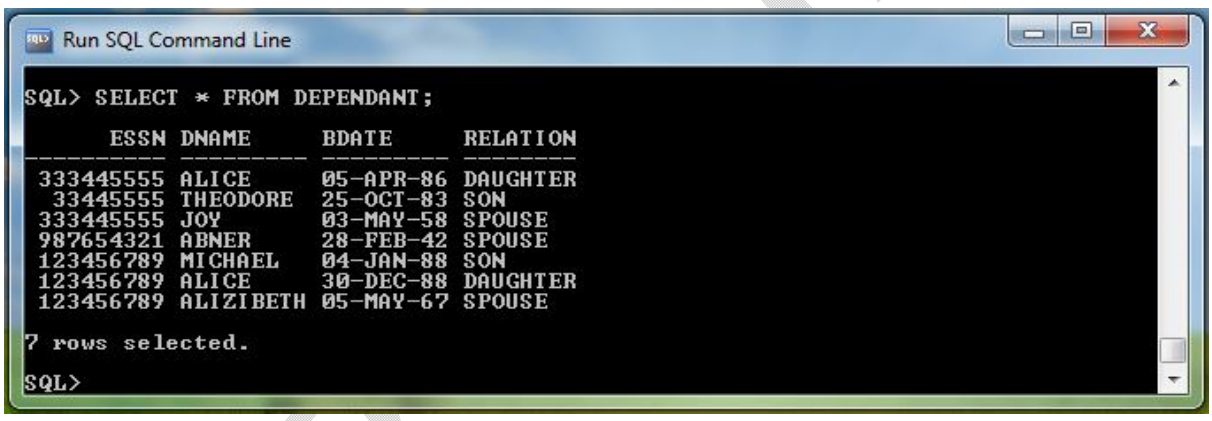
5.Create table **DEPENDENT** with the following attributes and insert the following data into **DEPENDENT** table.

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
333445555	Alice	F	05-APR-86	DUAGHTER
333445555	Theodore	M	25-OCT-83	SON

333445555	Joy	F	03-MAY-58	SPOUSE
987654321	Abner	M	28-FEB-42	SPOUSE
123456789	Michael	M	04-JAN-88	SON
123456789	Alice	F	30-DEC-88	DAUGHTER
123456789	Elizabeth	F	05-MAY-67	SPOUSE

Ans: create table dependent(essn varchar2(9),**dependent_name** varchar2(15), **sex** char(3),**bdate** date,**relationship** varchar2(12 **);**



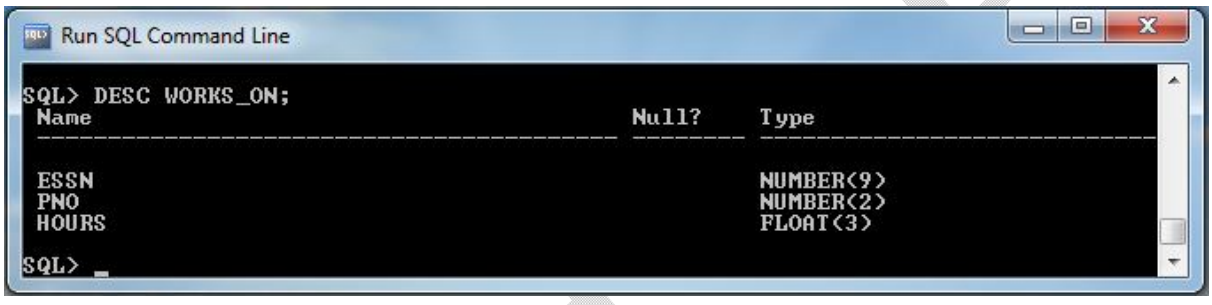


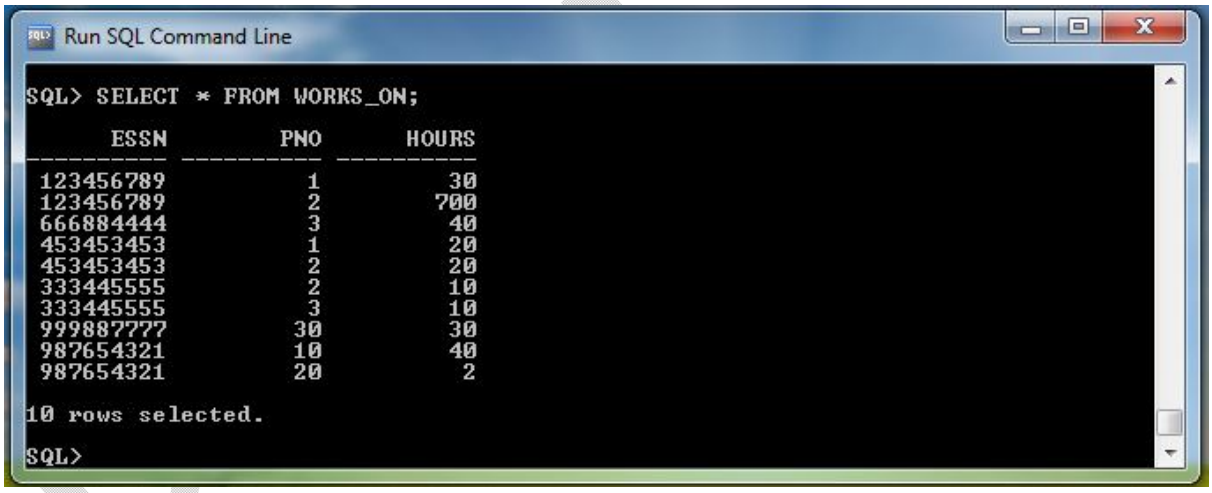
6. Create table **WORKS_ON** with the following attributes and then insert the following data into **WORKS_ON** table.

ESSN	PNO	HOURS
123456789	1	32.5
123456789	2	7.5
666884444	3	40
453453453	1	20
453453453	2	20
333445555	2	10
333445555	3	10

333445555	10	10
333445555	20	10
999887777	30	30
999887777	10	10
987987987	10	35
987987987	30	5
987654321	30	20
987654321	20	15
888665555	20	

Ans: create table works_on(essn varchar2(9,**pno** number(3),**hours** decimal(4,1) **);**



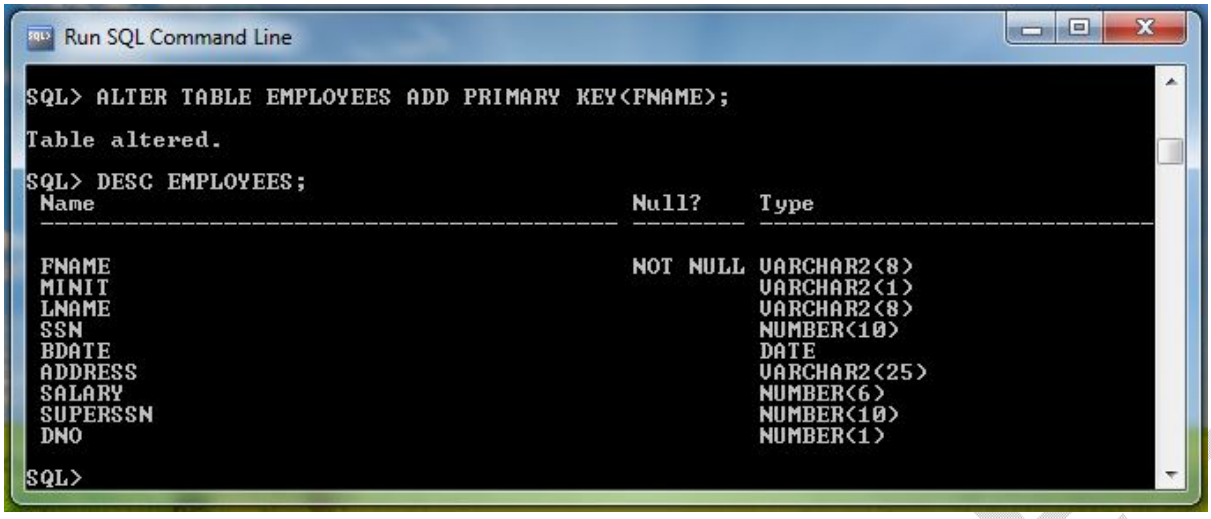


B) **Assigning Key Attributes to Created Tables**

1. **Create table employee (fname** varchar(15) NOT NULL,**minit** char(5),**lname** varchar2(15) NOT NULL,**ssn** varchar2(9) NOT NULL,**bdate** date,**address** varchar2(30),**sex** char(3),**salary** decimal(10,2),**superssn** varchar2(9),**dno** number(7) NOT NULL,**PRIMARY KEY(ssn));**

(Or)

Alter Table Employee add Primary key (ssn);



```
Run SQL Command Line
SQL> ALTER TABLE EMPLOYEES ADD PRIMARY KEY(FNAME);

Table altered.

SQL> DESC EMPLOYEES;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 FNAME                                     NOT NULL VARCHAR2(8)
 MINIT                                              VARCHAR2(1)
 LNAME                                              VARCHAR2(8)
 SSN                                                NUMBER(10)
 BDATE                                              DATE
 ADDRESS                                            VARCHAR2(25)
 SALARY                                             NUMBER(6)
 SUPERSSN                                           NUMBER(10)
 DNO                                                NUMBER(1)

SQL>
```

2. **Create table department (dname** varchar2(15) NOT NULL,**dnumber number(7) NOT NULL, mgrssn varchar2(9),mgrstartdate date,PRIMARY KEY(dnumber), UNIQUE(dname));**

   (Or)

   Alter Table department add Primary key (dnumber);

3. **Create table dept_locations(dnumber** number(7) NOT NULL,**dlocation** varchar2(15) NOT NULL, **PRIMARY KEY(dnumber,dlocation));**

   (Or)

   Alter Table dept_locations add Primary key (dnumber,dlocation);

4. **create table project(pname** varchar2(15) NOT NULL,**pnumber** number(7) NOT NULL,**plocation** varchar2(15),**dnum** number(4) NOT NULL,**PRIMARY KEY(pnumber),UNIQUE(pname));**

   (Or)

   Alter Table project add Primary key (pnumber);

5. **create table dependent(essn** varchar2(9) NOT NULL,**dependent_name** varchar2(15) NOT NULL,**sex** char(3),**bdate** date,**relationship** varchar2(12),**PRIMARY KEY(essn,dependent_name));**

   (Or)

   Alter Table dependent add Primary key (essn,dependent_name);

6. **create table works_on(essn** varchar2(9) NOT NULL,**pno** number(3) NOT NULL,**hours** decimal(4,1) **,PRIMARY KEY(essn,pno));**

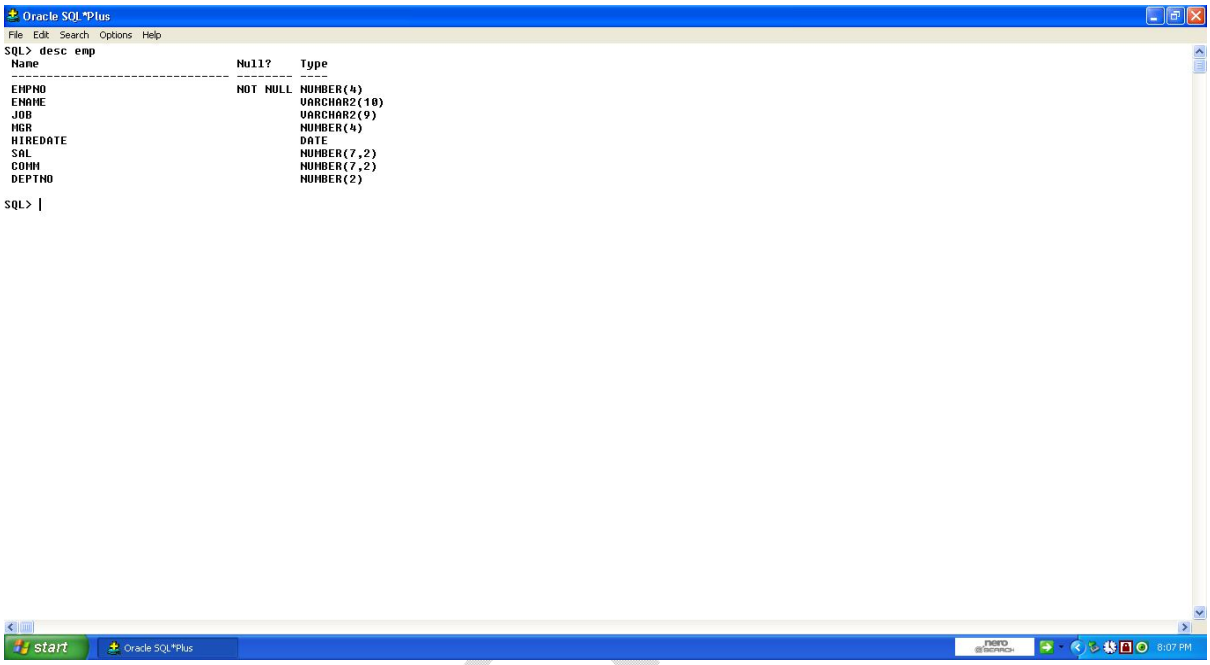
   (Or)

   Alter Table works_on add Primary key (essn,pno);

c) Simple queries:

1.write a query to display emp table.

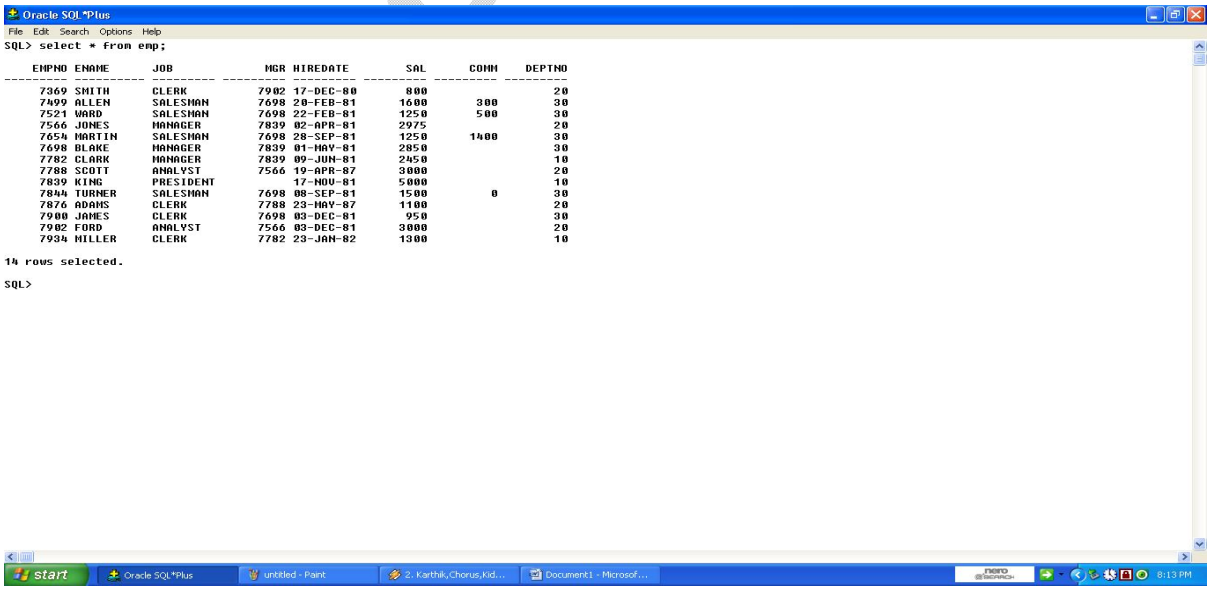
SQL> desc emp;



```
SQL> desc emp
 Name                                      Null?    Type
 ----------------------------------------- -------- ----
 EMPNO                                     NOT NULL NUMBER(4)
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(9)
 MGR                                                NUMBER(4)
 HIREDATE                                           DATE
 SAL                                                NUMBER(7,2)
 COMM                                               NUMBER(7,2)
 DEPTNO                                             NUMBER(2)

SQL>
```

2. List all employee values.

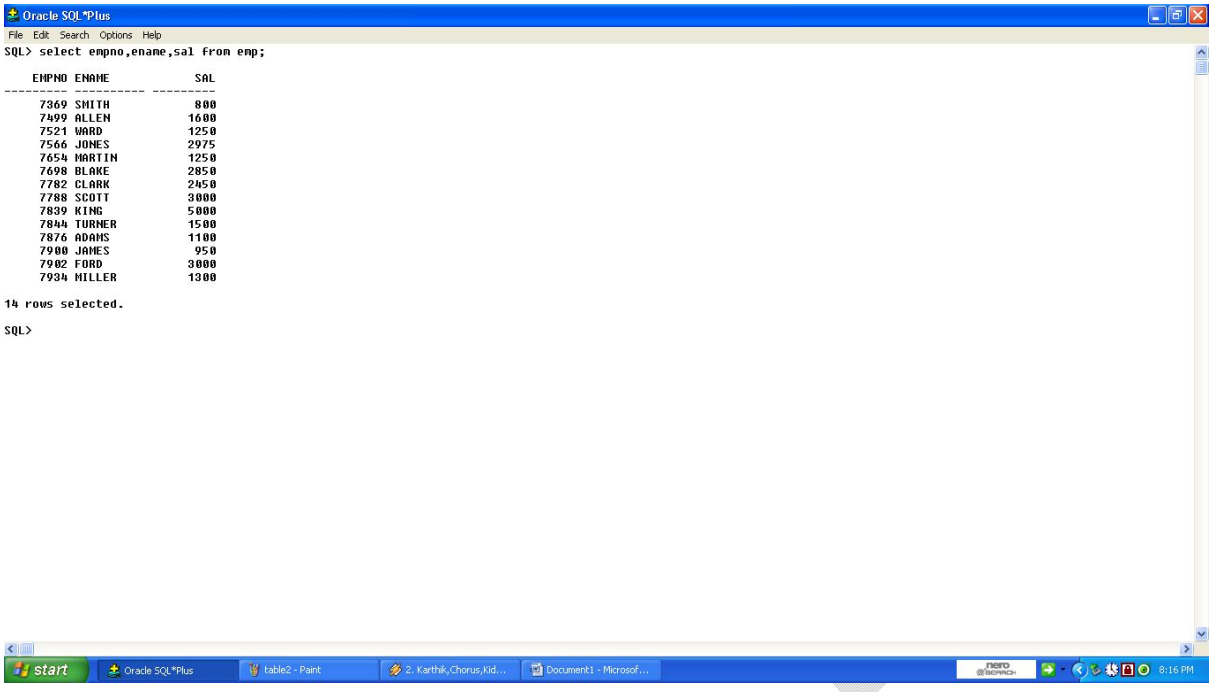
SQL>select * from emp;



```
SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE        SAL       COMM     DEPTNO
---------- ---------- --------- --------- --------- --------- ---------- ----------
      7369 SMITH      CLERK          7902 17-DEC-80       800                    20
      7499 ALLEN      SALESMAN       7698 20-FEB-81      1600        300         30
      7521 WARD       SALESMAN       7698 22-FEB-81      1250        500         30
      7566 JONES      MANAGER        7839 02-APR-81      2975                    20
      7654 MARTIN     SALESMAN       7698 28-SEP-81      1250       1400         30
      7698 BLAKE      MANAGER        7839 01-MAY-81      2850                    30
      7782 CLARK      MANAGER        7839 09-JUN-81      2450                    10
      7788 SCOTT      ANALYST        7566 19-APR-87      3000                    20
      7839 KING       PRESIDENT           17-NOV-81      5000                    10
      7844 TURNER     SALESMAN       7698 08-SEP-81      1500          0         30
      7876 ADAMS      CLERK          7788 23-MAY-87      1100                    20
      7900 JAMES      CLERK          7698 03-DEC-81       950                    30
      7902 FORD       ANALYST        7566 03-DEC-81      3000                    20
      7934 MILLER     CLERK          7782 23-JAN-82      1300                    10

14 rows selected.

SQL>
```

3. List empno,empname and salary.

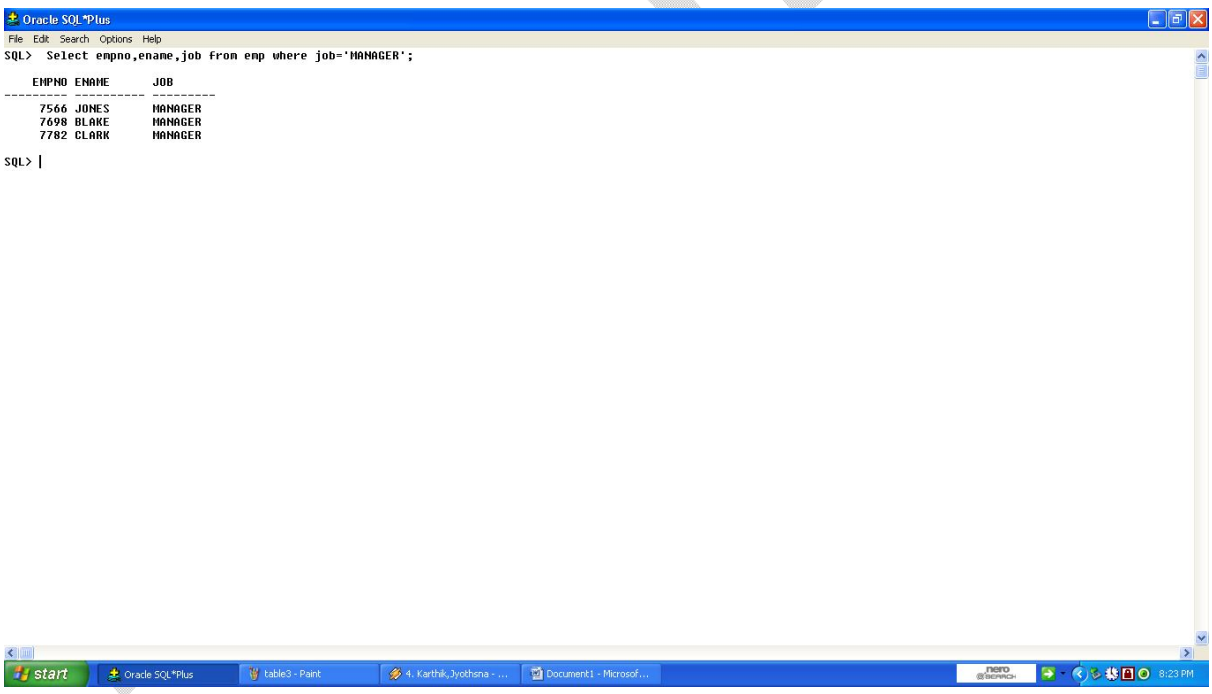
SQL>Select empno,ename,sal from emp;



```
SQL> select empno,ename,sal from emp;

     EMPNO ENAME            SAL
---------- ---------- ---------
      7369 SMITH            800
      7499 ALLEN           1600
      7521 WARD            1250
      7566 JONES           2975
      7654 MARTIN          1250
      7698 BLAKE           2850
      7782 CLARK           2450
      7788 SCOTT           3000
      7839 KING            5000
      7844 TURNER          1500
      7876 ADAMS           1100
      7900 JAMES            950
      7902 FORD            3000
      7934 MILLER          1300

14 rows selected.

SQL>
```

4.List the names of all MANAGERS.

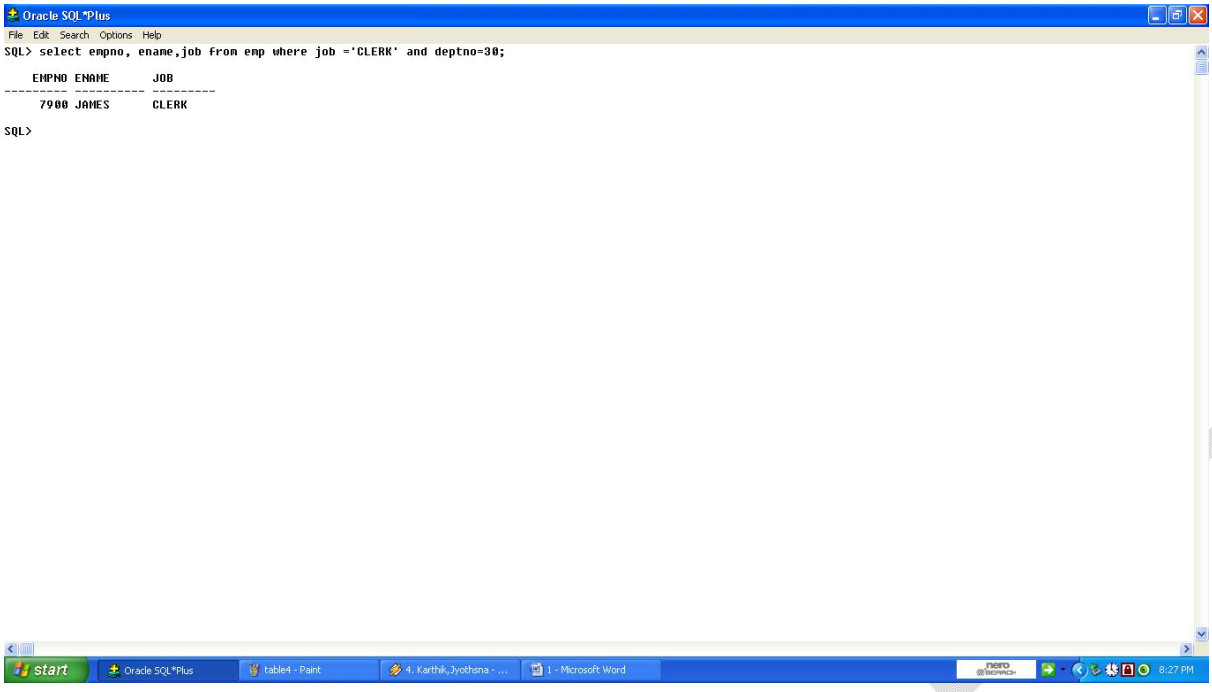
SQL>Select empno,ename,job from emp where job=' MANAGERS';



```
SQL>  Select empno,ename,job from emp where job='MANAGER';

     EMPNO ENAME      JOB
---------- ---------- ---------
      7566 JONES      MANAGER
      7698 BLAKE      MANAGER
      7782 CLARK      MANAGER

SQL> |
```

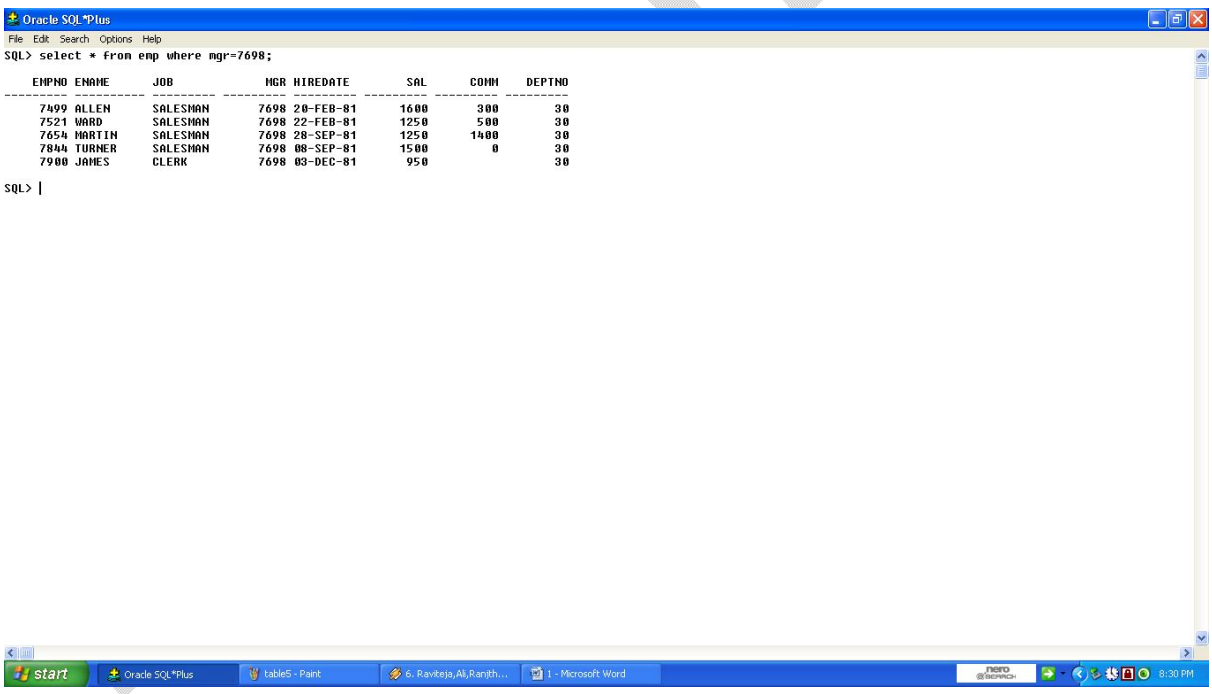
5.List all clerks in deptno. 30.

SQL> select empno, ename,job from emp where job ='CLERK' and deptno=30;



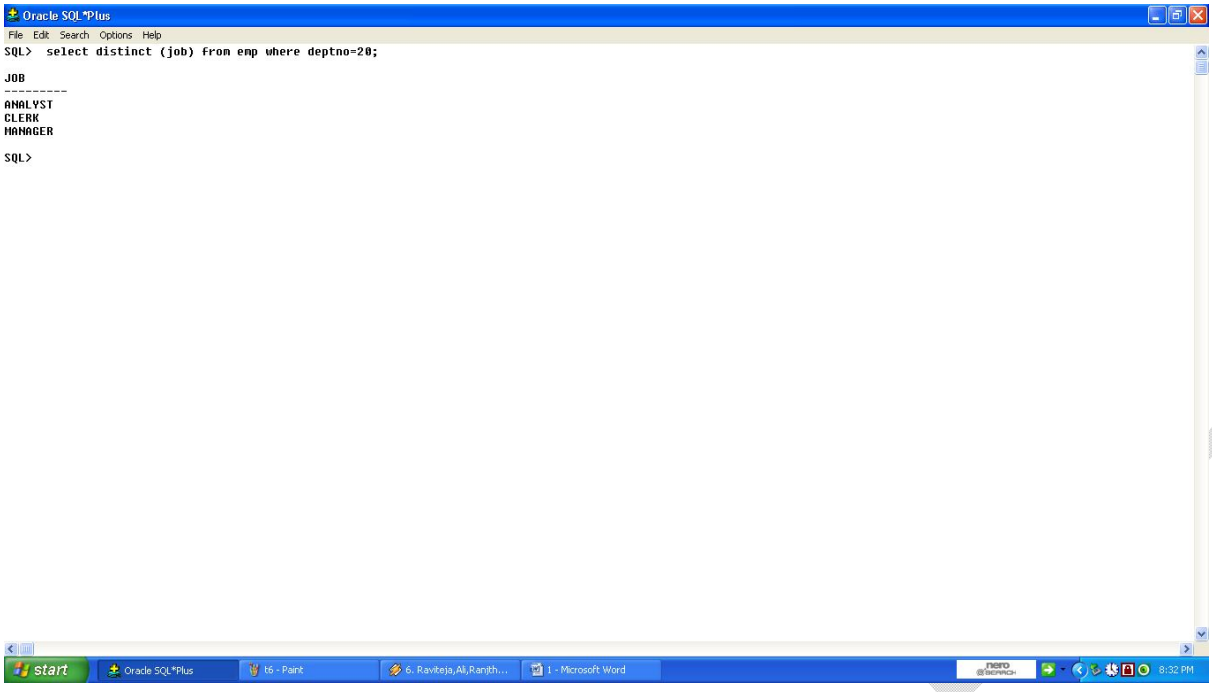
6.List all employee names whose mgr no is 7698.

SQL>select * from emp where mgr=7698;



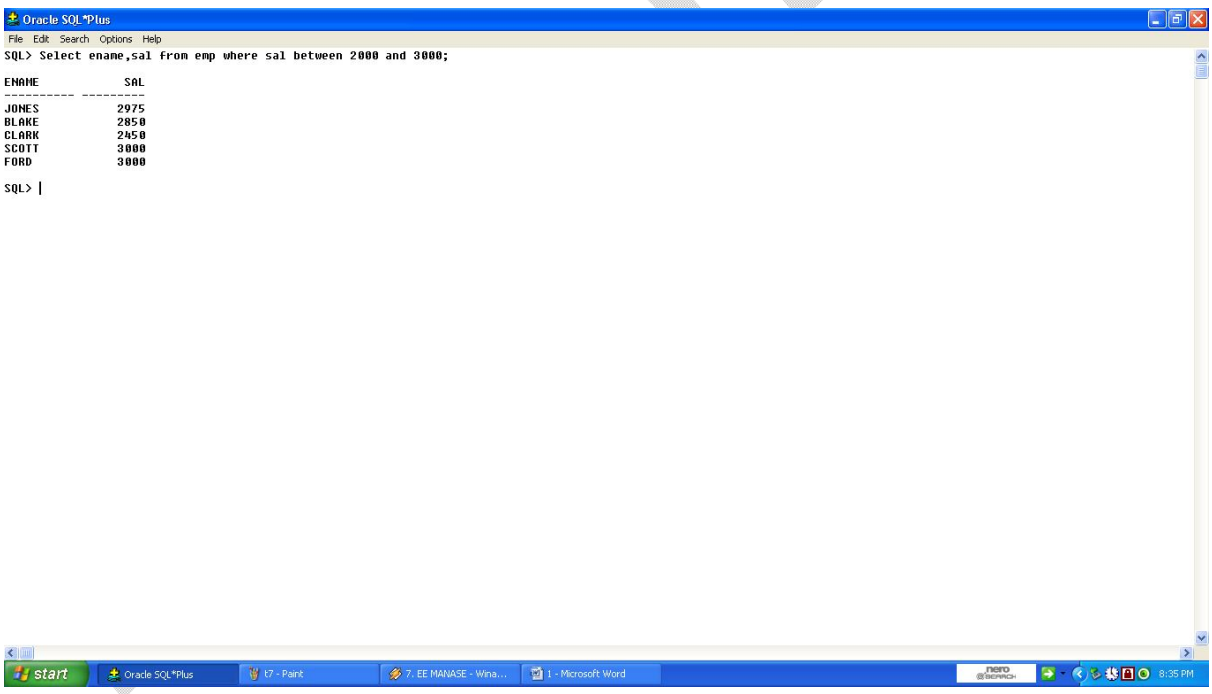
7.List jobs dept 20.

SQL> select distint(job) from emp where deptno=20;



8.List employee names whose salary is between 2000 and 3000.

SQL>Select ename,sal from emp where sal between 2000 and 3000;



9.List employee in the dependent 10,20.

SQL>select * from emp where deptno in(10,20);

```
SQL> select * from emp where deptno in(10,20);

    EMPNO ENAME      JOB              MGR HIREDATE        SAL       COMM     DEPTNO
--------- ---------- --------- --------- --------- --------- --------- ---------
     7369 SMITH      CLERK          7902 17-DEC-80       800                    20
     7566 JONES      MANAGER        7839 02-APR-81      2975                    20
     7782 CLARK      MANAGER        7839 09-JUN-81      2450                    10
     7788 SCOTT      ANALYST        7566 19-APR-87      3000                    20
     7839 KING       PRESIDENT           17-NOV-81      5000                    10
     7876 ADAMS      CLERK          7788 23-MAY-87      1100                    20
     7902 FORD       ANALYST        7566 03-DEC-81      3000                    20
     7934 MILLER     CLERK          7782 23-JAN-82      1300                    10

8 rows selected.

SQL>
```

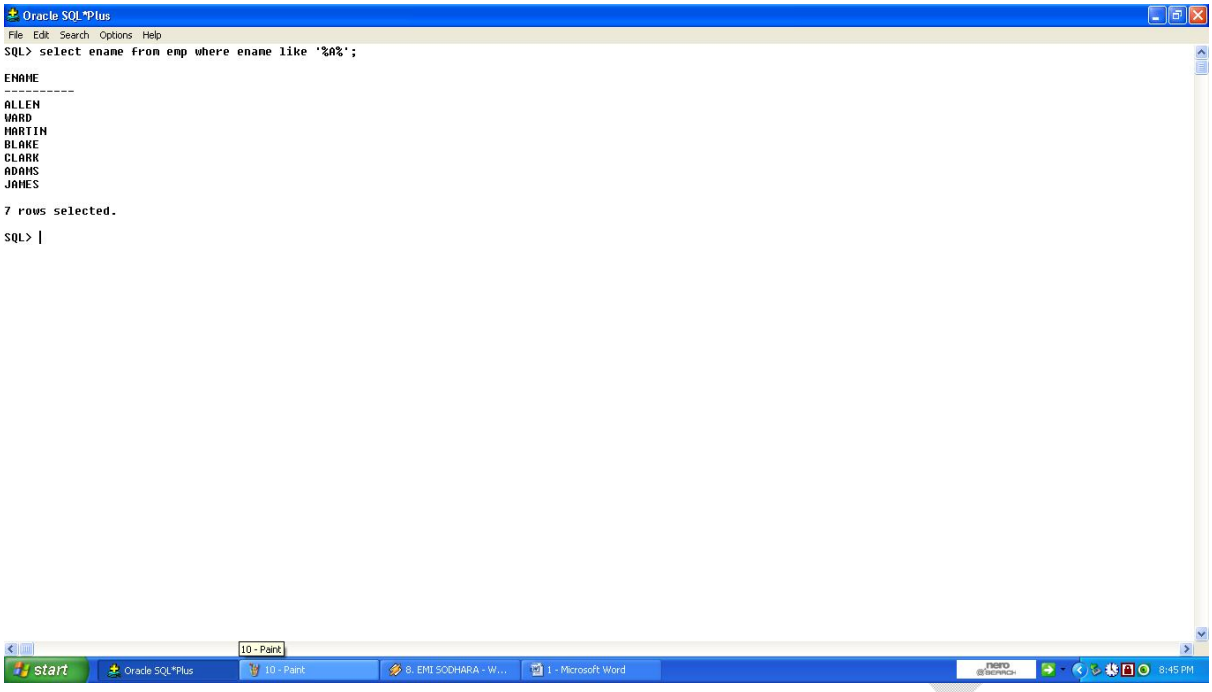
10.list employee names which begin with S.

SQL>select ename from emp where ename like 'S%';

```
SQL> select ename from emp where ename like 'S%';

ENAME
----------
SMITH
SCOTT

SQL>
```

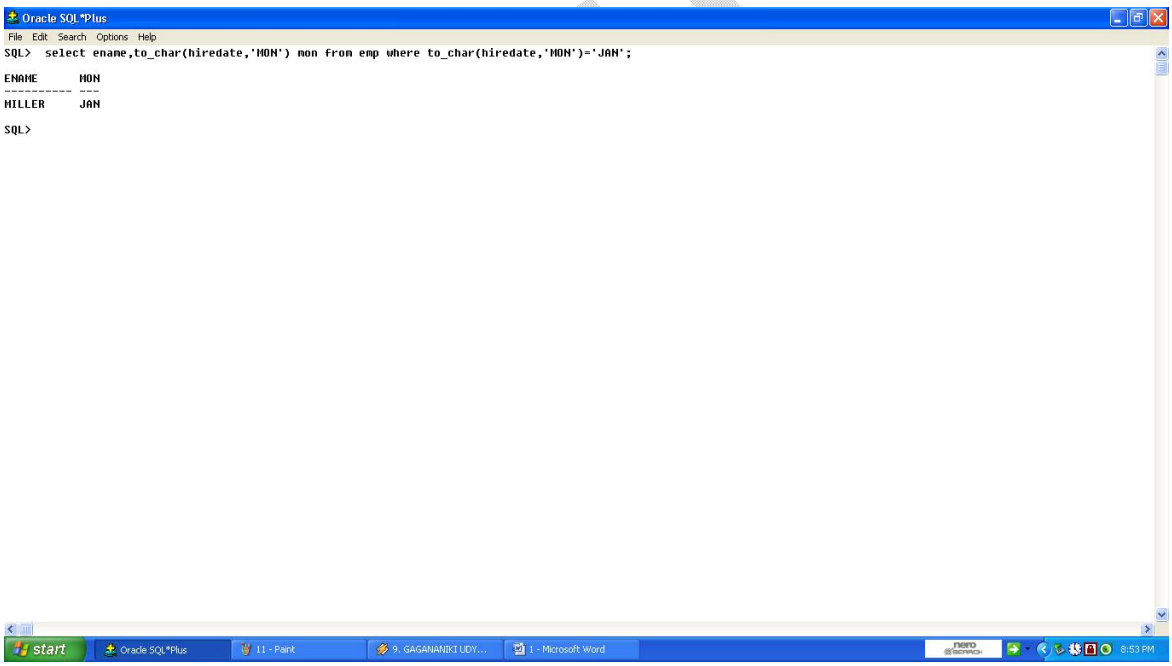
11.List employee names having 'A' in their names.

SQL>select ename from emp where ename like '%A%';



12.List employee who have joined in JAN.

SQL>select ename,to_char(hiredate,'MON') mon from emp where to_char(hiredate,'MON')='JAN';



13.List employees who have joined in the year 81.

SQL>select ename,to_char(hiredate,'YYYY') from emp where to_char(hiredate,'YYYY')=1981;

```
SQL> select ename,to_char(hiredate,'YYYY') from emp where to_char(hiredate,'YYYY')=1981;

ENAME      TO_C
---------- ----
ALLEN      1981
WARD       1981
JONES      1981
MARTIN     1981
BLAKE      1981
CLARK      1981
KING       1981
TURNER     1981
JAMES      1981
FORD       1981

10 rows selected.

SQL> |
```

14.List all distinct jobs.

SQL> select distinct(job) from emp;

```
SQL>  select distinct(job) from emp;

JOB
---------
ANALYST
CLERK
MANAGER
PRESIDENT
SALESMAN

SQL> |
```

15.List employee names in alphabetical order.

SQL>select ename from emp order by ename;

```
SQL> select ename from emp order by ename;

ENAME
----------
ADAMS
ALLEN
BLAKE
CLARK
FORD
JAMES
JONES
KING
MARTIN
MILLER
SCOTT
SMITH
TURNER
WARD

14 rows selected.

SQL>
```

16.List employee names alphabetically department by deptno.

SQL>select ename,deptno from emp order by deptno;

```
SQL> select ename,deptno from emp order by deptno;

ENAME         DEPTNO
---------- ---------
CLARK             10
KING              10
MILLER            10
SMITH             20
ADAMS             20
FORD              20
SCOTT             20
JONES             20
ALLEN             30
BLAKE             30
MARTIN            30
JAMES             30
TURNER            30
WARD              30

14 rows selected.

SQL>
```

17.List employee numbers,name sal,DA(15% OF SAL) and PF (10% of sal).

SQL>select empno,ename sal,(sal*0.15)DA,(SAL*0.1) PF from emp;

```
SQL> select empno,ename sal,(sal*0.15)DA,(SAL*0.1) PF from emp;

    EMPNO SAL                 DA         PF
--------- ---------- --------- ---------
     7369 SMITH             120         80
     7499 ALLEN             240        160
     7521 WARD            187.5        125
     7566 JONES          446.25      297.5
     7654 MARTIN          187.5        125
     7698 BLAKE           427.5        285
     7782 CLARK           367.5        245
     7788 SCOTT             450        300
     7839 KING              750        500
     7844 TURNER            225        150
     7876 ADAMS             165        110
     7900 JAMES           142.5         95
     7902 FORD              450        300
     7934 MILLER            195        130

14 rows selected.

SQL>
```

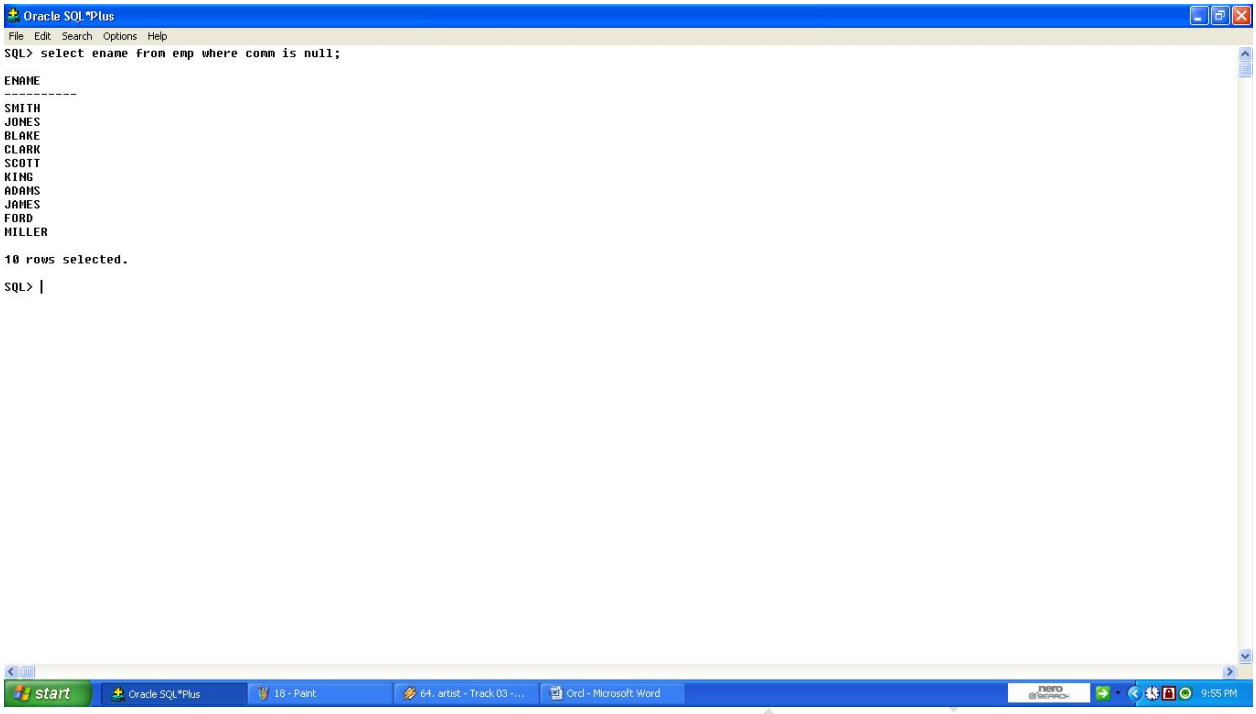
18.List employee names having an experience more than15 years.

SQL>select ename,round(months_between(sysdate,hiredate)/12)exp from emp where round (months_between(sysdate,hiredate)/12)>24;

```
SQL> select ename,round(months_between(sysdate,hiredate)/12)exp from emp where round (months_between
(sysdate,hiredate)/12)>24;

ENAME            EXP
---------- ---------
SMITH             32
ALLEN             31
WARD              31
JONES             31
MARTIN            31
BLAKE             31
CLARK             31
SCOTT             25
KING              31
TURNER            31
ADAMS             25
JAMES             31
FORD              31
MILLER            31

14 rows selected.

SQL> |
```

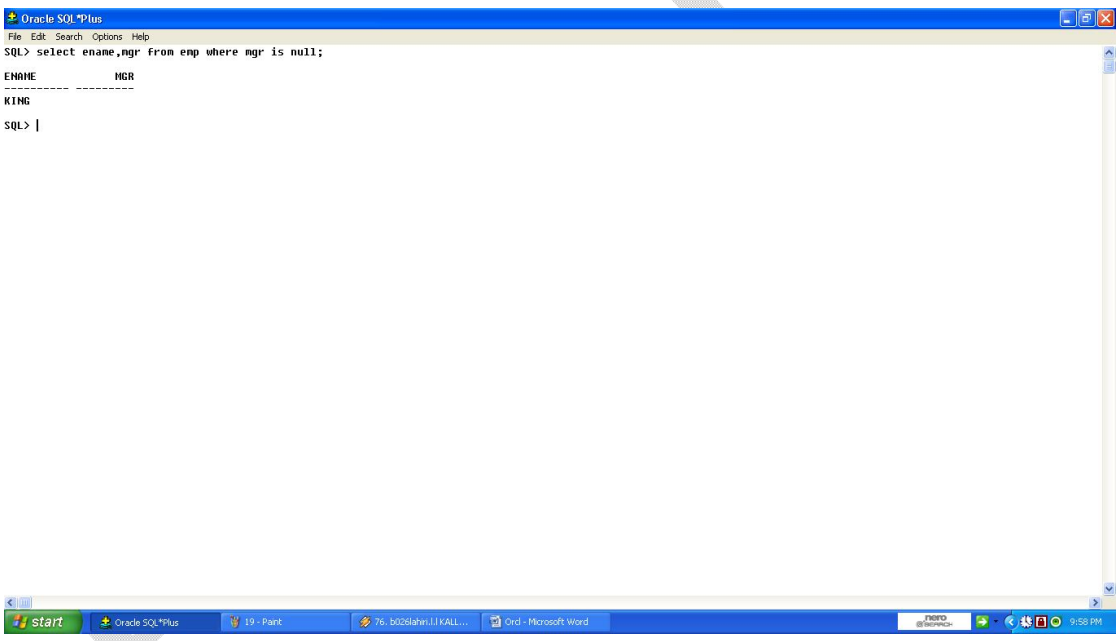
19.List employee names whose commission is NULL.

SQL>select ename from emp where comm is null;



20.List employee who do not report to anybody.

SQL>select ename,mgr from emp where mgr is null;



21.List maximum sal,minimum sal,average sal,
SQL>select max(sal),min(sal),avg(sal) from emp;

```
Oracle SQL*Plus
File Edit Search Options Help
SQL>  select max(sal),min(sal),avg(sal) from emp;

 MAX(SAL)  MIN(SAL)  AVG(SAL)
--------- --------- ---------
     5000       800 2073.2143

SQL>
```

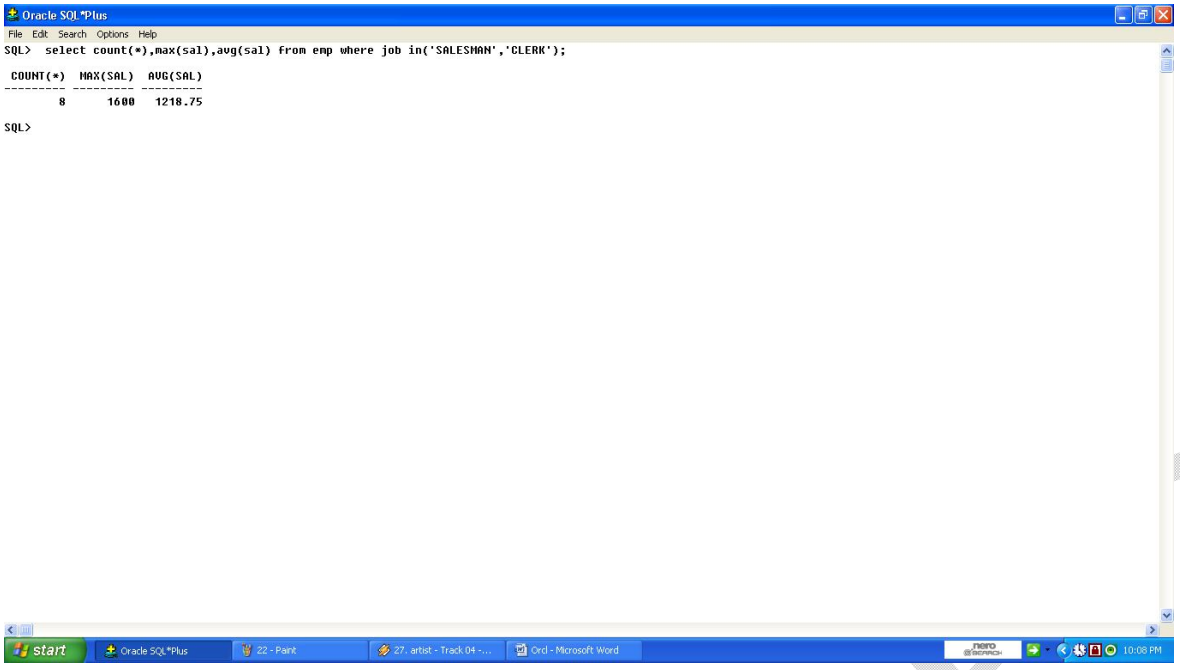
22.List the numbers of people and average slary in deptno 30.

SQL>select count(*),avg (sal) from emp where deptno=30;

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> select count(*),avg (sal) from emp where deptno=30;

 COUNT(*)  AVG(SAL)
--------- ---------
        6 1566.6667

SQL> |
```

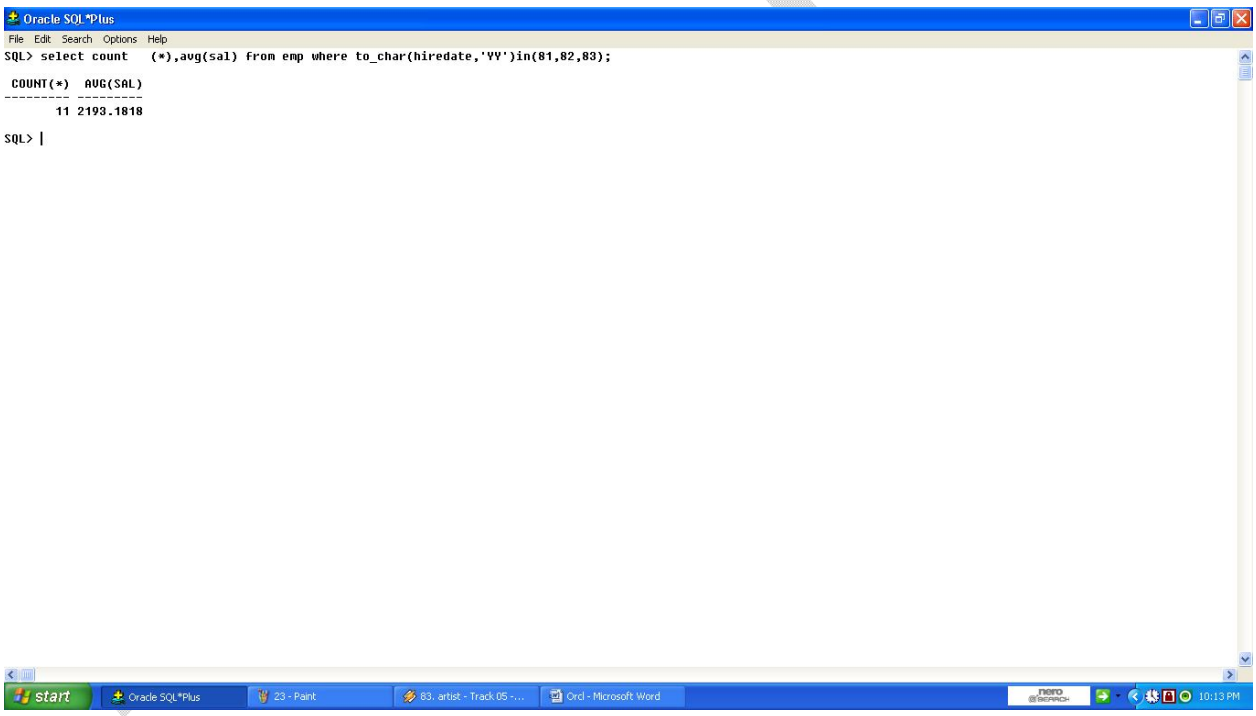
23.List maximum sal and minimum sal in the designation SALESMAN and CLERK.

SQL> select count(*),max(sal),avg(sal) from emp where job in('SALESMAN','CLERK');



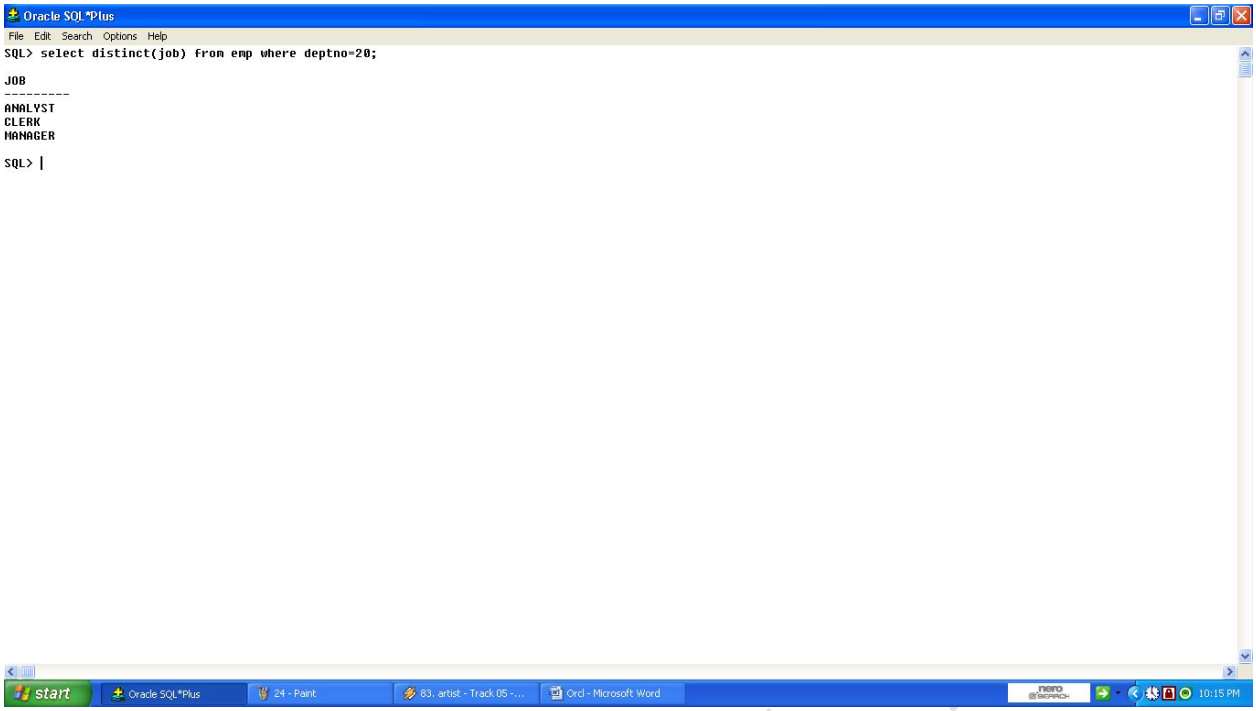
24.List the numbers of people and average salary of employee joined in 81,82 and 83.

SQL>select count (*),avg(sal) from emp where to_char(hiredate,'YY')in(81,82,83);



25.List jobs that are unique to deptno 20 .

SQL>select distinct(job) from emp where deptno=20;



26.List employee names and their joining date in the folloeing formats

A.SMITH 17th DEC NINETEEEN EIGHTY

B.SMITH SEVENTEENTH DEC NINTEEN EIGHTY

C.SMITH week day of joining

D.SMITH 17/12/80

A.select to_char(hiredate,'ddth mon year') from emp where ename like 'SMITH';

B. select to_char(hiredate,'ddspth mon year') from emp where ename like 'SMITH';

C. select to_char(hiredate,'day') from emp where ename like 'SMITH';

D. select to_char(hiredate,'dd/mm/yy') from emp where ename like 'SMITH';

```
SQL> select to_char(hiredate,'ddth mon year') from emp where ename like 'SMITH';

TO_CHAR(HIREDATE,'DDTHMONYEAR')
-------------------------------------------------
17th dec nineteen eighty

SQL>  select to_char(hiredate,'ddspth mon year') from emp where ename like 'SMITH';

TO_CHAR(HIREDATE,'DDSPTHMONYEAR')
---------------------------------------------------------------
seventeenth dec nineteen eighty

SQL> select to_char(hiredate,'day') from emp where ename like 'SMITH';

TO_CHAR(H
---------
wednesday

SQL> select to_char(hiredate,'dd/mm/yy') from emp where ename like 'SMITH';

TO_CHAR(
--------
17/12/80

SQL>
```

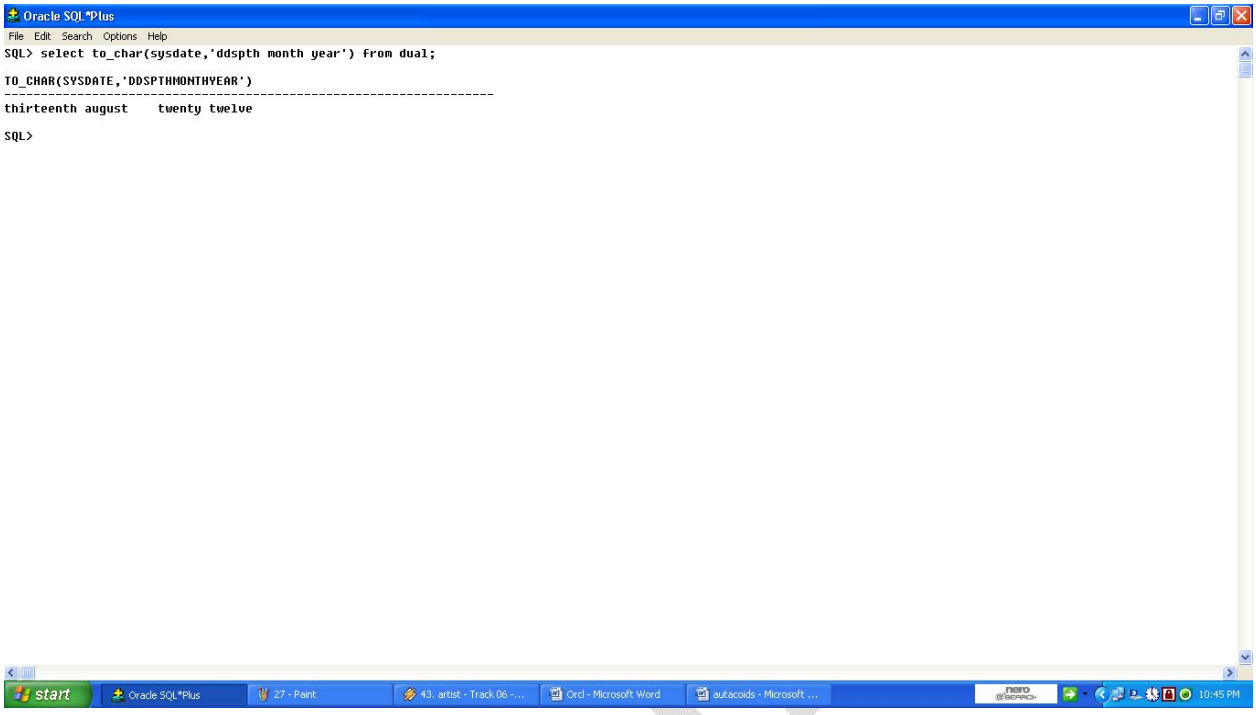
27.List employee names and their experience in yesrs.

SQL>select ename,round(months_between(sysdate,hiredate)/12) exp from ;

```
SQL> select ename,round(months_between(sysdate,hiredate)/12) exp from emp;

ENAME            EXP
---------- ---------
SMITH             32
ALLEN             31
WARD              31
JONES             31
MARTIN            31
BLAKE             31
CLARK             31
SCOTT             25
KING              31
TURNER            31
ADAMS             25
JAMES             31
FORD              31
MILLER            31

14 rows selected.

SQL> |
```

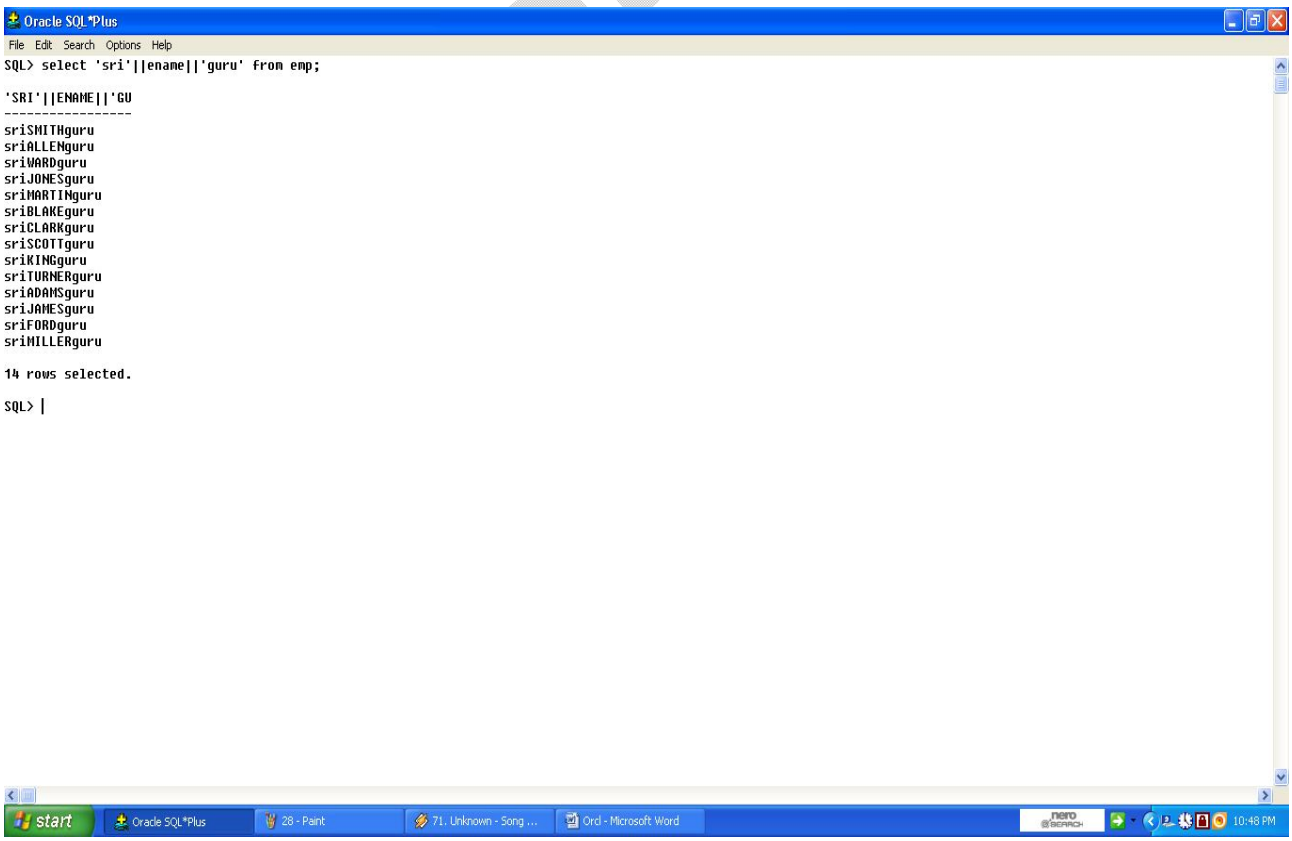
28.Display a given date as a string in different formats.

SQL>select to_char(sysdate,'ddspth month year') from dual;



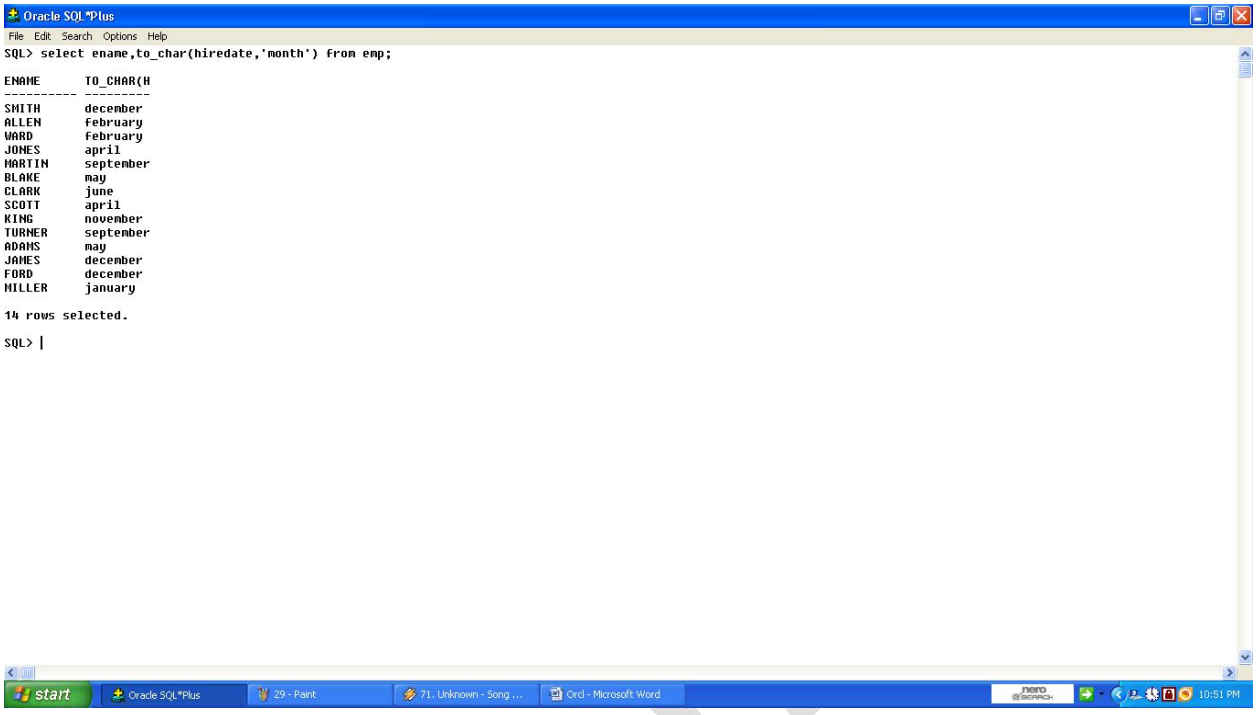
29. List employee names with length of the names sorted on length.

SQL>select 'sri'||ename||'guru' from emp;



30.List employee names with length of the name sorted on lengrh.

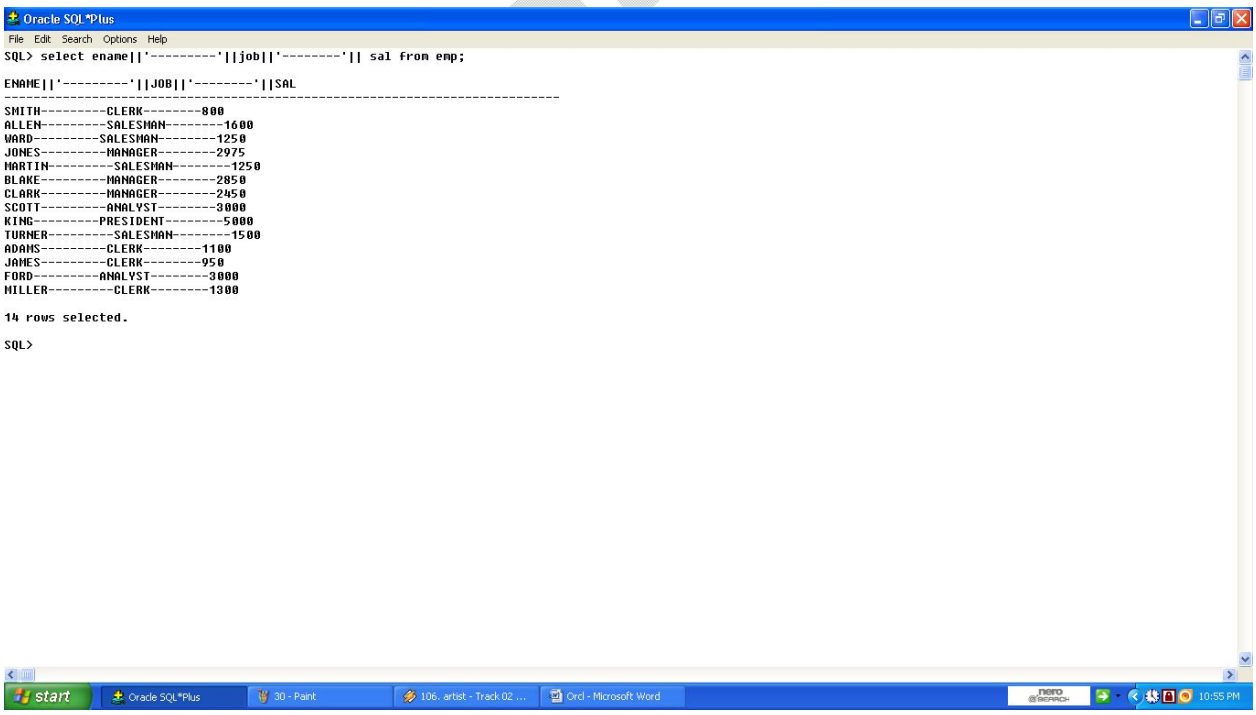
SQL>select ename,to_char(hiredate,'month') from emp;



```
SQL> select ename,to_char(hiredate,'month') from emp;

ENAME      TO_CHAR(H
---------- ---------
SMITH      december
ALLEN      february
WARD       february
JONES      april
MARTIN     september
BLAKE      may
CLARK      june
SCOTT      april
KING       november
TURNER     september
ADAMS      may
JAMES      december
FORD       december
MILLER     january

14 rows selected.

SQL> |
```

31.List employee names with length of the name sorted on lengrh.

SQL>select ename||'--------'||job||'-------'|| sal from emp;



```
SQL> select ename||'---------'||job||'--------'|| sal from emp;

ENAME||'---------'||JOB||'--------'||SAL
--------------------------------------------------------------------------------
SMITH---------CLERK--------800
ALLEN---------SALESMAN--------1600
WARD---------SALESMAN--------1250
JONES---------MANAGER--------2975
MARTIN---------SALESMAN--------1250
BLAKE---------MANAGER--------2850
CLARK---------MANAGER--------2450
SCOTT---------ANALYST--------3000
KING---------PRESIDENT--------5000
TURNER---------SALESMAN--------1500
ADAMS---------CLERK--------1100
JAMES---------CLERK--------950
FORD---------ANALYST--------3000
MILLER---------CLERK--------1300

14 rows selected.

SQL>
```

32.List employee names and the string with out first character and last character in their name.

SQL>select ename,instr(ename,'I') from emp;

```
SQL> select ename,instr(ename,'I') from emp;

ENAME      INSTR(ENAME,'I')
---------- ----------------
SMITH                     3
ALLEN                     0
WARD                      0
JONES                     0
MARTIN                    5
BLAKE                     0
CLARK                     0
SCOTT                     0
KING                      2
TURNER                    0
ADAMS                     0
JAMES                     0
FORD                      0
MILLER                    2

14 rows selected.

SQL>
```

33.SQL>select deptno,min(sal),max(sal),avg(sal) from emp where deptno in(10,30) group by deptno;

```
SQL> select deptno,min(sal),max(sal),avg(sal) from emp where deptno in(10,30) group by deptno;

   DEPTNO  MIN(SAL)  MAX(SAL)  AVG(SAL)
--------- --------- --------- ---------
       10      1300      5000 2916.6667
       30       950      2850 1566.6667

SQL>
```

34.SQL>select deptno,min(sal),max(sal),avg(sal) from emp group by deptno;

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> select deptno,min(sal),max(sal),avg(sal) from emp group by deptno;

   DEPTNO  MIN(SAL)  MAX(SAL)  AVG(SAL)
--------- --------- --------- ---------
       10      1300      5000 2916.6667
       20       800      3000      2175
       30       950      2850 1566.6667

SQL> |
```

35.SQL>select job,min(sal),max(sal),avg(sal) from emp group by job;

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> select job,min(sal),max(sal),avg(sal) from emp group by job;

JOB        MIN(SAL)  MAX(SAL)  AVG(SAL)
--------- --------- --------- ---------
ANALYST        3000      3000      3000
CLERK           800      1300    1037.5
MANAGER        2450      2975 2758.3333
PRESIDENT      5000      5000      5000
SALESMAN       1250      1600      1400

SQL>
```

36.SQL>select deptno,min(sal),max(sal),count(8) from emp group by deptno having count(*)>=2;

```
SQL> select deptno,min(sal),max(sal),count(8) from emp group by deptno having count(*)>=2;

   DEPTNO  MIN(SAL)  MAX(SAL)  COUNT(8)
--------- --------- --------- --------
       10      1300      5000        3
       20       800      3000        5
       30       950      2850        6

SQL>
```

37.List employee names and dept names with which they are associated.

SQL>select ename,dname from emp,dept where emp.deptno=dept.deptno;

```
SQL> select ename,dname from emp,dept where emp.deptno=dept.deptno;

ENAME      DNAME
---------- --------------
SMITH      RESEARCH
ALLEN      SALES
WARD       SALES
JONES      RESEARCH
MARTIN     SALES
BLAKE      SALES
CLARK      ACCOUNTING
SCOTT      RESEARCH
KING       ACCOUNTING
TURNER     SALES
ADAMS      RESEARCH
JAMES      SALES
FORD       RESEARCH
MILLER     ACCOUNTING

14 rows selected.

SQL>
```

D) Sql Operators (Simple-complex conditions):

1)Between,And:

SELECT FNAME, SALARY FROM EMPLOYEE WHERE SALARY BETWEEN 20000 AND 30000;

```
Run SQL Command Line
SQL> SELECT FNAME,SALARY FROM EMPLOYEE WHERE SALARY BETWEEN 20000 AND 30000;
FNAME        SALARY
-------- ----------
JOHN          30000
SQL>
```

2)Like:

1. SELECT FNAME FROM EMPLOYEE WHERE LNAME LIKE'S%';

```
Run SQL Command Line
SQL> SELECT FNAME FROM EMPLOYEE WHERE LNAME LIKE 'S%';
FNAME
--------
JOHN
SQL> _
```

2. Write a query to retrieve all employees who were born during the 1950s

.SQL> select * from employee where bdate like '%5_';

FNAME M LNAME SSN BDATE ADDRESS S SALARY SUPERSSN DNO

--- - --- --- --- --- - --- --- ---

Franklin T Wong 333445555 08-DEC-55 638 Voss, Houston, TX M 40000 888665555 5

3)IN:

SELECT FNAME FROM EMPLOYEE WHERE DNO IN(1,4,5);

```
Run SQL Command Line
SQL> SELECT FNAME FROM EMPLOYEE WHERE DNO IN (1,4,5);
FNAME
--------
JOHN
FRANKLIN
ALICIA
JENNIFER
JOYCE
AHMED

6 rows selected.
SQL>
```

4) NOT:

SELECT FNAME FROM EMPLOYEE WHERE DNO NOT IN(1,2,3);

E) ASC-DESC ordering combinations:

1) Write a query to retrieve names and salaries of employees in the descending order of their salaries.

A.) select fname,salary from employee order by salary desc;

SQL> /

FNAME	SALARY
Ramesh	38000
James	55000
Jennifer	43000
John	30000
Joyce	25000
Franklin	40000
Ahmad	25000

7 rows selected.

2) Write a query to retrieve names and salaries of employees in the Ascending order of their names.

A.) select fname,salary from employee order by salary Asc;

SQL>

FNAME	SALARY
Ahmad	25000
Franklin	40000
John	30000
Joyce	25000

Ramesh 38000

5 rows selected.

F) Renaming attributes:

1) Rename Employee table to Employees?

A) Rename Employee to Employees;

```
Run SQL Command Line

SQL> RENAME EMPLOYEE TO EMPLOYEES;

Table renamed.

SQL> SELECT * FROM EMPLOYEE;
SELECT * FROM EMPLOYEE
              *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL> SELECT * FROM EMPLOYEES;

FNAME    M LNAME           SSN BDATE     ADDRESS                        SALARY
-------- - -------- --------- --------- ------------------------- ----------
  SUPERSSN        DNO
---------- ----------
JOHN     B SMITH     123456789 09-JAN-65 731 FONDREN,HOUSTON,TX         30000
 333445555          5

FRANKLIN T WONG      333445555 08-DEC-55 638 VOSS,HOUSTON,TX            30006
 333445555          5

ALICIA   J ZELAYA    999887777 19-JAN-68 3321 CASTLE,SPRING,TX          45000
 987654321          4

JENNIFER S WALLACE   987654321 02-JUN-41 291 BERRY,BELLAIRE,TX          32000
 987654321          4

JOYCE    A ENGLISH  4534534531 31-JUL-72 5631 RICE,HOUSTON,TX           12000
 333445555          5

AHMED    V JABBAR   9817987987 29-MAR-62 980 DALLAS,HOUSTON,TX          45000
 987654321          5


6 rows selected.
```

G) Delete command:

Delete from employees where ssn=123456789;

```
Run SQL Command Line

SQL> DELETE FROM EMPLOYEES WHERE SSN=123456789;

1 row deleted.

SQL> SELECT * FROM EMPLOYEES;

FNAME    M LNAME           SSN BDATE     ADDRESS                        SALARY
-------- - -------- --------- --------- ------------------------- ----------
  SUPERSSN        DNO
---------- ----------
FRANKLIN T WONG      333445555 08-DEC-55 638 VOSS,HOUSTON,TX            30006
 333445555          5

ALICIA   J ZELAYA    999887777 19-JAN-68 3321 CASTLE,SPRING,TX          45000
 987654321          4

JENNIFER S WALLACE   987654321 02-JUN-41 291 BERRY,BELLAIRE,TX          32000
 987654321          4

JOYCE    A ENGLISH  4534534531 31-JUL-72 5631 RICE,HOUSTON,TX           12000
 333445555          5

AHMED    V JABBAR   9817987987 29-MAR-62 980 DALLAS,HOUSTON,TX          45000
 987654321          5

SQL>
```

H) ALTER: Used to modify or add or delete an attribute

1. To add a column for department table

Alter table department add dspl varchar2(5);

Table altered

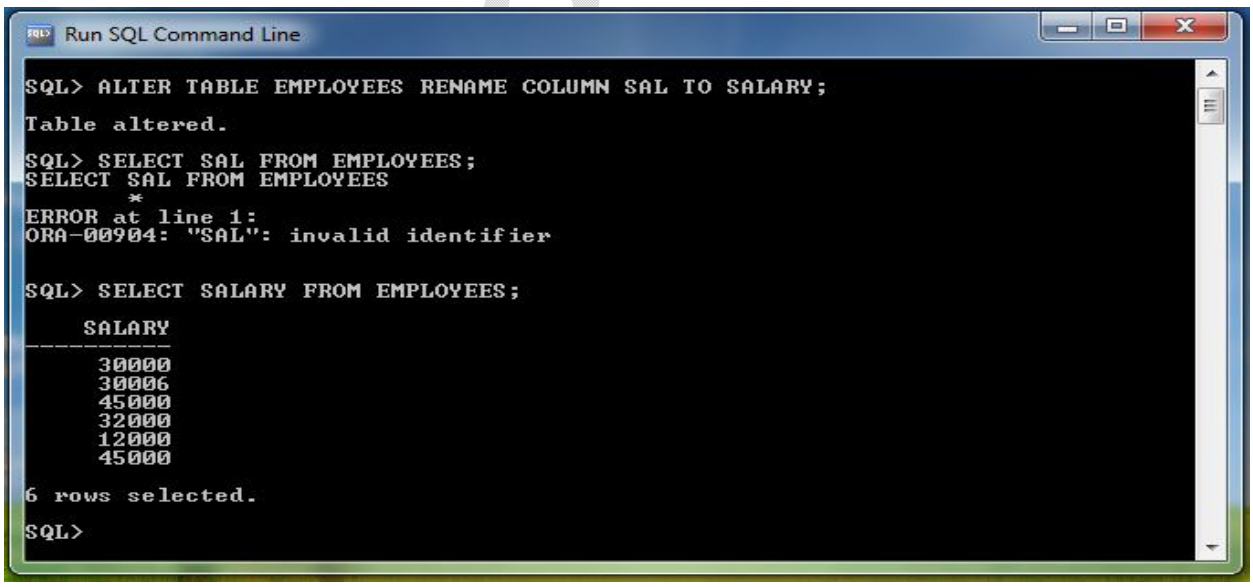
Name	Null?	Type
DNAME		VARCHAR2(5)
DNO		NUMBER(4)
DLOC		VARCHAR2(8)
DSPL		VARCHAR2(5)

2. To modify already exsisting column

Alter table department modify dloc varchar2(10);

Table altered

Name	Null?	Type
DNAME		VARCHAR2(5)
DNO		NUMBER(4)
DLOC		VARCHAR2(10)
DSPL		VARCHAR2(5)



3. To delete a column

Alter table dept drop column dspl;

Table altered

Name	Null?	Type
DNAME		VARCHAR2(5)
DNO		NUMBER(4)
DLOC		VARCHAR2(10)

I) DROP: used to delete a table.

Syntax: Drop table tablename;
Ex:
Drop table department;
Table deleted.

J) **SQL FUNCTIONS:**

These functions are used to manipulating data items and returning the results.

1. Group functions or Aggregate functions.
2. Single Row or scalar function.

Group functions or Aggregate functions:

These functions operated a set of values or rows

i. Sum()
ii. Avg()
iii. Min()
iv. Max()
v. Count()

Sum():used to find out the sum of the salary of employees.

Ex:List the sum of the salary of employees

Select sum(sal) from emp;

Avg():it find out the average salaries of employees.

Ex:List the average salary of the employees

Select avg(sal) from emp;

Min():used to find out the minimum salary of an employee in the given table.

Ex:list out the minimum salary of an employee in the emp table.

Select min(sal) from emp;

Max():used to find out the maximum salary of an employee in the given table.

Ex:list out the maxiimum salary of an employee in the emp table.

Select max(sal) from emp;

Count():used to list out the number of values in a particular table.

Ex:

1.List the numbers of jobs.

select count (job) from emp;

2.List the numbers of people and average salary in deptno 30.

select count(*),avg(sal) from emp where deptno=30;

Single Row or scalar function:These functions are operated a single row at a time.

Abs(): find the absolute value.

Select abs(10) from dual;

ABS(10)
10

Power():find the power.

Select power(2,3) from dual;

POWER(2,3)
8

Sqrt():find the square root of a given value.

Select sqrt(9) from dual;

SQRT(9)
3

Round()-find the round of the value.

Select round(12.35,1) from dual;

ROUND(12.35,1)
12.4

Truncate():find the truncate value.

Select trunc(12.35,1) from dual;

TRUNC(12.35,1)
12.3

Exp():used to find the exponential of given number.

Select exp(3) from dual;

EXP(3)
20.0855

Greastest():find out the greater value.

Select greatest(10,20,30) from dual;

GREATEST(10,20,30)
30

Least():find out the leastervalue.

Select least(10,20,30) from dual;

LEAST(10,20,30)
10

Mod():fina tha module of given numbers.

Select mod(3,2) from dual;

MOD(3,2)
1

Floor():find the floor value.

Select floor(12.56) from dual;

FLOOR(12.56)
12

Sign():find the sign of a number.

Select sign(-10) from dual;

SIGN(-10)
-1

Select sign(10) from dual;

SIGN(10)
1

Log():find logarthemic value.

Select log(3,2) from dual;

LOG(3,2)
.630929754

In these function we are using date functions also there are listed below:

Select months_between('26-jun-06','25-may-06') from dual;

MONTHS_BETWEEN('26-JUN-09','25-MAY-09')
1.03225806

Select add_months('26-jun-06',5') from dual;

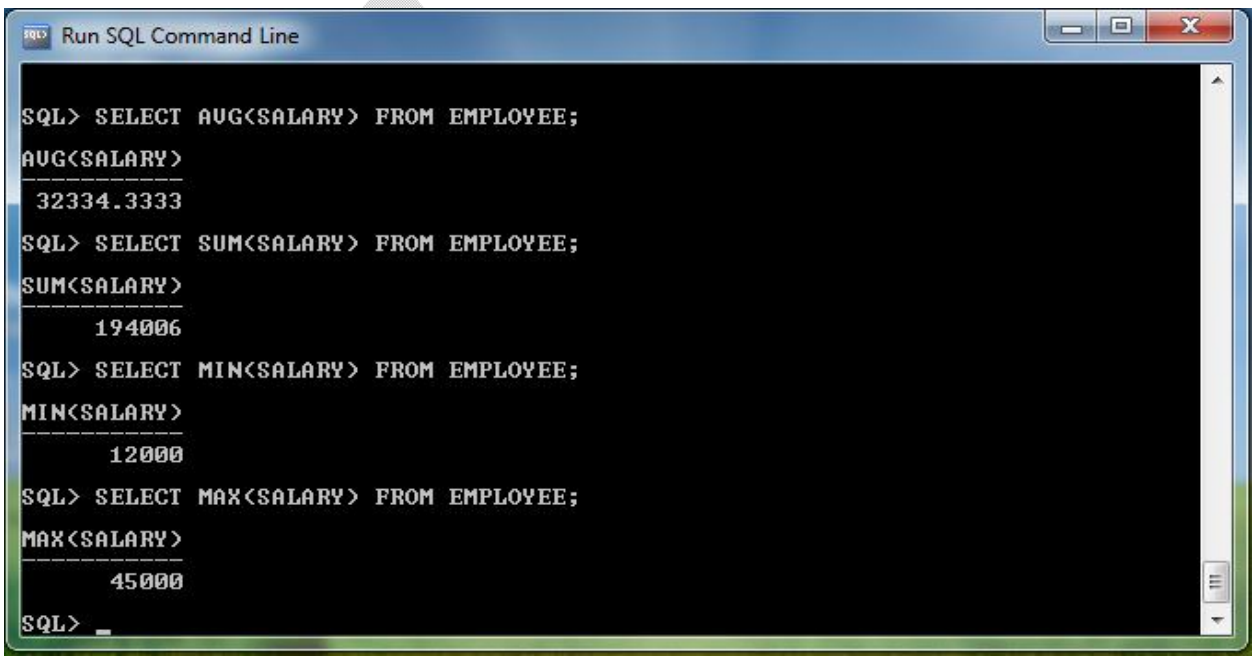
ADD_MONTH
26-NOV-06

Select next_day('26-jun-09','monday') from dual;

NEXT_DAY(
29-JUN-09

Select last_day('26-jun-09') from dual;

LAST_DAY(
30-JUN-09



```
SQL> SELECT COUNT(FNAME) FROM EMPLOYEE;
COUNT(FNAME)
------------
           6
SQL> SELECT POWER(3,2) FROM DUAL;
POWER(3,2)
----------
         9
SQL> SELECT SQRT(4) FROM DUAL;
   SQRT(4)
----------
         2
SQL> SELECT ROUND(100.2356,2) FROM DUAL;
ROUND(100.2356,2)
-----------------
           100.24
SQL> SELECT INITCAP('HELLO') FROM DUAL;
INITC
-----
Hello
SQL> SELECT LENGTH('ANIL') FROM DUAL;
LENGTH('ANIL')
--------------
             4
SQL> SELECT ASCII('A') FROM DUAL;
ASCII('A')
----------
        65
SQL>
```

Run SQL Command Line

```
SQL> SELECT LOWER('ABC') FROM DUAL;
LOW
---
abc
SQL> SELECT UPPER('abc') FROM DUAL;
UPP
---
ABC
SQL> SELECT MOD(22,20) FROM DUAL;
MOD(22,20)
----------
         2
SQL> SELECT DISTINCT(FNAME) FROM EMPLOYEE;
FNAME
--------
AHMED
JOHN
JOYCE
FRANKLIN
ALICIA
JENNIFER
6 rows selected.
SQL> _
```

EXP-III Multi-table queries (JOIN OPERATIONS)

1. Write a query to retrieve name and address of all employees who work for the 'Research' department.

```
FNAME          ADDRESS
--------------- ------------------------------
John           731 fondren, Houston, TX
Joyce          6531 Rice, Houston, TX
Ramesh         975, Fire Oak, Humble, TX
Franklin       638 Voss, Houston, TX
```

2. Write a query to retrieve employee's first and last name and first and last name of his or her immediate supervisor.

SQL>select e1.fname,e1.lname,e2.fname "supervisor fname",e2.lname "supervisor lname" from employee e1,employee e2 where e1.superssn=e2.ssn

```
FNAME          LNAME          supervisor fnam supervisor lnam
--------------- --------------- --------------- ---------------
John           Smith          Franklin        Wong
Ramesh         Narayan        Franklin        Wong
Joyce          English        Franklin        Wong
Franklin       Wong           James           Borg
Jennifer       Wallace        James           Borg
Alicia         Zelaya         Jennifer        Wallace
Ahmad          Jabbar         Jennifer        Wallace
```

OR

(USING OUTER JOIN)

SQL>select e1.fname ,e1.lname, e2.fname "supervisor fname", e2.lname "supervisor lname" from employee e1,employee e2 where e1.superssn= e2.ssn(+)

```
FNAME          LNAME          supervisor fnam supervisor lnam
--------------- --------------- --------------- ---------------
John           Smith          Franklin        Wong
Ramesh         Narayan        Franklin        Wong
Joyce          English        Franklin        Wong
```

Franklin	Wong	James	Borg
Jennifer	Wallace	James	Borg
Alicia	Zelaya	Jennifer	Wallace
Ahmad	Jabbar	Jennifer	Wallace
James	Borg		

8 rows selected.

NOTE: + is specified in above using 'outer join' to an attribute in join condition where we don't have null value.

Ex: ssn don't have null value but superssn has. So, I gave (+) to ssn.

3. Write a query to retrieve list of employees and the projects they are working on, ordered by department and with in each department, ordered alphabetically by last name, first name.

SQL>select e.ssn,e.fname,e.lname,w.pno,e.dno from employee e,works_on w where e.ssn=w.essn order by dno,fname,lname;

SSN	FNAME	LNAME	PNO	DNO
888665555	James	Borg	20	1
987987987	Ahmad	Jabbar	10	4
987987987	Ahmad	Jabbar	30	4
999887777	Alicia	Zelaya	30	4
999887777	Alicia	Zelaya	10	4
987654321	Jennifer	Wallace	30	4
987654321	Jennifer	Wallace	20	4
333445555	Franklin	Wong	2	5
333445555	Franklin	Wong	3	5
333445555	Franklin	Wong	10	5
333445555	Franklin	Wong	20	5
123456789	John	Smith	1	5
123456789	John	Smith	2	5
453453453	Joyce	English	1	5

16 rows selected.

4. For every project located in 'Stafford', list the project number, the controlling department number and the department manager's last name, birth date.

SQL>select p.pnumber,p.dnum,e.lname,e.bdate from employee e,department d,project p where p.plocation='Stafford' and p.dnum=d.dnumber and d.mgrssn=e.ssn

```
PNUMBER     DNUM LNAME          BDATE
-------- --------- --------------- ---------
      10         4 Wallace         20-JUN-41
      30         4 Wallace         20-JUN-41
```

5. Find the sum of the salaries of all employees, the maximum salary, the minimum salary and the average salary.

SQL> select sum(salary),max(salary),min(salary),avg(salary) from employee

```
SUM(SALARY) MAX(SALARY) MIN(SALARY) AVG(SALARY)
----------- ----------- ----------- -----------
    281000       55000       25000       35125
```

6. Find the sum of the salaries of all employees, the maximum salary, the minimum salary and the average salary of all employees of the 'Research' department.

SQL>select sum(salary),max(salary),min(salary),avg(salary) from employee e,department d where d.dname='Research' and d.dnumber=e.dno

```
SALARY) MAX(SALARY) MIN(SALARY) AVG(SALARY)
------- ----------- ----------- -----------
 133000       40000       25000       33250
```

7. Count the number of employees working in the 'Research' department.

SQL>select count(*) from employee e,department d where d.dname='Research' and d.dnumber=e.dno

```
  COUNT(*)
---------
         4
```

8. For each department, retrieve the department number, the number of employees in the department and their average salary.

```
SQL>select dno,count(dno),avg(salary) from employee group by dno
        DNO COUNT(DNO) AVG(SALARY)
--------- ---------- -----------
         1          1      55000
         4          3      31000
         5          4      33250  count(*) also  works
```

9. For each project, retrieve the project number, Project name and the number of employees who work on that project.

```
SQL>select pno,pname,count(pno) "Employees working" from project p,works_on w
where p.pnumber=w.pno group by pno,pname
        PNO PNAME          Employees working
--------- --------------- -----------------
         1 ProductX                    2
         2 ProductY                    3
         3 ProductZ                    2
        10 Computerization              3
        20 Reorganization               3
        30 Newbenefits                  3
6 rows selected.
```

10. For each project on which more than two employees work, retrieve the project number, project name and the number of employees who work on the project.

```
SQL>select pno,pname,count(pno) from project p,works_on w where p.pnumber=w.pno
group by pno,pname having count(pno)>2

        PNO PNAME          COUNT(PNO)
--------- ---------------      ----------
         2 ProductY                    3
        10 Computerization             3
        20 Reorganization              3
        30 Newbenefits                 3
```

11. For each project, retrieve the project number, Project name and the number of employees from department 5 who work on the project.

```
SQL>select pno,pname,count(pno) "Employees in dept:5" from employee e,project
 p,works_on w where w.pno=p.pnumber and w.essn=e.ssn and dno=5 group by
 pno,pname
```

PNO	PNAME	Employees in dept:5
1	ProductX	2
2	ProductY	3
3	ProductZ	2
10	Computerization	1
20	Reorganization	1

NOTE: “for each” given in problem implies to use group by clause.

count (any attribute)=count(*)

Ex: count(pno)=count(*)

EXP-IV Nested queries

1. Write a nested query to retrieve the name of each employee who has a dependent with the same first name and same sex as the employee.

A. SQL>select fname from employee e1 where ssn in (select ssn from dependant d where e1.fname=d.dependent_name and e1.sex=d.sex and e1.ssn=d.essn)

output:

no rows selected

2. [Using exists]

A . SQL>select fname from employee e1 where exists (select ssn from dependant d where e1.fname=d.dependent_name and e1.sex=d.sex and e1.ssn=d.essn)

OUTPUT:

No rows selected

3. Write a query to show resulting salaries if every employee working on 'ProductX' project is given a 10 percent raise.

A . SQL> select ssn,salary+.1*salary "10 % raise of salary" from employee where ssn in (select essn

```
        from works_on w
        where pno in (select pnumber
                      from project
                      where pname='ProductX'));
```

OUTPUT:

```
SSN       10 % raise of salary
--------- --------------------
123456789          33000
453453453          27500
```

4. For each department that has more than two employees, retrieve the department number and the number of its employees who are making more than or equal to $30,000.

A . SQL> select dno,dname,count(*) from employee e,department d where e.dno=d.dnumber and salary>=30000 and dno in(select e2.dno from employee e2 group by e2.dno having count(*) >2) group by dno,dname

OUTPUT:

```
     DNO DNAME          COUNT(*)
---------- --------------- ----------
       4 Administration         1
       5 Research               3
```

5. Write a nested query to retrieve the names of employees who have no dependents.

SQL> select fname from employee e1 where e1.ssn not in(select essn from dependant d where e1.ssn=d.essn)

OUTPUT:

```
FNAME
---------------
Alicia
Ramesh
Joyce
Ahmad
James
```

6. Write a nested query to list the names of managers who have at least one dependent.

A . SQL> select fname from employee,department where mgrssn=ssn and exists(select * from dependant where mgrssn=essn)

OUTPUT:

```
 FNAME
---------------
Franklin
Jennifer
```

7. Write a nested query to retrieve the names of all employees who have two or more dependents.

A: SQL>

select fname from employee where ssn in(select essn from dependant where essn=ssn group by(essn) having count(*)>2)

OUTPUT:

```
FNAME
---------------
```

John

Franklin

8. Write a nested query to retrieve the SSNs of all employees who work on project number 1, 2 or 3.

A: SQL>

```
select ssn from employee where ssn in( select distinct essn  from works_on
        where pno in(1,2,3))
```

OUTPUT:

```
SSN
---------
123456789
333445555
453453453
666884444
```

NOTE: 'distinct' may or mayn't be present

9. Write a nested query to retrieve the names of all employees whose salary is greater than the salary of all the employees in department 5.

A: SQL> select fname from employee where salary > all (select salary from employee where dno=5)

OUTPUT:

```
FNAME
---------------
Jennifer
James
```

Note: while doing nested queries consider the output as inner and apply conditions one by one with each condition satisfying in one condition.

EXP-V Set Oriented Operations

1. Write a query to make a list of all project numbers for projects that involve an employee whose last name is 'Smith', either as a worker or as a manager of the department that controls the project.

SQL> select pnumber from project p where pnumber in(select pno from works_on w,employee e where w.essn=e.ssn and e.lname='Smith')UNION(select pnumber from project p1, department d,employee e where e.lname='Smith' and p1.dnum=d.dnumber and e.dno=d.dnumber)

OUTPUT:

```
  PNUMBER
----------
         1
         2
         3
```

2. Write a query to retrieve the SSNs of employees who worked on projects 1 and 2 but not on 3.

SQL> select ssn from employee e where e.ssn in ((select distinct essn from works_on w,employee e where w.essn=e.ssn and w.pno=1) UNION (select distinct essn from works_on w,employee e where w.essn=e.ssn and w.pno=2) MINUS(select distinct essn from works_on w,employee e where w.essn=e.ssn and w.pno=3))

OUTPUT:

```
SSN
---------
123456789
453453453
```

3. Write a query to retrieve the names of employee who worked on all the projects controlled by department 5.

SQL>

4. Without using a nested query, retrieve the names of employees who have no dependents.

SQL> select fname from employee e where e.ssn in ((select ssn from employee) MINUS (select distinct essn from dependant))

OUTPUT:

FNAME

Joyce
Ramesh
James
Ahmad
Alicia

5. SELECT * FROM student UNION SELECT * FROM std;

SNO	SNAME
1	kumar
2	ravi
3	ramu
5	lalitha
9	devi

6. SELECT * FROM student UNIONALL SELECT * FROM std;

SNO	SNAME
1	kumar
2	ravi
3	ramu
5	lalitha
9	devi

7. SQL> SELECT * FROM student INTERSECT SELECT * FROM std;

SNO	SNAME
1	Kumar

8. SELECT * FROM student MINUS SELECT * FROM std;

SNO	SNAME
2	RAVI
3	RAMU

EXP-VI DDL & TCL Commands

DATA DEFINITION LANGUAGE (DDL): The Data Definition Language (DDL) is used to create and destroy databases and database objects. These commands will primarily be used by database administrators during the setup and removal phases of a database project. Let's take a look at the structure and usage of four basic DDL commands:

1. CREATE 2. ALTER 3. DROP 4. RENAME

1. CREATE:

(a)CREATE TABLE: This is used to create a new relation and the corresponding

Syntax: CREATE TABLE relation_name

(field_1 data_type(Size),field_2 data_type(Size), .. .);

Example:

SQL>CREATE TABLE Student (sno NUMBER(3),sname CHAR(10),class CHAR(5));

(b)CREATE TABLE..AS SELECT....: This is used to create the structure of a new relation from the structure of an existing relation.

Syntax: CREATE TABLE (relation_name_1, field_1,field_2,.....field_n) AS SELECT field_1,field_2,.....field_n FROM relation_name_2;

Example: SQL>CREATE TABLE std(rno,sname) AS SELECT sno,sname FROM student;

2. ALTER:

(a)ALTER TABLE ...ADD...: This is used to add some extra fields into existing relation.

Syntax: ALTER TABLE relation_name ADD(new field_1 data_type(size), new field_2 data_type(size),..);

Example : SQL>ALTER TABLE std ADD(Address CHAR(10));

(b)ALTER TABLE...MODIFY...: This is used to change the width as well as data type of fields of existing relations.

Syntax: ALTER TABLE relation_name MODIFY (field_1 newdata_type(Size), field_2 newdata_type(Size),....field_newdata_type(Size));

Example:SQL>ALTER TABLE student MODIFY(sname VARCHAR(10),class VARCHAR(5));

3. DROP TABLE: This is used to delete the structure of a relation. It permanently deletes the records in the table.

Syntax: DROP TABLE relation_name;

Example: SQL>DROP TABLE std;

4. RENAME: It is used to modify the name of the existing database object.

Syntax: RENAME TABLE old_relation_name TO new_relation_name;

Example: SQL>RENAME TABLE std TO std1;

5. TRUNCATE: This command will remove the data permanently. But structure will not be removed.

Syntax: TRUNCATE TABLE <Table name>

Example TRUNCATE TABLE student;

Difference between Truncate & Delete:-

(A) By using truncate command data will be removed permanently & will not get back where as by using delete command data will be removed temporally & get back by using roll back command.

(B) By using delete command data will be removed based on the condition where as by using truncate command there is no condition.

(C) Truncate is a DDL command & delete is a DML command.

TRANSATIONAL CONTROL LANGUAGE (T.C.L):

A transaction is a logical unit of work. All changes made to the database can be referred to as a transaction. Transaction changes can be mode permanent to the database only if they are committed a transaction begins with an executable SQL statement & ends explicitly with either role back or commit statement.

1. COMMIT: This command is used to end a transaction only with the help of the commit command transaction changes can be made permanent to the database.

Syntax: SQL>COMMIT;

Example: SQL>COMMIT;

2. SAVE POINT: Save points are like marks to divide a very lengthy transaction to smaller once. They are used to identify a point in a transaction to which we can latter role back. Thus, save point is used in conjunction with role back.

Syntax: SQL>SAVE POINT ID;

Example: SQL>SAVE POINT xyz;

3. ROLE BACK: A role back command is used to undo the current transactions. We can role back the entire transaction so that all changes made by SQL statements are undo (or) role back a transaction to a save point so that the SQL statements after the save point are role back.

Syntax: ROLE BACK(current transaction can be role back)

ROLE BACK to save point ID;

Example: SQL>ROLE BACK;

SQL>ROLE BACK TO SAVE POINT xyz;

GRAND & REVOKE(DCL)

GRANT: The GRANT command allows granting various privileges to other users and allowing them to perform operations with in their privileges

For Example, if a uses is granted as 'SELECT' privilege then he/she can only view data but cannot perform any other DML operations on the data base object GRANTED privileges can also be withdrawn by the DBA at any time

Syntax: SQL>GRANT PRIVILEGES on object_name To user_name;

Example: SQL>GRANT SELECT, UPDATE on emp To hemanth;

2. REVOKE: To with draw the privileges that has been GRANTED to a uses, we use the REVOKE command

Syntax: SQL>REVOKE PRIVILEGES ON object-name FROM user_name;

Example: SQL>REVOKE SELECT, UPDATE ON emp FROM ravi;

VIEW: In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

A view is a virtual table, which consists of a set of columns from one or more tables. It is similar to a table but it doest not store in the database. View is a query stored as an object.

Syntax: CREATE VIEW view_name AS SELECT set of fields FROM relation_name WHERE (Condition)

1. Example:

```
SQL>CREATE VIEW employee AS SELECT empno,ename,job FROM EMP
        WHERE job = 'clerk';
        View created.
SQL> SELECT * FROM EMPLOYEE;
        EMPNO          ENAME          JOB
        ----           ------         -------
        7369           SMITH          CLERK
        7876           ADAMS          CLERK
        7900           JAMES          CLERK
        7934           MILLER         CLERK
```

2.Example:

```
CREATE VIEW [Current Product List] AS
SELECT ProductID,ProductName
FROM Products
WHERE Discontinued=No
```

DROP VIEW: This query is used to delete a view , which has been already created.

Syntax: DROP VIEW View_name;

Example : SQL> DROP VIEW EMPLOYEE;

View dropped

EXP-VII PL/SQL INTRODUCTION

PL/SQL stands for Procedural Language extension of SQL. PL/SQL is a combination of SQL along with the procedural features of programming languages. It was developed by Oracle Corporation in the early 90's to enhance the capabilities of SQL.

Oracle uses a PL/SQL engine to processes the PL/SQL statements. A PL/SQL code can be stored in the client system (client-side) or in the database (server-side).

Advantages of PL/SQL:

- ***Block Structures***: PL SQL consists of blocks of code, which can be nested within each other. Each block forms a unit of a task or a logical module. PL/SQL Blocks can be stored in the database and reused.
- ***Procedural Language Capability***: PL SQL consists of procedural language constructs such as conditional statements (if else statements) and loops like (FOR loops).
- ***Better Performance***: PL SQL engine processes multiple SQL statements simultaneously as a single block, thereby reducing network traffic.
- ***Error Handling***: PL/SQL handles errors or exceptions effectively during the execution of a PL/SQL program. Once an exception is caught, specific actions can be taken depending upon the type of the exception or it can be displayed to the user with a message.

Syntax of PL/SQL program:

```
Declare
        Variable declaration;
Begin
        Executable statements;
end;
```

Conditional Statements in PL/SQL

As the name implies, PL/SQL supports programming language features like conditional statements, iterative statements.

The programming constructs are similar to how you use in programming languages like Java and C++. In this section I will provide you syntax of how to use conditional statements in PL/SQL programming.

IF THEN ELSE STATEMENT:

1)
```
IF condition THEN
        Statement 1;
ELSE
        Statement 2;
END IF;
```

2)
```
IF condition 1 THEN
        Statement 1;
        Statement 2;
ELSIF condtion2 THEN
        Statement 3;
      ELSE
        Statement 4;
END IF
```

Loops in PL/SQL

There are three types of loops in PL/SQL:

1. Simple Loop
2. While Loop
3. For Loop

1. Simple Loop: A Simple Loop is used when a set of statements is to be executed at least once before the loop terminates. An EXIT condition must be specified in the loop, otherwise the loop will get into an infinite number of iterations. When the EXIT condition is satisfied the process exits from the loop.

Syntax:
```
LOOP
        Statements;
 EXIT;
        {or EXIT WHEN condition ;}
END LOOP;
```

2. While Loop: A WHILE LOOP is used when a set of statements has to be executed as long as a condition is true. The condition is evaluated at the beginning of each iteration. The iteration continues until the condition becomes false.

Syntax:
```
WHILE <condition>
        LOOP statements;
END LOOP;
```

3. FOR Loop: A FOR LOOP is used to execute a set of statements for a predetermined number of times. Iteration occurs between the start and end integer values given. The counter is always incremented by 1. The loop exits when the counter reaches the value of the end integer.

Syntax:
```
FOR counter IN val1..val2
        LOOP statements;
END LOOP;
```

VIII PL/SQL PROGRAMMING 1

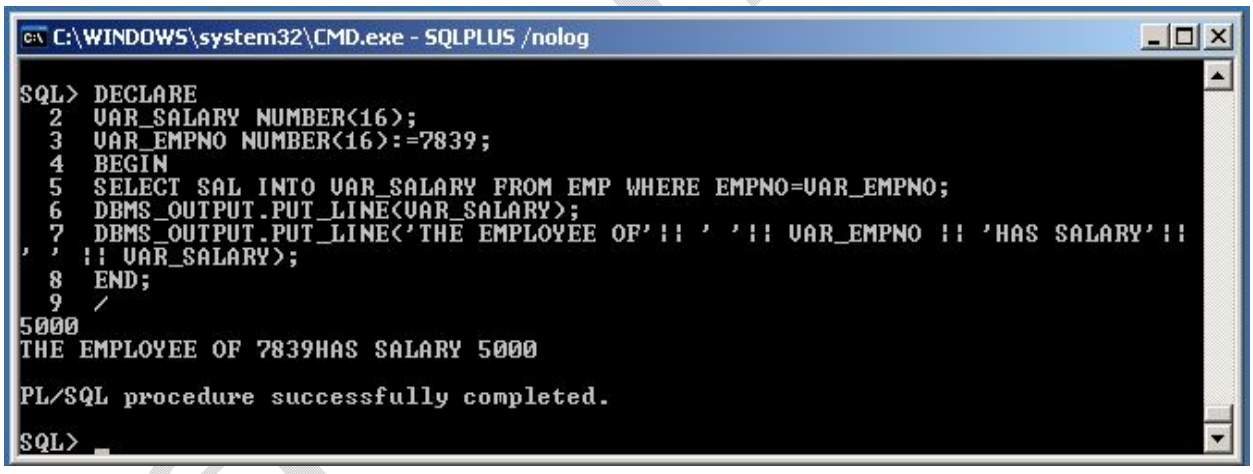
(i) Programs using named and unnamed blocks

Examples:

1) Write PL/SQL code for finding specific Employee salary in given table.

```
DECLARE
VAR_SALARY NUMBER(16);
VAR_EMPNO NUMBER(16):=7839;
BEGIN
SELECT SAL INTO VAR_SALARY FROM EMP WHERE EMPNO=VAR_EMPNO;
DBMS_OUTPUT.PUT_LINE(VAR_SALARY);
DBMS_OUTPUT.PUT_LINE('THE EMPLOYEE OF'|| ' '|| VAR_EMPNO || 'HAS SALARY'||
                                    ' ' || VAR_SALARY);
END;
/
```

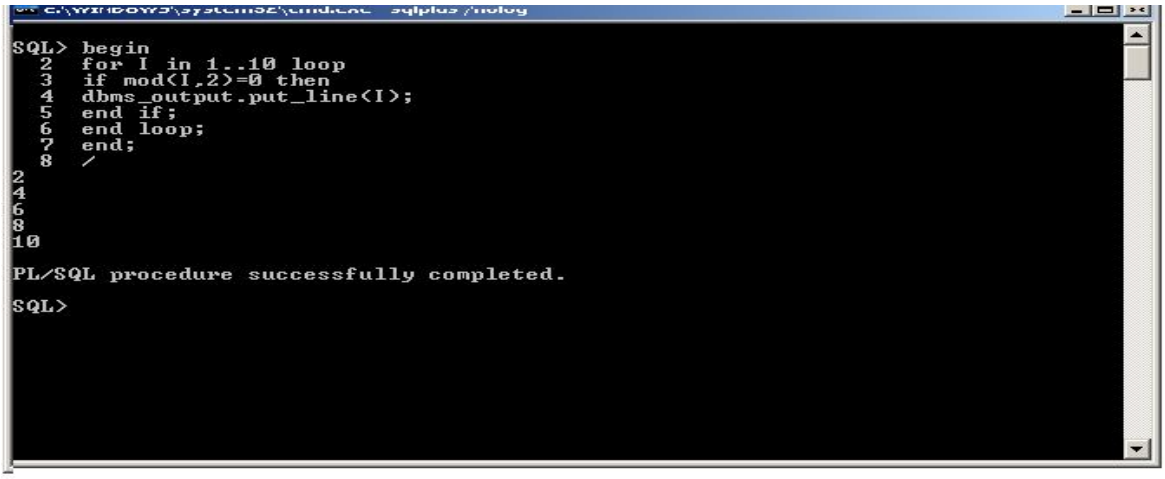
Output:



2) Write PL/SQL code for finding Even Numbers.

```
BEGIN
FOR I IN 1..100 LOOP
IF MOD(I,2)=0 THEN
DBMS_OUTPUT.PUT_LINE(I);
END IF;
END LOOP;
END;
/
```

Output:



3) Write PL/SQL code to find Largest of three numbers.

```
DECLARE
A NUMBER;
B NUMBER;
C NUMBER;
BEGIN
A:=&A;
B:=&B;
C:=&C;
IF A=B AND B=C AND C=A THEN
DBMS_OUTPUT.PUT_LINE('ALL ARE EQUAL');
ELSE IF A>B AND A>C THEN
DBMS_OUTPUT.PUT_LINE('A IS GREATER');
ELSE IF B>C THEN
DBMS_OUTPUT.PUT_LINE('B IS GREATER');
ELSE
DBMS_OUTPUT.PUT_LINE('C IS GREATER');
END IF;
END IF;
END IF;
END;
/
```

Output:

```
C:\WINDOWS\system32\cmd.exe - sqlplus /nolog

SQL> declare
  2  a number;
  3  b number;
  4  c number;
  5  begin
  6  a:=&a;
  7  b:=&b;
  8  c:=&c;
  9  if a=b and b=c and c=a then
 10  dbms_output.put_line('all are equal');
 11  else if a>b and a>c then
 12  dbms_output.put_line('a is greater');
 13  else if b>c then
 14  dbms_output.put_line('b is greater');
 15  else
 16  dbms_output.put_line('c is greater');
 17  end if;
 18  end if;
 19  end if;
 20  end;
 21  /
Enter value for a: 2
old   6: a:=&a;
new   6: a:=2;
Enter value for b: 4
old   7: b:=&b;
new   7: b:=4;
Enter value for c: 5
old   8: c:=&c;
new   8: c:=5;
c is greater

PL/SQL procedure successfully completed.

SQL>
```

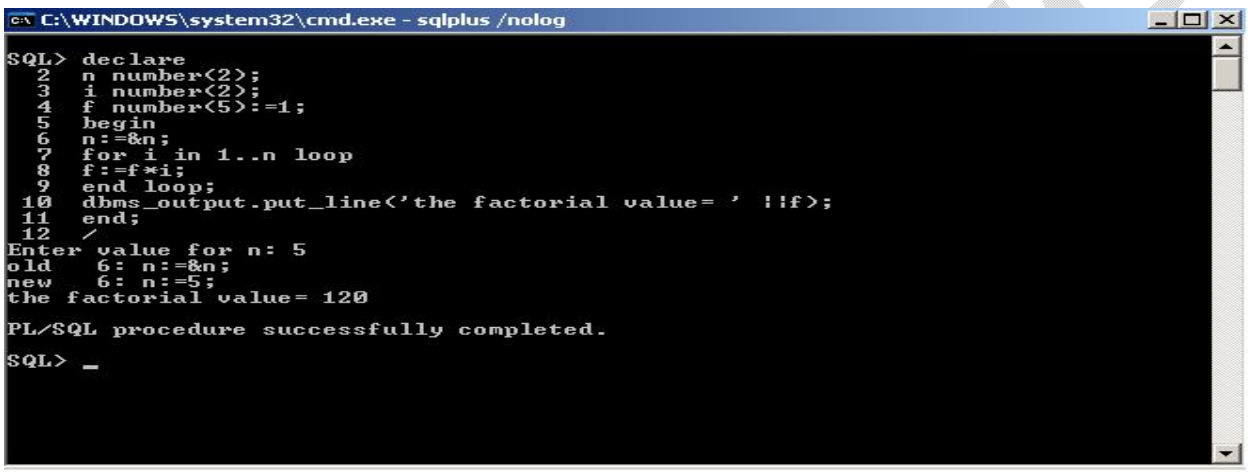
```
C:\WINDOWS\system32\CMD.exe - SQLPLUS /nolog
Enter value for a: 10
old   6: A:=&A;
new   6: A:=10;
Enter value for b: 10
old   7: B:=&B;
new   7: B:=10;
Enter value for c: 10
old   8: C:=&C;
new   8: C:=10;
ALL ARE EQUAL

PL/SQL procedure successfully completed.

SQL> /
Enter value for a: 10
old   6: A:=&A;
new   6: A:=10;
Enter value for b: 20
old   7: B:=&B;
new   7: B:=20;
Enter value for c: 30
old   8: C:=&C;
new   8: C:=30;
C IS GREATER

PL/SQL procedure successfully completed.

SQL> /
Enter value for a: 30
old   6: A:=&A;
new   6: A:=30;
Enter value for b: 20
old   7: B:=&B;
new   7: B:=20;
Enter value for c: 10
old   8: C:=&C;
new   8: C:=10;
A IS GREATER

PL/SQL procedure successfully completed.
```

4) Write PL/SQL code to find Factorial of a given number.

```
DECLARE
N NUMBER(2);
I NUMBER(2);
F NUMBER(5):=1;
BEGIN
```

```
N:=&N;
FOR I IN 1..N LOOP
F:=F*I;
END LOOP;
DBMS_OUTPUT.PUT_LINE('THE FACTORIAL VALUE IS =' || F);
END;
/
```

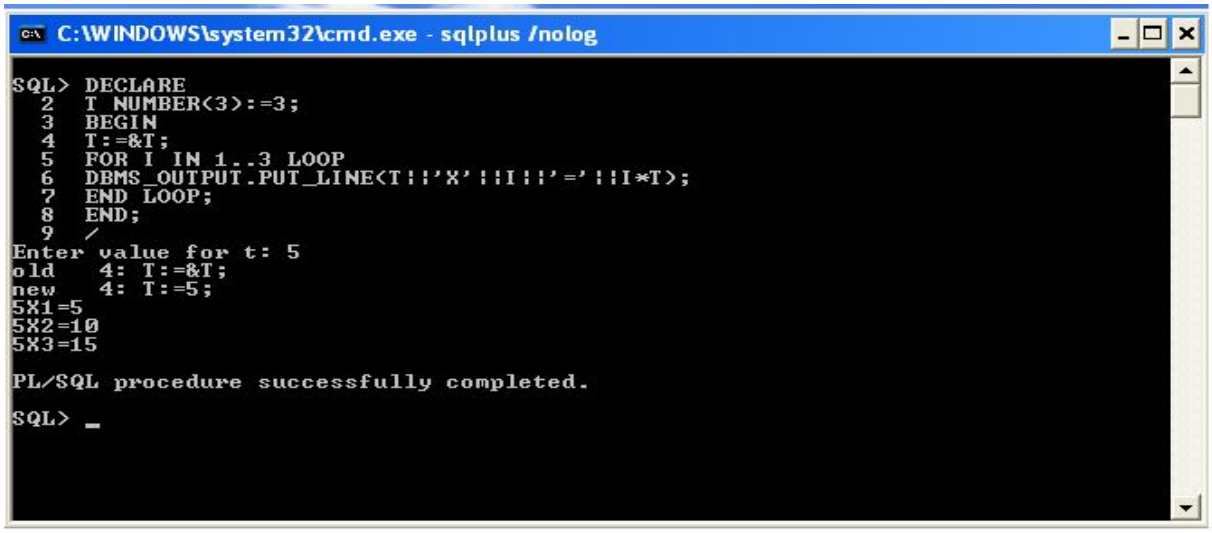
Output:



5) Write PL/SQL code to Read number and prints its Multiplication Table.

```
DECLARE
T NUMBER(3):=3;
BEGIN
T:=&T;
FOR I IN 1..3 LOOP
DBMS_OUTPUT.PUT_LINE(T||'X'||I||'='||I*T);
END LOOP;
END;
/
```

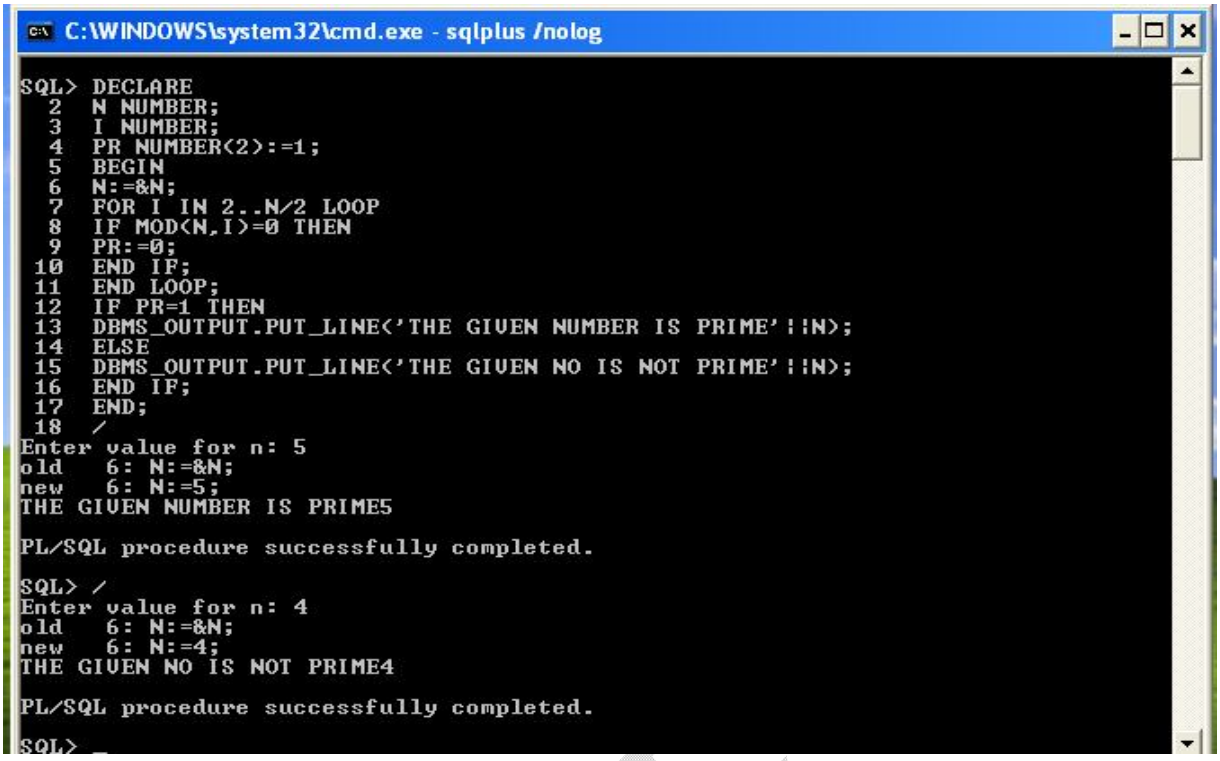
Output:



6) Write PL/SQL code to find given number is Prime or not.

```
DECLARE
N NUMBER;
I NUMBER;
PR NUMBER(2):=1;
BEGIN
N:=&N;
FOR I IN 2..N/2 LOOP
IF MOD(N,I)=0 THEN
PR:=0;
END IF;
END LOOP;
IF PR=1 THEN
DBMS_OUTPUT.PUT_LINE('THE GIVEN NUMBER IS PRIME'||N);
ELSE
DBMS_OUTPUT.PUT_LINE('THE GIVEN NO IS NOT PRIME'||N);
END IF;
END;
/
```

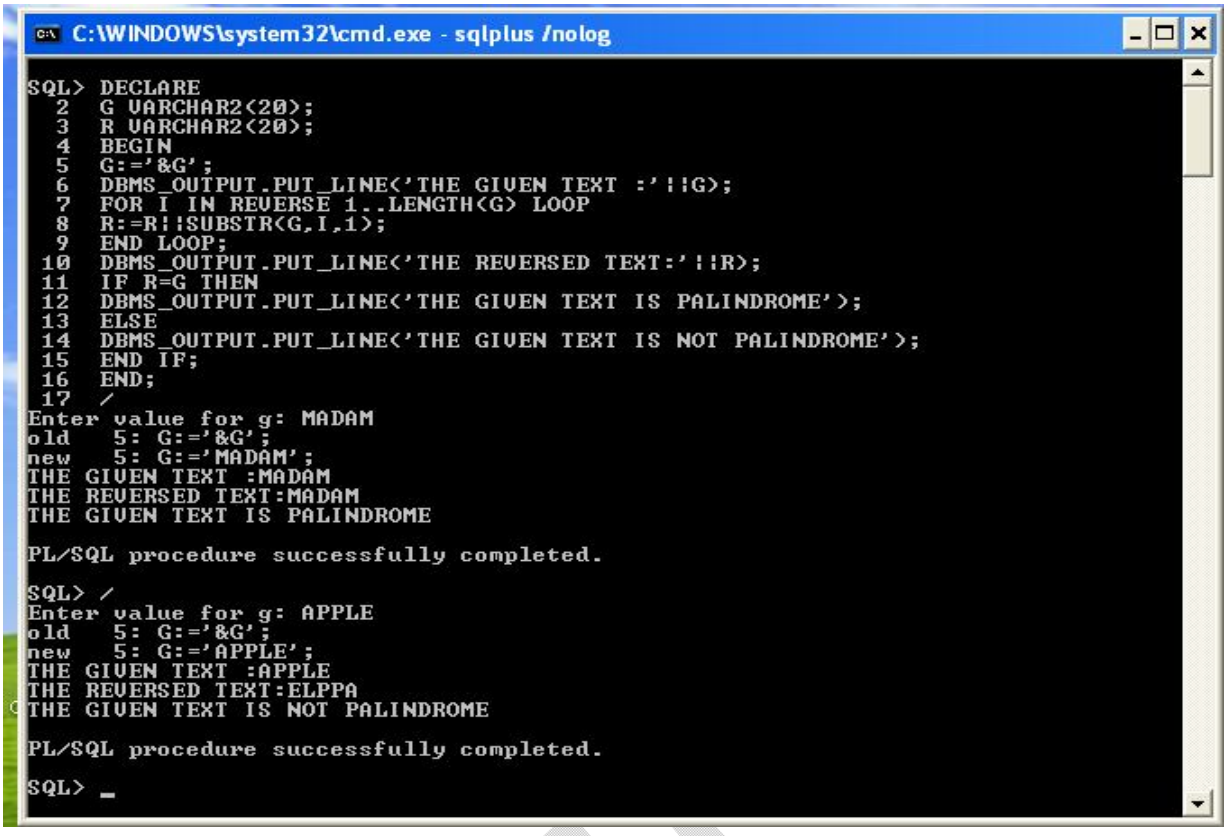
Output:



7) Write PL/SQL code to accept the text and reverse the text and test whether the given character is Palandrome or not.

```
DECLARE
G VARCHAR2(20);
R VARCHAR2(20);
BEGIN
G:='&G';
DBMS_OUTPUT.PUT_LINE('THE GIVEN TEXT :'||G);
FOR I IN REVERSE 1..LENGTH(G) LOOP
R:=R||SUBSTR(G,I,1);
END LOOP;
DBMS_OUTPUT.PUT_LINE('THE REVERSED TEXT:'||R);
IF R=G THEN
DBMS_OUTPUT.PUT_LINE('THE GIVEN TEXT IS PALINDROME');
ELSE
DBMS_OUTPUT.PUT_LINE('THE GIVEN TEXT IS NOT PALINDROME');
END IF;
END;
/
```

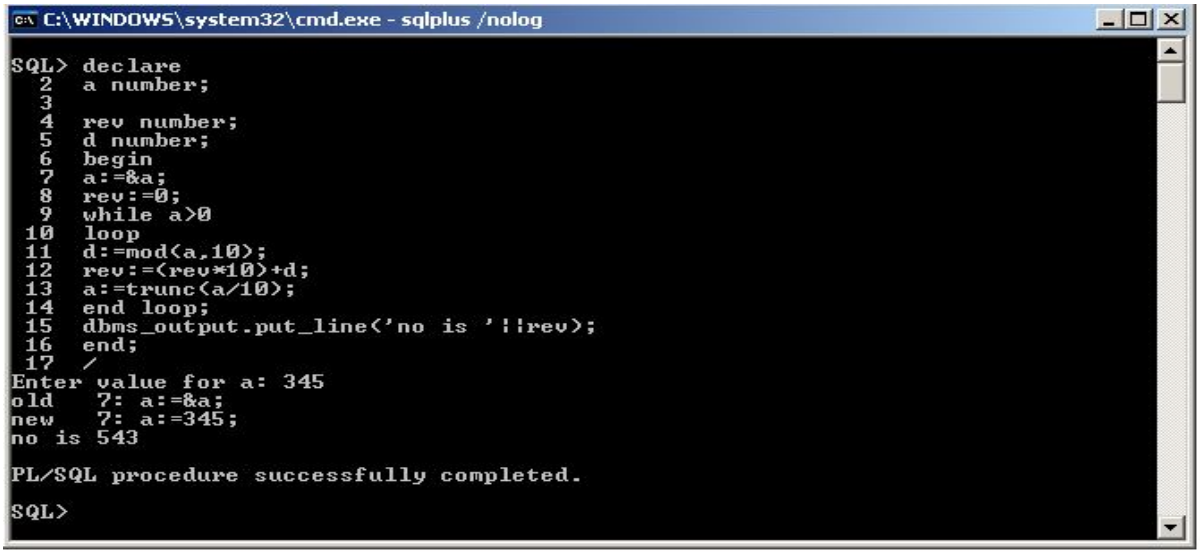
Output:



8) Write PL/SQL code to find Reverse of a given number.

```
DECLARE
A NUMBER;
REV NUMBER;
D NUMBER;
BEGIN
A:=&A;
REV:=0;
WHILE A>0
LOOP
D:=MOD(A,10);
REV:=(REV * 10) + D;
A:=TRUNC(A/10);
END LOOP;
DBMS_OUTPUT.PUT_LINE('NO IS'||REV);
END;
/
```

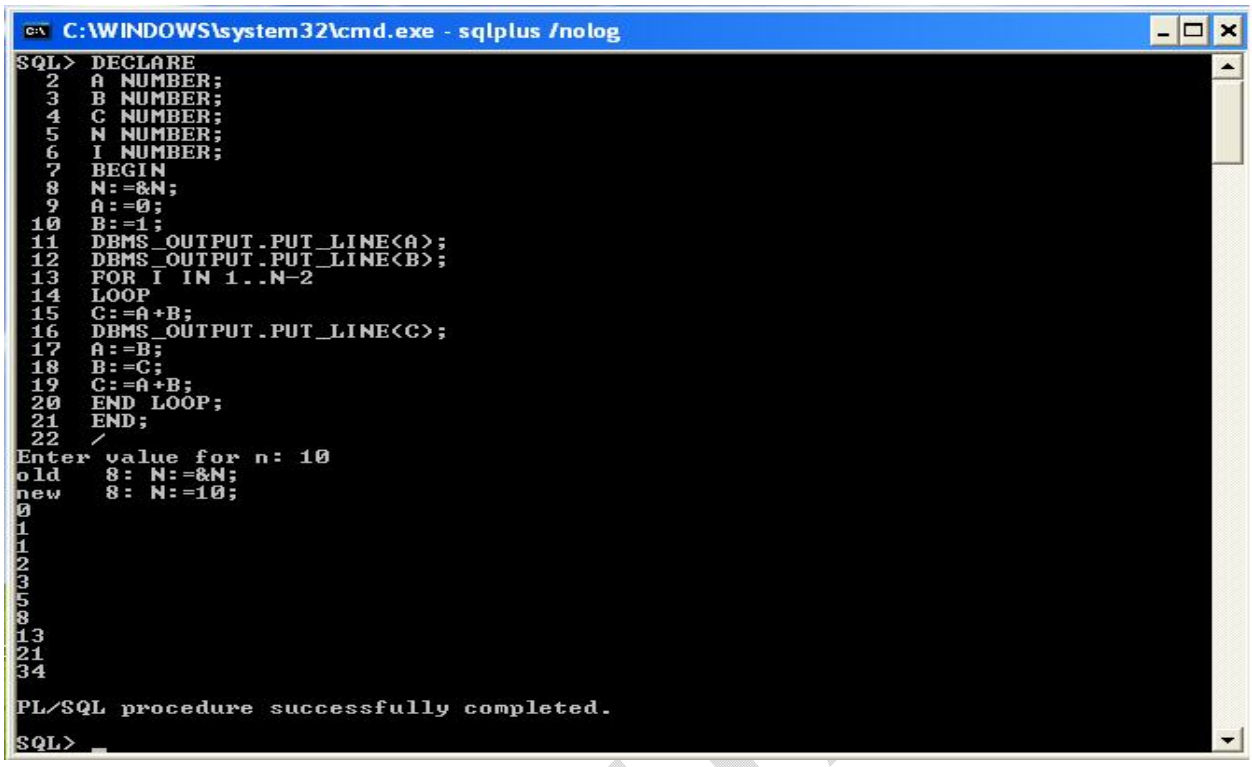
Output:



9) Write PL/SQL code to generate Fibonacci series for given number.

```
DECLARE
A NUMBER;
B NUMBER;
C NUMBER;
N NUMBER;
I NUMBER;
BEGIN
N:=&N;
A:=0;
B:=1;
DBMS_OUTPUT.PUT_LINE(A);
DBMS_OUTPUT.PUT_LINE(B);
FOR I IN 1..N-2
LOOP
C:=A+B;
DBMS_OUTPUT.PUT_LINE(C);
A:=B;
B:=C;
C:=A+B;
END LOOP;
END;
/
```

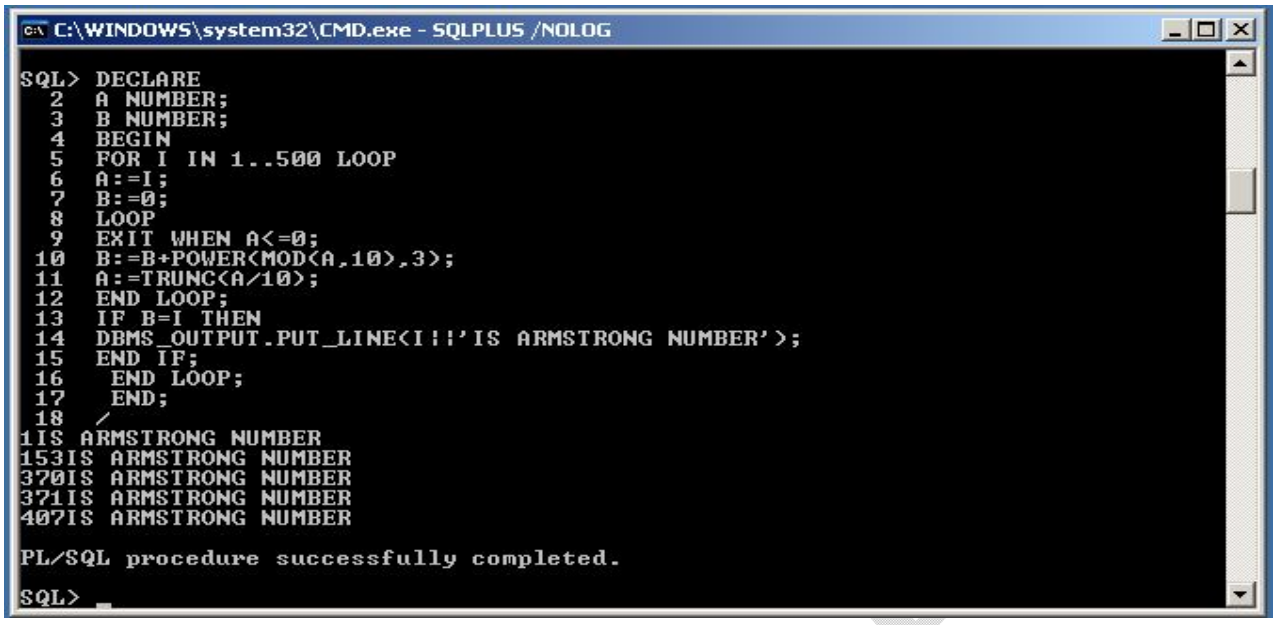
Output:



10)Write PL/SQL code to find Armstrong numbers from 1 to 500.

```
DECLARE
A NUMBER;
B NUMBER;
BEGIN
FOR I IN 1..500 LOOP
A:=I;
B:=0;
LOOP
EXIT WHEN A<=0;
B:=B+POWER(MOD(A,10),3);
A:=TRUNC(A/10);
END LOOP;
IF B=I THEN
DBMS_OUTPUT.PUT_LINE(I||'IS ARMSTRONG NUMBER');
END IF;
 END LOOP;
 END;
/
```

Output:



11)Write PL/SQL code to print the numbers in this form

1
1 2
123

```
DECLARE
I NUMBER;
J NUMBER;
N NUMBER;
BEGIN
N:=&N;
FOR I IN 1..N LOOP
FOR J IN 1..I LOOP
DBMS_OUTPUT.PUT(J);
END LOOP;
DBMS_OUTPUT.PUT_LINE('');
END LOOP;
END;
/
```

Output:

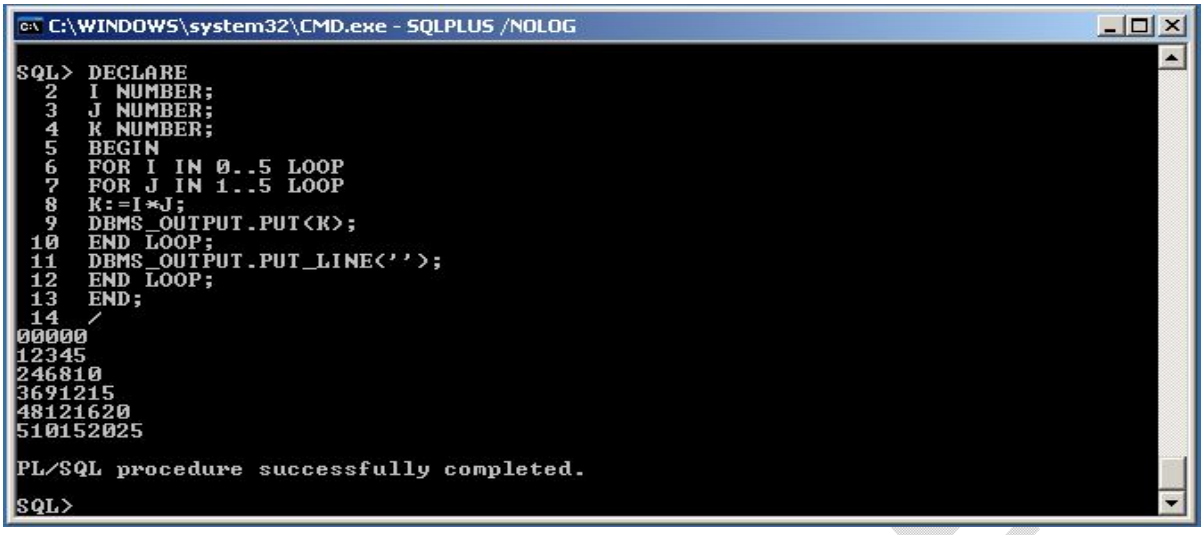
```
C:\WINDOWS\system32\CMD.exe - SQLPLUS /NOLOG

SQL> DECLARE
  2  I NUMBER;
  3  J NUMBER;
  4  N NUMBER;
  5  BEGIN
  6  N:=&N;
  7  FOR I IN 1..N LOOP
  8  FOR J IN 1..I LOOP
  9  DBMS_OUTPUT.PUT(J);
 10  END LOOP;
 11  DBMS_OUTPUT.PUT_LINE('');
 12  END LOOP;
 13  END;
 14  /
Enter value for n: 5
old   6: N:=&N;
new   6: N:=5;
1
12
123
1234
12345

PL/SQL procedure successfully completed.

SQL>
```

12) Write PL/SQL code to print the numbers in this form

0 0 0 0 0
1 2 3 4 5
2 4 6 8 10

```
DECLARE
I NUMBER;
J NUMBER;
K NUMBER;
BEGIN
FOR I IN 0..5 LOOP
FOR J IN 1..5 LOOP
K:=I*J;
DBMS_OUTPUT.PUT(K);
END LOOP;
DBMS_OUTPUT.PUT_LINE('');
END LOOP;
END;
/
```

Output:



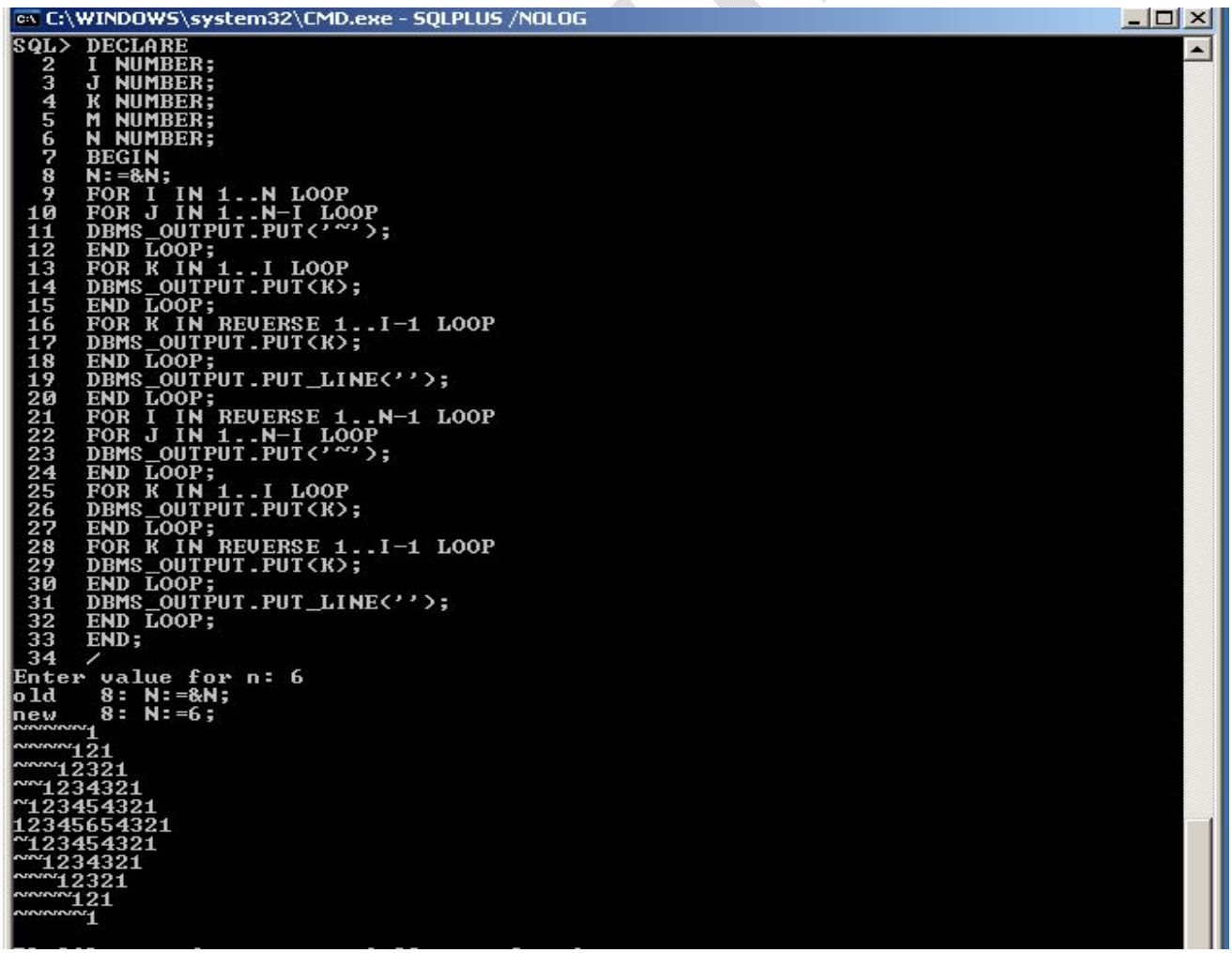
13)Write PL/SQL code to print the numbers in this form

```
    1
   1 2 1
  1 2 3 2 1
   1 2 1
    1
```

```
DECLARE
I NUMBER;
J NUMBER;
K NUMBER;
M NUMBER;
N NUMBER;
BEGIN
N:=&N;
FOR I IN 1..N LOOP
FOR J IN 1..N-I LOOP
DBMS_OUTPUT.PUT('~');
END LOOP;
FOR K IN 1..I LOOP
DBMS_OUTPUT.PUT(K);
END LOOP;
FOR K IN REVERSE 1..I-1 LOOP
DBMS_OUTPUT.PUT(K);
END LOOP;
DBMS_OUTPUT.PUT_LINE('');
END LOOP;
```

```
FOR I IN REVERSE 1..N-1 LOOP
FOR J IN 1..N-I LOOP
DBMS_OUTPUT.PUT('~');
END LOOP;
FOR K IN 1..I LOOP
DBMS_OUTPUT.PUT(K);
END LOOP;
FOR K IN REVERSE 1..I-1 LOOP
DBMS_OUTPUT.PUT(K);
END LOOP;
DBMS_OUTPUT.PUT_LINE('');
END LOOP;
END;
/
```

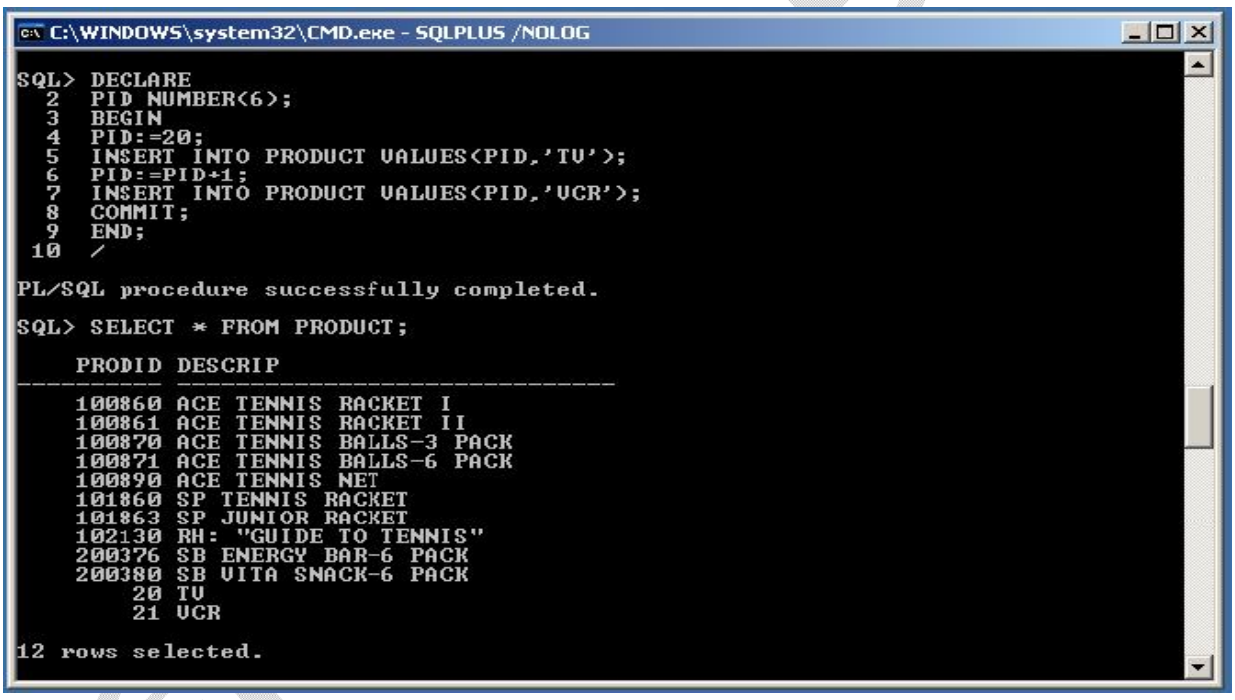
Output:



14) Write PL/SQL code to Insert values in created tables.

```
DECLARE
PID NUMBER(6);
BEGIN
PID:=20;
INSERT INTO PRODUCT VALUES(PID,'TV');
PID:=PID+1;
INSERT INTO PRODUCT VALUES(PID,'VCR');
COMMIT;
END;
/
```

Output:



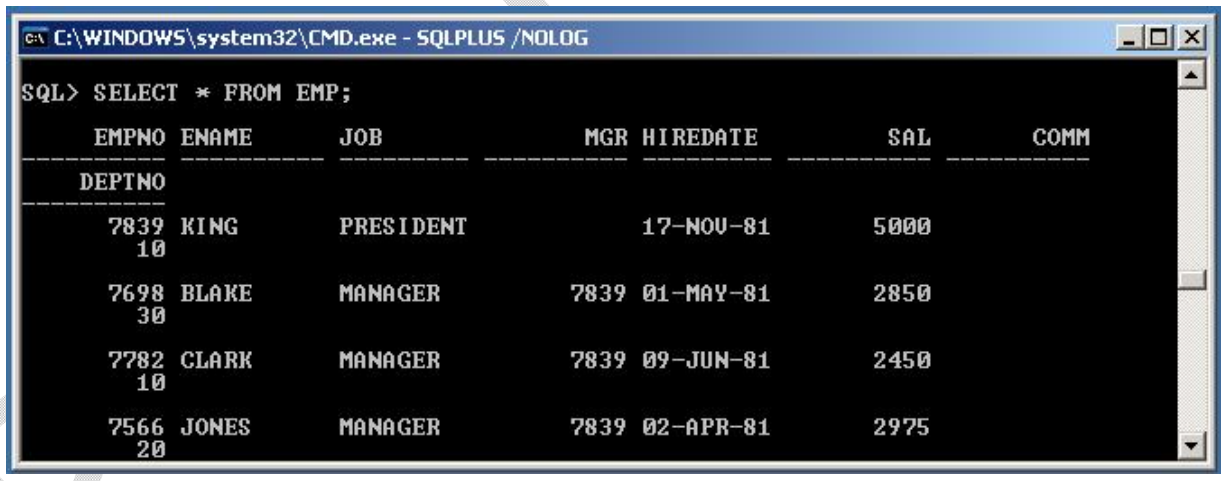
```
C:\WINDOWS\system32\CMD.exe - SQLPLUS /NOLOG

SQL> DECLARE
  2  PID NUMBER(6);
  3  BEGIN
  4  PID:=20;
  5  INSERT INTO PRODUCT VALUES(PID,'TV');
  6  PID:=PID+1;
  7  INSERT INTO PRODUCT VALUES(PID,'VCR');
  8  COMMIT;
  9  END;
 10  /

PL/SQL procedure successfully completed.

SQL> SELECT * FROM PRODUCT;

    PRODID DESCRIP
---------- ------------------------------
    100860 ACE TENNIS RACKET I
    100861 ACE TENNIS RACKET II
    100870 ACE TENNIS BALLS-3 PACK
    100871 ACE TENNIS BALLS-6 PACK
    100890 ACE TENNIS NET
    101860 SP TENNIS RACKET
    101863 SP JUNIOR RACKET
    102130 RH: "GUIDE TO TENNIS"
    200376 SB ENERGY BAR-6 PACK
    200380 SB VITA SNACK-6 PACK
        20 TV
        21 VCR

12 rows selected.
```

(ii) Programs using Cursors, Cursor loops and records

1) Write PL/SQL code to UPDATE values in created tables by using Implicit Cursors.

```
DECLARE
VAR_ROWS NUMBER(5);
BEGIN
UPDATE EMP SET SAL=SAL+100;
IF SQL%NOTFOUND THEN
DBMS_OUTPUT.PUT_LINE('NONE OF THE SALARIES WERE UPDATED');
ELSE IF SQL%FOUND THEN
VAR_ROWS:=SQL%ROWCOUNT;
DBMS_OUTPUT.PUT_LINE('SALARIES FOR'||VAR_ROWS||'EMPLOYEES ARE
UPDATED');
END IF ;
END IF;
END;
/
```

Output:



```
C:\WINDOWS\system32\CMD.exe - SQLPLUS /NOLOG
SQL> SELECT * FROM EMP;
     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM
---------- ---------- --------- ---------- --------- ---------- ----------
    DEPTNO
----------
      7839 KING       PRESIDENT            17-NOV-81       5000
        10

      7698 BLAKE      MANAGER         7839 01-MAY-81       2850
        30

      7782 CLARK      MANAGER         7839 09-JUN-81       2450
        10

      7566 JONES      MANAGER         7839 02-APR-81       2975
        20
```

```
C:\WINDOWS\system32\CMD.exe - SQLPLUS /NOLOG

SQL> DECLARE
  2  VAR_ROWS NUMBER(5);
  3  BEGIN
  4  UPDATE EMP SET SAL=SAL+100;
  5  IF SQL%NOTFOUND THEN
  6  DBMS_OUTPUT.PUT_LINE('NONE OF THE SALARIES WERE UPDATED');
  7  ELSE IF SQL%FOUND THEN
  8  VAR_ROWS:=SQL%ROWCOUNT;
  9  DBMS_OUTPUT.PUT_LINE('SALARIES FOR'||VAR_ROWS||'EMPLOYEES ARE UPDATED');
 10  END IF ;
 11  END IF;
 12  END;
 13  /

PL/SQL procedure successfully completed.

SQL> SELECT * FROM EMP;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM
---------- ---------- --------- ---------- --------- ---------- ----------
    DEPTNO
----------
      7839 KING       PRESIDENT            17-NOV-81       5100
        10

      7698 BLAKE      MANAGER         7839 01-MAY-81       2950
        30

      7782 CLARK      MANAGER         7839 09-JUN-81       2550
        10

      7566 JONES      MANAGER         7839 02-APR-81       3075
        20
```

2) Write PL/SQL code to display Employee details using Explicit Cursors.

```
DECLARE
CURSOR EMP_CUR IS SELECT * FROM EMP;
EMP_REC EMP%ROWTYPE;
BEGIN
OPEN EMP_CUR;
LOOP
FETCH EMP_CUR INTO EMP_REC;
EXIT WHEN EMP_CUR%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(EMP_REC.EMPNO || '   ' ||EMP_REC.ENAME || ' '
                                ||EMP_REC.SAL);
END LOOP;
CLOSE EMP_CUR;
END;
/
```

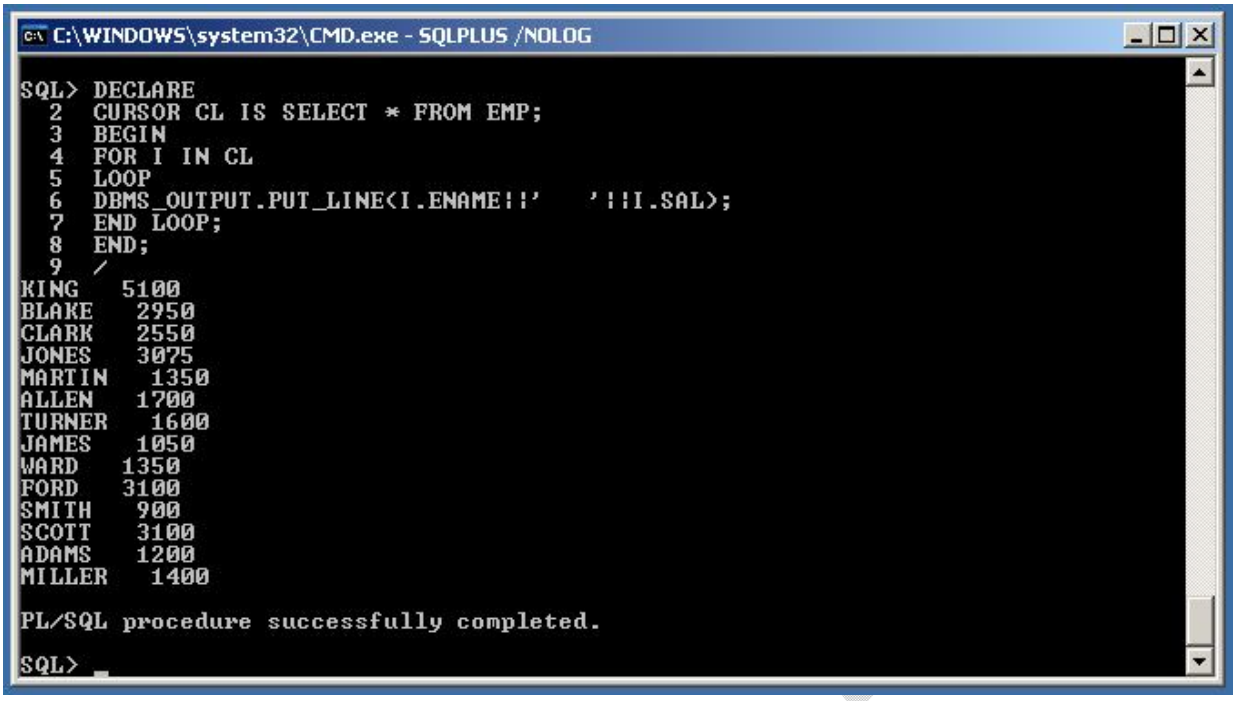
Output:

```
C:\WINDOWS\system32\CMD.exe - SQLPLUS /NOLOG

SQL> DECLARE
  2  CURSOR EMP_CUR IS SELECT * FROM EMP;
  3  EMP_REC EMP%ROWTYPE;
  4  BEGIN
  5  OPEN EMP_CUR;
  6  LOOP
  7  FETCH EMP_CUR INTO EMP_REC;
  8  EXIT WHEN EMP_CUR%NOTFOUND;
  9  DBMS_OUTPUT.PUT_LINE(EMP_REC.EMPNO || '    ' ||EMP_REC.ENAME || '  ' ||EMP_
REC.SAL);
 10  END LOOP;
 11  CLOSE EMP_CUR;
 12  END;
 13  /
7839    KING  5100
7698    BLAKE  2950
7782    CLARK  2550
7566    JONES  3075
7654    MARTIN  1350
7499    ALLEN  1700
7844    TURNER  1600
7900    JAMES  1050
7521    WARD  1350
7902    FORD  3100
7369    SMITH  900
7788    SCOTT  3100
7876    ADAMS  1200
7934    MILLER  1400

PL/SQL procedure successfully completed.

SQL> _
```

3) Write PL/SQL code in Cursor to display employee names and salary.

```
DECLARE
CURSOR CL IS SELECT * FROM EMP;
BEGIN
FOR I IN CL
LOOP
DBMS_OUTPUT.PUT_LINE(I.ENAME||'  '||I.SAL);
END LOOP;
END;
/
```

Output:



```
C:\WINDOWS\system32\CMD.exe - SQLPLUS /NOLOG

SQL> DECLARE
  2  CURSOR CL IS SELECT * FROM EMP;
  3  BEGIN
  4  FOR I IN CL
  5  LOOP
  6  DBMS_OUTPUT.PUT_LINE(I.ENAME||'    '||I.SAL);
  7  END LOOP;
  8  END;
  9  /
KING   5100
BLAKE   2950
CLARK   2550
JONES   3075
MARTIN   1350
ALLEN   1700
TURNER   1600
JAMES   1050
WARD   1350
FORD   3100
SMITH   900
SCOTT   3100
ADAMS   1200
MILLER   1400

PL/SQL procedure successfully completed.

SQL>
```

4) Write PL/SQL Programs in Cursors using two cursors at a time.

```
DECLARE
CURSOR D IS SELECT * FROM DEPT;
CURSOR E(DNO NUMBER) IS SELECT * FROM EMP WHERE DEPTNO=DNO;
BEGIN
FOR DEPT IN D
LOOP
DBMS_OUTPUT.PUT_LINE('DEPARTMENT NUMBER'|| DEPT.DEPTNO);
DBMS_OUTPUT.PUT_LINE('.........');
FOR EMP IN E(DEPT.DEPTNO)
LOOP
DBMS_OUTPUT.PUT_LINE('MR'||'  '||EMP.ENAME||'  '|| 'IS WORKING IN
DEPARTMENT '||DEPT.DNAME||'  '|| 'AT'||'  '||DEPT.LOC||'   AS   '
||EMP.JOB);
END LOOP;
END LOOP;
END;
/
```

Output:

```
C:\WINDOWS\system32\CMD.exe - SQLPLUS /NOLOG
SQL> DECLARE
  2  CURSOR D IS SELECT * FROM DEPT;
  3  CURSOR E(DNO NUMBER) IS SELECT * FROM EMP WHERE DEPTNO=DNO;
  4  BEGIN
  5  FOR DEPT IN D
  6  LOOP
  7  DBMS_OUTPUT.PUT_LINE('DEPARTMENT NUMBER'|| DEPT.DEPTNO);
  8  DBMS_OUTPUT.PUT_LINE('...........');
  9  FOR EMP IN E(DEPT.DEPTNO)
 10  LOOP
 11  DBMS_OUTPUT.PUT_LINE('MR'||'   '||EMP.ENAME||'  '|| 'IS WORKING IN DEPARTME
NT '||DEPT.DNAME||'  '|| 'AT'||'  '||DEPT.LOC||'   AS   '  ||EMP.JOB);
 12  END LOOP;
 13  END LOOP;
 14  END;
 15  /
DEPARTMENT NUMBER10
...........
MR   KING  IS WORKING IN DEPARTMENT ACCOUNTING  AT  NEW YORK   AS   PRESIDENT
MR   CLARK  IS WORKING IN DEPARTMENT ACCOUNTING  AT  NEW YORK   AS   MANAGER
MR   MILLER  IS WORKING IN DEPARTMENT ACCOUNTING  AT  NEW YORK   AS   CLERK
DEPARTMENT NUMBER20
...........
MR   JONES  IS WORKING IN DEPARTMENT RESEARCH  AT  DALLAS   AS   MANAGER
MR   FORD  IS WORKING IN DEPARTMENT RESEARCH  AT  DALLAS   AS   ANALYST
MR   SMITH  IS WORKING IN DEPARTMENT RESEARCH  AT  DALLAS   AS   CLERK
MR   SCOTT  IS WORKING IN DEPARTMENT RESEARCH  AT  DALLAS   AS   ANALYST
MR   ADAMS  IS WORKING IN DEPARTMENT RESEARCH  AT  DALLAS   AS   CLERK
DEPARTMENT NUMBER30
...........
MR   BLAKE  IS WORKING IN DEPARTMENT SALES  AT  CHICAGO   AS   MANAGER
MR   MARTIN  IS WORKING IN DEPARTMENT SALES  AT  CHICAGO   AS   SALESMAN
MR   ALLEN  IS WORKING IN DEPARTMENT SALES  AT  CHICAGO   AS   SALESMAN
MR   TURNER  IS WORKING IN DEPARTMENT SALES  AT  CHICAGO   AS   SALESMAN
MR   JAMES  IS WORKING IN DEPARTMENT SALES  AT  CHICAGO   AS   CLERK
MR   WARD  IS WORKING IN DEPARTMENT SALES  AT  CHICAGO   AS   SALESMAN
DEPARTMENT NUMBER40
...........

PL/SQL procedure successfully completed.

SQL> _
```

5) Write PL/SQL Programs in Cursors using Loops.

A)

```
DECLARE
CURSOR ALL_EMPS IS SELECT EMPNO,ENAME FROM EMP ORDER BY
EMPNO;
EMP1 ALL_EMPS%ROWTYPE;
BEGIN
OPEN ALL_EMPS;
LOOP
EXIT WHEN ALL_EMPS%NOTFOUND;
FETCH ALL_EMPS INTO EMP1;
DBMS_OUTPUT.PUT_LINE(EMP1.ENAME||'    '||EMP1.EMPNO);
END LOOP;
CLOSE ALL_EMPS;
END;
/
```

Output:

```
C:\WINDOWS\system32\CMD.exe - SQLPLUS /NOLOG

SQL> DECLARE
  2  CURSOR ALL_EMPS IS SELECT EMPNO,ENAME FROM EMP ORDER BY EMPNO;
  3  EMP1 ALL_EMPS%ROWTYPE;
  4  BEGIN
  5  OPEN ALL_EMPS;
  6  LOOP
  7  EXIT WHEN ALL_EMPS%NOTFOUND;
  8  FETCH ALL_EMPS INTO EMP1;
  9  DBMS_OUTPUT.PUT_LINE(EMP1.ENAME||'      '||EMP1.EMPNO);
 10  END LOOP;
 11  CLOSE ALL_EMPS;
 12  END;
 13  /
SMITH      7369
ALLEN      7499
WARD      7521
JONES      7566
MARTIN      7654
BLAKE      7698
CLARK      7782
SCOTT      7788
KING      7839
TURNER      7844
ADAMS      7876
JAMES      7900
FORD      7902
MILLER      7934
MILLER      7934

PL/SQL procedure successfully completed.

SQL> _
```

B)

```
DECLARE
CURSOR C_EMPLOYEE IS
SELECT EMPNO,ENAME,JOB,SAL,DEPTNO FROM EMP WHERE
                                        EMPNO=7839;
V_EMPLOYEEDATA C_EMPLOYEE%ROWTYPE;
BEGIN
OPEN C_EMPLOYEE;
FETCH C_EMPLOYEE INTO V_EMPLOYEEDATA;
WHILE C_EMPLOYEE%FOUND LOOP
DBMS_OUTPUT.PUT_LINE(V_EMPLOYEEDATA.ENAME||'
                        '||V_EMPLOYEEDATA.EMPNO||'
                        '||V_EMPLOYEEDATA.SAL);
FETCH C_EMPLOYEE INTO V_EMPLOYEEDATA;
END LOOP;
CLOSE C_EMPLOYEE;
COMMIT;
END;
/
```

Output:

```
C:\WINDOWS\system32\CMD.exe - SQLPLUS /NOLOG

SQL> DECLARE
  2  CURSOR C_EMPLOYEE IS
  3  SELECT EMPNO,ENAME,JOB,SAL,DEPTNO FROM EMP WHERE EMPNO=7839;
  4  V_EMPLOYEEDATA C_EMPLOYEE%ROWTYPE;
  5  BEGIN
  6  OPEN C_EMPLOYEE;
  7  FETCH C_EMPLOYEE INTO V_EMPLOYEEDATA;
  8  WHILE C_EMPLOYEE%FOUND LOOP
  9  DBMS_OUTPUT.PUT_LINE(V_EMPLOYEEDATA.ENAME||'     '||V_EMPLOYEEDATA.EMPNO||'
    '||V_EMPLOYEEDATA.SAL);
 10  FETCH C_EMPLOYEE INTO V_EMPLOYEEDATA;
 11  END LOOP;
 12  CLOSE C_EMPLOYEE;
 13  COMMIT;
 14  END;
 15  /
KING     7839      5100

PL/SQL procedure successfully completed.

SQL>
```

6) To write a Cursor to display the list of employees who are Working as a Managers or Analyst.

```
DECLARE
        cursor c(jb varchar2) is select ename from emp where job=jb;
        em emp.job%type;
BEGIN
        open c('MANAGER');
        dbms_output.put_line(' EMPLOYEES WORKING AS MANAGERS ARE:');
        loop
                fetch c into em;
                exit when c%notfound;
                dbms_output.put_line(em);
        end loop;
        close c;
        open c('ANALYST');
        dbms_output.put_line(' EMPLOYEES WORKING AS ANALYST ARE:');
        loop
                fetch c into em;
                exit when c%notfound;
                dbms_output.put_line(em);
        end loop;
        close c;
END;
```

/

Output:

```
C:\WINDOWS\system32\CMD.exe - sqlplus /nolog
SQL> set serveroutput on;
SQL> DECLARE
  2      cursor c(jb varchar2) is select ename from emp where job=jb;
  3      em emp.job%type;
  4  BEGIN
  5      open c('MANAGER');
  6      dbms_output.put_line(' EMPLOYEES WORKING AS MANAGERS ARE:');
  7      loop
  8              fetch c into em;
  9              exit when c%notfound;
 10              dbms_output.put_line(em);
 11      end loop;
 12      close c;
 13      open c('ANALYST');
 14      dbms_output.put_line(' EMPLOYEES WORKING AS ANALYST ARE:');
 15      loop
 16              fetch c into em;
 17              exit when c%notfound;
 18              dbms_output.put_line(em);
 19      end loop;
 20      close c;
 21  END;
 22  /
EMPLOYEES WORKING AS MANAGERS ARE:
BLAKE
CLARK
JONES
EMPLOYEES WORKING AS ANALYST ARE:
FORD
SCOTT

PL/SQL procedure successfully completed.

SQL>
```

7. **Write pl/sql code in cursor by using while loop.**

```
set serveroutput on;
Declare
cursor c_emp is
select sal,ename from emp where sal=5200;
v_empdata c_emp%rowtype;
Begin
open c_emp;
fetch c_emp INTO v_empdata;
while c_emp%found loop
dbms_output.put_line(v_empdata.sal);
fetch c_emp INTO v_empdata;
end loop;
close c_emp;
commit;
end;
/
```

Output:

```
C:\WINDOWS\system32\CMD.exe - SQLplus /nolog

SQL> set serveroutput on;
SQL> Declare
  2  cursor c_emp is
  3  select sal,ename from emp where sal=5200;
  4  v_empdata c_emp%rowtype;
  5  Begin
  6  open c_emp;
  7  fetch c_emp INTO v_empdata;
  8  while c_emp%found loop
  9  dbms_output.put_line(v_empdata.sal);
 10  fetch c_emp INTO v_empdata;
 11  end loop;
 12  close c_emp;
 13  commit;
 14  end;
 15  /
5200

PL/SQL procedure successfully completed.

SQL> _
```

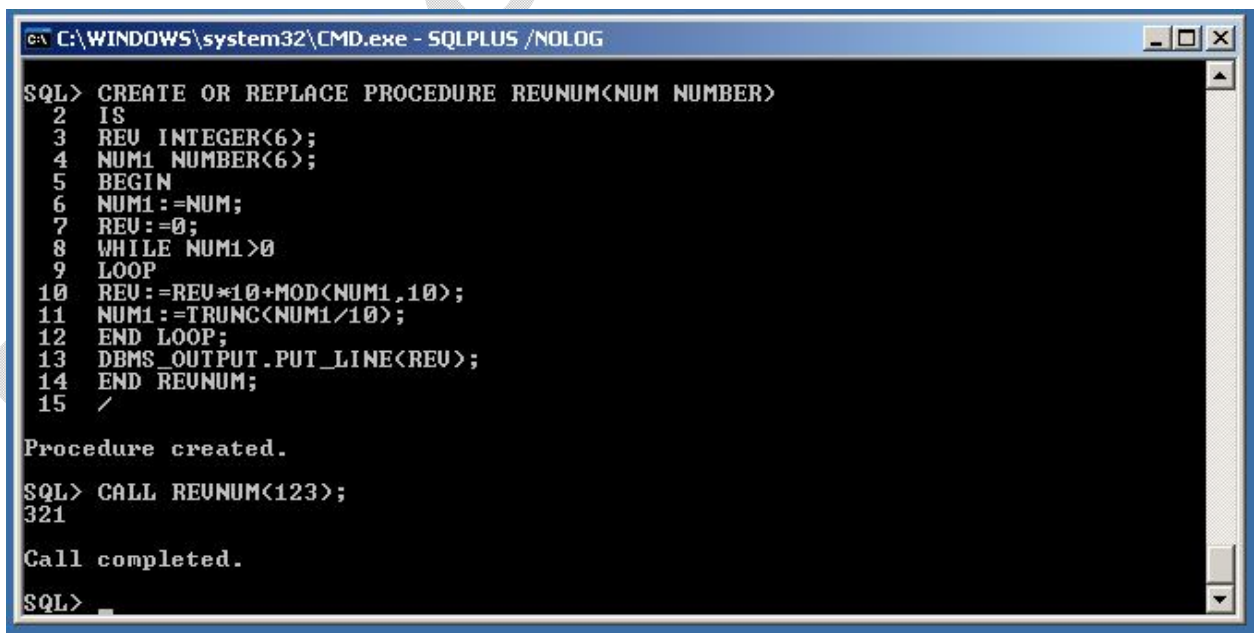
EXP-IX PL/SQL Programming II

(i) Creating stored procedures and functions

1) Write PL/SQL code in Procedure to find Reverse number

```
CREATE OR REPLACE PROCEDURE REVNUM(NUM NUMBER)
IS
REV INTEGER(6);
NUM1 NUMBER(6);
BEGIN
NUM1:=NUM;
REV:=0;
WHILE NUM1>0
LOOP
REV:=REV*10+MOD(NUM1,10);
NUM1:=TRUNC(NUM1/10);
END LOOP;
DBMS_OUTPUT.PUT_LINE(REV);
END REVNUM;
/
```

Output:

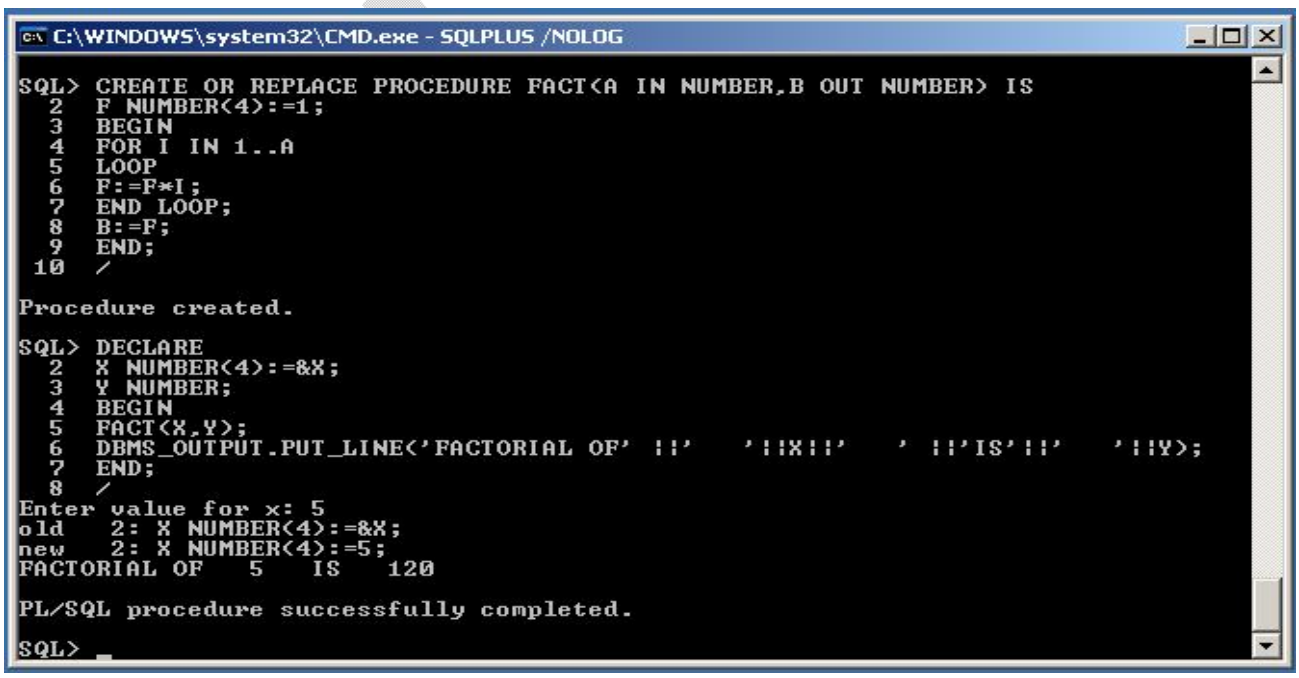


2) Write PL/SQL code in Procedure to find Factorial of a given number by using call procedure.

a)
```
CREATE OR REPLACE PROCEDURE FACT(A IN NUMBER,B OUT NUMBER) IS
F NUMBER(4):=1;
BEGIN
FOR I IN 1..A
LOOP
F:=F*I;
END LOOP;
B:=F;
END;
/
```

b)
```
DECLARE
X NUMBER(4):=&X;
Y NUMBER;
BEGIN
FACT(X,Y);
DBMS_OUTPUT.PUT_LINE('FACTORIAL OF' ||'  '||X||'  ' ||'IS'||'  '||Y);
END;
/
```

Output:



3) Write a Procedure to check the given number is prime or not by using call procedure.

PROCEDURE DEFINITION:

```
CREATE or REPLACE PROCEDURE isprimepro(num in number,chec out number) IS
        temp NUMBER;
        BEGIN
                temp:=num;
      FOR itr IN 2..(num-1)
                LOOP
                IF(mod(num,itr)=0) THEN
                chec:=1;
                END IF;
                END LOOP;
         END;
/
```

PL/SQL BLOCK TO CALL THE PROCEDURE:

```
        DECLARE
                chec NUMBER;
                  given_number NUMBER;
        BEGIN
                  given_number:=&given_number;
                   isprimepro(given_number,chec);
                  IF chec=1  THEN
                  dbms_output.put_line('number is not prime');
                  ELSE
                  dbms_output.put_line('number is   prime');
          END IF;
          END;
        /
```

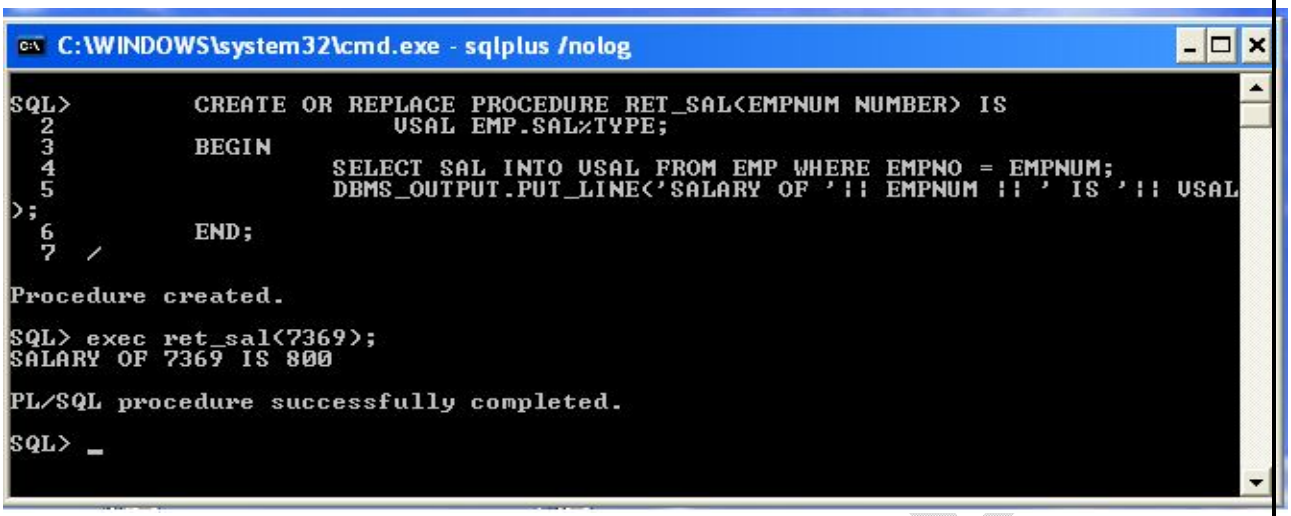
Output:

```
C:\WINDOWS\system32\cmd.exe - sqlplus /nolog
SQL> set serveroutput on;
SQL> CREATE or REPLACE PROCEDURE isprimepro(num in number,chec out number) IS
  2              temp NUMBER;
  3      BEGIN
  4                      temp:=num;
  5            FOR itr IN 2..(num-1)
  6                      LOOP
  7                      IF(mod(num,itr)=0) THEN
  8                      chec:=1;
  9                      END IF;
 10                      END LOOP;
 11               END;
 12   /

Procedure created.

SQL> DECLARE
  2                      chec NUMBER;
  3      given_number NUMBER;
  4   BEGIN
  5      given_number:=&given_number;
  6        isprimepro(given_number,chec);
  7       IF chec=1  THEN
  8       dbms_output.put_line('number is not prime');
  9       ELSE
 10       dbms_output.put_line('number is   prime');
 11     END IF;
 12     END;
 13   /
Enter value for given_number: 5
old   5:     given_number:=&given_number;
new   5:     given_number:=5;
number is   prime

PL/SQL procedure successfully completed.

SQL> /
Enter value for given_number: 4
old   5:     given_number:=&given_number;
new   5:     given_number:=4;
number is not prime
```

4) Write a procedure to retrieve the salary of a particular employee

```
CREATE OR REPLACE PROCEDURE RET_SAL(EMPNUM NUMBER) IS
        VSAL EMP.SAL%TYPE;
BEGIN
      SELECT SAL INTO VSAL FROM EMP WHERE EMPNO = EMPNUM;
      DBMS_OUTPUT.PUT_LINE('SALARY OF '|| EMPNUM || ' IS '|| VSAL);
END;
/
```

Output:

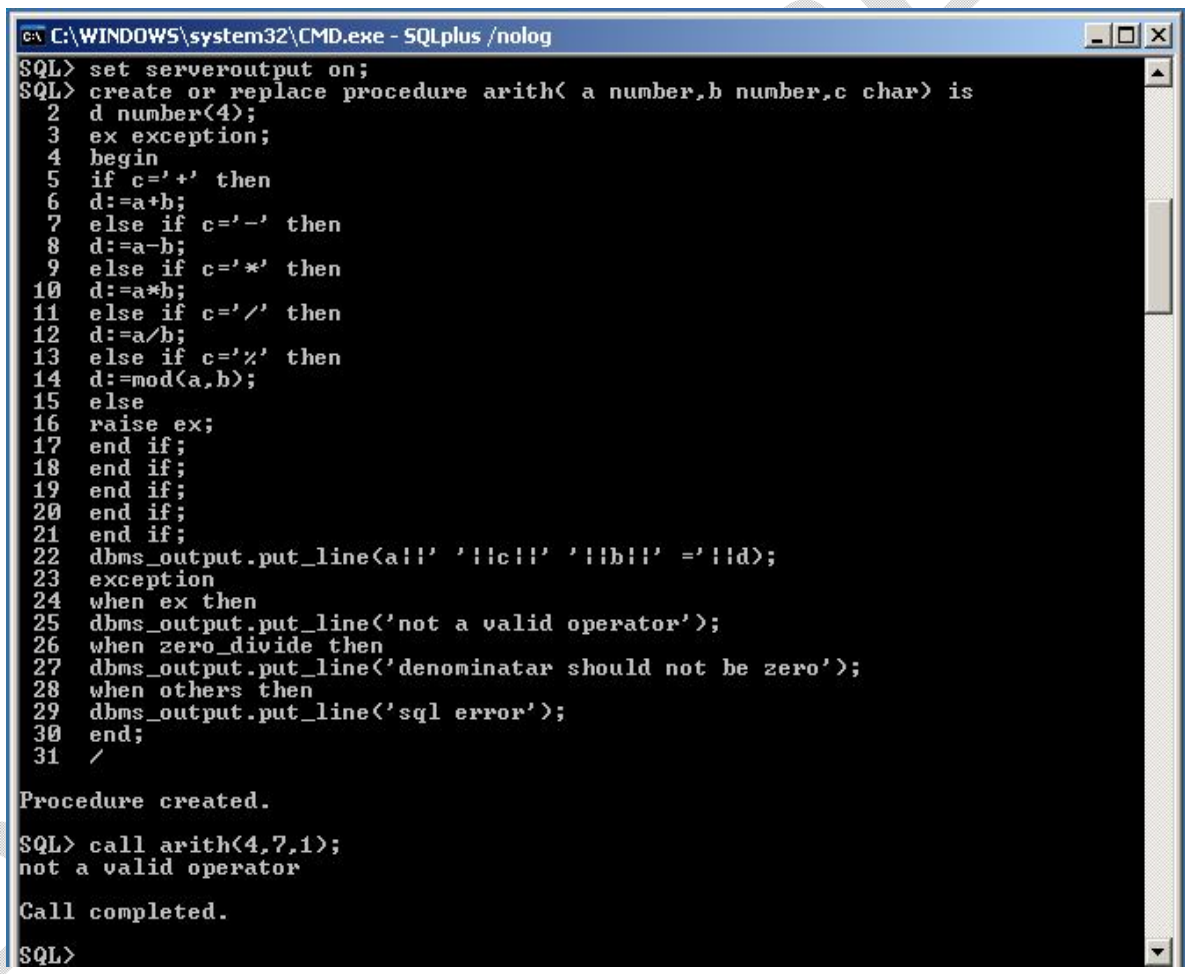


5 Write a procedure to work with Arthimetic operations.

```
create or replace procedure arith( a number,b number,c char) is
d number(4);
ex exception;
begin
if c='+' then
d:=a+b;
else if c='-' then
d:=a-b;
else if c='*' then
d:=a*b;
else if c='/' then
d:=a/b;
else if c='%' then
d:=mod(a,b);
else
raise ex;
end if;
end if;
end if;
end if;
end if;
dbms_output.put_line(a||' '||c||' '||b||' ='||d);
```

```
exception
when ex then
dbms_output.put_line('not a valid operator');
when zero_divide then
dbms_output.put_line('denominatar should not be zero');
when others then
dbms_output.put_line('sql error');
end;
/
```

Output:



```
C:\WINDOWS\system32\CMD.exe - SQLplus /nolog
SQL> set serveroutput on;
SQL> create or replace procedure arith( a number,b number,c char) is
  2  d number(4);
  3  ex exception;
  4  begin
  5  if c='+' then
  6  d:=a+b;
  7  else if c='-' then
  8  d:=a-b;
  9  else if c='*' then
 10  d:=a*b;
 11  else if c='/' then
 12  d:=a/b;
 13  else if c='%' then
 14  d:=mod(a,b);
 15  else
 16  raise ex;
 17  end if;
 18  end if;
 19  end if;
 20  end if;
 21  end if;
 22  dbms_output.put_line(a||' '||c||' '||b||' ='||d);
 23  exception
 24  when ex then
 25  dbms_output.put_line('not a valid operator');
 26  when zero_divide then
 27  dbms_output.put_line('denominatar should not be zero');
 28  when others then
 29  dbms_output.put_line('sql error');
 30  end;
 31  /

Procedure created.

SQL> call arith(4,7,1);
not a valid operator

Call completed.

SQL>
```

Functions:

1. Write a Function to check the given number is prime or not.

FUNCTION DEFINITION :

CREATE or REPLACE FUNCTION isprime(num in number) RETURN number IS

```
        temp NUMBER;
BEGIN
            temp:=num;
        FOR itr  IN 2..(num-1)
        LOOP
                IF (mod(temp,itr)=0)  THEN
                return 1;
            END IF;
        END LOOP;
          return 0;
          END;
    /
```

PL/SQL BLOCK TO CALL THE FUNCTION:

```
DECLARE
                given_number  NUMBER;
                prime_check  NUMBER;
BEGIN
   given_number:=&given_number;
  prime_check:=isprime(given_number);
        IF prime_check=0 THEN
                dbms_output.put_line('NUMBER IS PRIME');
 ELSE
   dbms_output.put_line('NUMBER IS  NOT PRIME');
 END IF;
        END;
/
```

Output:

```
C:\WINDOWS\system32\CMD.exe - sqlplus /nolog

SQL>              CREATE or  REPLACE FUNCTION isprime(num in number)  RETURN
  2                number IS
  3                temp NUMBER;
  4    BEGIN
  5                      temp:=num;
  6                    FOR itr  IN 2..(num-1)
  7                    LOOP
  8                     IF (mod(temp,itr)=0)  THEN
  9                              return 1;
 10                    END IF;
 11                    END LOOP;
 12                 return 0;
 13                 END;
 14  /

Function created.

SQL> DECLARE
  2                              given_number  NUMBER;
  3                               prime_check  NUMBER;
  4  BEGIN
  5      given_number:=&given_number;
  6     prime_check:=isprime(given_number);
  7                IF prime_check=0 THEN
  8                        dbms_output.put_line('NUMBER IS PRIME');
  9   ELSE
 10      dbms_output.put_line('NUMBER IS  NOT PRIME');
 11    END IF;
 12                 END;
 13  /
Enter value for given_number: 10
old   5:     given_number:=&given_number;
new   5:     given_number:=10;
NUMBER IS  NOT PRIME

PL/SQL procedure successfully completed.

SQL> /
Enter value for given_number: 7
old   5:     given_number:=&given_number;
new   5:     given_number:=7;
NUMBER IS PRIME

PL/SQL procedure successfully completed.

SQL> _
```

2. Write pl./sql code in Function for Factorialnumber.

(a) set serveroutput on;

```
create or replace Function FunFact(a number) return number IS
 f number(4):=1;
 begin
 for i in 1..a
 loop
 f:=f*i;
 end loop;
 return f;
 end;
 /
```

(b) set serveroutput on;

```
Declare
```

```
n number(2):=&n;
r number(4);
Begin
r:=FunFact(n);
dbms_output.put_line('factorial of'||n||'is:'||r);
end;
/
```

Output:

```
C:\WINDOWS\system32\CMD.exe - SQLplus /nolog

SQL> set serveroutput on;
SQL> create or replace Function FunFact(a number) return number IS
  2  f number(4):=1;
  3  begin
  4  for i in 1..a
  5  loop
  6  f:=f*i;
  7  end loop;
  8  return f;
  9  end;
 10  /

Function created.

SQL> set serveroutput on;
SQL> Declare
  2  n number(2):=&n;
  3  r number(4);
  4  Begin
  5  r:=FunFact(n);
  6  dbms_output.put_line('factorial of'||n||'is:'||r);
  7  end;
  8  /
Enter value for n: 5
old   2: n number(2):=&n;
new   2: n number(2):=5;
factorial of5is:120

PL/SQL procedure successfully completed.

SQL>
```

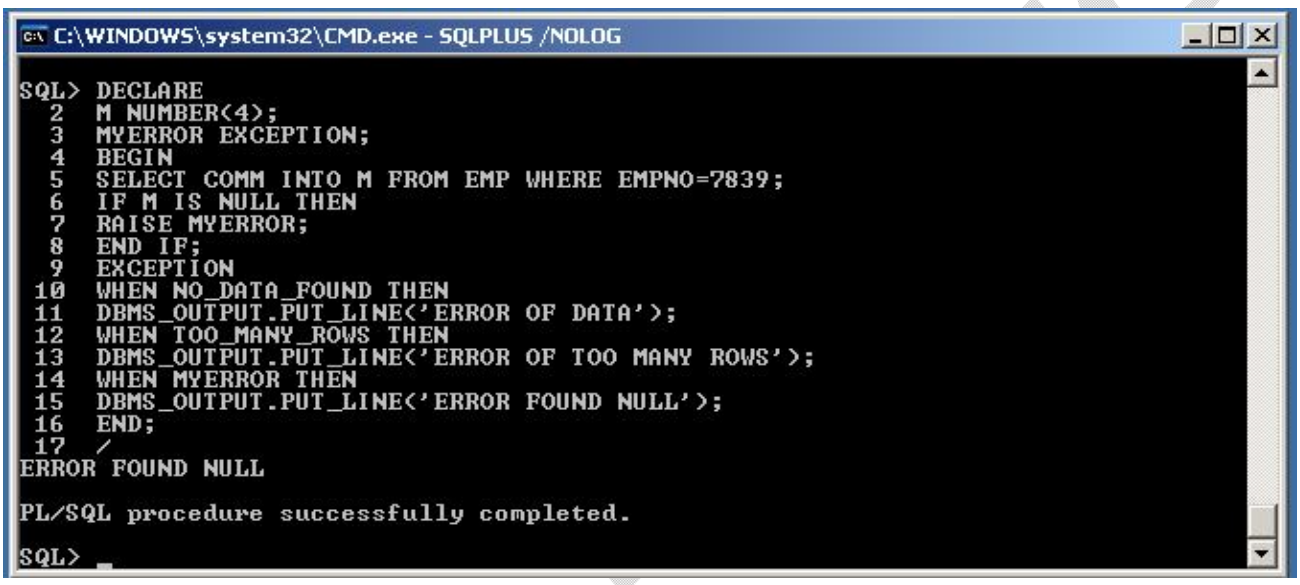
(ii) Error handling and Exception

1) Write a PL/SQL block to handle the following BUILT-IN EXCEPTIONS.

```
DECLARE
M NUMBER(4);
MYERROR EXCEPTION;
BEGIN
SELECT COMM INTO M FROM EMP WHERE EMPNO=7839;
IF M IS NULL THEN
RAISE MYERROR;
END IF;
EXCEPTION
WHEN NO_DATA_FOUND THEN
```

```
DBMS_OUTPUT.PUT_LINE('ERROR OF DATA');
WHEN TOO_MANY_ROWS THEN
DBMS_OUTPUT.PUT_LINE('ERROR OF TOO MANY ROWS');
WHEN MYERROR THEN
DBMS_OUTPUT.PUT_LINE('ERROR FOUND NULL');
END;
/
```

Output:



(ii) Triggers

1) Write pl/sql code for before insert Trigger program

Steps for doing Trigger Program

a. First create a table called Person(Don't insert any values into created table)

b.Write code for Trigger and run the trigger program

c.Insert values into the created table after sucessfully completion of trigger program and see the trigger output.

(a) CREATE TABLE PERSON1(ID INTEGER,NAME VARCHAR2(30),DOB DATE,PRIMARY KEY(ID));

(b) CREATE OR REPLACE TRIGGER PERSON_INSERT_BEFORE

```
BEFORE INSERT ON PERSON1 FOR EACH ROW
BEGIN
DBMS_OUTPUT.PUT_LINE('BEFORE INSERT OF'||:NEW.NAME);
```

```
END;
/
```

(c) INSERT INTO PERSON1 VALUES(1,'JOHN DOE',SYSDATE);

Output:

```
C:\WINDOWS\system32\CMD.exe - SQLPLUS /NOLOG
SQL> CREATE TABLE PERSON1(ID INTEGER,NAME VARCHAR2(30),DOB DATE,PRIMARY KEY(ID))
;

Table created.
```

```
C:\WINDOWS\system32\CMD.exe - SQLPLUS /NOLOG
SQL> CREATE OR REPLACE TRIGGER PERSON_INSERT_BEFORE
  2  BEFORE INSERT ON PERSON1 FOR EACH ROW
  3  BEGIN
  4  DBMS_OUTPUT.PUT_LINE('BEFORE INSERT OF'||:NEW.NAME);
  5  END;
  6  /

Trigger created.

SQL> INSERT INTO PERSON1 VALUES(1,'JOHN DOE',SYSDATE);
BEFORE INSERT OFJOHN DOE

1 row created.

SQL> SELECT * FROM PERSON1;

        ID NAME                           DOB
---------- ------------------------------ ---------
         1 JOHN DOE                       29-AUG-12

SQL> _
```

2) Write pl/sql code for After inser Trigger

(a) CREATE OR REPLACE TRIGGER PERSON_INSERT_AFTER

```
AFTER INSERT ON PERSON1 FOR EACH ROW
BEGIN
DBMS_OUTPUT.PUT_LINE('AFTER INSERT OF'||:NEW.NAME);
END;
/
```

(b) INSERT INTO PERSON1 VALUES(2,'JAME DOE',SYSDATE);

Output:

```
C:\WINDOWS\system32\CMD.exe - SQLPLUS /NOLOG

SQL> CREATE OR REPLACE TRIGGER PERSON_INSERT_AFTER
  2  AFTER INSERT ON PERSON1 FOR EACH ROW
  3  BEGIN
  4  DBMS_OUTPUT.PUT_LINE('AFTER INSERT OF'||:NEW.NAME);
  5  END;
  6  /

Trigger created.

SQL> INSERT INTO PERSON1 VALUES(2,'JAME DOE',SYSDATE);
BEFORE INSERT OFJAME DOE
AFTER INSERT OFJAME DOE

1 row created.

SQL> SELECT * FROM PERSON1;

        ID NAME                           DOB
---------- ------------------------------ ---------
         1 JOHN DOE                       29-AUG-12
         2 JAME DOE                       29-AUG-12

SQL>
```

3) Write pl/sql code for Before Update statement Trigger

(a) CREATE OR REPLACE TRIGGER PERSON_UPDATE_BEFORE

BEFORE UPDATE ON PERSON1

BEGIN

DBMS_OUTPUT.PUT_LINE('BEFORE UPDATING SOME PERSONS');

END;

/

(b) UPDATE PERSON1 SET DOB='21-MAY-2000';

Output:

```
C:\WINDOWS\system32\CMD.exe - SQLPLUS /NOLOG

SQL> CREATE OR REPLACE TRIGGER PERSON_UPDATE_BEFORE
  2  BEFORE UPDATE ON PERSON1
  3  BEGIN
  4  DBMS_OUTPUT.PUT_LINE('BEFORE UPDATING SOME PERSONS');
  5  END;
  6  /

Trigger created.

SQL> UPDATE PERSON1 SET DOB='21-MAY-2000';
BEFORE UPDATING SOME PERSONS

2 rows updated.

SQL> SELECT * FROM PERSON1;

        ID NAME                           DOB
---------- ------------------------------ ---------
         1 JOHN DOE                       21-MAY-00
         2 JAME DOE                       21-MAY-00

SQL>
```

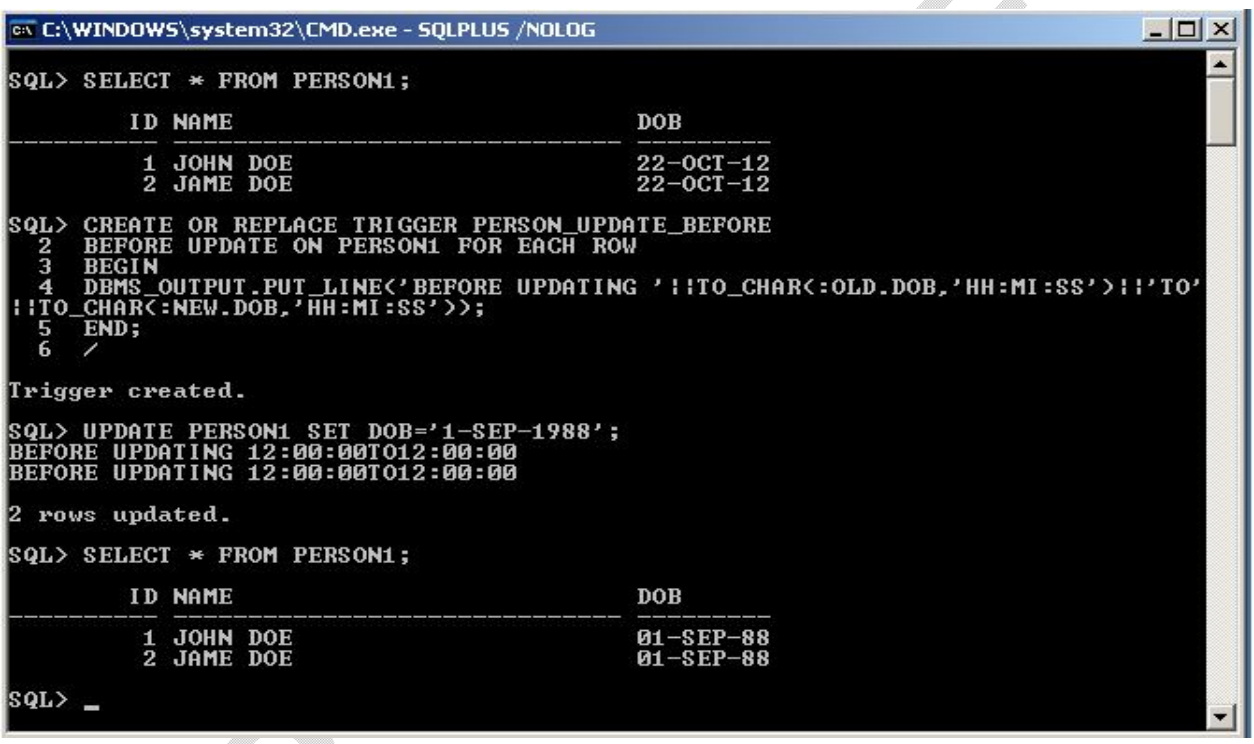
4) Write pl/sql code for each row Before update Trigger.

(a) CREATE OR REPLACE TRIGGER PERSON_UPDATE_BEFORE

```
BEFORE UPDATE ON PERSON1 FOR EACH ROW
BEGIN
DBMS_OUTPUT.PUT_LINE('BEFORE UPDATING
        '||TO_CHAR(:OLD.DOB,'HH:MI:SS')||'TO'||TO_CHAR(:NEW.DOB,'HH:MI:SS'));
END;
/
```

(b) UPDATE PERSON1 SET DOB='1-sep-1988';

Output:



```
C:\WINDOWS\system32\CMD.exe - SQLPLUS /NOLOG

SQL> SELECT * FROM PERSON1;

        ID NAME                           DOB
---------- ------------------------------ ---------
         1 JOHN DOE                       22-OCT-12
         2 JAME DOE                       22-OCT-12

SQL> CREATE OR REPLACE TRIGGER PERSON_UPDATE_BEFORE
  2  BEFORE UPDATE ON PERSON1 FOR EACH ROW
  3  BEGIN
  4  DBMS_OUTPUT.PUT_LINE('BEFORE UPDATING '||TO_CHAR(:OLD.DOB,'HH:MI:SS')||'TO'
||TO_CHAR(:NEW.DOB,'HH:MI:SS'));
  5  END;
  6  /

Trigger created.

SQL> UPDATE PERSON1 SET DOB='1-SEP-1988';
BEFORE UPDATING 12:00:00TO12:00:00
BEFORE UPDATING 12:00:00TO12:00:00

2 rows updated.

SQL> SELECT * FROM PERSON1;

        ID NAME                           DOB
---------- ------------------------------ ---------
         1 JOHN DOE                       01-SEP-88
         2 JAME DOE                       01-SEP-88

SQL> _
```

5) Write pl/sql code for If Statement Trigger.

(a) CREATE OR REPLACE TRIGGER PERSON_BIUD

```
BEFORE INSERT OR UPDATE OR DELETE ON PERSON1 FOR EACH ROW
BEGIN
IF INSERTING THEN
DBMS_OUTPUT.PUT_LINE('INSERTING PERSON:'||:NEW.NAME);
ELSE IF UPDATING THEN
DBMS_OUTPUT.PUT_LINE('UPDATING
PERSON:'||:OLD.NAME||'TO'||:NEW.NAME);
ELSE IF DELETING THEN
DBMS_OUTPUT.PUT_LINE('DELETING PERSON:'||:OLD.NAME);
```

```
END IF;
END IF;
END IF;
END;
/
```

(b) INSERT INTO PERSON1 VALUES(4,'CSE','2-SEP-2009');

(C) UPDATE PERSON1 SET NAME='COMPUTER' WHERE NAME='CSE';

(D) DELETE PERSON1 WHERE NAME='COMPUTER';

(E) SELECT * FROM PERSON1;

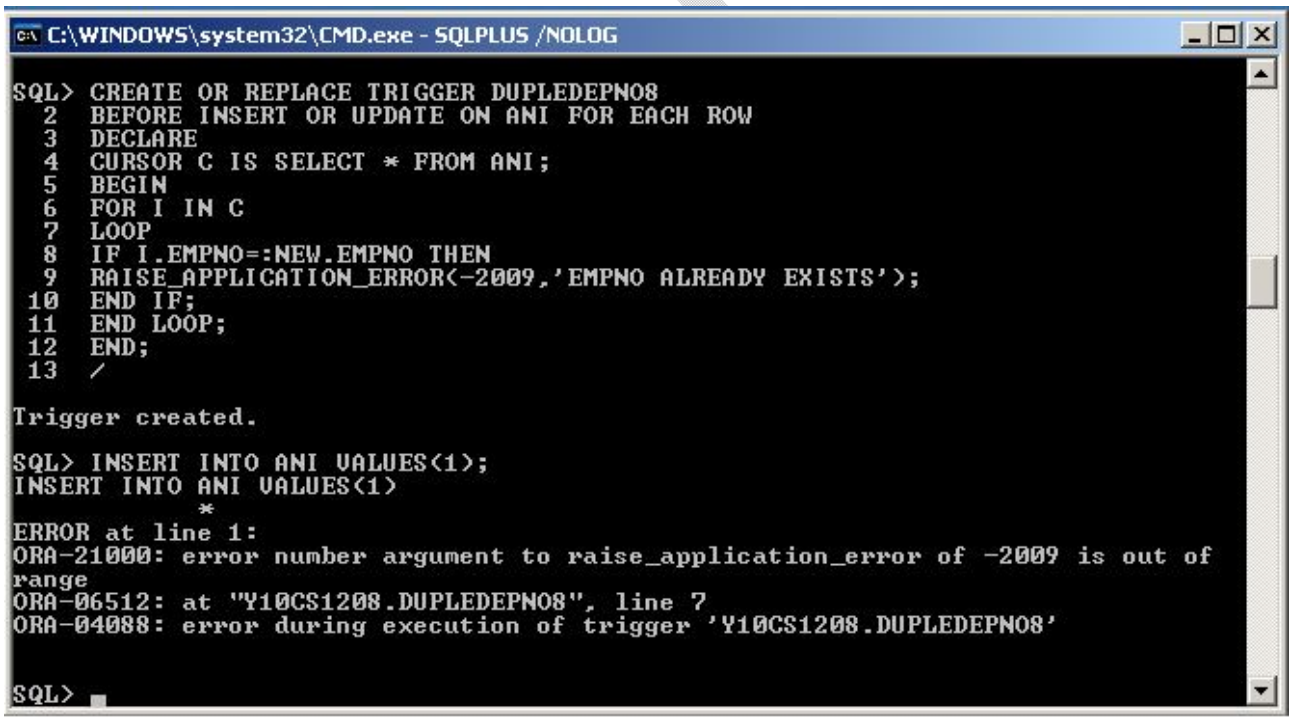
Output:

```
C:\WINDOWS\system32\CMD.exe - SQLPLUS /NOLOG

SQL> CREATE OR REPLACE TRIGGER PERSON_BIUD
  2  BEFORE INSERT OR UPDATE OR DELETE ON PERSON1 FOR EACH ROW
  3  BEGIN
  4  IF INSERTING THEN
  5  DBMS_OUTPUT.PUT_LINE('INSERTING PERSON:'||:NEW.NAME);
  6  ELSE IF UPDATING THEN
  7  DBMS_OUTPUT.PUT_LINE('UPDATING PERSON:'||:OLD.NAME||'TO'||:NEW.NAME);
  8  ELSE IF DELETING THEN
  9  DBMS_OUTPUT.PUT_LINE('DELETING PERSON:'||:OLD.NAME);
 10  END IF;
 11  END IF;
 12  END IF;
 13  END;
 14  /

Trigger created.

SQL> INSERT INTO PERSON1 VALUES(4,'CSE','2-SEP-2009');
INSERTING PERSON:CSE
BEFORE INSERT OFCSE
AFTER INSERT OFCSE

1 row created.

SQL> UPDATE PERSON1 SET NAME='COMPUTER' WHERE NAME='CSE';
UPDATING PERSON:CSETOCOMPUTER
BEFORE UPDATING 12:00:00TO12:00:00

1 row updated.

SQL> DELETE TABLE PERSON1 WHERE NAME='COMPUTER';
DELETE TABLE PERSON1 WHERE NAME='COMPUTER'
       *
ERROR at line 1:
ORA-00903: invalid table name


SQL> DELETE PERSON1 WHERE NAME='COMPUTER';
DELETING PERSON:COMPUTER

1 row deleted.

SQL> SELECT * FROM PERSON1;

        ID NAME                           DOB
---------- ------------------------------ ---------
         1 JOHN DOE                       01-SEP-88
```

6. Write pl/sql code in Trigger not to accept the existing Empno (Unique no)

```
CREATE OR REPLACE TRIGGER DUPLEDEPNO8
BEFORE INSERT OR UPDATE ON ANI FOR EACH ROW
DECLARE
CURSOR C IS SELECT * FROM ANI;
BEGIN
FOR I IN C
LOOP
IF I.EMPNO=:NEW.EMPNO THEN
RAISE_APPLICATION_ERROR(-2009,'EMPNO ALREADY EXISTS');
END IF;
END LOOP;
END;
/
```

Output:



```
C:\WINDOWS\system32\CMD.exe - SQLPLUS /NOLOG

SQL> CREATE OR REPLACE TRIGGER DUPLEDEPNO8
  2  BEFORE INSERT OR UPDATE ON ANI FOR EACH ROW
  3  DECLARE
  4  CURSOR C IS SELECT * FROM ANI;
  5  BEGIN
  6  FOR I IN C
  7  LOOP
  8  IF I.EMPNO=:NEW.EMPNO THEN
  9  RAISE_APPLICATION_ERROR(-2009,'EMPNO ALREADY EXISTS');
 10  END IF;
 11  END LOOP;
 12  END;
 13  /

Trigger created.

SQL> INSERT INTO ANI VALUES(1);
INSERT INTO ANI VALUES(1)
            *
ERROR at line 1:
ORA-21000: error number argument to raise_application_error of -2009 is out of
range
ORA-06512: at "Y10CS1208.DUPLEDEPNO8", line 7
ORA-04088: error during execution of trigger 'Y10CS1208.DUPLEDEPNO8'

SQL>
```

8. Write pl/sql code using Trigger to salary with more than old salary.

```
CREATE OR REPLACE TRIGGER SALUPDATE9
BEFORE UPDATE ON C FOR EACH ROW
BEGIN
```

```
IF :NEW.SAL<:OLD.SAL THEN
RAISE_APPLICATION_ERROR(-20005,'NEW SALARY IS LESSER THAN OLD SALARY');
END IF;
END;
/
```

Output:

```
C:\WINDOWS\system32\CMD.exe - SQLPLUS /NOLOG

SQL> CREATE OR REPLACE TRIGGER SALUPDATE9
  2  BEFORE UPDATE ON C FOR EACH ROW
  3  BEGIN
  4  IF :NEW.SAL<:OLD.SAL THEN
  5  RAISE_APPLICATION_ERROR(-20005,'NEW SALARY IS LESSER THAN OLD SALARY');
  6  END IF;
  7  END;
  8  /

Trigger created.

SQL> UPDATE C SET SAL=2 WHERE EMPNO=5;
UPDATE C SET SAL=2 WHERE EMPNO=5
       *
ERROR at line 1:
ORA-20005: NEW SALARY IS LESSER THAN OLD SALARY
ORA-06512: at "Y10CS1208.SALUPDATE9", line 3
ORA-04088: error during execution of trigger 'Y10CS1208.SALUPDATE9'


SQL> SELECT * FROM C;

       SAL      EMPNO
---------- ----------
       500          5
       400          6
         5          5

SQL>
```