SOMAIYA
VIDYAVIHAR UNIVERSITY
K J Somaiya Institute of Management

DBMS JOURNAL

FULL NAME: Karan Dinesh Panchal
ROLL No: 33
Course : FY MCA / Sem I / Database Application [Course Code : 217P09L101]
Subject: Database Application Lab
Subject-In-Charge: Prof. Sangeetha Rajesh

INDEX

Practical 1 : Data Definition Language

Data description language is a syntax for creating and modifying database objects such as tables, indices, and users.

a. Create

Syntax - create table table_name(column1 datatype, column2 datatype,...)

Code:

create table customers(

ID int primary key,

cname varchar2(20),

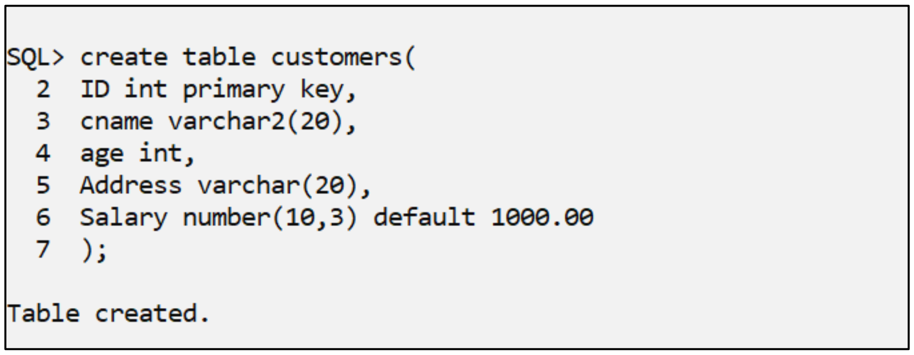
age int,

Address varchar(20),

Salary number(10,3) default 1000.00

);

Output:



```
SQL> create table customers(
  2  ID int primary key,
  3  cname varchar2(20),
  4  age int,
  5  Address varchar(20),
  6  Salary number(10,3) default 1000.00
  7  );

Table created.
```

Syntax - create table table_name(column1 datatype constraint, column2 datatype constraint,...)

Code:

Create table supplier (

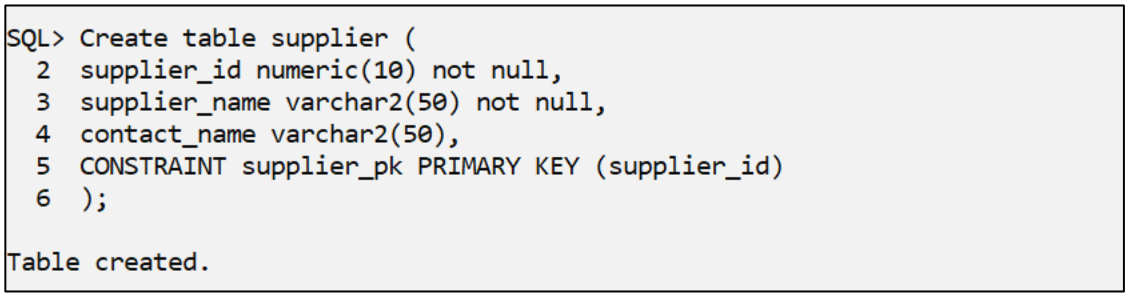
supplier_id numeric(10) not null,

supplier_name varchar2(50) not null,

contact_name varchar2(50),

CONSTRAINT supplier_pk PRIMARY KEY (supplier_id));

Output:


```
SQL> Create table supplier (
  2  supplier_id numeric(10) not null,
  3  supplier_name varchar2(50) not null,
  4  contact_name varchar2(50),
  5  CONSTRAINT supplier_pk PRIMARY KEY (supplier_id)
  6  );

Table created.
```

Query: With Composite Key

Syntax - create table table_name(column1 datatype constraint, column2 datatype constraint,...)

Code:

Create table supplier2 (

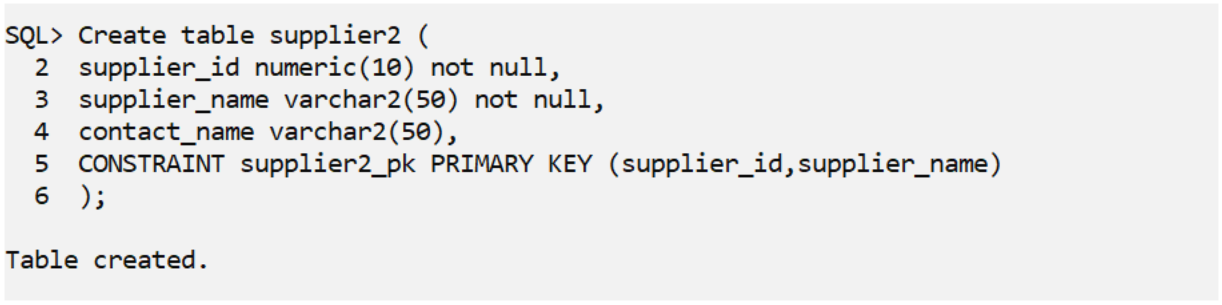
supplier_id numeric(10) not null,

supplier_name varchar2(50) not null,

contact_name varchar2(50),

CONSTRAINT supplier2_pk PRIMARY KEY (supplier_id,supplier_name));

Output:

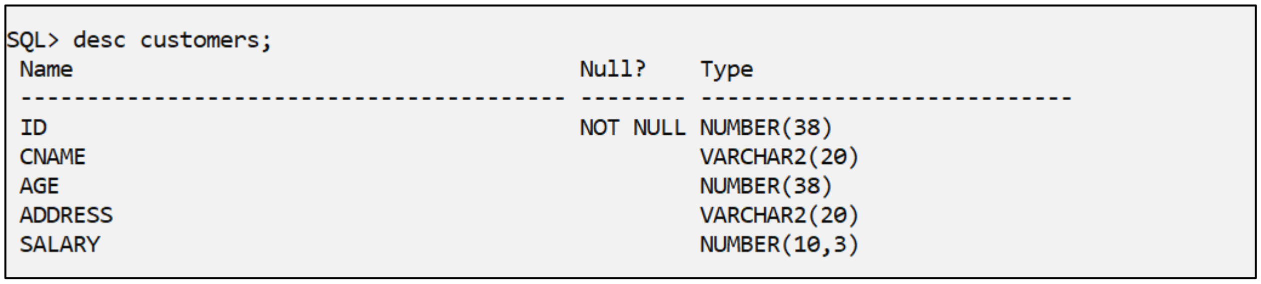

```
SQL> Create table supplier2 (
  2  supplier_id numeric(10) not null,
  3  supplier_name varchar2(50) not null,
  4  contact_name varchar2(50),
  5  CONSTRAINT supplier2_pk PRIMARY KEY (supplier_id,supplier_name)
  6  );

Table created.
```

Describe Relation

Syntax: DESCRIBE table_name;

desc customers;

describe customers;


```
SQL> desc customers;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER(38)
 CNAME                                              VARCHAR2(20)
 AGE                                                NUMBER(38)
 ADDRESS                                            VARCHAR2(20)
 SALARY                                             NUMBER(10,3)
```

Data Types

Character Datatypes

- Char(size)
- VARCHAR
- VARCHAR2

Numeric Datatypes

- INTEGER
- FLOAT
- NUMBER(p,s) ex NUMERIC(10,3)
- DECIMAL
- BOOLEAN

Date/Time Datatypes

- DATE
- TIMESTAMP
- Large Object LOB
- BLOB

Syntax - create table table_name(column1 datatype constraint, column2 datatype constraint,...)

Code:

Create table student(

student_roll_no integer not null

, student_name varchar2(15) not null

, student_programme varchar2 (5)

, student_semester varchar2 (3)

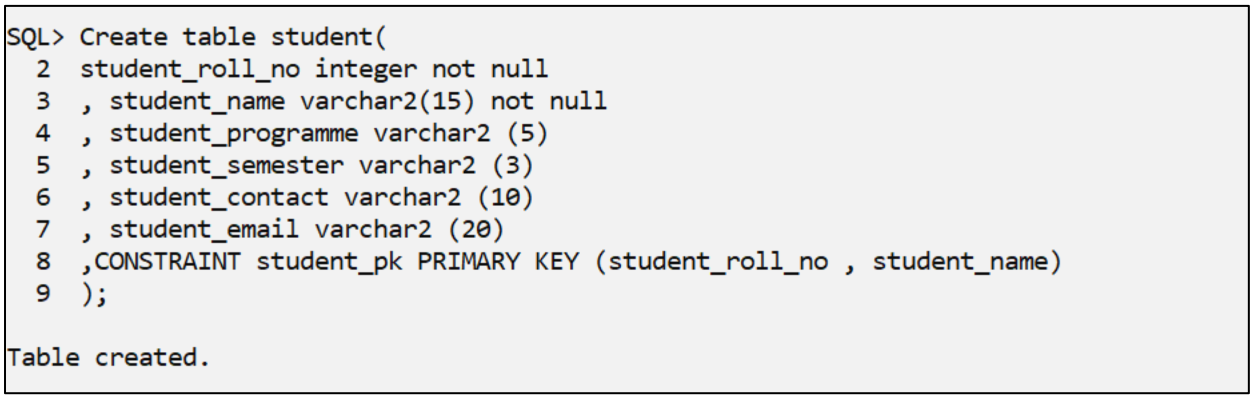
, student_contact varchar2 (10)

, student_email varchar2 (20)

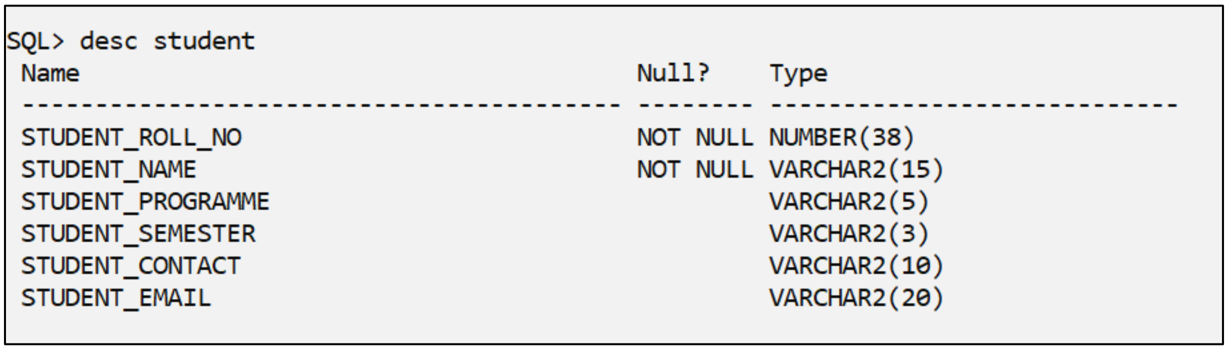
,CONSTRAINT student_pk PRIMARY KEY (student_roll_no , student_name)

);

Output:



```
SQL> Create table student(
  2  student_roll_no integer not null
  3  , student_name varchar2(15) not null
  4  , student_programme varchar2 (5)
  5  , student_semester varchar2 (3)
  6  , student_contact varchar2 (10)
  7  , student_email varchar2 (20)
  8  ,CONSTRAINT student_pk PRIMARY KEY (student_roll_no , student_name)
  9  );

Table created.
```



```
SQL> desc student
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENT_ROLL_NO                           NOT NULL NUMBER(38)
 STUDENT_NAME                              NOT NULL VARCHAR2(15)
 STUDENT_PROGRAMME                                  VARCHAR2(5)
 STUDENT_SEMESTER                                   VARCHAR2(3)
 STUDENT_CONTACT                                    VARCHAR2(10)
 STUDENT_EMAIL                                      VARCHAR2(20)
```

Query: Create EMPLOYEE table with following

ID(not null,number(5),primary key)

Empname(not null,varchar(20))

Address (varchar(30))

Age > 18 integer

Salary(Number (10,2))

Code: Create table employee(

eid number(5) not null

,ename varchar(15) not null

,address varchar(30)

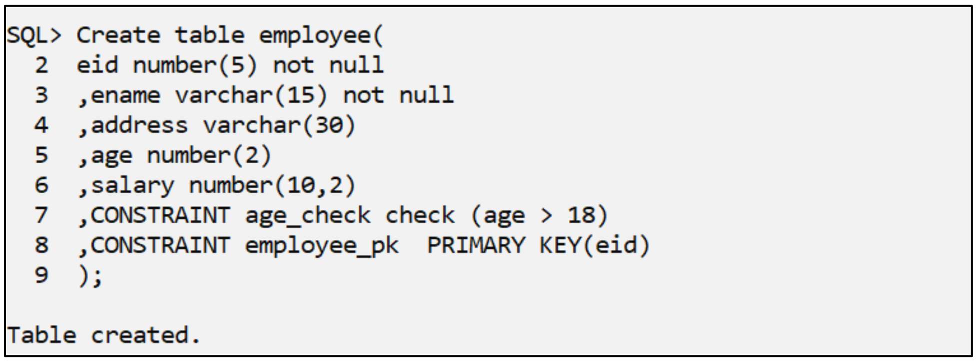
,age number(2)

,salary number(10,2)

,CONSTRAINT age_check check (age > 18)

,CONSTRAINT employee_pk PRIMARY KEY(eid));

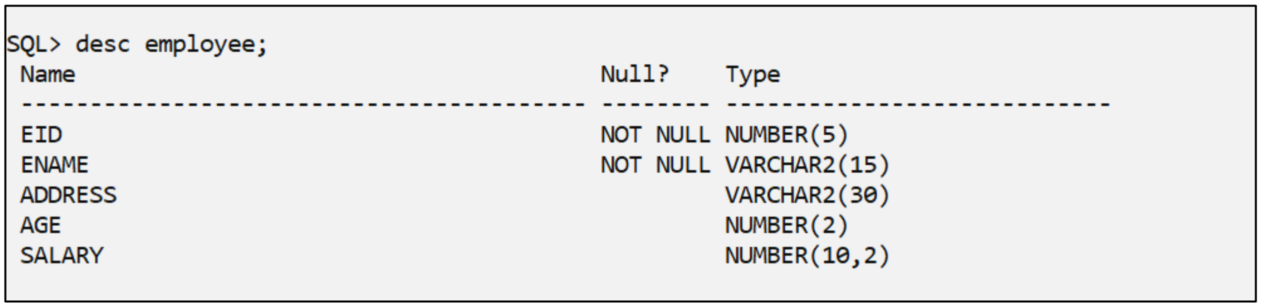
Output:



```
SQL> Create table employee(
  2  eid number(5) not null
  3  ,ename varchar(15) not null
  4  ,address varchar(30)
  5  ,age number(2)
  6  ,salary number(10,2)
  7  ,CONSTRAINT age_check check (age > 18)
  8  ,CONSTRAINT employee_pk  PRIMARY KEY(eid)
  9  );

Table created.
```

Syntax - desc table_name; / describe table_name
Code: desc employee;

Ouput:



```
SQL> desc employee;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EID                                       NOT NULL NUMBER(5)
 ENAME                                     NOT NULL VARCHAR2(15)
 ADDRESS                                            VARCHAR2(30)
 AGE                                                NUMBER(2)
 SALARY                                             NUMBER(10,2)
```

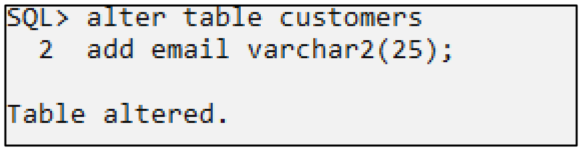
b. Alter : this statement is used to add, delete, or modify columns in an existing table.

Syntax: Add Column (ALTER TABLE table_name ADD column_name data_type;)

Code: alter table customers

add email varchar2(25);

Output:

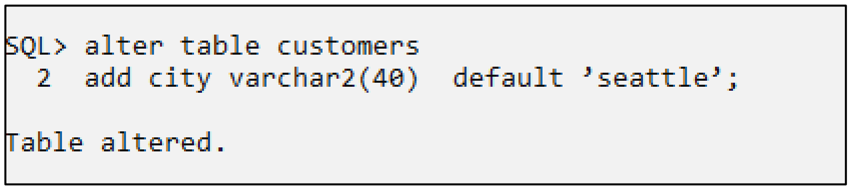


```
SQL> alter table customers
  2  add email varchar2(25);

Table altered.
```

Code: alter table customers

add city varchar2(40) default 'seattle';

Output:



```
SQL> alter table customers
  2  add city varchar2(40)  default 'seattle';

Table altered.
```

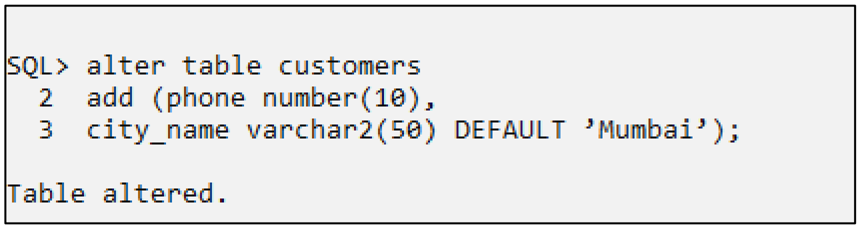
Query: Add more than one

Syntax - Alter Table Table_name Add (Column_1 Column Definition, Column_2 Column Definition, … Column_n Column Definition);
Code: alter table customers

add (phone number(10),

city_name varchar2(50) DEFAULT 'Mumbai');

Output:



```
SQL> alter table customers
  2  add (phone number(10),
  3  city_name varchar2(50) DEFAULT 'Mumbai');

Table altered.
```

Modify Column:

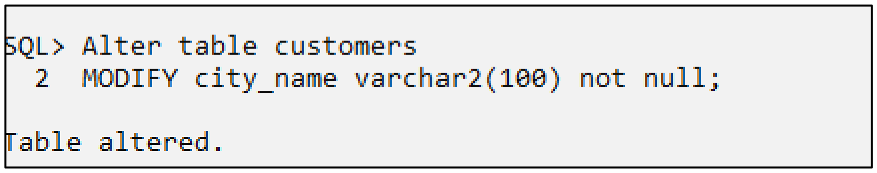
Syntax:Alter Table table_nameMODIFY column_name column_type;

Query: Modify existing one column

Code: Alter table customers

MODIFY city_name varchar2(100) not null;

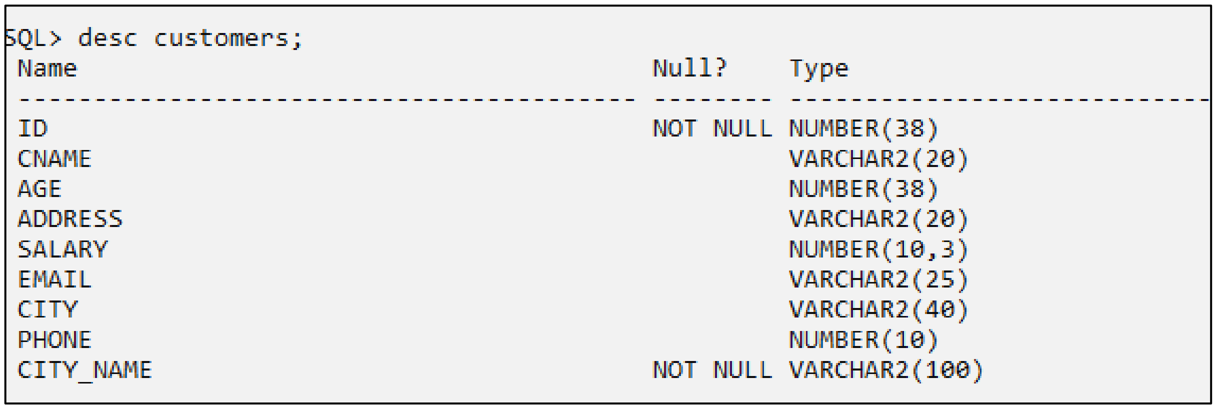
Output:



```
SQL> Alter table customers
  2  MODIFY city_name varchar2(100) not null;

Table altered.
```

Code: Desc customers;

Output:



```
SQL> desc customers;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER(38)
 CNAME                                              VARCHAR2(20)
 AGE                                                NUMBER(38)
 ADDRESS                                            VARCHAR2(20)
 SALARY                                             NUMBER(10,3)
 EMAIL                                              VARCHAR2(25)
 CITY                                               VARCHAR2(40)
 PHONE                                              NUMBER(10)
 CITY_NAME                                 NOT NULL VARCHAR2(100)
```

Query: Modify existing more than one column

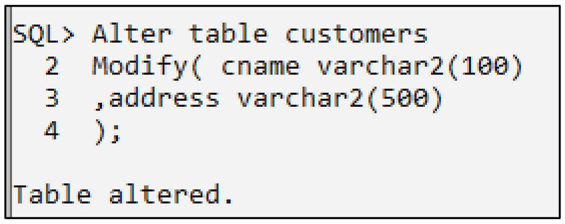
Syntax : Alter table table_name Modify(col_name column_type, col_name1 column_type);

Code: Alter table customers

Modify(cname varchar2(100)

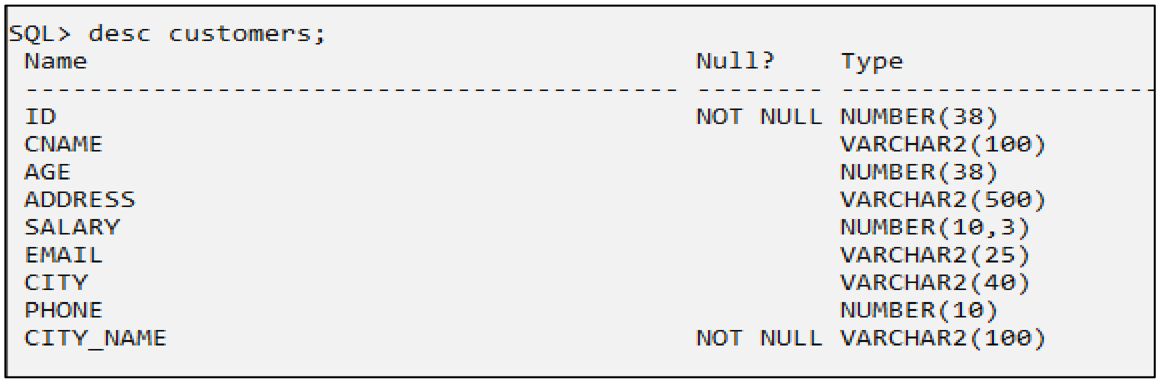
,address varchar2(500));

Output:



```
SQL> Alter table customers
  2  Modify( cname varchar2(100)
  3  ,address varchar2(500)
  4  );

Table altered.
```

Code: Desc customers;
Output:



```
SQL> desc customers;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 ID                                        NOT NULL NUMBER(38)
 CNAME                                              VARCHAR2(100)
 AGE                                                NUMBER(38)
 ADDRESS                                            VARCHAR2(500)
 SALARY                                             NUMBER(10,3)
 EMAIL                                              VARCHAR2(25)
 CITY                                               VARCHAR2(40)
 PHONE                                              NUMBER(10)
 CITY_NAME                                 NOT NULL VARCHAR2(100)
```

Query : Alter Constraint

Syntax : Alter table table_name

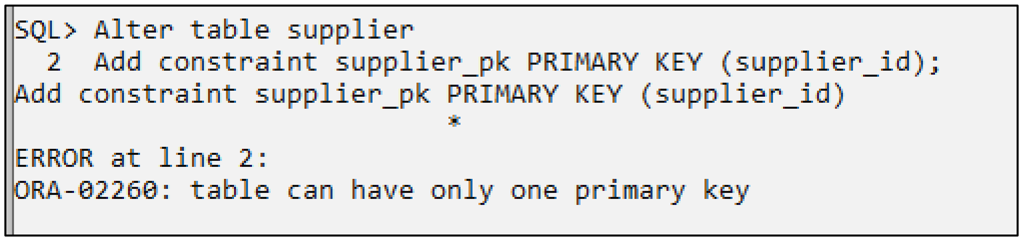
Add CONSTRAINT constraint_name primary Key(col_name1,column2,…n);

Code:

Alter table supplier

Add constraint supplier_pk PRIMARY KEY (supplier_id);

Output:



```
SQL> Alter table supplier
  2  Add constraint supplier_pk PRIMARY KEY (supplier_id);
Add constraint supplier_pk PRIMARY KEY (supplier_id)
                           *
ERROR at line 2:
ORA-02260: table can have only one primary key
```

Query : DROP CONSTRAINT:

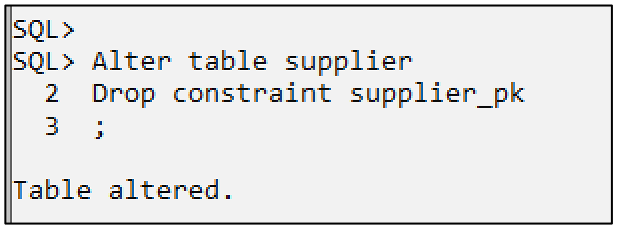
Syntax : Alter table table_name Drop constraint constraint_name;

Code:

Alter table supplier

Drop constraint supplier_pk;

Output:



```
SQL>
SQL> Alter table supplier
  2  Drop constraint supplier_pk
  3  ;

Table altered.
```

Query : create table products

Code:

Create table Product(

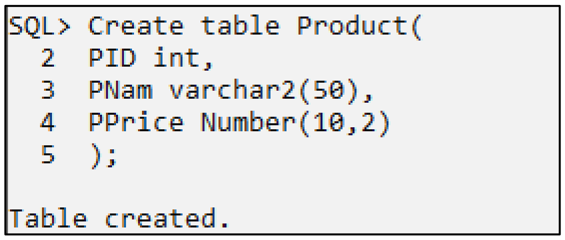
PID int,

PNam varchar2(50),

PPrice Number(10,2)

);

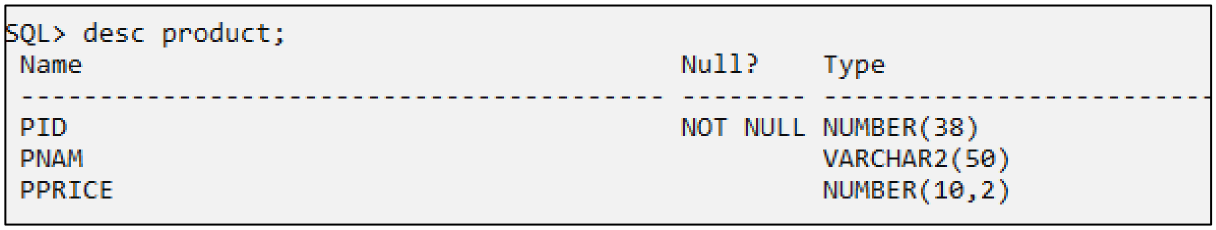
Output:



```
SQL> Create table Product(
  2  PID int,
  3  PNam varchar2(50),
  4  PPrice Number(10,2)
  5  );

Table created.
```

Code: desc product;

Output:



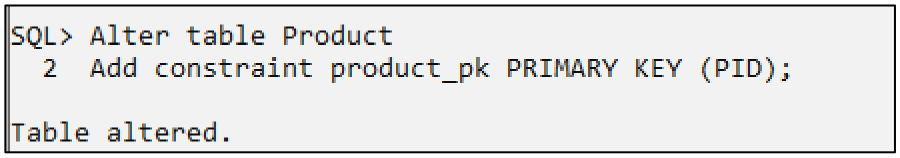
```
SQL> desc product;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PID                                       NOT NULL NUMBER(38)
 PNAM                                               VARCHAR2(50)
 PPRICE                                             NUMBER(10,2)
```

Query : Add Constraint

Code: Alter table Product

Add constraint product_pk PRIMARY KEY (PID);

Output:



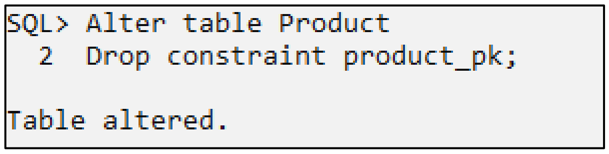
```
SQL> Alter table Product
  2  Add constraint product_pk PRIMARY KEY (PID);

Table altered.
```

Query : Drop Constraint

Syntax- Alter Table Table_name Drop Constraint Constraint_name;
Code: Alter table Product

Drop constraint product_pk;

Output:



```
SQL> Alter table Product
  2  Drop constraint product_pk;

Table altered.
```

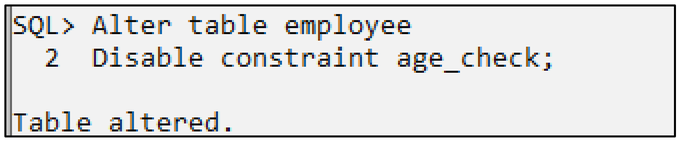
Query : Disable Constraint

Syntax- Alter Table Table_name Disable Constraint Constraint Name;

Code: Alter table employee

Disable constraint age_check;

Output:



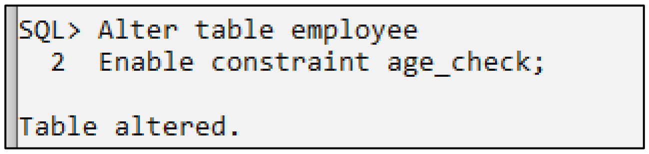
```
SQL> Alter table employee
  2  Disable constraint age_check;

Table altered.
```

Query : Enable Constraint

Syntax- Alter Table Table_name Enable Constraint Constraint Name;
Code: Alter table employee

Enable constraint age_check;

Output:



```
SQL> Alter table employee
  2  Enable constraint age_check;

Table altered.
```

c. Drop

<u>**SYNTAX:**</u>

Alter table table_name

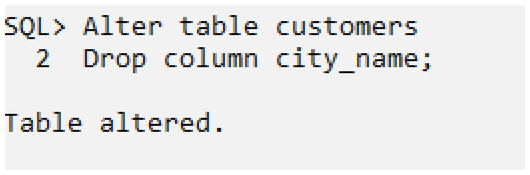
Drop column_name;

Code:

Alter table customers

Drop column city_name;

Output:



```
SQL> Alter table customers
  2  Drop column city_name;

Table altered.
```

d. Rename

SYNTAX:

Alter table table_name

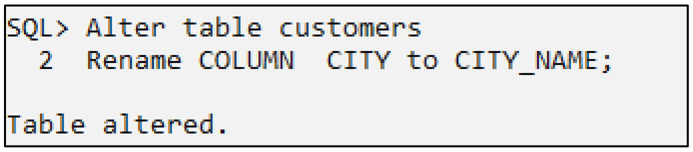
Rename COLUMN old_name TO new_name;

Code:

Alter table customers

Rename COLUMN CITY to CITY_NAME;

Output:

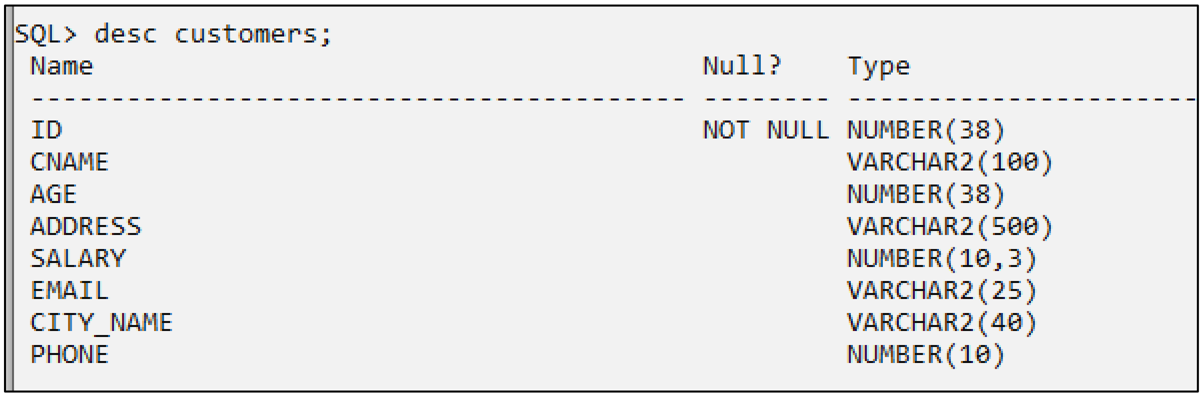


```
SQL> Alter table customers
  2  Rename COLUMN  CITY to CITY_NAME;

Table altered.
```

Code:

Desc customers;

Output:



```
SQL> desc customers;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 ID                                        NOT NULL NUMBER(38)
 CNAME                                              VARCHAR2(100)
 AGE                                                NUMBER(38)
 ADDRESS                                            VARCHAR2(500)
 SALARY                                             NUMBER(10,3)
 EMAIL                                              VARCHAR2(25)
 CITY_NAME                                          VARCHAR2(40)
 PHONE                                              NUMBER(10)
```

Rename Table

SYNTAX:

Alter table table_name

Rename to new_name;

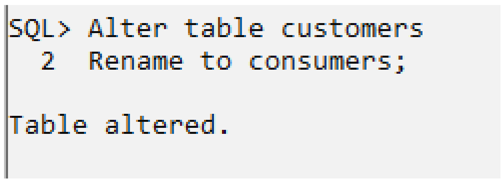
Query: change customer table name to consumer

Code:

Alter table customers

Rename to consumers;

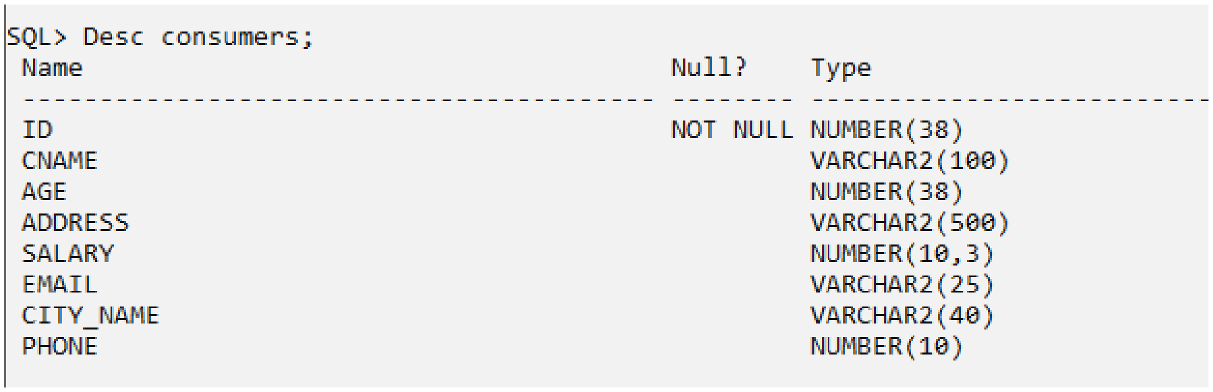
Output:



```
SQL> Alter table customers
  2  Rename to consumers;

Table altered.
```

Code: Desc consumers;

Output:



```
SQL> Desc consumers;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER(38)
 CNAME                                              VARCHAR2(100)
 AGE                                                NUMBER(38)
 ADDRESS                                            VARCHAR2(500)
 SALARY                                             NUMBER(10,3)
 EMAIL                                              VARCHAR2(25)
 CITY_NAME                                          VARCHAR2(40)
 PHONE                                              NUMBER(10)
```

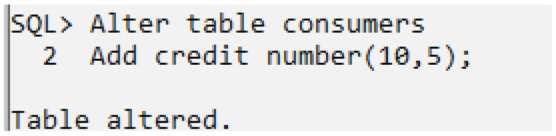
<u>Exercise</u>

Query : . Add column credit to table

Code: Alter table consumers

Add credit number(10,5);

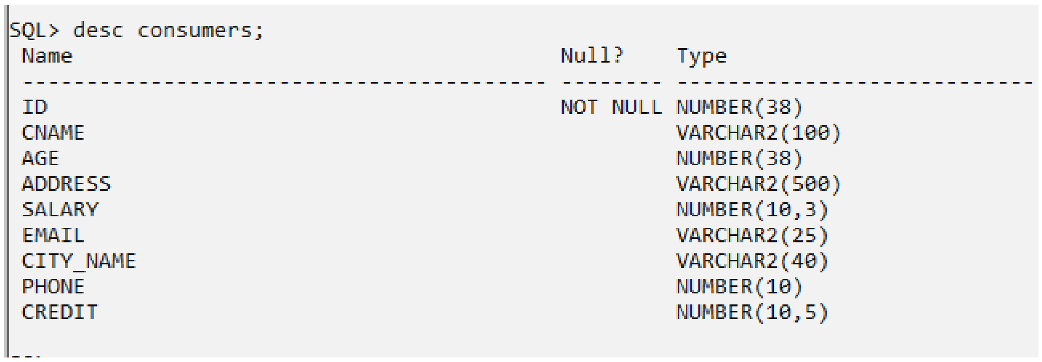
Output:



```
SQL> Alter table consumers
  2  Add credit number(10,5);

Table altered.
```

Code: Desc consumers;

Output:



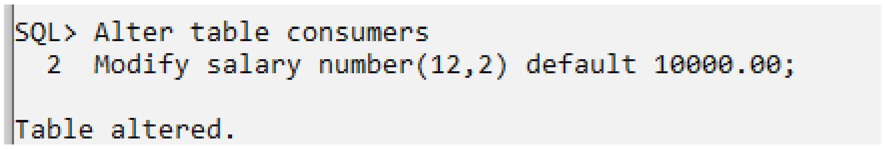
```
SQL> desc consumers;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER(38)
 CNAME                                              VARCHAR2(100)
 AGE                                                NUMBER(38)
 ADDRESS                                            VARCHAR2(500)
 SALARY                                             NUMBER(10,3)
 EMAIL                                              VARCHAR2(25)
 CITY_NAME                                          VARCHAR2(40)
 PHONE                                              NUMBER(10)
 CREDIT                                             NUMBER(10,5)
```

Query : . Change the size of salary to (12,2) and default value to 10000

Code: Alter table consumers

Modify salary number(12,2) default 10000.00;

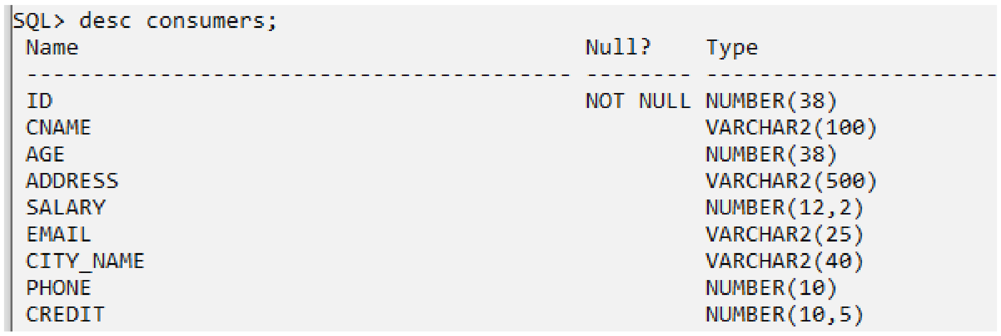
Output:



```
SQL> Alter table consumers
  2  Modify salary number(12,2) default 10000.00;

Table altered.
```

Code: Desc consumers;

Output:



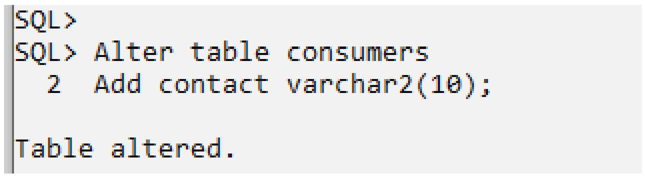
```
SQL> desc consumers;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 ID                                        NOT NULL NUMBER(38)
 CNAME                                              VARCHAR2(100)
 AGE                                                NUMBER(38)
 ADDRESS                                            VARCHAR2(500)
 SALARY                                             NUMBER(12,2)
 EMAIL                                              VARCHAR2(25)
 CITY_NAME                                          VARCHAR2(40)
 PHONE                                              NUMBER(10)
 CREDIT                                             NUMBER(10,5)
```

Query : .Add column contact

Code: Alter table consumers

Add contact varchar2(10);

Output:



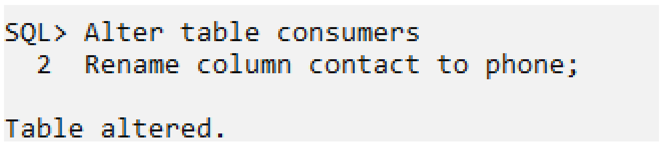
```
SQL>
SQL> Alter table consumers
  2  Add contact varchar2(10);

Table altered.
```

Query : Rename contact to phone in consumers table

Code: Alter table consumers

Rename column contact to phone;

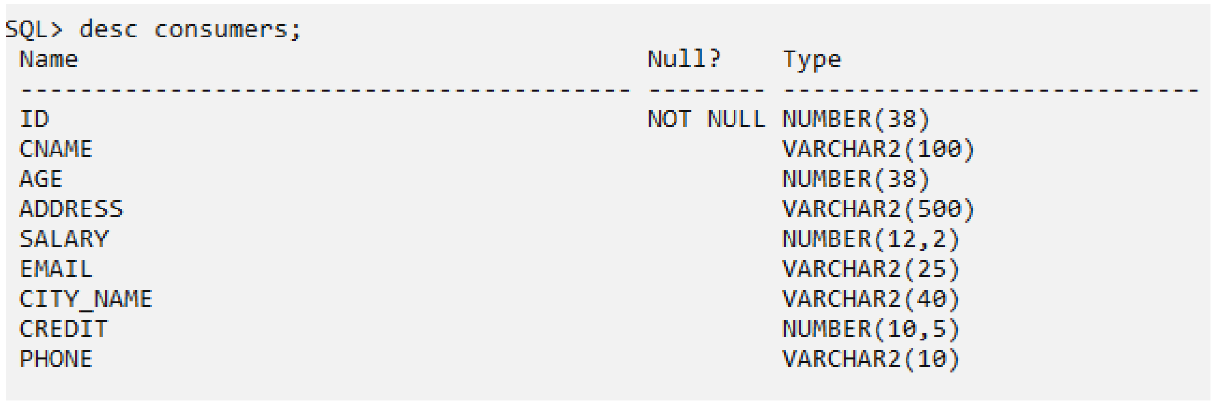
Output:



```
SQL> Alter table consumers
  2  Rename column contact to phone;

Table altered.
```

Code: Desc consumers;

Output:



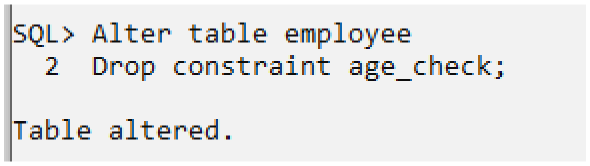
```
SQL> desc consumers;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER(38)
 CNAME                                              VARCHAR2(100)
 AGE                                                NUMBER(38)
 ADDRESS                                            VARCHAR2(500)
 SALARY                                             NUMBER(12,2)
 EMAIL                                              VARCHAR2(25)
 CITY_NAME                                          VARCHAR2(40)
 CREDIT                                             NUMBER(10,5)
 PHONE                                              VARCHAR2(10)
```

Query : drop constraint for age

Code: Alter table employee

Drop constraint age_check;

Output:



```
SQL> Alter table employee
  2  Drop constraint age_check;

Table altered.
```

Query : add constraint

1. Salary should be more than 2000

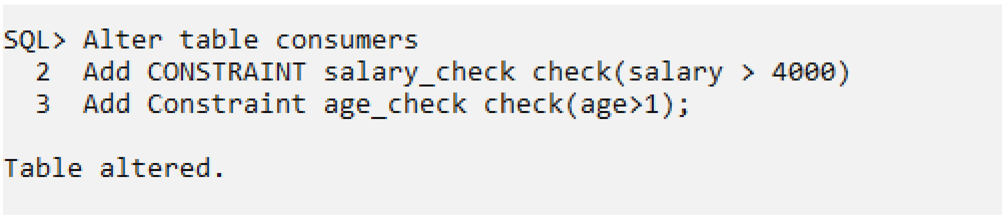
2. Age more than 1

Code: Alter table consumers

Add CONSTRAINT salary_check check(salary > 4000)

Add Constraint age_check check(age>1);

Output:



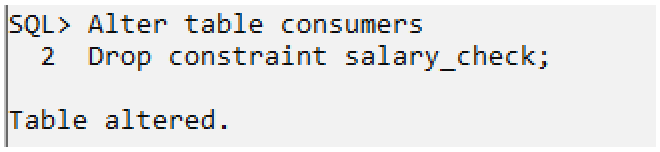
```
SQL> Alter table consumers
  2  Add CONSTRAINT salary_check check(salary > 4000)
  3  Add Constraint age_check check(age>1);

Table altered.
```

Query : Remove constraint : salary should be more than 4000

Code: Alter table consumers

Drop constraint salary_check;

Output:



```
SQL> Alter table consumers
  2  Drop constraint salary_check;

Table altered.
```

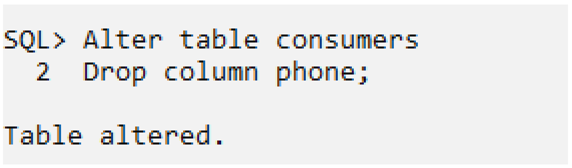
Query : Delete phone from consumer;

Code:

Alter table consumers

Drop column phone;

Output:

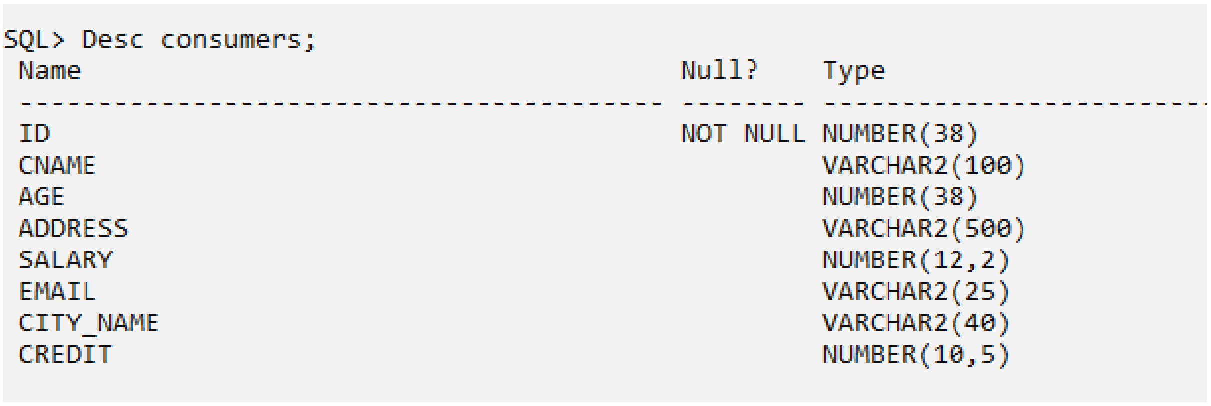


```
SQL> Alter table consumers
  2  Drop column phone;

Table altered.
```

Code:

Desc consumers;

Output:



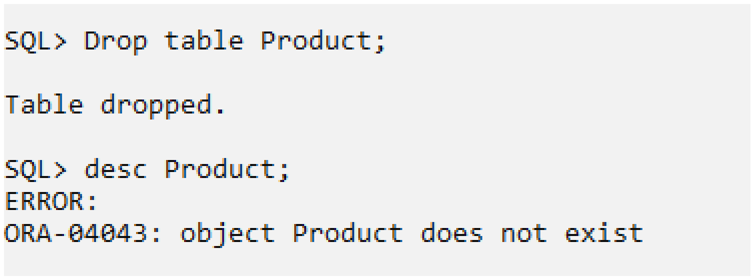
```
SQL> Desc consumers;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
 ID                                        NOT NULL NUMBER(38)
 CNAME                                              VARCHAR2(100)
 AGE                                                NUMBER(38)
 ADDRESS                                            VARCHAR2(500)
 SALARY                                             NUMBER(12,2)
 EMAIL                                              VARCHAR2(25)
 CITY_NAME                                          VARCHAR2(40)
 CREDIT                                             NUMBER(10,5)
```

Query : delete table from database

Syntax : Drop table table_name [CASCADE];

Code: Drop table Product;

Output:



```
SQL> Drop table Product;

Table dropped.

SQL> desc Product;
ERROR:
ORA-04043: object Product does not exist
```

Query : Create table with given attributes

Code:

create table ott(

 contentID varchar2(5)

,title varchar2(20) not null

,description varchar2(20)

,releaseDate varchar2(20)

,genre varchar2(10)

,director varchar2(20)

,seasons int

,episodeCount int default 25

,duration int

,rating varchar2(5) not null

,Price int

,availablity varchar2(5)

,langugae varchar2(20)

,userWatched varchar2(1)

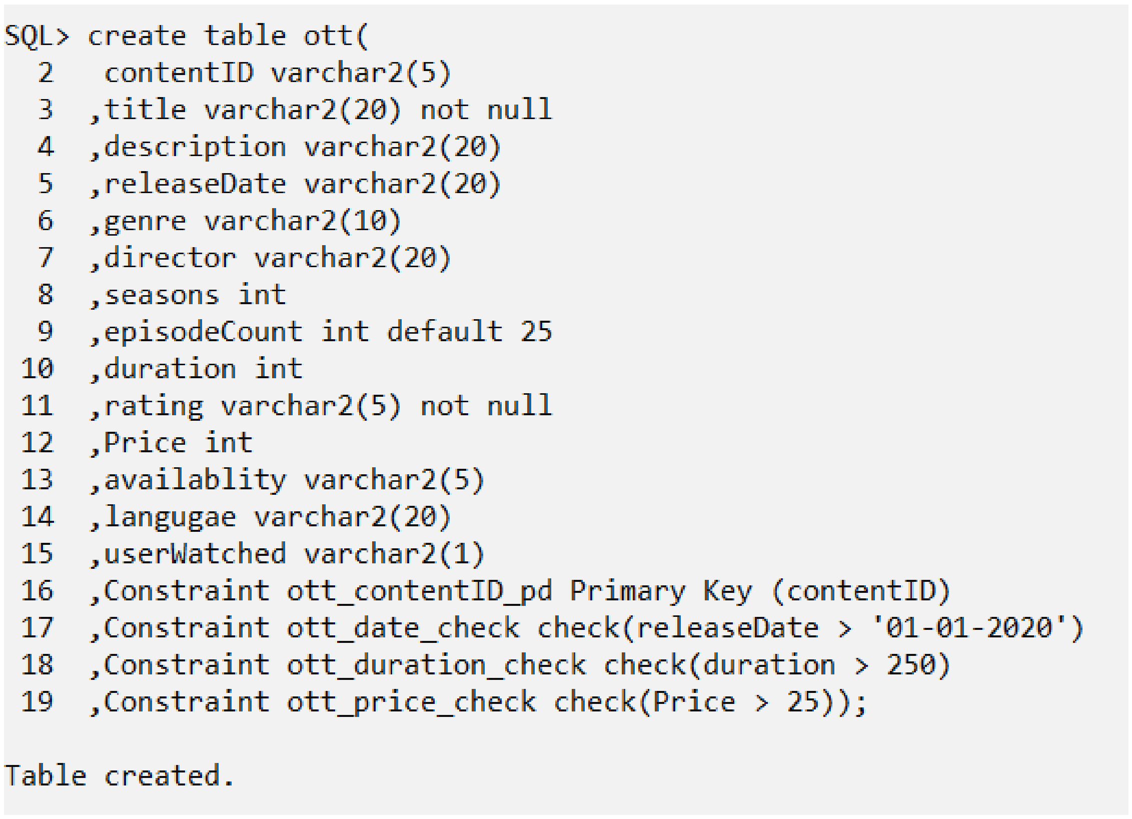
,Constraint ott_contentID_pd Primary Key (contentID)

,Constraint ott_date_check check(releaseDate > '01-01-2020')

,Constraint ott_duration_check check(duration > 250)

,Constraint ott_price_check check(Price > 25));

Output:



```
SQL> create table ott(
 contentID varchar2(5)
,title varchar2(20) not null
,description varchar2(20)
,releaseDate varchar2(20)
,genre varchar2(10)
,director varchar2(20)
,seasons int
,episodeCount int default 25
,duration int
,rating varchar2(5) not null
,Price int
,availablity varchar2(5)
,langugae varchar2(20)
,userWatched varchar2(1)
,Constraint ott_contentID_pd Primary Key (contentID)
,Constraint ott_date_check check(releaseDate > '01-01-2020')
,Constraint ott_duration_check check(duration > 250)
,Constraint ott_price_check check(Price > 25));

Table created.
```

Code:

Desc ott;

Output:

```
SQL> desc ott;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CONTENTID                                 NOT NULL VARCHAR2(5)
 TITLE                                     NOT NULL VARCHAR2(20)
 DESCRIPTION                                        VARCHAR2(20)
 RELEASEDATE                                        VARCHAR2(20)
 GENRE                                              VARCHAR2(10)
 DIRECTOR                                           VARCHAR2(20)
 SEASONS                                            NUMBER(38)
 EPISODECOUNT                                       NUMBER(38)
 DURATION                                           NUMBER(38)
 RATING                                    NOT NULL VARCHAR2(5)
 PRICE                                              NUMBER(38)
 AVAILABLITY                                        VARCHAR2(5)
 LANGUGAE                                           VARCHAR2(20)
 USERWATCHED                                        VARCHAR2(1)
```

Practical 2 : Data Manipulation Language

A data manipulation language is a computer programming language used for adding, deleting, and modifying data in a database.

a) Insert

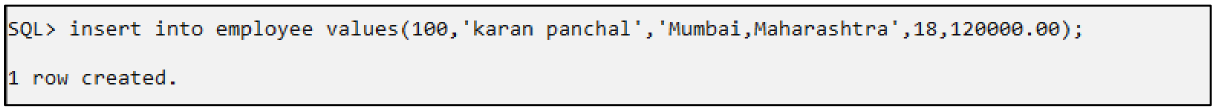
Syntax :

Insert into table_name (col1,col2,col3…..N)Values(value1,value2,value…N);

Code:

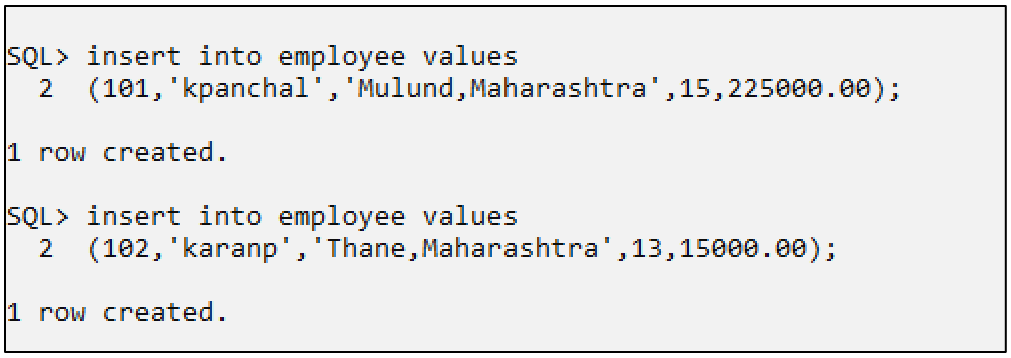
insert into employee values(100,'karan panchal','Mumbai,Maharashtra',18,120000.00);

Output:



```
SQL> insert into employee values(100,'karan panchal','Mumbai,Maharashtra',18,120000.00);

1 row created.
```

Output:



```
SQL> insert into employee values
  2  (101,'kpanchal','Mulund,Maharashtra',15,225000.00);

1 row created.

SQL> insert into employee values
  2  (102,'karanp','Thane,Maharashtra',13,15000.00);

1 row created.
```

Query : <u>Insert 3 tuples in student</u>

Syntax :

Insert into table_name (col1,col2,col3…..N)Values(value1,value2,value…N);

Code:

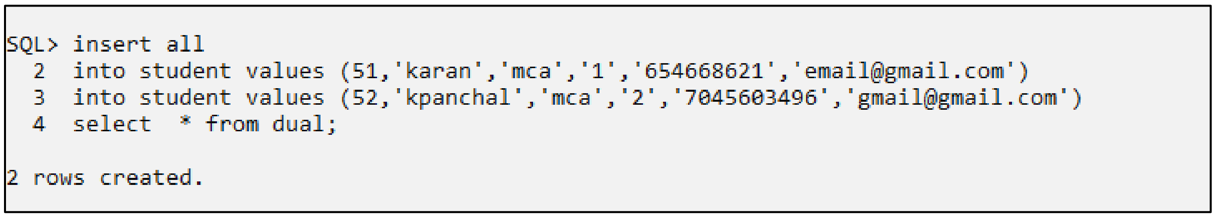
insert all

into student values (51,'karan','mca','1','654668621','email@gmail.com')

into student values (52,'kpanchal','mca','2','7045603496','gmail@gmail.com')

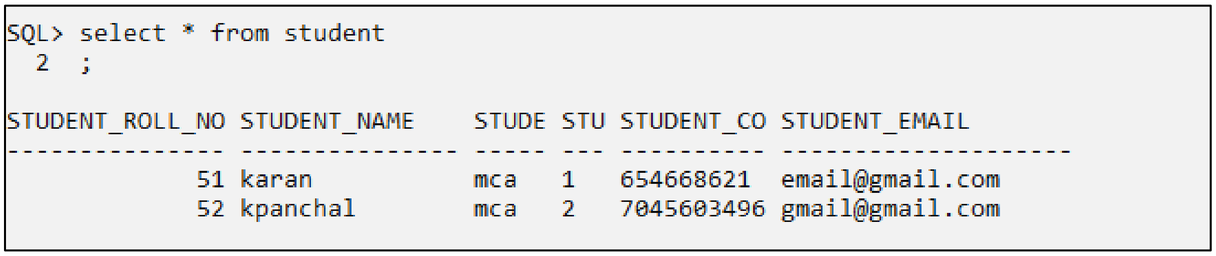
select * from dual;

Output:


```
SQL> insert all
  2  into student values (51,'karan','mca','1','654668621','email@gmail.com')
  3  into student values (52,'kpanchal','mca','2','7045603496','gmail@gmail.com')
  4  select  * from dual;

2 rows created.
```

Syntax – select * from table_name;
Code: Select * from student;

Output:


```
SQL> select * from student
  2  ;

STUDENT_ROLL_NO STUDENT_NAME    STUDE STU STUDENT_CO STUDENT_EMAIL
--------------- --------------- ----- --- ---------- --------------------
             51 karan           mca   1   654668621  email@gmail.com
             52 kpanchal        mca   2   7045603496 gmail@gmail.com
```

Query : Insert 5 records into supplier

Syntax :

Syntax - INSERT ALL
```
                INTO mytable (column1, column2, column_n) values (expr1, expr2,…exprN)
                INTO mytable (column1, column2, column_n) values (expr1, expr2,…exprN)
                INTO mytable (column1, column2, column_n) values (expr1, expr2,…exprN)
        SELECT * FROM dual;
```
Code insert all

into supplier values (1,'S-name-1','0987654321')

into supplier values (2,'S-name-2','0987654321')

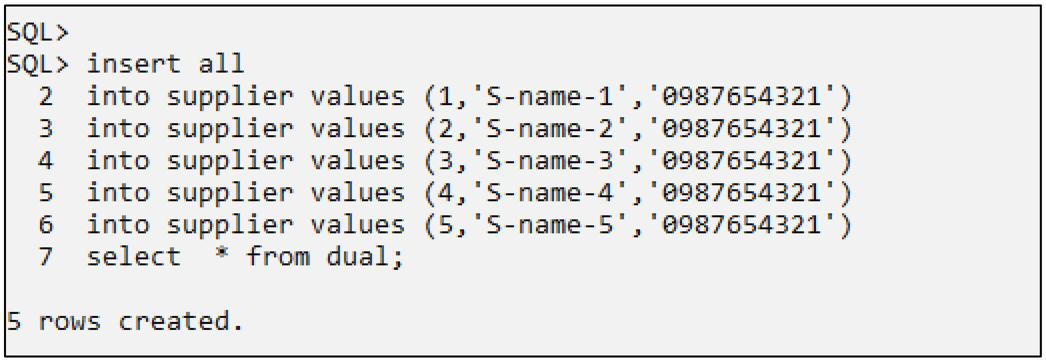
into supplier values (3,'S-name-3','0987654321')

into supplier values (4,'S-name-4','0987654321')

into supplier values (5,'S-name-5','0987654321')

select * from dual;

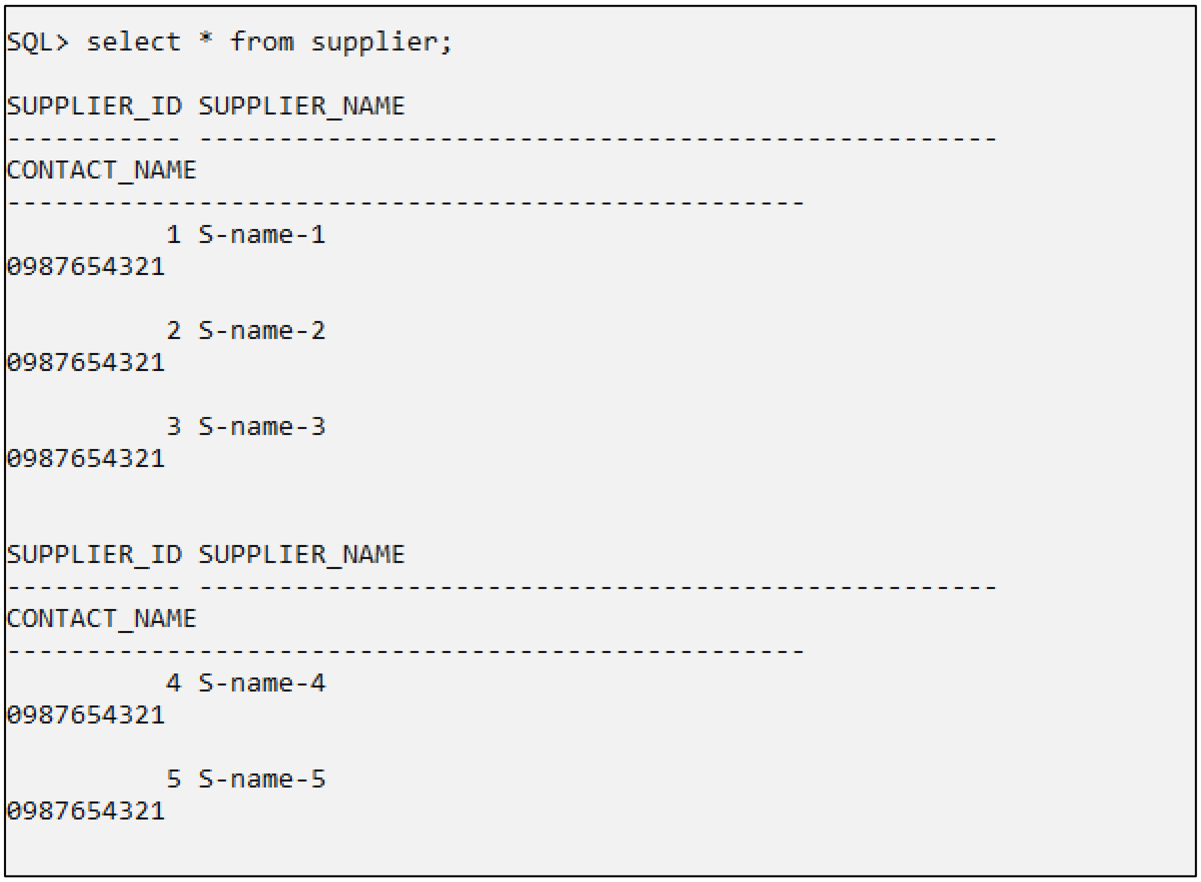
Output:



```
SQL>
SQL> insert all
  2  into supplier values (1,'S-name-1','0987654321')
  3  into supplier values (2,'S-name-2','0987654321')
  4  into supplier values (3,'S-name-3','0987654321')
  5  into supplier values (4,'S-name-4','0987654321')
  6  into supplier values (5,'S-name-5','0987654321')
  7  select  * from dual;

5 rows created.
```

Syntax – select * from table_name;
Code select * from supplier;

Output:



```
SQL> select * from supplier;

SUPPLIER_ID SUPPLIER_NAME
----------- --------------------------------------------------
CONTACT_NAME
--------------------------------------------------
          1 S-name-1
0987654321

          2 S-name-2
0987654321

          3 S-name-3
0987654321


SUPPLIER_ID SUPPLIER_NAME
----------- --------------------------------------------------
CONTACT_NAME
--------------------------------------------------
          4 S-name-4
0987654321

          5 S-name-5
0987654321
```

Query : Insert records into netflix table

Syntax - INSERT ALL
            INTO mytable (column1, column2, column_n) values (expr1, expr2,…exprN)
      SELECT * FROM dual;

Code into netflix values ('101','Star','Web show','19/10/2021','comedy','director',6,6,255,'3.4',499,'Yes','english','Italy')

into netflix values ('102','Star Wars','Web show','19/10/2021','Fiction','aaa',6,6,255,'3.4',499,'Yes','english','Italy')

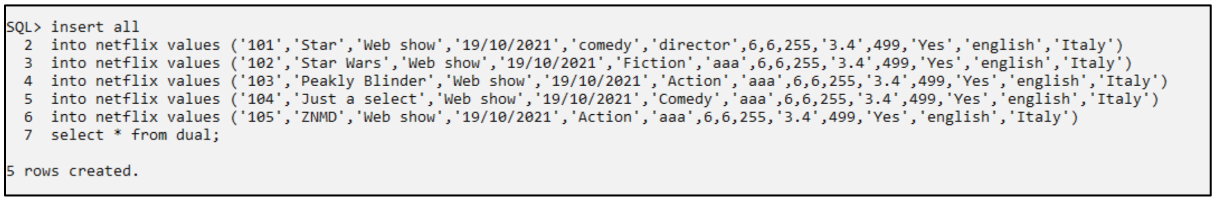
into netflix values ('103','Peakly Blinder','Web show','19/10/2021','Action','aaa',6,6,255,'3.4',499,'Yes','english','Italy')

into netflix values ('104','Just a select','Web show','19/10/2021','Comedy','aaa',6,6,255,'3.4',499,'Yes','english','Italy')

into netflix values ('105','ZNMD','Web show','19/10/2021','Action','aaa',6,6,255,'3.4',499,'Yes','english','Italy')

select * from dual;

Output:



```
SQL> insert all
into netflix values ('101','Star','Web show','19/10/2021','comedy','director',6,6,255,'3.4',499,'Yes','english','Italy')
into netflix values ('102','Star Wars','Web show','19/10/2021','Fiction','aaa',6,6,255,'3.4',499,'Yes','english','Italy')
into netflix values ('103','Peakly Blinder','Web show','19/10/2021','Action','aaa',6,6,255,'3.4',499,'Yes','english','Italy')
into netflix values ('104','Just a select','Web show','19/10/2021','Comedy','aaa',6,6,255,'3.4',499,'Yes','english','Italy')
into netflix values ('105','ZNMD','Web show','19/10/2021','Action','aaa',6,6,255,'3.4',499,'Yes','english','Italy')
select * from dual;

5 rows created.
```

b) Update : to modify or change the existing records in a table.

Syntax: Update table_name

Set column1 = val1,

column2 = val2,

column3 = val3

where [CONDITION]

Example:

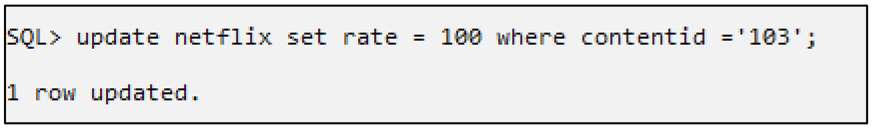
Update employee set salary = 10000 where id = 102;

Update employee set salary = salary + salary*0.2;

Query: update price when contentId is given

Code: update netflix set rate = 100 where contentid ='103';

Output:

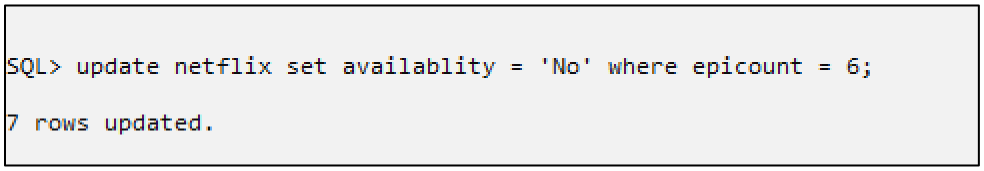


```
SQL> update netflix set rate = 100 where contentid ='103';

1 row updated.
```

Query: change the availiablity for content with episode count more than 30.

Code: update netflix set availablity = 'No' where epicount = 6;

Output:



```
SQL> update netflix set availablity = 'No' where epicount = 6;

7 rows updated.
```

c) Delete : The **DELETE statement** is used to **delete** existing records in a table.

<u>**Syntax:**</u>

Delete from table_name

Where[condition]

Query: Delete a customer whose id is 102

Code : Delete from customer where content_id = 102

Query: Delete all records from customers

Code : Delete from customers;

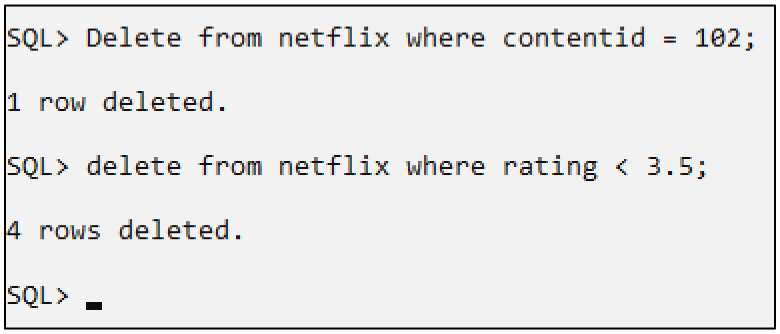
Query: Delete content details when content_id is provided

Code : Delete from netflix where contentid = 102;

Query: Delete the contents with less rating

Code : delete from netflix where rating < 3.5;

Output :



```
SQL> Delete from netflix where contentid = 102;

1 row deleted.

SQL> delete from netflix where rating < 3.5;

4 rows deleted.

SQL>
```

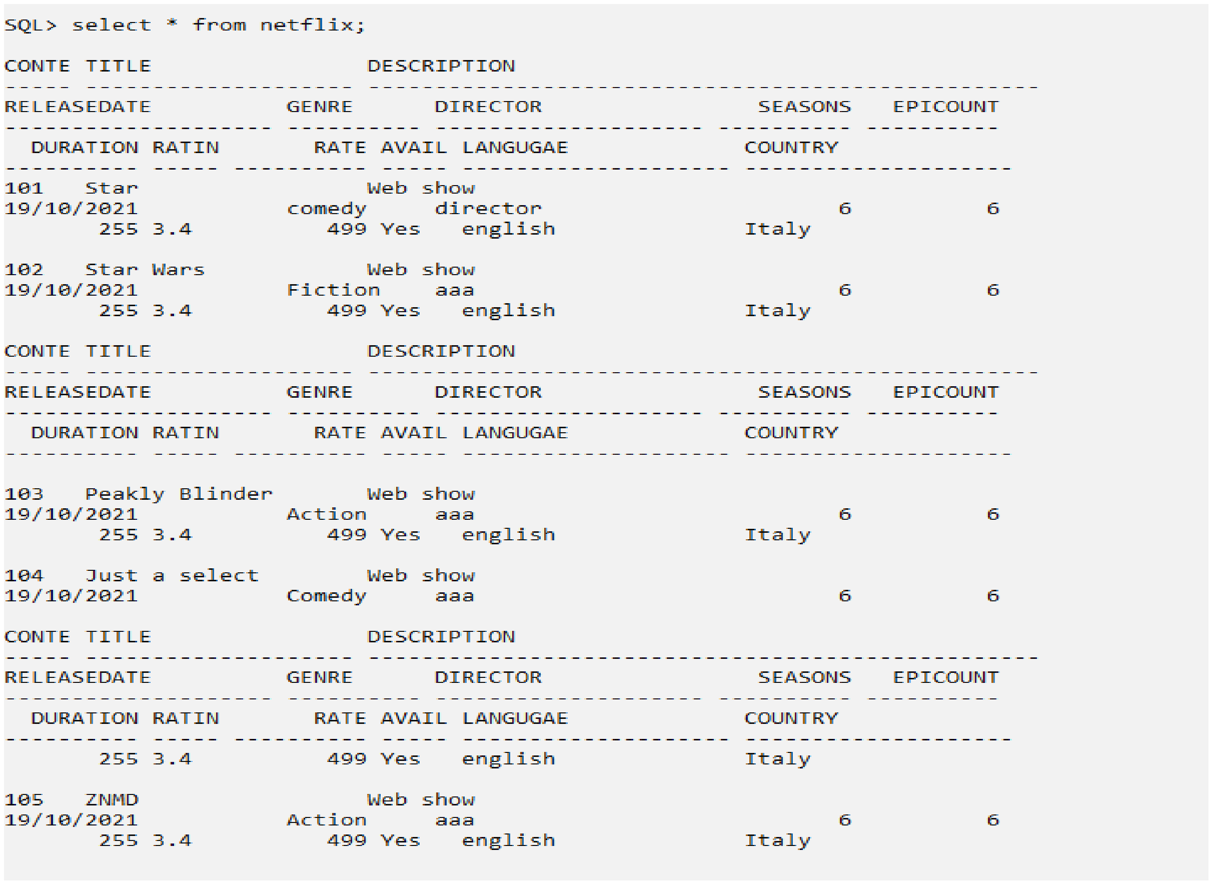
Practical 3 : Sql Select Statements

The **SQL SELECT Statement**. The SELECT statement is used to select data from a database.

a. Selecting all columns

Code select * from netflix;

Output:



```
SQL> select * from netflix;

CONTE TITLE                DESCRIPTION
----- -------------------- --------------------------------------------------
RELEASEDATE          GENRE      DIRECTOR                SEASONS   EPICOUNT
-------------------- ---------- -------------------- ---------- ----------
  DURATION RATIN       RATE AVAIL LANGUGAE             COUNTRY
---------- ----- ---------- ----- -------------------- --------------------
101   Star                 Web show
19/10/2021           comedy     director                      6          6
       255 3.4          499 Yes   english              Italy

102   Star Wars            Web show
19/10/2021           Fiction    aaa                           6          6
       255 3.4          499 Yes   english              Italy

CONTE TITLE                DESCRIPTION
----- -------------------- --------------------------------------------------
RELEASEDATE          GENRE      DIRECTOR                SEASONS   EPICOUNT
-------------------- ---------- -------------------- ---------- ----------
  DURATION RATIN       RATE AVAIL LANGUGAE             COUNTRY
---------- ----- ---------- ----- -------------------- --------------------

103   Peakly Blinder       Web show
19/10/2021           Action     aaa                           6          6
       255 3.4          499 Yes   english              Italy

104   Just a select        Web show
19/10/2021           Comedy     aaa                           6          6

CONTE TITLE                DESCRIPTION
----- -------------------- --------------------------------------------------
RELEASEDATE          GENRE      DIRECTOR                SEASONS   EPICOUNT
-------------------- ---------- -------------------- ---------- ----------
  DURATION RATIN       RATE AVAIL LANGUGAE             COUNTRY
---------- ----- ---------- ----- -------------------- --------------------
       255 3.4          499 Yes   english              Italy

105   ZNMD                 Web show
19/10/2021           Action     aaa                           6          6
       255 3.4          499 Yes   english              Italy
```

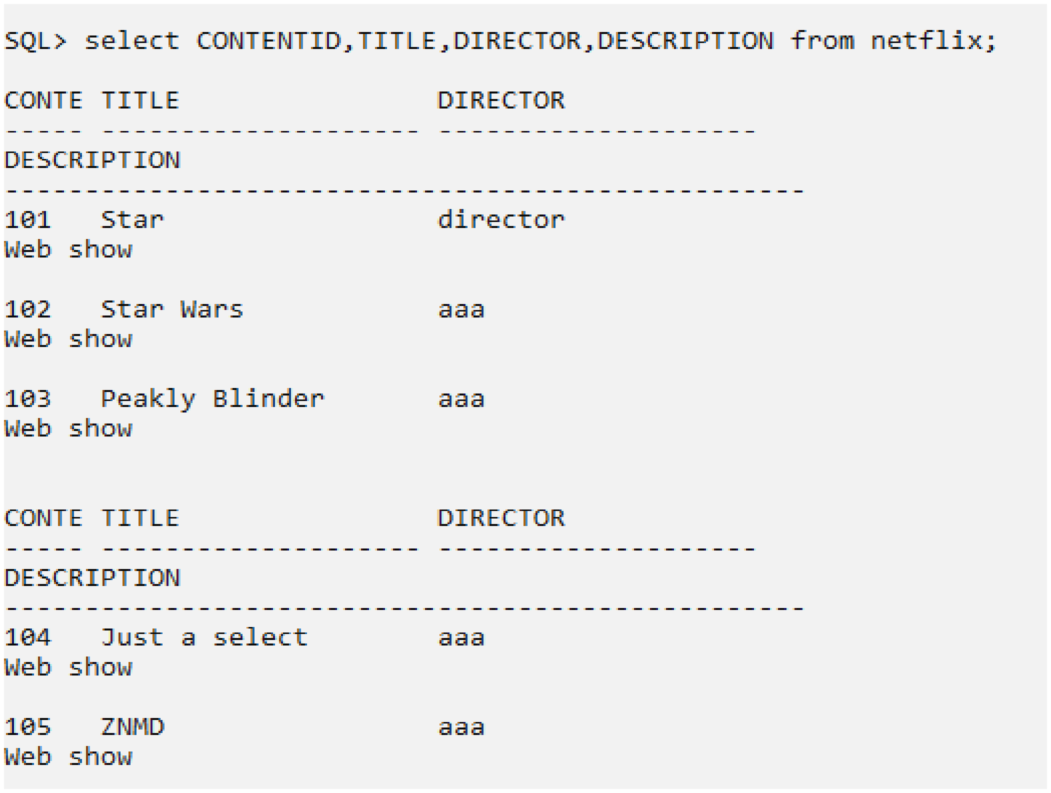
b. Selecting Specific Columns

Query : Fetch content id, title, director and description from ott

Syntax : select colum_name,col_name1 from table_name;

Code: select CONTENTID,TITLE,DIRECTOR,DESCRIPTION from netflix;

Output:

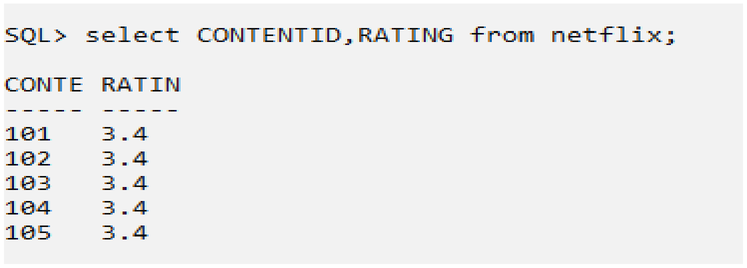

```
SQL> select CONTENTID,TITLE,DIRECTOR,DESCRIPTION from netflix;

CONTE TITLE                DIRECTOR
----- -------------------- --------------------
DESCRIPTION
--------------------------------------------------
101   Star                 director
Web show

102   Star Wars            aaa
Web show

103   Peakly Blinder       aaa
Web show


CONTE TITLE                DIRECTOR
----- -------------------- --------------------
DESCRIPTION
--------------------------------------------------
104   Just a select        aaa
Web show

105   ZNMD                 aaa
Web show
```

Query : Fetch contented, rating

Syntax : select colum_name,col_name1 from table_name;

Code: select CONTENTID,RATING from netflix;

Output:

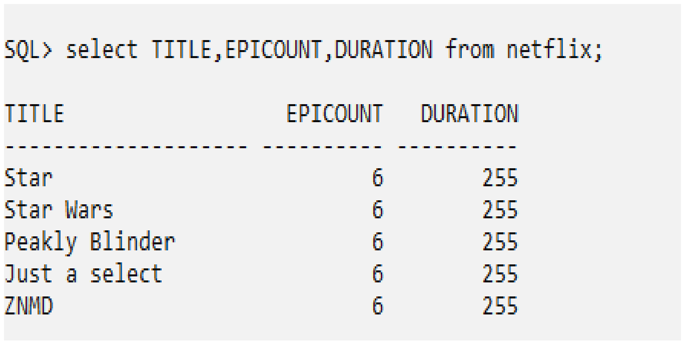

```
SQL> select CONTENTID,RATING from netflix;

CONTE RATIN
----- -----
101   3.4
102   3.4
103   3.4
104   3.4
105   3.4
```

Query : Get title, episode count and duration

Syntax : select colum_name,col_name1 from table_name;

Code: select TITLE,EPICOUNT,DURATION from netflix;

Output:

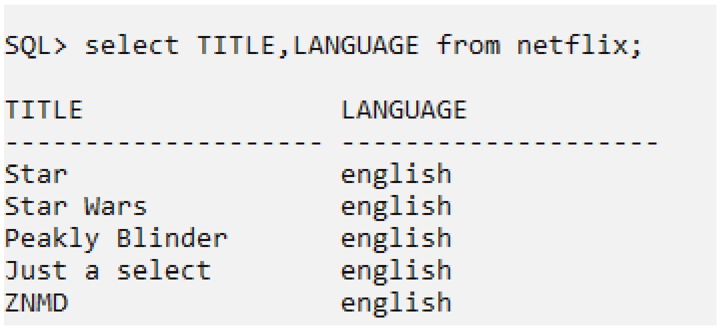

```
SQL> select TITLE,EPICOUNT,DURATION from netflix;

TITLE                  EPICOUNT   DURATION
------------------- ---------- ----------
Star                          6        255
Star Wars                     6        255
Peakly Blinder                6        255
Just a select                 6        255
ZNMD                          6        255
```

Query : Get title,language from Netflix

Syntax : select colum_name,col_name1 from table_name;

Code: select TITLE,LANGUAGE from netflix;

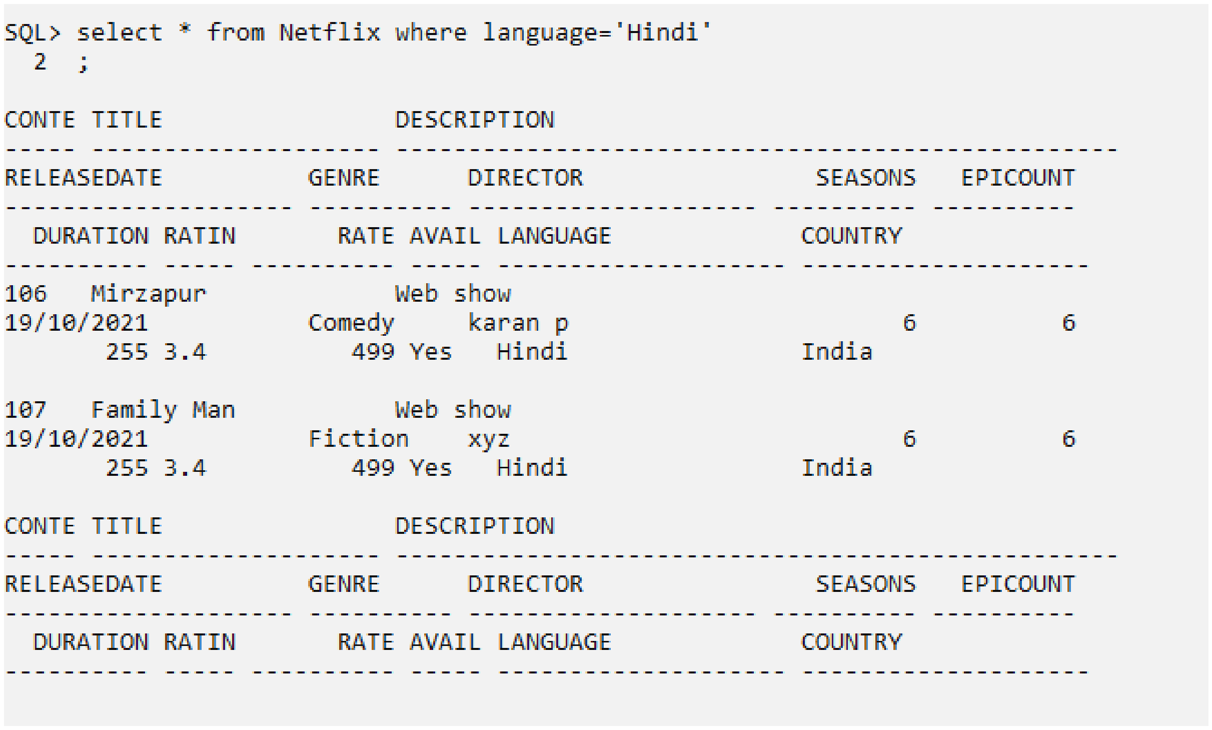
Output:


```
SQL> select TITLE,LANGUAGE from netflix;

TITLE                LANGUAGE
-------------------- --------------------
Star                 english
Star Wars            english
Peakly Blinder       english
Just a select        english
ZNMD                 english
```

Query : Get hindi series details:

Code: select * from Netflix where language='Hindi';

Output:

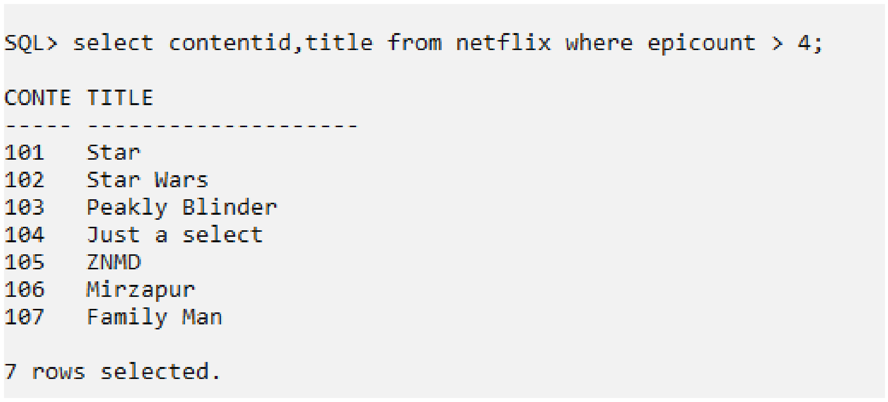


```
SQL> select * from Netflix where language='Hindi'
  2  ;

CONTE TITLE                DESCRIPTION
----- -------------------- --------------------------------------------------
RELEASEDATE          GENRE      DIRECTOR                SEASONS    EPICOUNT
-------------------- ---------- -------------------- ---------- ----------
  DURATION RATIN       RATE AVAIL LANGUAGE             COUNTRY
---------- ----- ---------- ----- -------------------- --------------------
106   Mirzapur             Web show
19/10/2021           Comedy     karan p                       6          6
       255 3.4          499 Yes   Hindi                India

107   Family Man           Web show
19/10/2021           Fiction    xyz                           6          6
       255 3.4          499 Yes   Hindi                India

CONTE TITLE                DESCRIPTION
----- -------------------- --------------------------------------------------
RELEASEDATE          GENRE      DIRECTOR                SEASONS    EPICOUNT
-------------------- ---------- -------------------- ---------- ----------
  DURATION RATIN       RATE AVAIL LANGUAGE             COUNTRY
---------- ----- ---------- ----- -------------------- --------------------
```

Query : Get content id and title of series with more than 4 episodes:

Code: select contentid,title from netflix where epicount > 4;

Output:

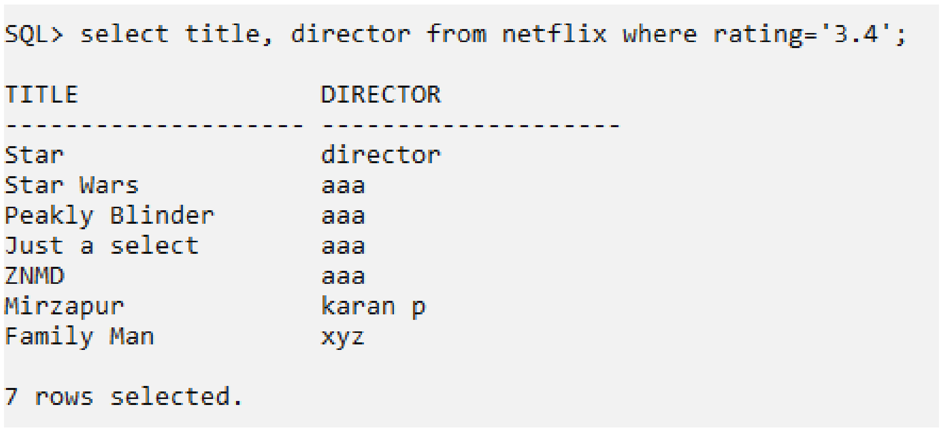


```
SQL> select contentid,title from netflix where epicount > 4;

CONTE TITLE
----- --------------------
101   Star
102   Star Wars
103   Peakly Blinder
104   Just a select
105   ZNMD
106   Mirzapur
107   Family Man

7 rows selected.
```

Query : Get the title,director and description of series with ‘3.4’.

Code: select title, director from netflix where rating='3.4';

Output:



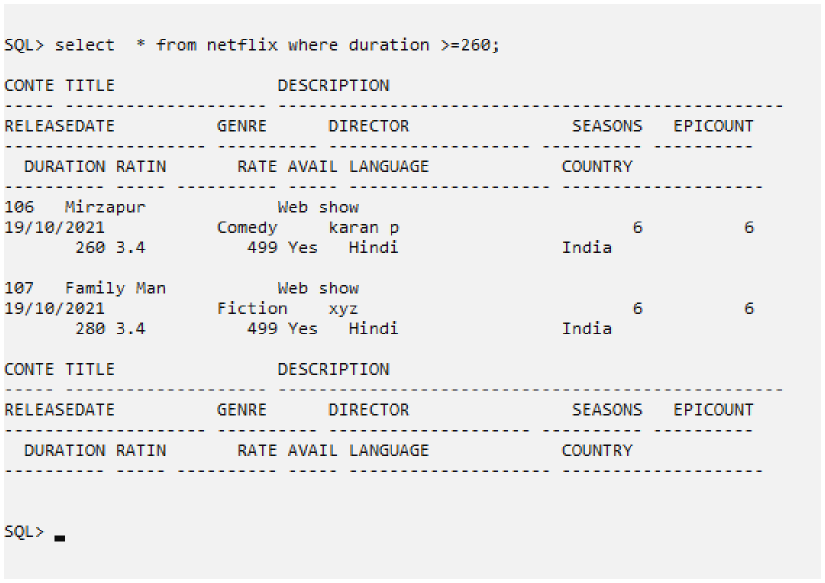
```
SQL> select title, director from netflix where rating='3.4';

TITLE                DIRECTOR
-------------------- --------------------
Star                 director
Star Wars            aaa
Peakly Blinder       aaa
Just a select        aaa
ZNMD                 aaa
Mirzapur             karan p
Family Man           xyz

7 rows selected.
```

Query : get details of series with duration more than 260.

Syntax : select colum_name,col_name1 from table_name;

Code: select * from netflix where duration >=260;

Output:

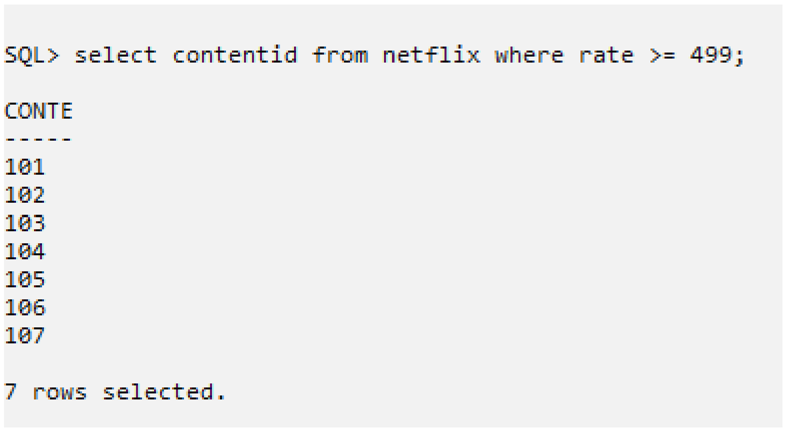


```
SQL> select  * from netflix where duration >=260;

CONTE TITLE                DESCRIPTION
----- -------------------- --------------------------------------------------
RELEASEDATE          GENRE      DIRECTOR                SEASONS   EPICOUNT
-------------------- ---------- -------------------- ---------- ----------
  DURATION RATIN       RATE AVAIL LANGUAGE             COUNTRY
---------- ----- ---------- ----- -------------------- --------------------
106   Mirzapur             Web show
19/10/2021           Comedy     karan p                       6          6
       260 3.4          499 Yes   Hindi                India

107   Family Man           Web show
19/10/2021           Fiction    xyz                           6          6
       280 3.4          499 Yes   Hindi                India

CONTE TITLE                DESCRIPTION
----- -------------------- --------------------------------------------------
RELEASEDATE          GENRE      DIRECTOR                SEASONS   EPICOUNT
-------------------- ---------- -------------------- ---------- ----------
  DURATION RATIN       RATE AVAIL LANGUAGE             COUNTRY
---------- ----- ---------- ----- -------------------- --------------------


SQL>
```

Query : Fetch content id of series with price/rate more than 100

Code: select contentid from netflix where rate >= 499;

Output:

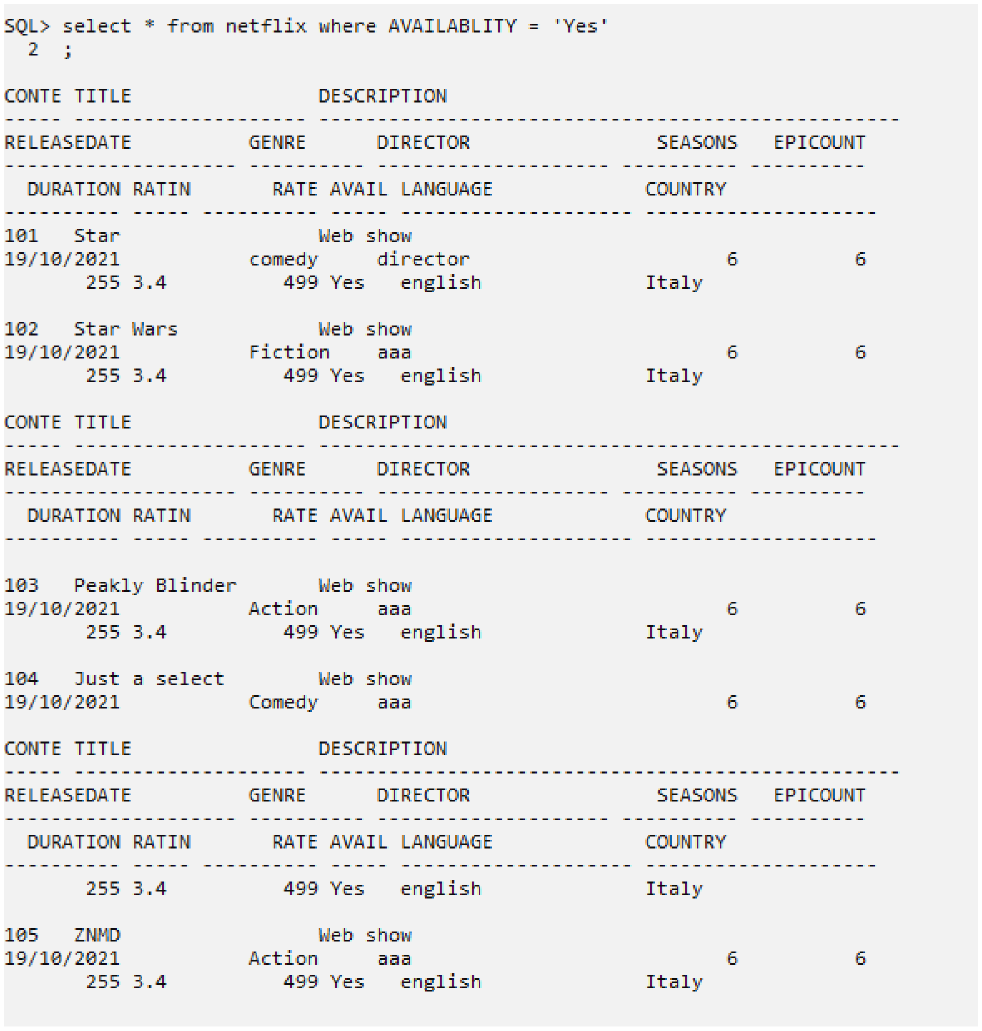


```
SQL> select contentid from netflix where rate >= 499;

CONTE
-----
101
102
103
104
105
106
107

7 rows selected.
```

Query : Fetch the available series details.

Code: select * from netflix where AVAILABLITY = 'Yes';

Output:



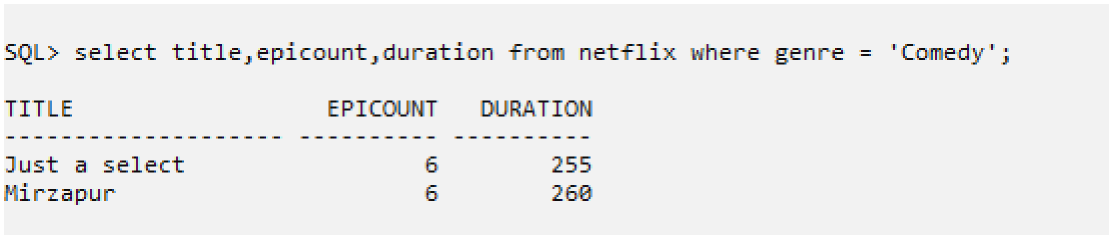
```
SQL> select * from netflix where AVAILABLITY = 'Yes'
  2  ;

CONTE TITLE                DESCRIPTION
----- -------------------- --------------------------------------------------
RELEASEDATE          GENRE      DIRECTOR                SEASONS   EPICOUNT
-------------------- ---------- -------------------- ---------- ----------
  DURATION RATIN       RATE AVAIL LANGUAGE             COUNTRY
---------- ----- ---------- ----- -------------------- --------------------
101   Star                 Web show
19/10/2021           comedy     director                      6          6
       255 3.4          499 Yes   english              Italy

102   Star Wars            Web show
19/10/2021           Fiction    aaa                           6          6
       255 3.4          499 Yes   english              Italy

CONTE TITLE                DESCRIPTION
----- -------------------- --------------------------------------------------
RELEASEDATE          GENRE      DIRECTOR                SEASONS   EPICOUNT
-------------------- ---------- -------------------- ---------- ----------
  DURATION RATIN       RATE AVAIL LANGUAGE             COUNTRY
---------- ----- ---------- ----- -------------------- --------------------

103   Peakly Blinder       Web show
19/10/2021           Action     aaa                           6          6
       255 3.4          499 Yes   english              Italy

104   Just a select        Web show
19/10/2021           Comedy     aaa                           6          6

CONTE TITLE                DESCRIPTION
----- -------------------- --------------------------------------------------
RELEASEDATE          GENRE      DIRECTOR                SEASONS   EPICOUNT
-------------------- ---------- -------------------- ---------- ----------
  DURATION RATIN       RATE AVAIL LANGUAGE             COUNTRY
---------- ----- ---------- ----- -------------------- --------------------
       255 3.4          499 Yes   english              Italy

105   ZNMD                 Web show
19/10/2021           Action     aaa                           6          6
       255 3.4          499 Yes   english              Italy
```

Query : Get the title,episode count and duration of comedy series

Syntax : select colum_name,col_name1 from table_name;

Code:select title,epicount,duration from netflix where genre = 'Comedy';

Output:



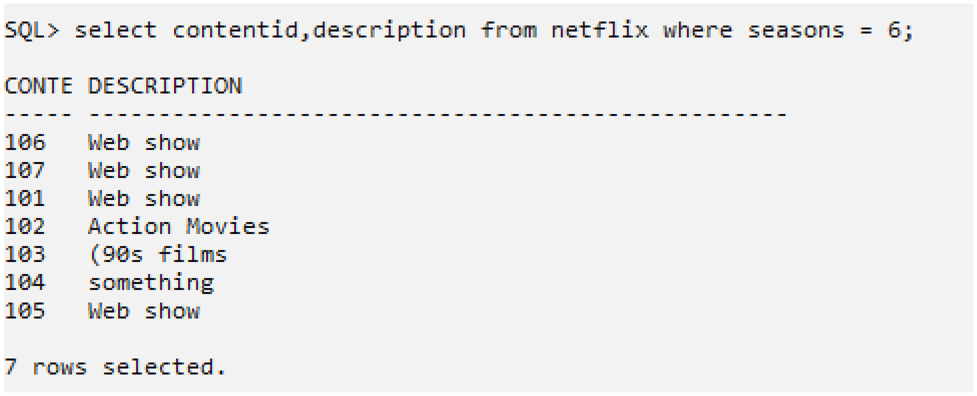
```
SQL> select title,epicount,duration from netflix where genre = 'Comedy';

TITLE                  EPICOUNT   DURATION
-------------------- ---------- ----------
Just a select                 6        255
Mirzapur                      6        260
```

Query : Get the content id, description of series with 5 seasons.

Syntax : select colum_name,col_name1 from table_name;

Code:select contentid,description from netflix where seasons = 6;

Output:



```
SQL> select contentid,description from netflix where seasons = 6;

CONTE DESCRIPTION
----- --------------------------------------------------
106   Web show
107   Web show
101   Web show
102   Action Movies
103   (90s films
104   something
105   Web show

7 rows selected.
```

c. Concatenation Operator
d. Logical Conditions : logical operators are used to test for the truth of the condition. A logical operator like the Comparison operator returns a boolean value of TRUE, FALSE, or UNKNOWN (AND and OR operator)

Syntax:
Select col1 , col2, colN from table_name

Where [condition] and [condition]..and [conditionN]

Code:
Select * from suppliers where (state = “MAHARASHTRA” and supplier_name=”SAHARA”)

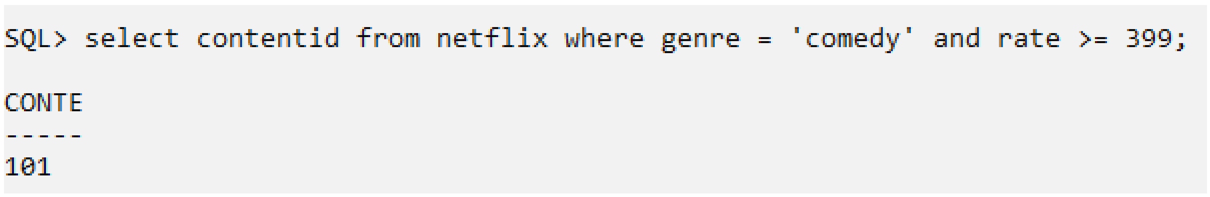
OR (supplier_id < 5000);

QUERY: Retrieve the id and title of the series in comedy genre with price more than 399

Syntax : select colum_name,col_name1 from table_name where {condition};

Code : select contentid from Netflix where genre = ‘comedy’ and rate >=399;

Output:



```
SQL> select contentid from netflix where genre = 'comedy' and rate >= 399;

CONTE
-----
101
```

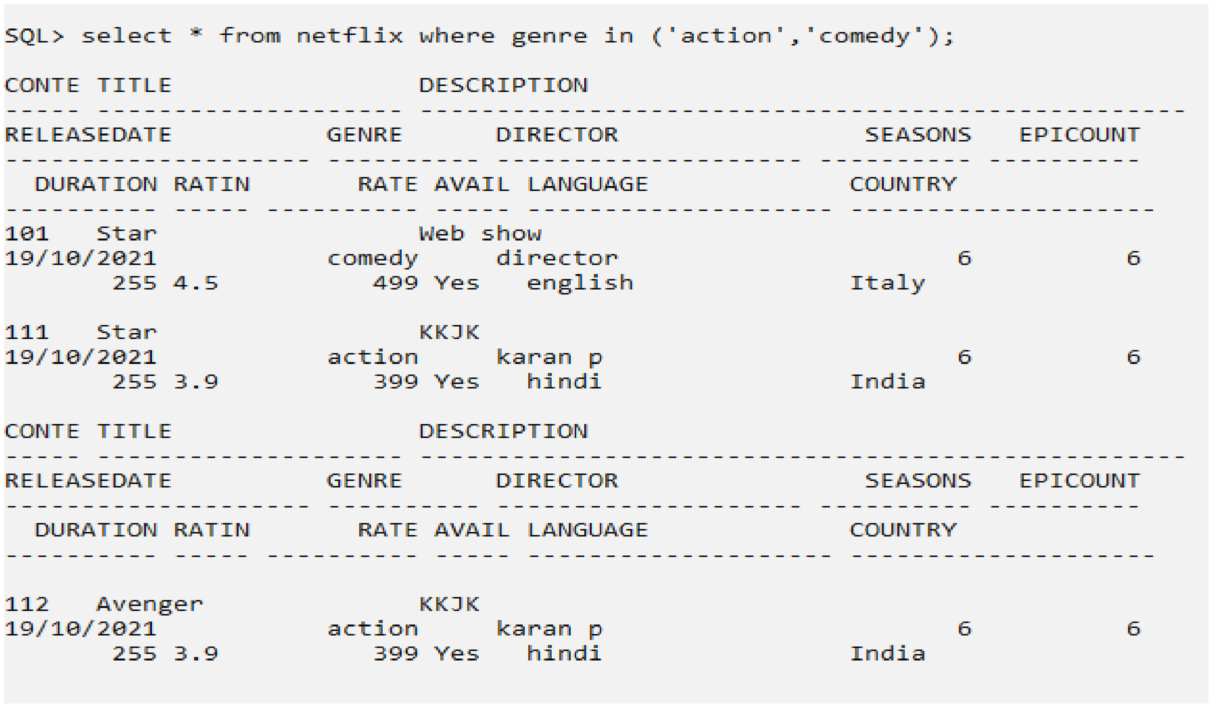
Query: Get the details of series of thriller or horror genre;

Syntax : select colum_name,col_name1 from table_name where {condition};

Code :select * from netflix where genre = 'action' or genre = 'comedy';

select * from netflix where genre in ('action','comedy');

Output:

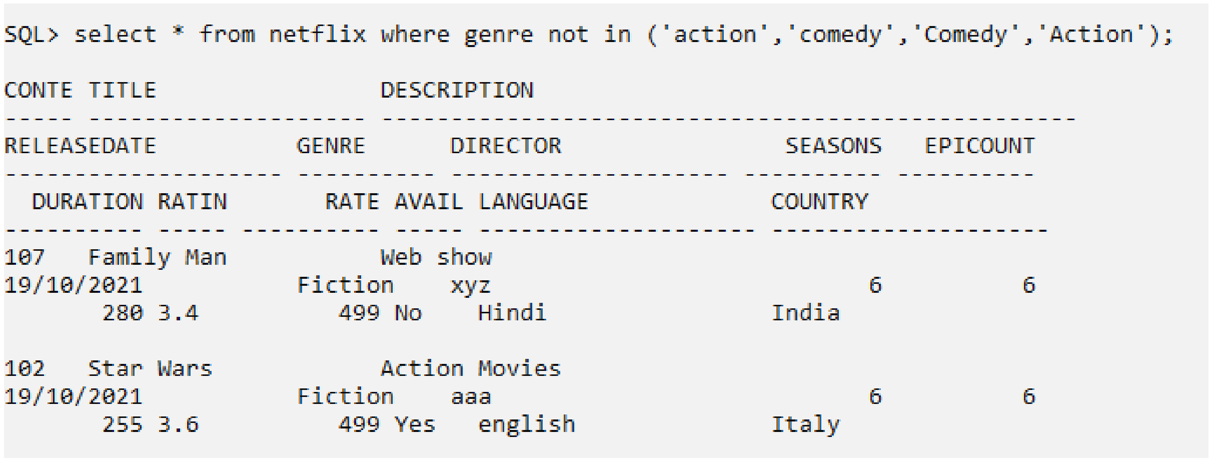


```
SQL> select * from netflix where genre in ('action','comedy');

CONTE TITLE                DESCRIPTION
----- -------------------- --------------------------------------------------
RELEASEDATE          GENRE      DIRECTOR                SEASONS   EPICOUNT
-------------------- ---------- -------------------- ---------- ----------
  DURATION RATIN       RATE AVAIL LANGUAGE             COUNTRY
---------- ----- ---------- ----- -------------------- --------------------
101   Star                 Web show
19/10/2021           comedy     director                      6          6
       255 4.5          499 Yes   english              Italy

111   Star                 KKJK
19/10/2021           action     karan p                       6          6
       255 3.9          399 Yes   hindi                India

CONTE TITLE                DESCRIPTION
----- -------------------- --------------------------------------------------
RELEASEDATE          GENRE      DIRECTOR                SEASONS   EPICOUNT
-------------------- ---------- -------------------- ---------- ----------
  DURATION RATIN       RATE AVAIL LANGUAGE             COUNTRY
---------- ----- ---------- ----- -------------------- --------------------

112   Avenger              KKJK
19/10/2021           action     karan p                       6          6
       255 3.9          399 Yes   hindi                India
```

Query: get details of series other than action genre;

Code : select * from netflix where genre not in ('action','comedy','Comedy','Action');

Output:



```
SQL> select * from netflix where genre not in ('action','comedy','Comedy','Action');

CONTE TITLE                DESCRIPTION
----- -------------------- --------------------------------------------------
RELEASEDATE          GENRE      DIRECTOR                SEASONS   EPICOUNT
-------------------- ---------- -------------------- ---------- ----------
  DURATION RATIN       RATE AVAIL LANGUAGE             COUNTRY
---------- ----- ---------- ----- -------------------- --------------------
107   Family Man           Web show
19/10/2021           Fiction    xyz                           6          6
       280 3.4          499 No    Hindi                India

102   Star Wars            Action Movies
19/10/2021           Fiction    aaa                           6          6
       255 3.6          499 Yes   english              Italy
```

DISTINCT

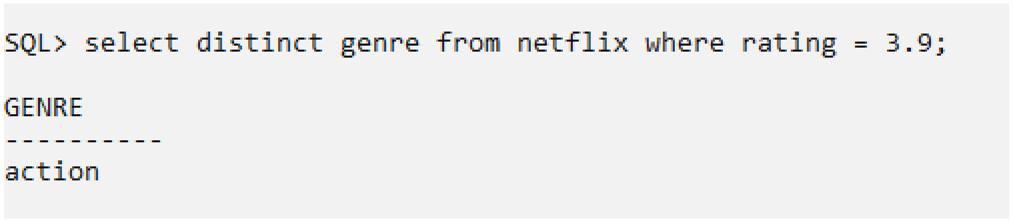
Syntax :Select distinct expressions from tables [where conditions];

Select distinct state from customers where last_name = “Panchal”;

Query: Get the genre of series with rating = 3.9

Code : select distinct genre from Netflix where rating = 3.9;

Output:



```
SQL> select distinct genre from netflix where rating = 3.9;

GENRE
----------
action
```

BETWEEN/ NOT BETWEEN

Syntax

Expression BETWEEN value1 and value2;

Expression NOT BETWEEN value1 and value2;

Example:

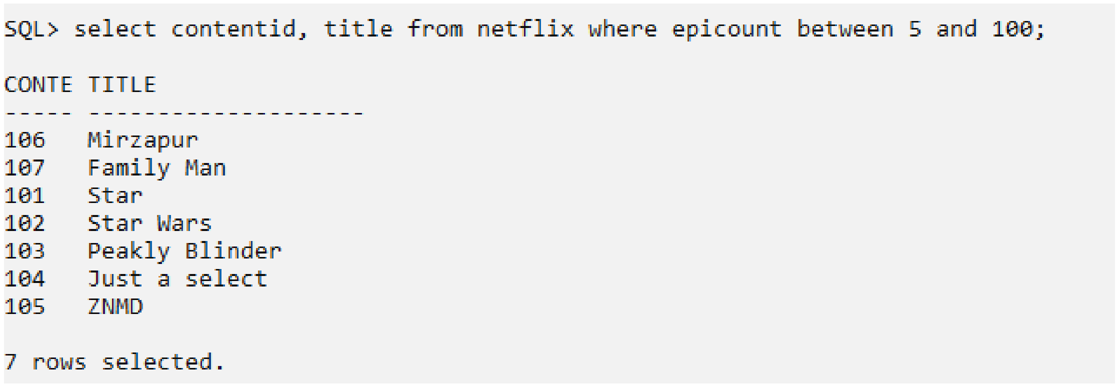
select * from customers where customer_id between 4000 and 4999;

select * from customers where customer_id not between 4000 and 4999;

QUERY: Get the content ID and title of series with episode count more than 5 and less than 100.

Code :select contentid, title from netflix where epicount between 5 and 100;

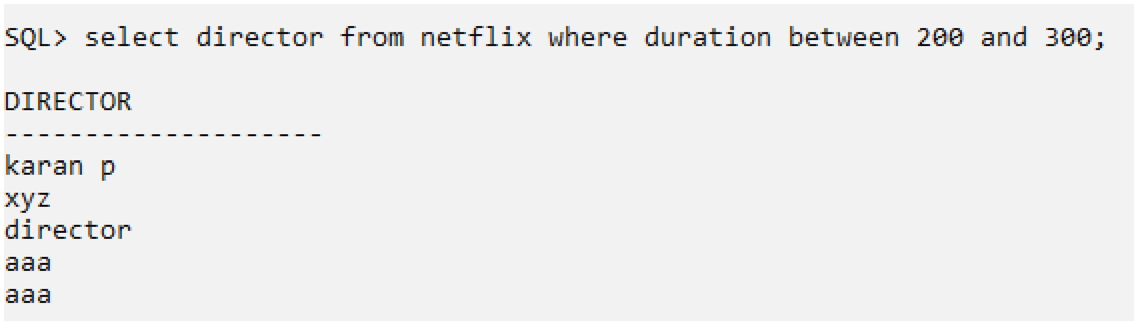
Output:



```
SQL> select contentid, title from netflix where epicount between 5 and 100;

CONTE TITLE
----- --------------------
106   Mirzapur
107   Family Man
101   Star
102   Star Wars
103   Peakly Blinder
104   Just a select
105   ZNMD

7 rows selected.
```

QUERY: Get the directors of series with duration between 200 to 300;

Code : select director from Netflix where duration between 200 and 300;

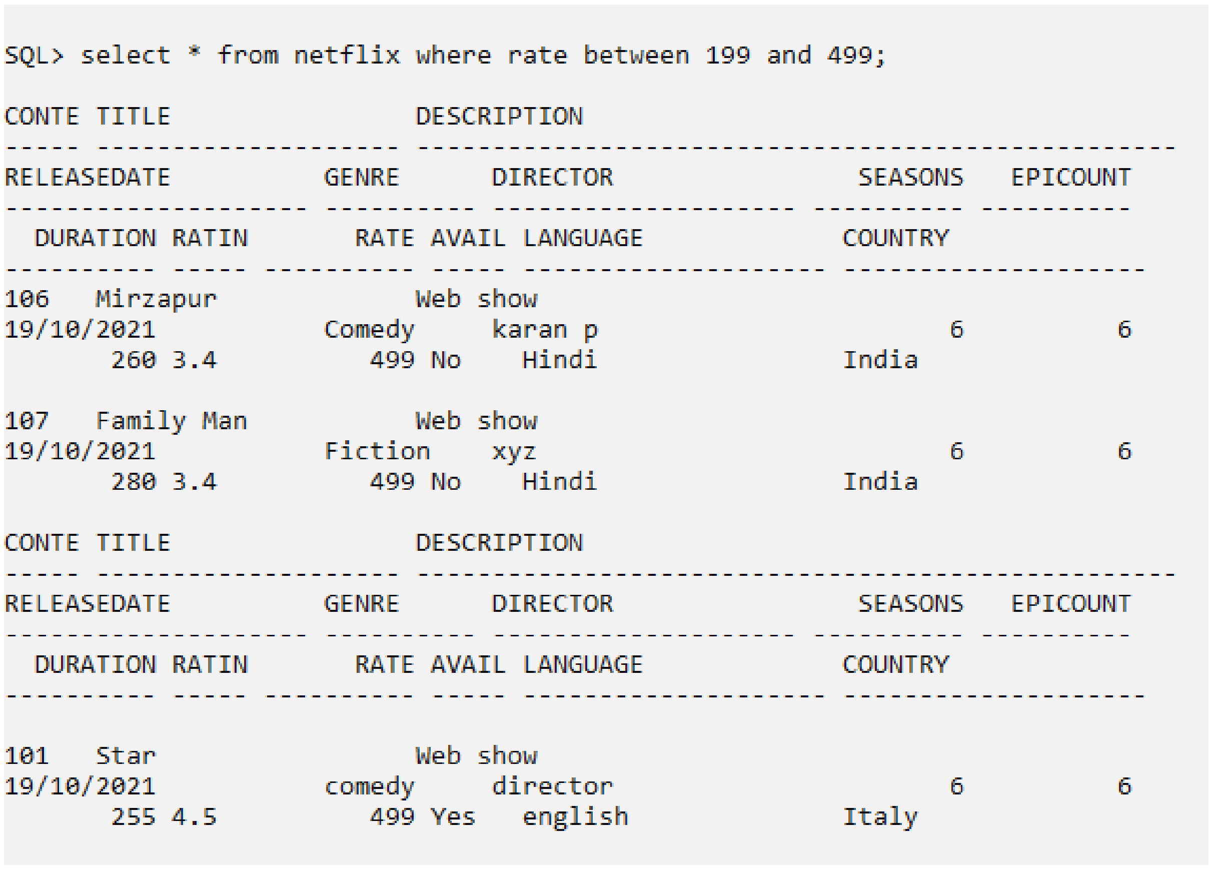


```
SQL> select director from netflix where duration between 200 and 300;

DIRECTOR
--------------------
karan p
xyz
director
aaa
aaa
```

QUERY: Get the series with price ranges in 199 to 499;

Code : select * from Netflix where rate between 199 and 499;

Output:

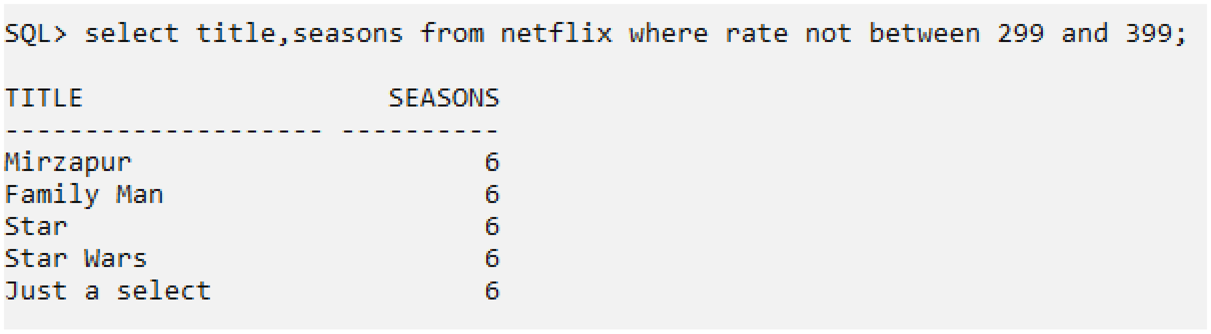


```
SQL> select * from netflix where rate between 199 and 499;

CONTE TITLE                DESCRIPTION
----- -------------------- --------------------------------------------------
RELEASEDATE          GENRE      DIRECTOR             SEASONS    EPICOUNT
-------------------- ---------- -------------------- ---------- ----------
  DURATION RATIN       RATE AVAIL LANGUAGE             COUNTRY
---------- ----- ---------- ----- -------------------- --------------------
106   Mirzapur             Web show
19/10/2021           Comedy     karan p                       6          6
       260 3.4          499 No    Hindi                India

107   Family Man           Web show
19/10/2021           Fiction    xyz                           6          6
       280 3.4          499 No    Hindi                India

CONTE TITLE                DESCRIPTION
----- -------------------- --------------------------------------------------
RELEASEDATE          GENRE      DIRECTOR             SEASONS    EPICOUNT
-------------------- ---------- -------------------- ---------- ----------
  DURATION RATIN       RATE AVAIL LANGUAGE             COUNTRY
---------- ----- ---------- ----- -------------------- --------------------

101   Star                 Web show
19/10/2021           comedy     director                      6          6
       255 4.5          499 Yes   english              Italy
```

QUERY: get the title and sesaons where rate not in 299 to 399

Code : select title,seasons from Netflix where rate not between 299 and 399;

Output:

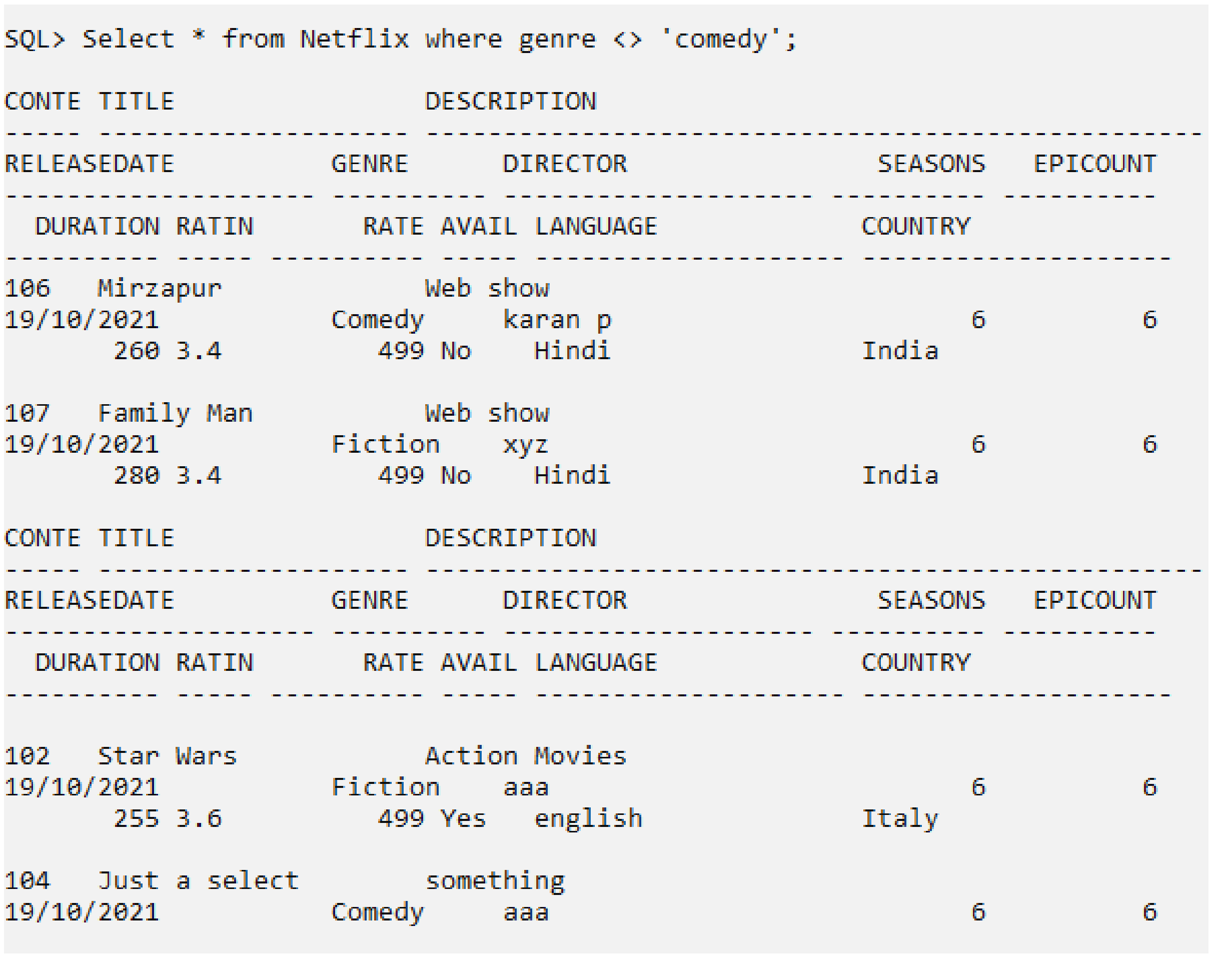


```
SQL> select title,seasons from netflix where rate not between 299 and 399;

TITLE                   SEASONS
-------------------- ----------
Mirzapur                      6
Family Man                    6
Star                          6
Star Wars                     6
Just a select                 6
```

QUERY: Get the details of series not in comedy genre

Code : Select * from Netflix where genre <> 'comedy';

Output:



```
SQL> Select * from Netflix where genre <> 'comedy';

CONTE TITLE                DESCRIPTION
----- -------------------- --------------------------------------------------
RELEASEDATE          GENRE      DIRECTOR                SEASONS    EPICOUNT
-------------------- ---------- -------------------- ---------- ----------
  DURATION RATIN       RATE AVAIL LANGUAGE             COUNTRY
---------- ----- ---------- ----- -------------------- --------------------
106   Mirzapur             Web show
19/10/2021           Comedy     karan p                       6          6
       260 3.4          499 No    Hindi                India

107   Family Man           Web show
19/10/2021           Fiction    xyz                           6          6
       280 3.4          499 No    Hindi                India

CONTE TITLE                DESCRIPTION
----- -------------------- --------------------------------------------------
RELEASEDATE          GENRE      DIRECTOR                SEASONS    EPICOUNT
-------------------- ---------- -------------------- ---------- ----------
  DURATION RATIN       RATE AVAIL LANGUAGE             COUNTRY
---------- ----- ---------- ----- -------------------- --------------------

102   Star Wars            Action Movies
19/10/2021           Fiction    aaa                           6          6
       255 3.6          499 Yes   english              Italy

104   Just a select        something
19/10/2021           Comedy     aaa                           6          6
```

LIKE : % or _:

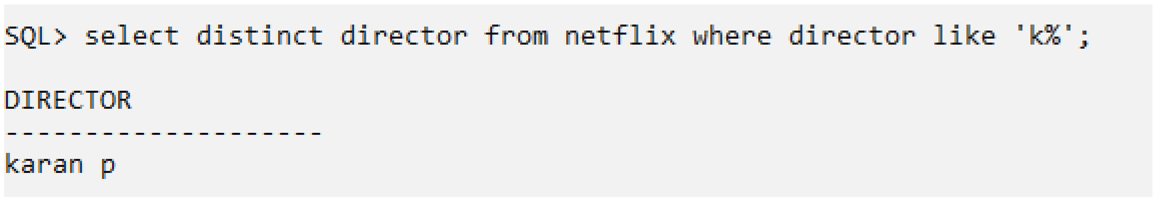
Syntax:

[expression] where col1 like ‘%value1%’

Query: get the name of directors whose name starts with ‘k’

select distinct director from netflix where director like 'k%';

Output:

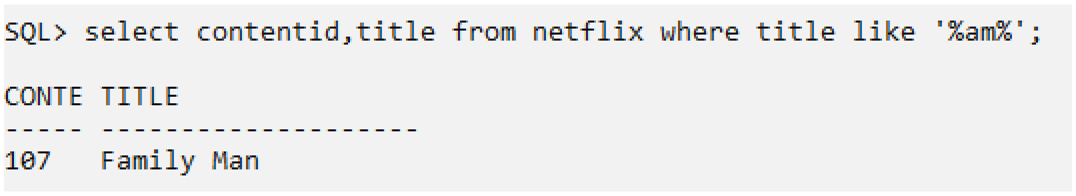


```
SQL> select distinct director from netflix where director like 'k%';

DIRECTOR
--------------------
karan p
```

Query: Get the series and title which contains ‘man’

Code : select contented ,title from Netflix where title like ‘%am%’;

Output:

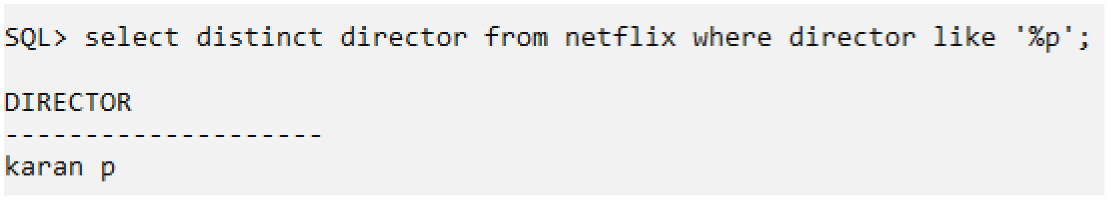


```
SQL> select contentid,title from netflix where title like '%am%';

CONTE TITLE
----- --------------------
107   Family Man
```

Query: get the directors name ending with p

Code : select distinct director from netflix where director like '%p';

Output:



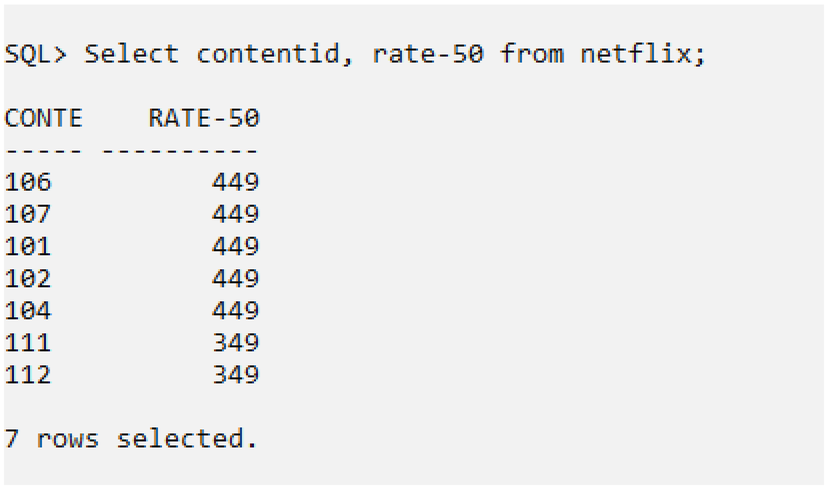
```
SQL> select distinct director from netflix where director like '%p';

DIRECTOR
--------------------
karan p
```

e. Arithmetic Operators

Query: Display the price after deducting 50 rupees

Select contentid, price-50 from netflix;

Output:

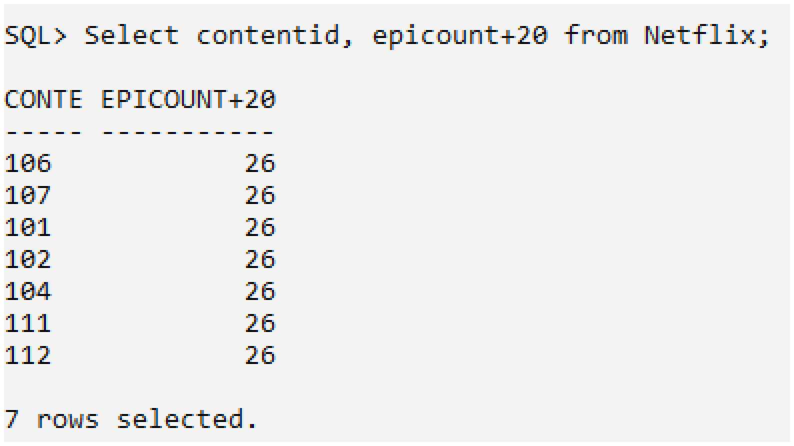


```
SQL> Select contentid, rate-50 from netflix;

CONTE    RATE-50
----- ----------
106          449
107          449
101          449
102          449
104          449
111          349
112          349

7 rows selected.
```

Query: Display the episode count increased by 20;

Select contentid, epicount+20 from Netflix;

Output:

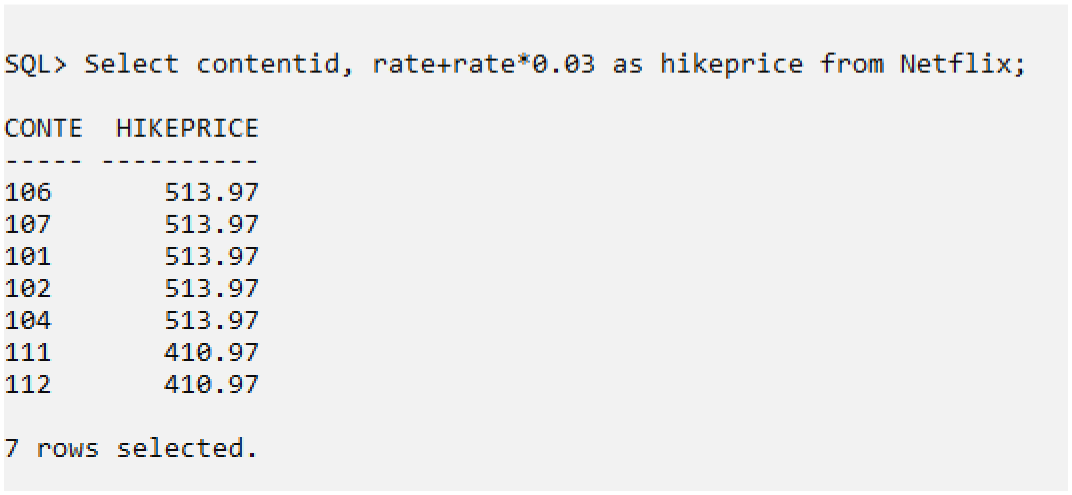


```
SQL> Select contentid, epicount+20 from Netflix;

CONTE EPICOUNT+20
----- -----------
106            26
107            26
101            26
102            26
104            26
111            26
112            26

7 rows selected.
```

Query: Display the price after increasing by 3%

Select contentid, rate+rate*0.03 as hikeprice from Netflix;

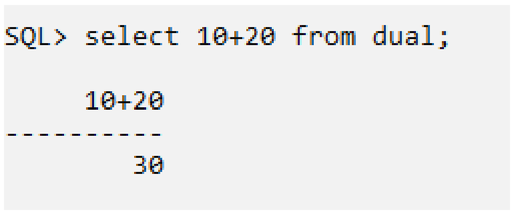
Output:



```
SQL> Select contentid, rate+rate*0.03 as hikeprice from Netflix;

CONTE  HIKEPRICE
----- ----------
106       513.97
107       513.97
101       513.97
102       513.97
104       513.97
111       410.97
112       410.97

7 rows selected.
```

DUAL Temp table in Oracle:

Output:



```
SQL> select 10+20 from dual;

     10+20
----------
        30
```

f. Comparison Conditions

<u>Equal (=)</u>

Syntax : Select * from customers where last_name = ‘Maria’;

<u>Not Equal (!= , <>)</u>

Syntax : Select * from customers where last_name <> ‘Maria’;

<u>Less than , Greater than</u>

Example:

Select * from customers where age > 60;

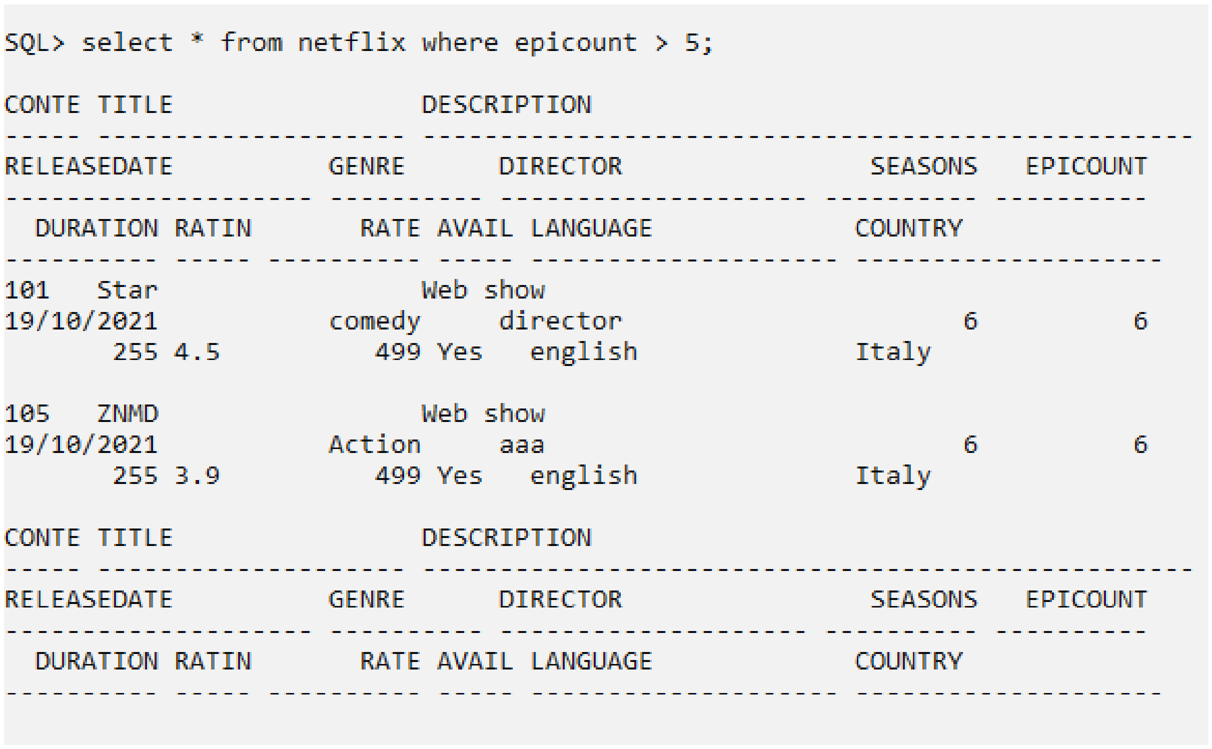
Select * from customers where age >= 18;

Select * from customers where age < 45;

Query: get the content details with number of episodes more than 5

Code: select * from netflix where epicount > 5;

Output:

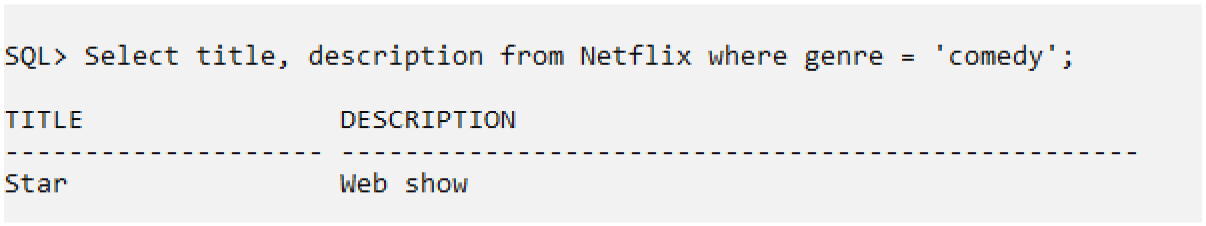


```
SQL> select * from netflix where epicount > 5;

CONTE TITLE                DESCRIPTION
----- -------------------- --------------------------------------------------
RELEASEDATE          GENRE      DIRECTOR                SEASONS   EPICOUNT
-------------------- ---------- -------------------- ---------- ----------
  DURATION RATIN       RATE AVAIL LANGUAGE             COUNTRY
---------- ----- ---------- ----- -------------------- --------------------
101   Star                 Web show
19/10/2021           comedy     director                      6          6
       255 4.5          499 Yes   english              Italy

105   ZNMD                 Web show
19/10/2021           Action     aaa                           6          6
       255 3.9          499 Yes   english              Italy

CONTE TITLE                DESCRIPTION
----- -------------------- --------------------------------------------------
RELEASEDATE          GENRE      DIRECTOR                SEASONS   EPICOUNT
-------------------- ---------- -------------------- ---------- ----------
  DURATION RATIN       RATE AVAIL LANGUAGE             COUNTRY
---------- ----- ---------- ----- -------------------- --------------------
```

Query: get the title and description of comedy series

Code: Select title, description from Netflix where genre = 'comedy';

Output:

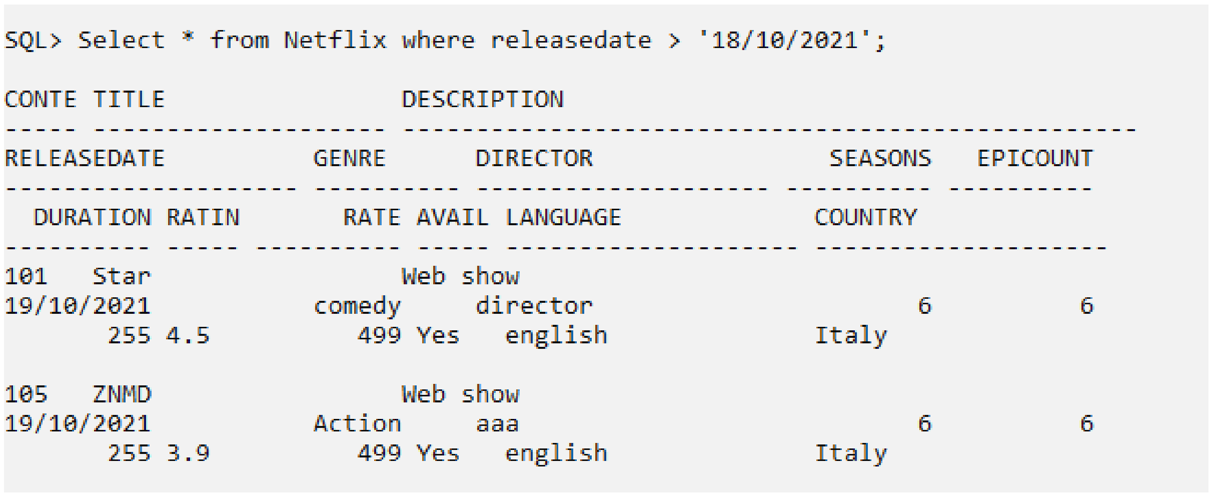


```
SQL> Select title, description from Netflix where genre = 'comedy';

TITLE                DESCRIPTION
-------------------- --------------------------------------------------
Star                 Web show
```

Query:get the serires details released before 19/10/21

Code: Select * from Netflix where releasedate > '18/10/2021';

Output:

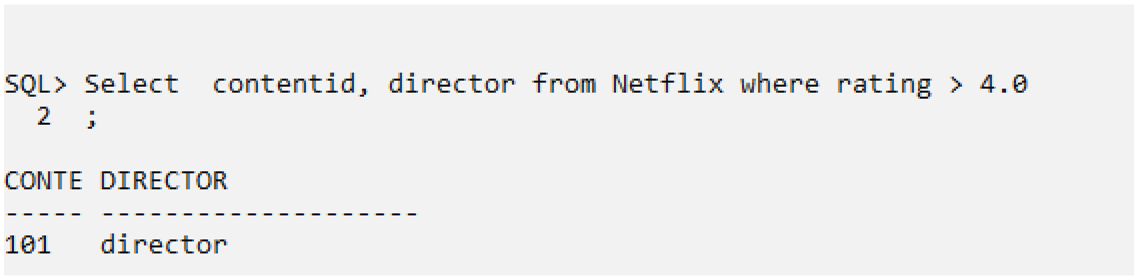


```
SQL> Select * from Netflix where releasedate > '18/10/2021';

CONTE TITLE                DESCRIPTION
----- -------------------- --------------------------------------------------
RELEASEDATE          GENRE      DIRECTOR                SEASONS   EPICOUNT
-------------------- ---------- -------------------- ---------- ----------
  DURATION RATIN      RATE AVAIL LANGUAGE             COUNTRY
---------- ----- ---------- ----- -------------------- --------------------
101   Star                 Web show
19/10/2021           comedy     director                      6          6
       255 4.5         499 Yes   english              Italy

105   ZNMD                 Web show
19/10/2021           Action     aaa                           6          6
       255 3.9         499 Yes   english              Italy
```

Query:get the id and director of series with rating 4.0

Code: Select contentid, director from Netflix where rating > 4.0;

Output:

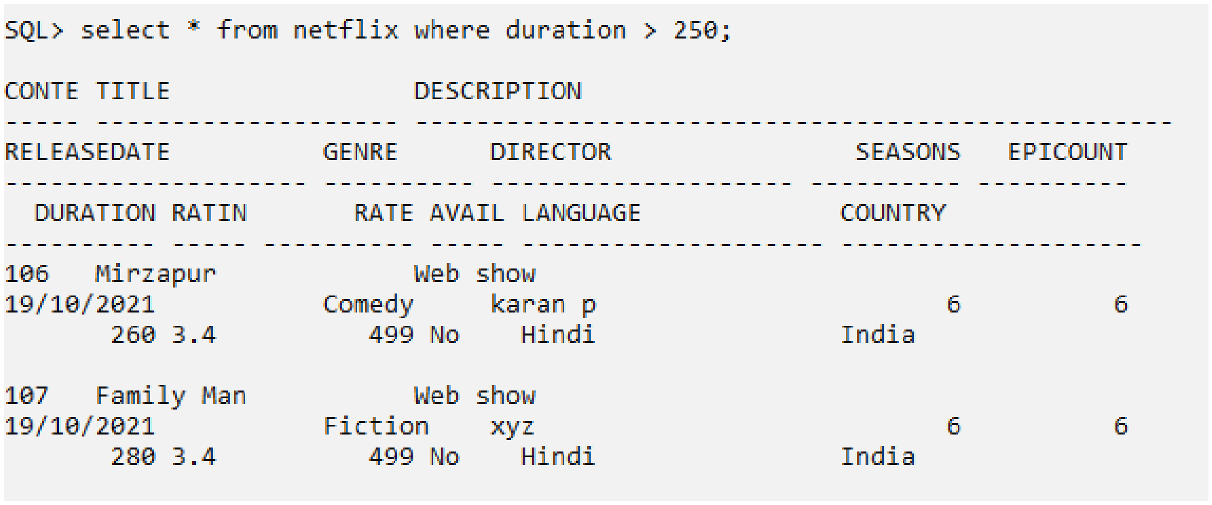


```
SQL> Select  contentid, director from Netflix where rating > 4.0
  2  ;

CONTE DIRECTOR
----- --------------------
101   director
```

Query: Get all the details where duration is less than 500

Code:select * from netflix where duration > 250;

Output:

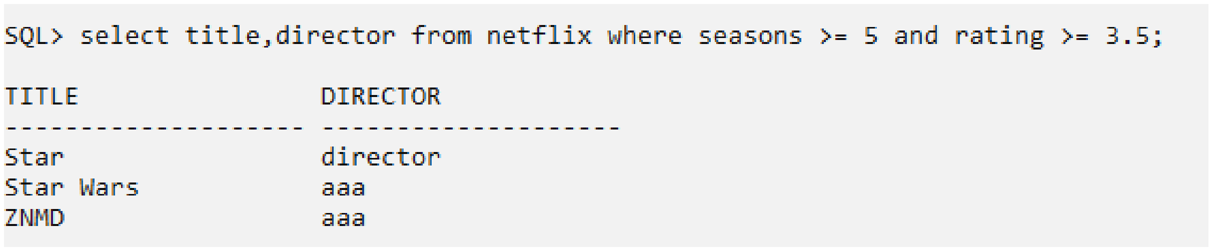


```
SQL> select * from netflix where duration > 250;

CONTE TITLE                DESCRIPTION
----- -------------------- --------------------------------------------------
RELEASEDATE          GENRE      DIRECTOR                SEASONS   EPICOUNT
-------------------- ---------- -------------------- ---------- ----------
  DURATION RATIN       RATE AVAIL LANGUAGE             COUNTRY
---------- ----- ---------- ----- -------------------- --------------------
106   Mirzapur             Web show
19/10/2021           Comedy     karan p                       6          6
       260 3.4          499 No    Hindi                India

107   Family Man           Web show
19/10/2021           Fiction    xyz                           6          6
       280 3.4          499 No    Hindi                India
```

Query: Get the title and director of contents with 3 seasons and rating > 4.0

Code:select title,director from netflix where seasons >= 5 and rating >= 3.5;

Output:

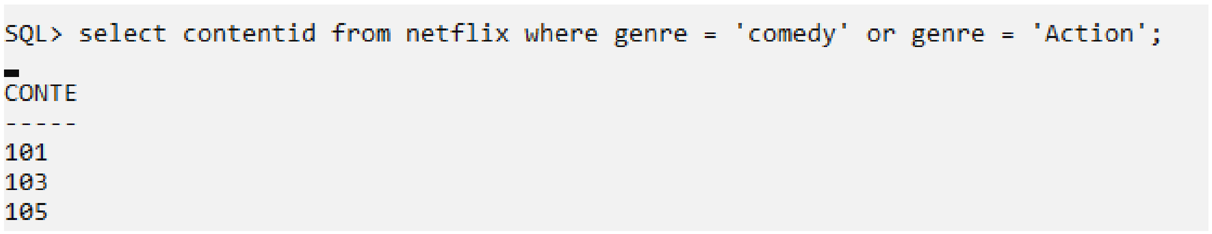


```
SQL> select title,director from netflix where seasons >= 5 and rating >= 3.5;

TITLE                DIRECTOR
-------------------- --------------------
Star                 director
Star Wars            aaa
ZNMD                 aaa
```

Query: Get the contentids of comedy or action series.

Code: select contentid from netflix where genre = 'comedy' or genre = 'Action';

Output:



```
SQL> select contentid from netflix where genre = 'comedy' or genre = 'Action';

CONTE
-----
101
103
105
```

g. Order by

Default ascending order

Order By clause is used to sort the data in ascending or descending order

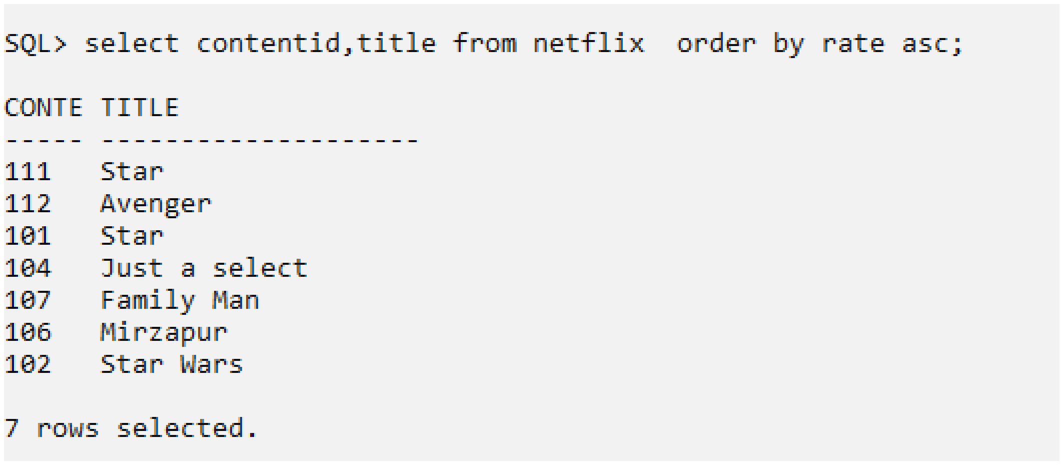
Select column-list

From table_name [where condition] ORDER BY [col1, col2,…] [asc|desc]

Query: Fetch the contented and title of series in ascending order of price

Code: select contentid,title from netflix order by rate asc;

Output:

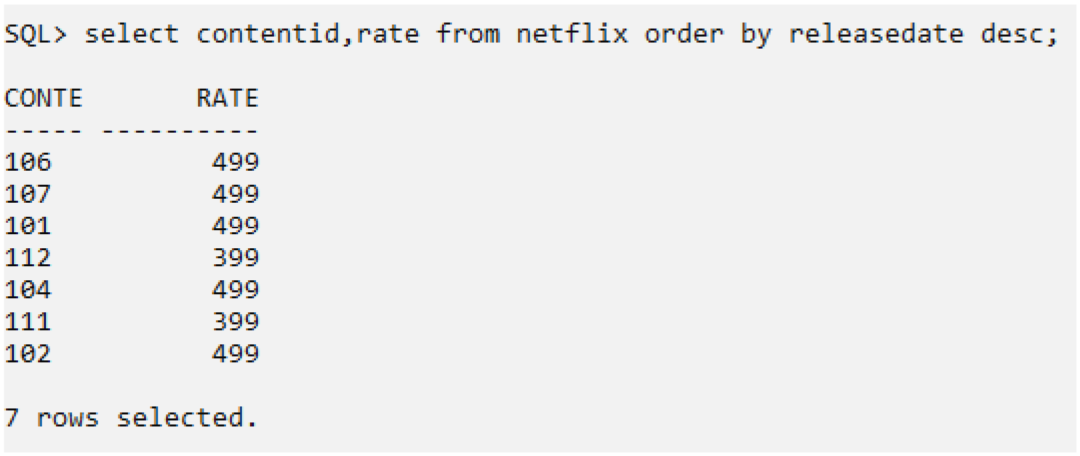


```
SQL> select contentid,title from netflix  order by rate asc;

CONTE TITLE
----- --------------------
111   Star
112   Avenger
101   Star
104   Just a select
107   Family Man
106   Mirzapur
102   Star Wars

7 rows selected.
```

Query: Fetch the contented and price in descending order of release data;

Code :select contentid,rate from netflix order by releasedate desc;

Output:



```
SQL> select contentid,rate from netflix order by releasedate desc;

CONTE       RATE
----- ----------
106          499
107          499
101          499
112          399
104          499
111          399
102          499

7 rows selected.
```

Code :select contentid,director from netflix order by genre;

Output:

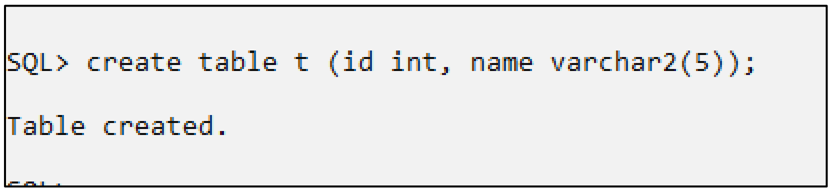
```
SQL> select contentid,director from netflix order by genre;

CONTE DIRECTOR
----- --------------------
104   aaa
106   karan p
107   xyz
102   aaa
111   karan p
112   karan p
101   director

7 rows selected.
```

Practical 4 : Transaction Control

A transaction is a unit or sequence of work that is performed on a database. Transactions are accomplished in a logical order, whether in a manual fashion by a user or automatically by some sort of a database program.

Code: create table t (id int , name varchar2(5));

Output:



```
SQL> create table t (id int, name varchar2(5));

Table created.
```

Code: create table t (id int , name varchar2(5));

Insert into t values (1, ‘aaa’);

Insert into t values (2, ‘bbb’);

Commit;

Rollback;

Output:

```
SQL> insert into t values (1, 'aaa');

1 row created.

SQL> insert into t values (2, 'bbb');

1 row created.

SQL> commit;

Commit complete.

SQL> delete from t where id = 2;

1 row deleted.

SQL> select * from t;

        ID NAME
---------- -----
         1 aaa

SQL> rollback;

Rollback complete.

SQL> select * from t;

        ID NAME
---------- -----
         1 aaa
         2 bbb
```

Practical 5 : Functions

SQL scalar functions are user-defined or built-in functions that take one or more parameters and return a single value.

a) Single Row Functions

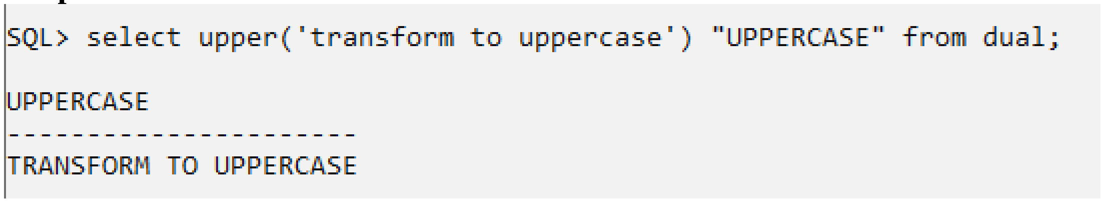
- manipulate data items
- accept arg and return one value
- act on each returned
- return one result per row
- may modify the datatype
- can be nested

function_name (col | expression, [arg1,arg2])
can work on number,character, string

<u>case conversion</u>

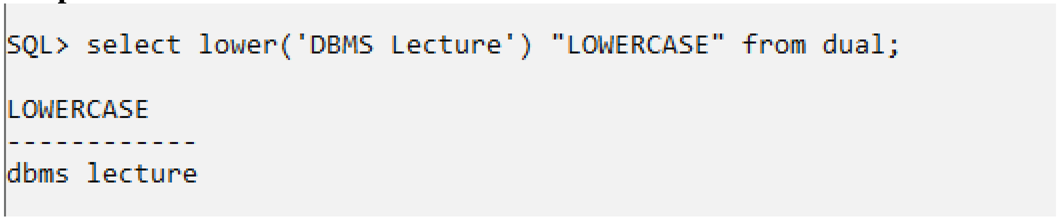
LOWER('SQL course')	sql course
UPPER('SQL course')	SQL COURSE
Initcap('sql course')	Sql Course

Code: select upper('transform to uppercase') "UPPERCASE" from dual;
Output:



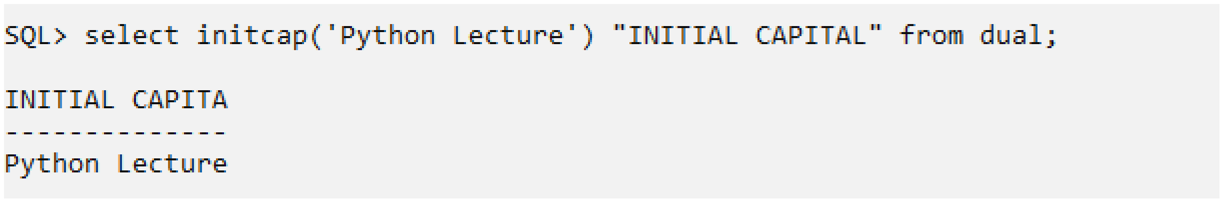
```
SQL> select upper('transform to uppercase') "UPPERCASE" from dual;

UPPERCASE
----------------------
TRANSFORM TO UPPERCASE
```

Code: select lower('DBMS Lecture') "LOWERCASE" from dual;
Output:



```
SQL> select lower('DBMS Lecture') "LOWERCASE" from dual;

LOWERCASE
------------
dbms lecture
```

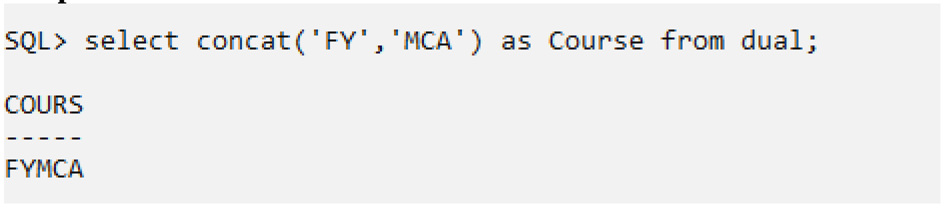
Code: select initcap('Python Lecture') "INITIAL CAPITAL" from dual;
Output:



```
SQL> select initcap('Python Lecture') "INITIAL CAPITAL" from dual;

INITIAL CAPITA
--------------
Python Lecture
```

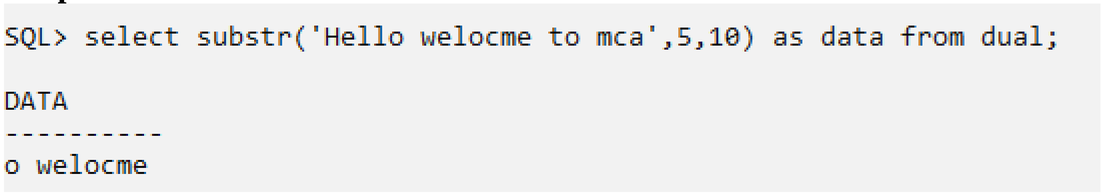
b) Character Functions

Code: select concat('FY','MCA') as Course from dual;
Output:



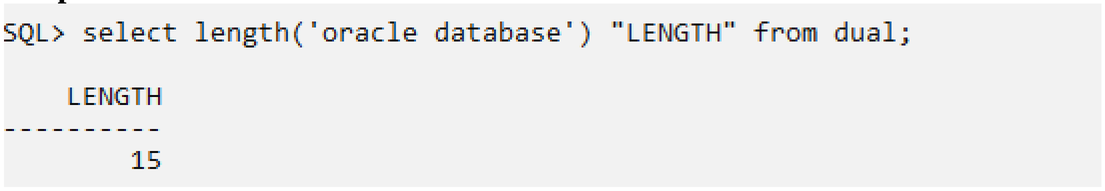
```
SQL> select concat('FY','MCA') as Course from dual;

COURS
-----
FYMCA
```

Code: select substr('Hello welocme to mca',5,10) as data from dual;
Output:



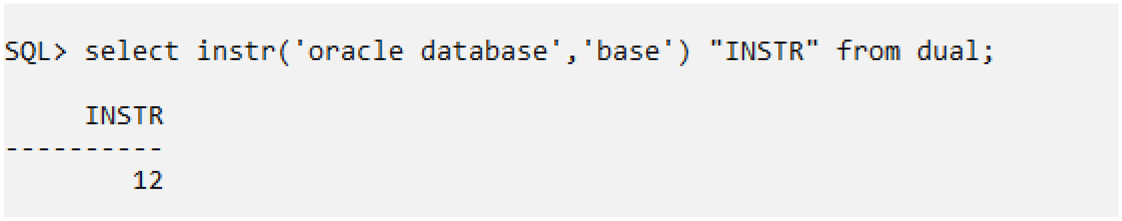
```
SQL> select substr('Hello welocme to mca',5,10) as data from dual;

DATA
----------
o welocme
```

Code: select length('oracle database') "LENGTH" from dual;
Output:



```
SQL> select length('oracle database') "LENGTH" from dual;

    LENGTH
----------
        15
```

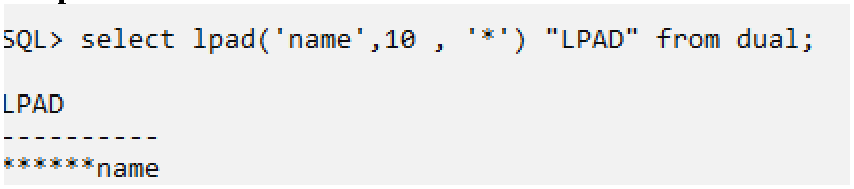
Code: select instr('oracle database','base') "INSTR" from dual;
Output:



```
SQL> select instr('oracle database','base') "INSTR" from dual;

     INSTR
----------
        12
```

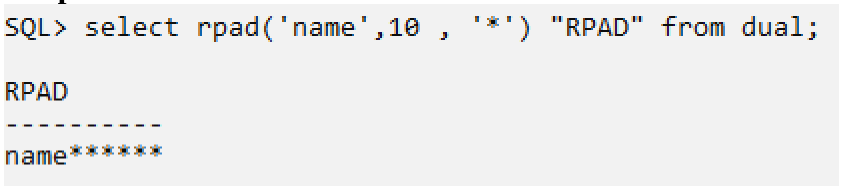
Code: select lpad('name',10 , '*') "LPAD" from dual;

Output:



```
SQL> select lpad('name',10 , '*') "LPAD" from dual;

LPAD
----------
******name
```

Code: select rpad('name',10 , '*') "RPAD" from dual;
Output:

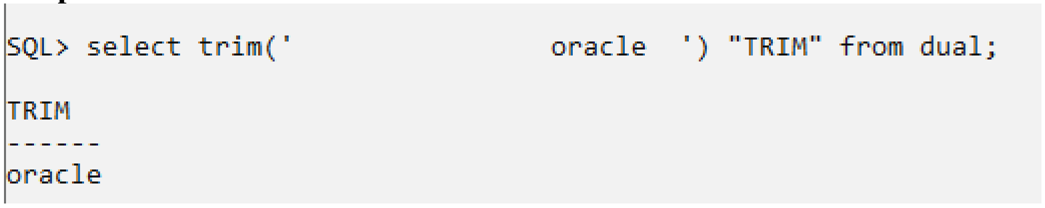


```
SQL> select rpad('name',10 , '*') "RPAD" from dual;

RPAD
----------
name******
```

Ascii
Manipulate Character strings

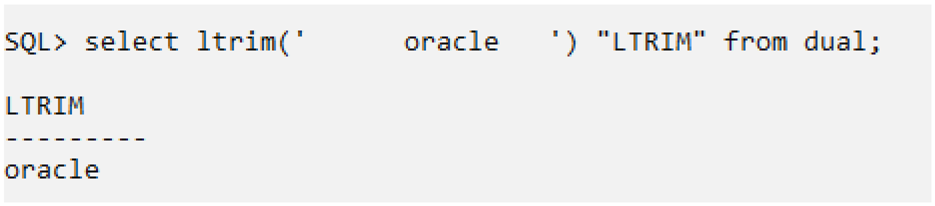
Function	Result
Translate('abcdef123','134','')	abcABCxyzXYZ
LTRIM('nicky','n')	lcky
RTRIM('nicky','n')	nicky
TRIM(' nicky ')	nicky

Code: select trim('   oracle ') "TRIM" from dual;
Output:



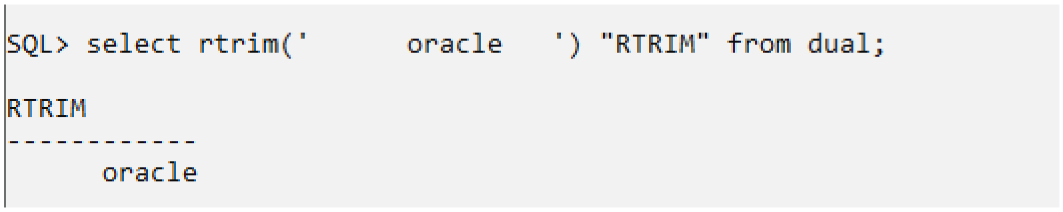
```
SQL> select trim('                oracle   ') "TRIM" from dual;

TRIM
------
oracle
```

Code: select ltrim('   oracle   ') "LTRIM" from dual;
Output:



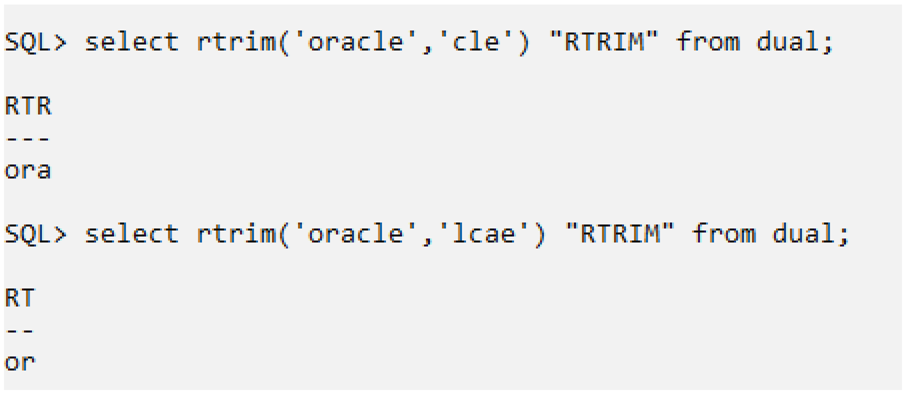
```
SQL> select ltrim('      oracle   ') "LTRIM" from dual;

LTRIM
---------
oracle
```

Code: select rtrim('   oracle   ') "RTRIM" from dual;
Output:



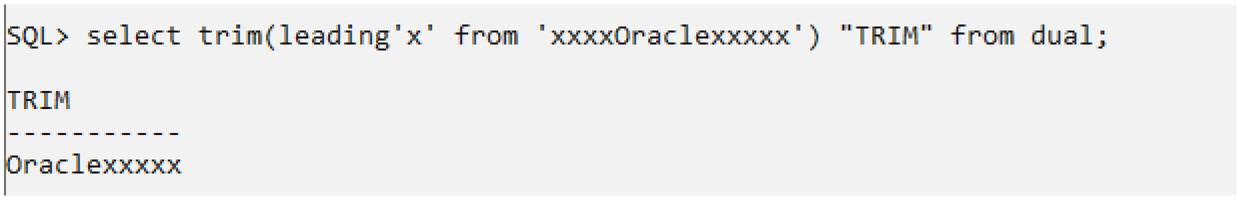
```
SQL> select rtrim('      oracle   ') "RTRIM" from dual;

RTRIM
------------
      oracle
```

Code: select rtrim('oracle','cle') "RTRIM" from dual;
 select rtrim('oracle','lcae') "RTRIM" from dual;
Output:



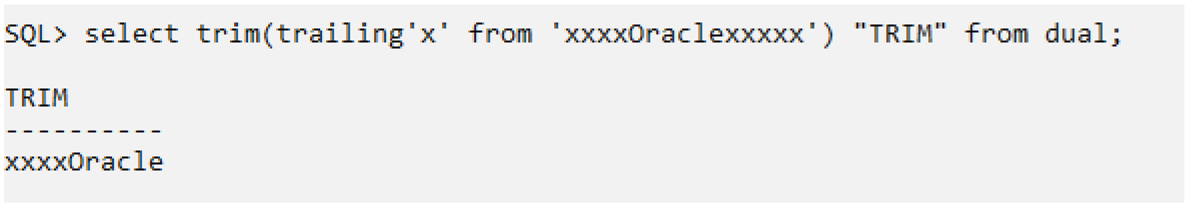
```
SQL> select rtrim('oracle','cle') "RTRIM" from dual;

RTR
---
ora

SQL> select rtrim('oracle','lcae') "RTRIM" from dual;

RT
--
or
```

Code: select trim(leading'x' from 'xxxxOraclexxxxx') "TRIM" from dual;
Output:



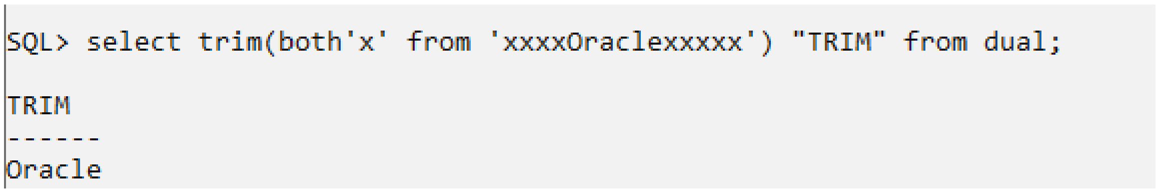
```
SQL> select trim(leading'x' from 'xxxxOraclexxxxx') "TRIM" from dual;

TRIM
-----------
Oraclexxxxx
```

Code: select trim(trailing'x' from 'xxxxOraclexxxxx') "TRIM" from dual;
Output:



```
SQL> select trim(trailing'x' from 'xxxxOraclexxxxx') "TRIM" from dual;

TRIM
----------
xxxxOracle
```

Code: select trim(both'x' from 'xxxxOraclexxxxx') "TRIM" from dual;
Output:

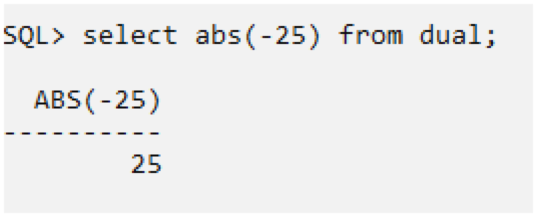


```
SQL> select trim(both'x' from 'xxxxOraclexxxxx') "TRIM" from dual;

TRIM
------
Oracle
```

c) Numeric Functions

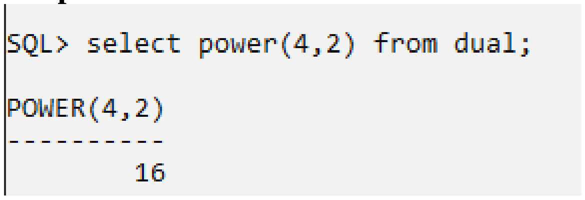
ABS(n)	absolute value of n
POWER(m,n)	M raised to n
ROUND(n,m)	n rounded to m places
SQRT(n)	square root of n
EXP(n)	en
EXTRACT()	value extracted from a date
GREATEST()	greatest value in the list
LEAST()	least value in the list
MOD(m,n)	remainder of m/n
TRUNC()	number truncated to a certain number of decimal places
FLOOR(n)	largest integer<=n
CEIL(n)	smallest integer>= n

Code: select abs(-25) from dual;
Output:



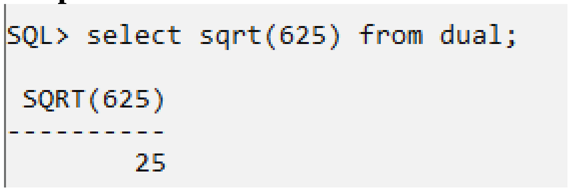
```
SQL> select abs(-25) from dual;

  ABS(-25)
----------
        25
```

Code:select power(4,2) from dual;
Output:



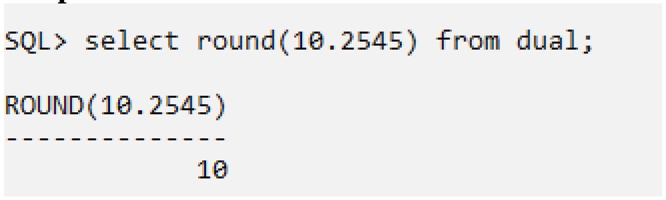
```
SQL> select power(4,2) from dual;

POWER(4,2)
----------
        16
```

Code:select sqrt(625) from dual;
Output:



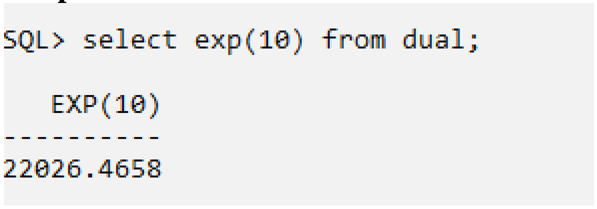
```
SQL> select sqrt(625) from dual;

 SQRT(625)
----------
        25
```

Code: select round(10.2545) from dual;
Output:



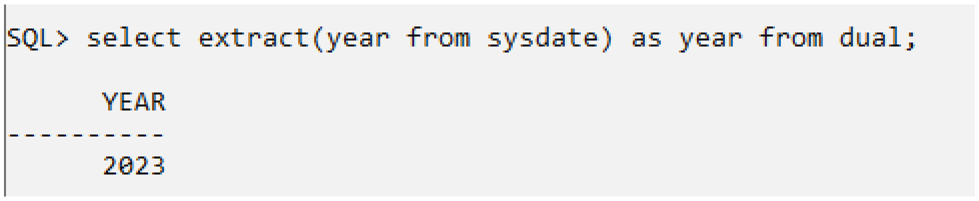
```
SQL> select round(10.2545) from dual;

ROUND(10.2545)
--------------
            10
```

Code:select exp(10) from dual;
Output:

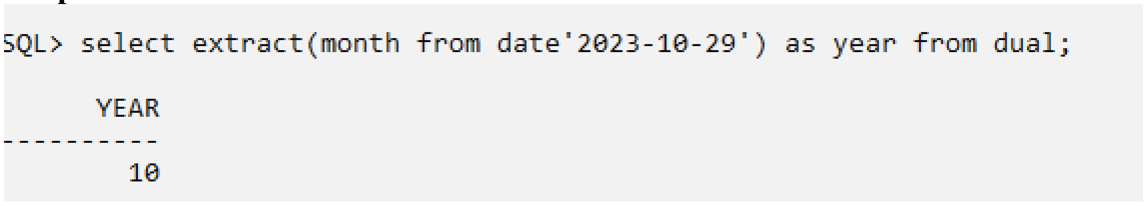


```
SQL> select exp(10) from dual;

   EXP(10)
----------
22026.4658
```

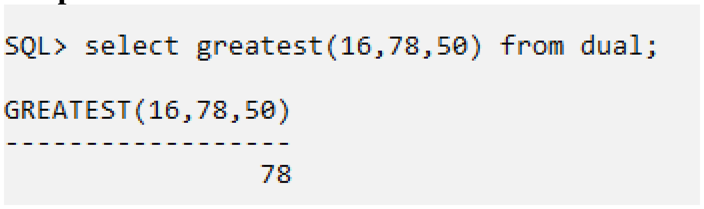
Code: select extract(year from sysdate) as year from dual;
Output:


```
SQL> select extract(year from sysdate) as year from dual;

      YEAR
----------
      2023
```

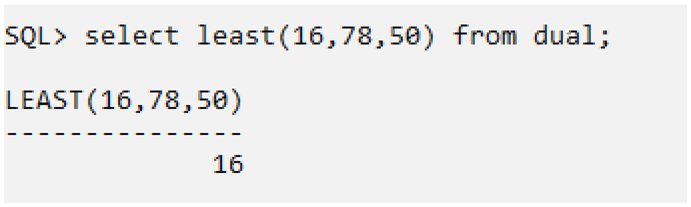
Code:select extract(month from date'2023-10-29') as year from dual;
Output:


```
SQL> select extract(month from date'2023-10-29') as year from dual;

      YEAR
----------
        10
```

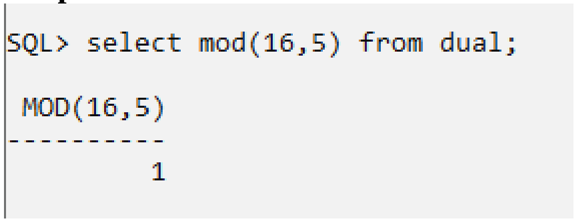
Code:select greatest(16,78,50) from dual;
Output:


```
SQL> select greatest(16,78,50) from dual;

GREATEST(16,78,50)
------------------
                78
```

Code:select least(16,78,50) from dual;
Output:

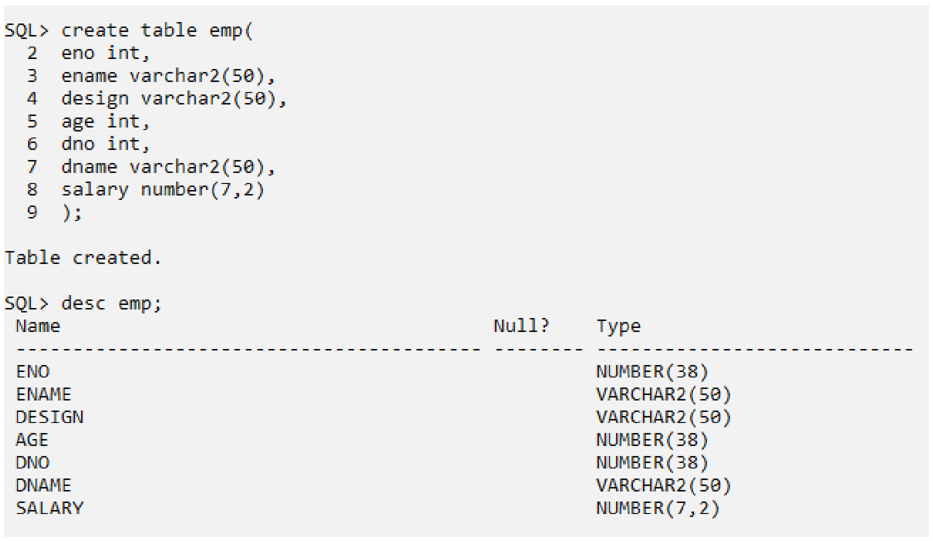

```
SQL> select least(16,78,50) from dual;

LEAST(16,78,50)
---------------
             16
```

Code: select mod(16,5) from dual;
Output:


```
SQL> select mod(16,5) from dual;

 MOD(16,5)
----------
         1
```

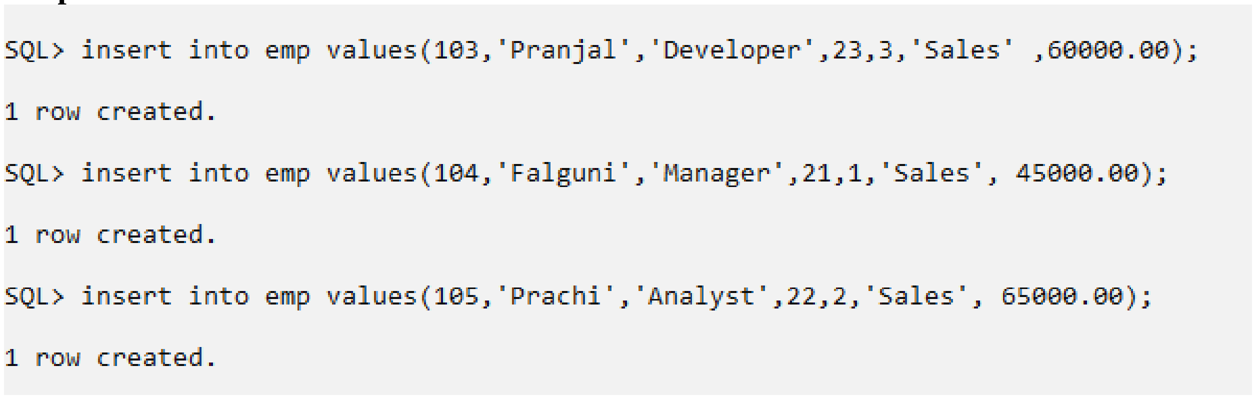
d) Date Functions
e) Conversion Functions
f) General Functions
g) Multiple Row Functions

Code:
create table emp(eno int,ename varchar2(50),design varchar2(50),age int,dno int,dname varchar2(50),salary number(7,2));
Output:



```
SQL> create table emp(
  2  eno int,
  3  ename varchar2(50),
  4  design varchar2(50),
  5  age int,
  6  dno int,
  7  dname varchar2(50),
  8  salary number(7,2)
  9  );

Table created.

SQL> desc emp;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ENO                                                NUMBER(38)
 ENAME                                              VARCHAR2(50)
 DESIGN                                             VARCHAR2(50)
 AGE                                                NUMBER(38)
 DNO                                                NUMBER(38)
 DNAME                                              VARCHAR2(50)
 SALARY                                             NUMBER(7,2)
```

Code:
insert into emp values(101,'Karan','Analyst',23,2,'Sales', 50000.00);
insert into emp values(102,'Darshan','Manager',24,1,'IT', 75000.00);
insert into emp values(103,'Pranjal','Developer',23,3,'Finance' ,60000.00);
insert into emp values(104,'Falguni','Manager',21,1,'Sales', 45000.00);
insert into emp values(105,'Prachi','Analyst',22,2,'IT', 65000.00);
Output:



```
SQL> insert into emp values(103,'Pranjal','Developer',23,3,'Sales' ,60000.00);

1 row created.

SQL> insert into emp values(104,'Falguni','Manager',21,1,'Sales', 45000.00);

1 row created.

SQL> insert into emp values(105,'Prachi','Analyst',22,2,'Sales', 65000.00);

1 row created.
```

<u>Table Level</u>
Code: Create table Orders(
    OrderID int not null,
    OrderNumber int not null,

```
        PersonID int, Primary Key (OrderID),
Foreign Key(PersonID) References Person(PersonID));
```

<u>Column Level</u>

Code:

```
Create table Orders(
        OrderID int not null PRIMARY KEY,
        OrderNumber int not null,
        PersonID int,
Int Foreign Key References Persons(PersonID));
```

Practical 6 : Subqueries

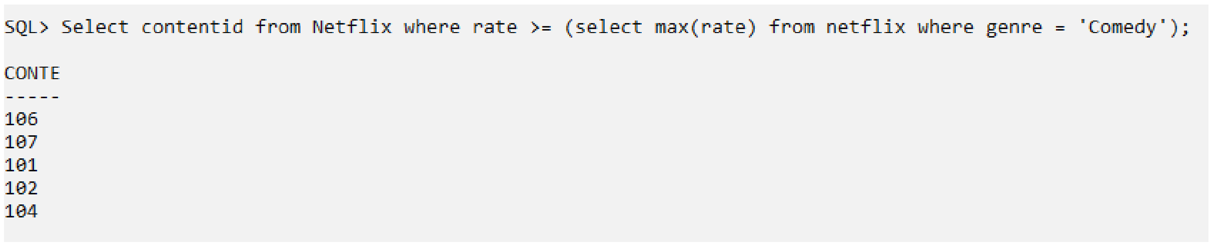
a) Subquery

Sub Queries

Query: fetch the id series with price greater than the price of series in comedy genere

Code: Select contentid from Netflix where rate >= (select max(rate) from netflix where genre = 'Comedy');

Output:

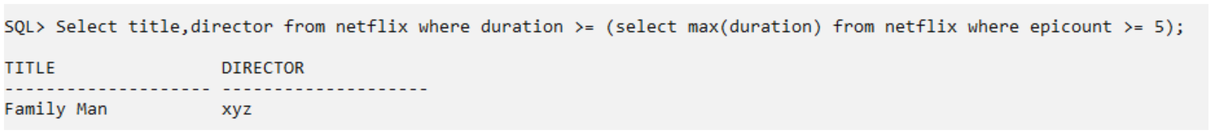


```
SQL> Select contentid from Netflix where rate >= (select max(rate) from netflix where genre = 'Comedy');

CONTE
-----
106
107
101
102
104
```

Query: Fetch the title and director of the series with duration more than the duration of the series of the series with episode count more than 5.

Code: Select title,director from netflix where duration >= (select max(duration) from netflix where epicount >= 5);

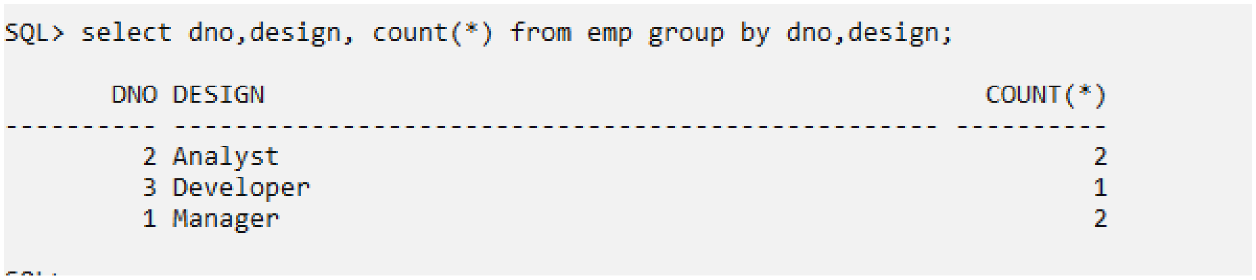
Output:



```
SQL> Select title,director from netflix where duration >= (select max(duration) from netflix where epicount >= 5);

TITLE                DIRECTOR
-------------------- --------------------
Family Man           xyz
```

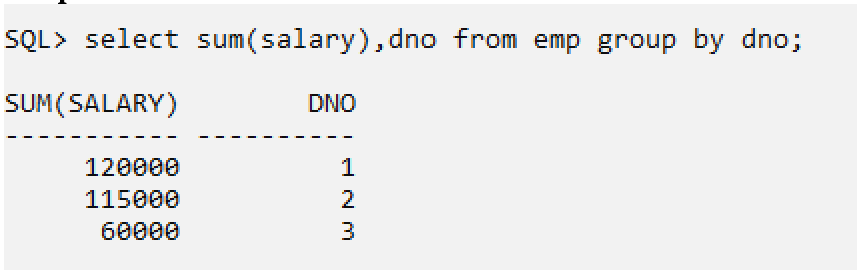
b) Types of Subquery
c) Group Function

Query: How many employees working in each department?
Code: select dno,design, count(*) from emp group by dno,design;
Output:



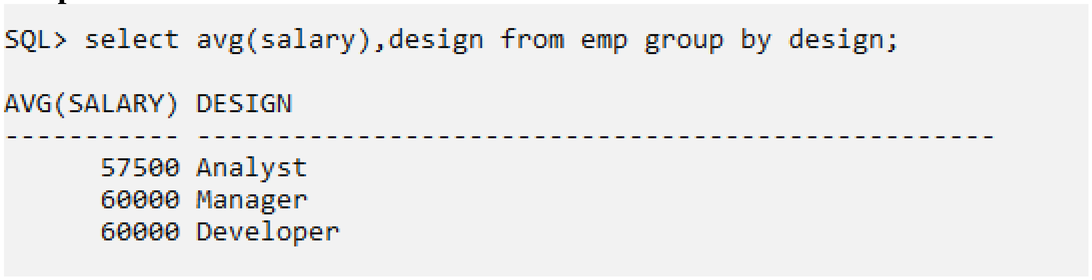
```
SQL> select dno,design, count(*) from emp group by dno,design;

       DNO DESIGN                                              COUNT(*)
---------- -------------------------------------------------- ----------
         2 Analyst                                                     2
         3 Developer                                                   1
         1 Manager                                                     2
```

Query: total salary for each department
Code: select sum(salary),dno from emp group by dno;
Output:



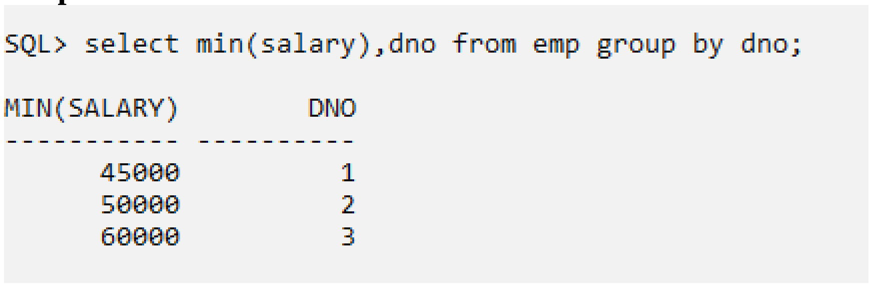
```
SQL> select sum(salary),dno from emp group by dno;

SUM(SALARY)        DNO
----------- ----------
     120000          1
     115000          2
      60000          3
```

Query: Average salary for each department
Code: select avg(salary),design from emp group by design;
Output:



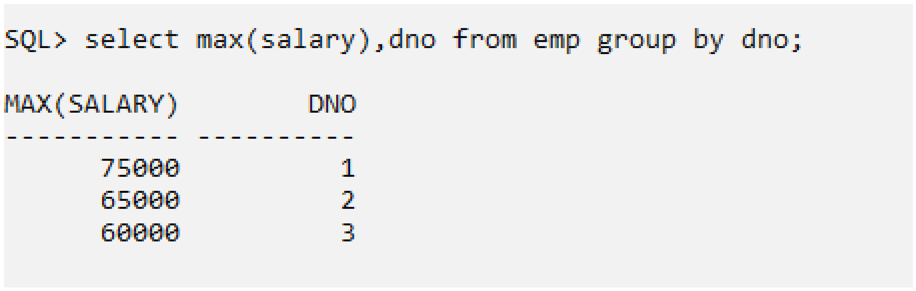
```
SQL> select avg(salary),design from emp group by design;

AVG(SALARY) DESIGN
----------- --------------------------------------------------
      57500 Analyst
      60000 Manager
      60000 Developer
```

Query: Minimum salary for each department
Code: select min(salary),dno from emp group by dno;
Output:



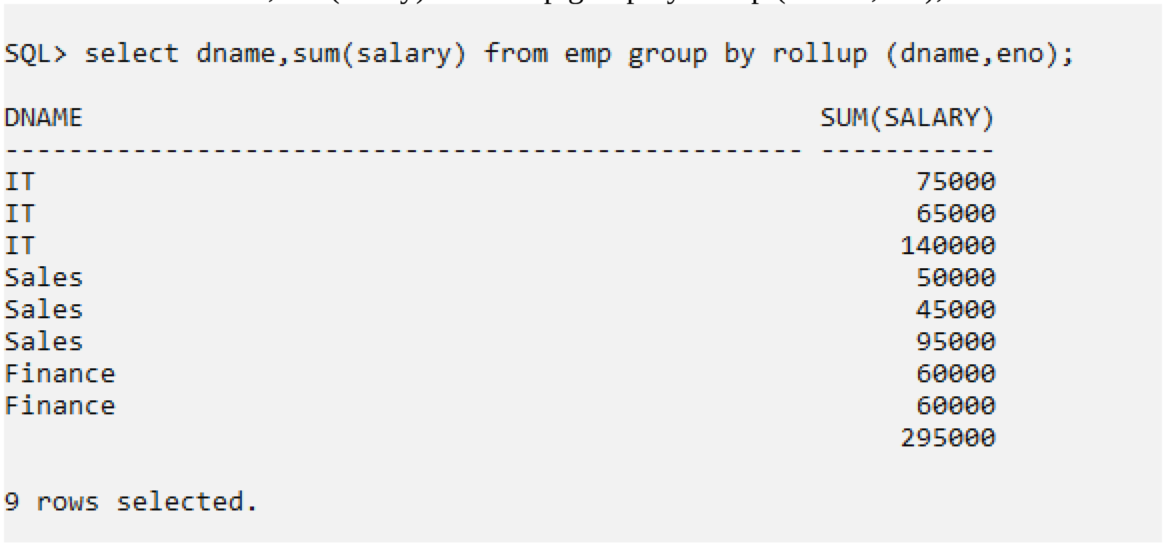
```
SQL> select min(salary),dno from emp group by dno;

MIN(SALARY)        DNO
----------- ----------
      45000          1
      50000          2
      60000          3
```

Query: Maximum salary for each department
Code: select max(salary),dno from emp group by dno;
Output:



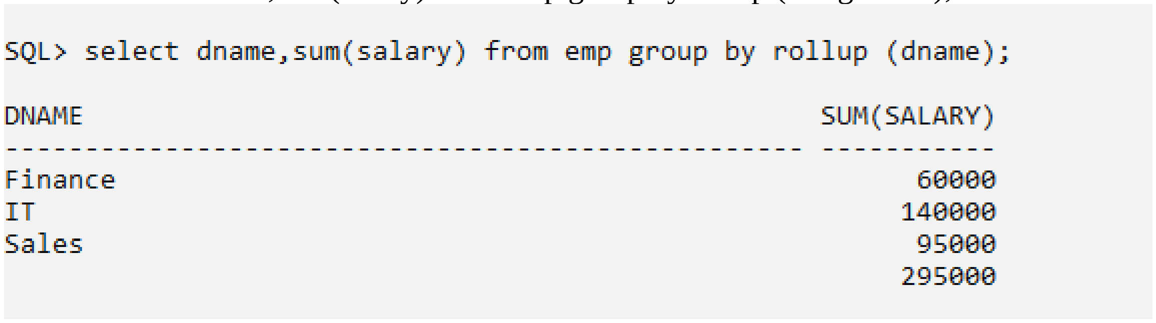
```
SQL> select max(salary),dno from emp group by dno;

MAX(SALARY)        DNO
----------- ----------
      75000          1
      65000          2
      60000          3
```

Code: select dname,sum(salary) from emp group by rollup (dname,eno);



```
SQL> select dname,sum(salary) from emp group by rollup (dname,eno);

DNAME                                              SUM(SALARY)
-------------------------------------------------- -----------
IT                                                       75000
IT                                                       65000
IT                                                      140000
Sales                                                    50000
Sales                                                    45000
Sales                                                    95000
Finance                                                  60000
Finance                                                  60000
                                                        295000

9 rows selected.
```

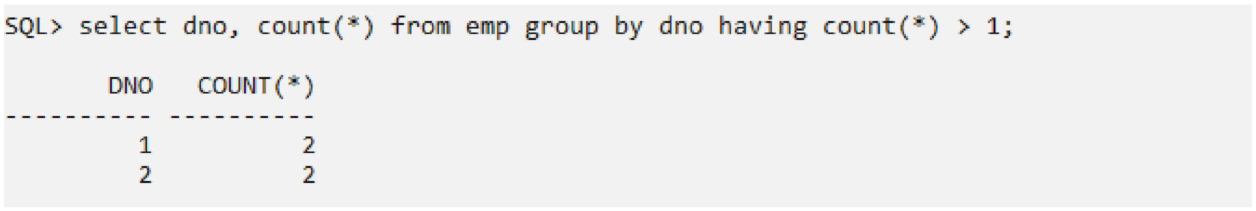
Query: Find the subtotal of salary for each designation
Code: select dname,sum(salary) from emp group by rollup (designation);



```
SQL> select dname,sum(salary) from emp group by rollup (dname);

DNAME                                              SUM(SALARY)
-------------------------------------------------- -----------
Finance                                                  60000
IT                                                      140000
Sales                                                    95000
                                                        295000
```

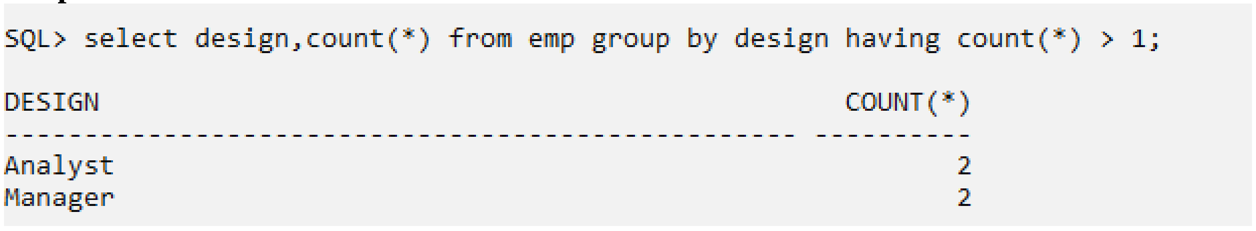
d) Having Clause

Query: display the department with number of employees more than one
Code: select dno, count(*) from emp group by dno having count(*) > 1;
Output:



```
SQL> select dno, count(*) from emp group by dno having count(*) > 1;

       DNO   COUNT(*)
---------- ----------
         1          2
         2          2
```

Query: Display the designation with number of employees more than 1
Code: select design,count(*) from emp group by design having count(*) > 1;
Output:

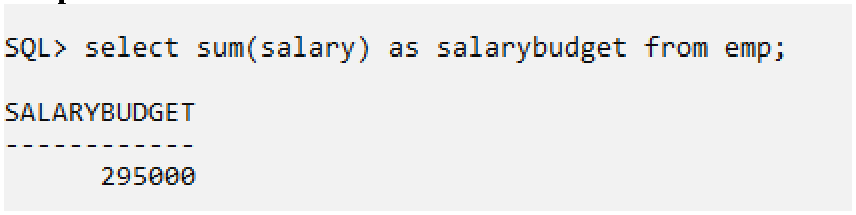


```
SQL> select design,count(*) from emp group by design having count(*) > 1;

DESIGN                                               COUNT(*)
-------------------------------------------------- ----------
Analyst                                                     2
Manager                                                     2
```

e) Aggregate function

Query: Write query to display sum of salary of all employees

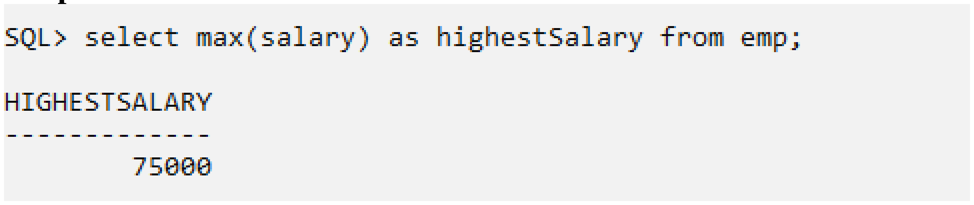
Code: select sum(salary) as salarybudget from emp;
Output:



```
SQL> select sum(salary) as salarybudget from emp;

SALARYBUDGET
------------
      295000
```

Query: maximum salary of all employees

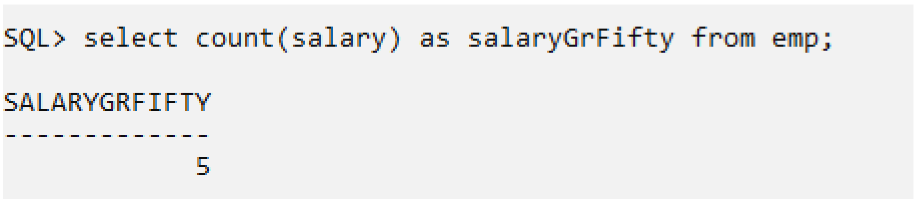
Code: select max(salary) as highestSalary from emp;
Output:



```
SQL> select max(salary) as highestSalary from emp;

HIGHESTSALARY
-------------
        75000
```

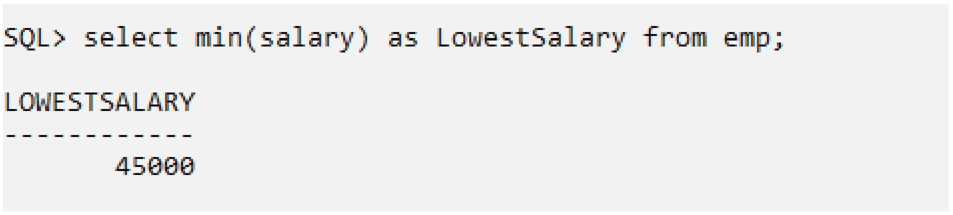
Query: count the number of employees getting salary > 50000
Code: select count(salary) as salaryGrFifty from emp;

Output:



```
SQL> select count(salary) as salaryGrFifty from emp;

SALARYGRFIFTY
-------------
            5
```

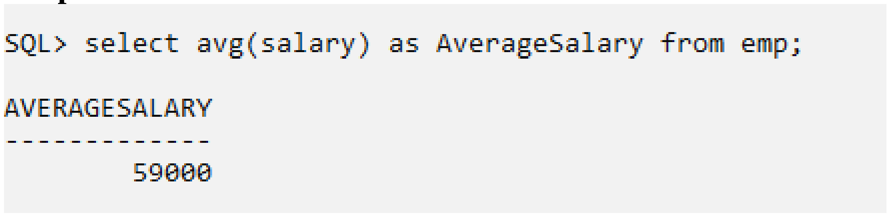
Query: minimum salary from employee
Code: select min(salary) as LowestSalary from emp;
Output:



```
SQL> select min(salary) as LowestSalary from emp;

LOWESTSALARY
------------
       45000
```

Query: average salary for all employees

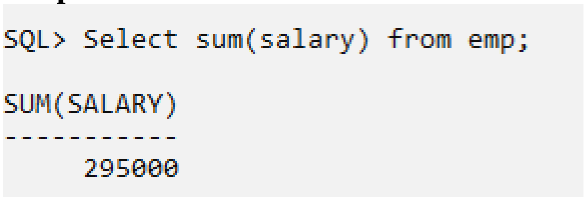
Code: select avg(salary) as AverageSalary from emp;
Output:



```
SQL> select avg(salary) as AverageSalary from emp;

AVERAGESALARY
-------------
        59000
```

f) Window function

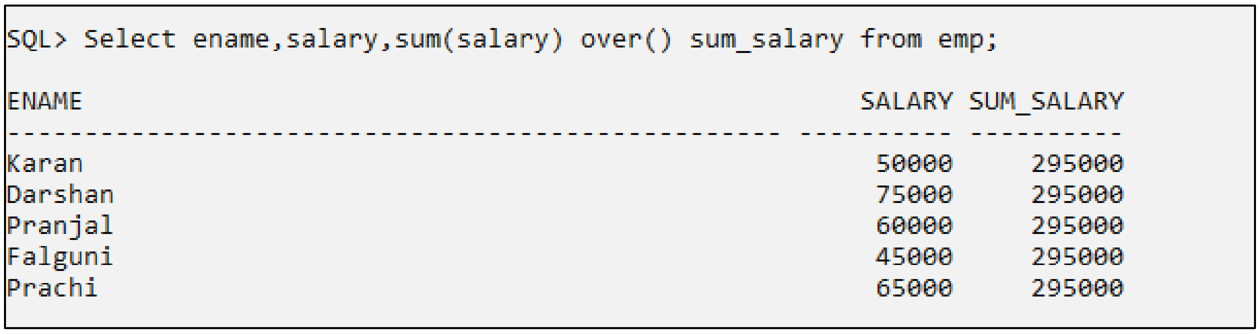
Code: Select sum(salary) from emp
Output:



```
SQL> Select sum(salary) from emp;

SUM(SALARY)
-----------
     295000
```

Code: Select ename,salary,sum(salary) over() sum_salary from emp;

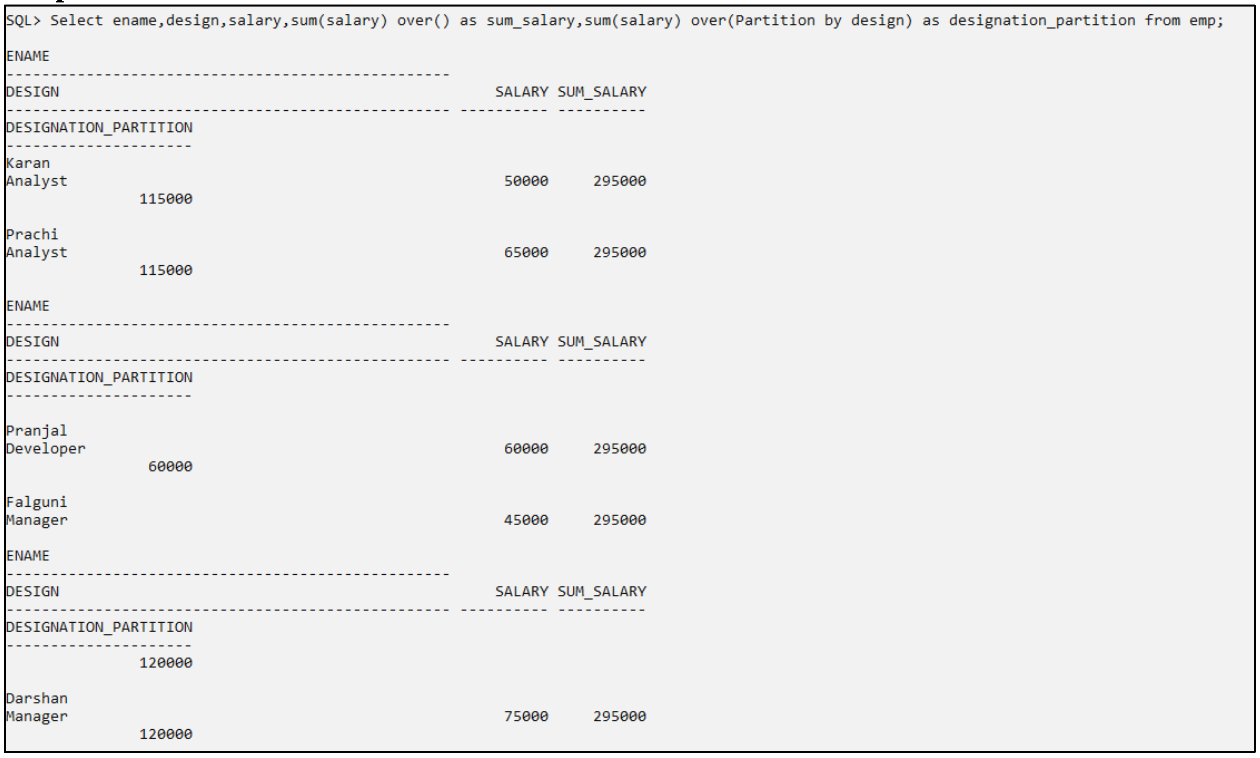
Output:



```
SQL> Select ename,salary,sum(salary) over() sum_salary from emp;

ENAME                                                  SALARY SUM_SALARY
-------------------------------------------------- ---------- ----------
Karan                                                   50000     295000
Darshan                                                 75000     295000
Pranjal                                                 60000     295000
Falguni                                                 45000     295000
Prachi                                                  65000     295000
```

Code: Select ename, design,salary,sum(salary) over() as sum_salary,sum(salary) over(Partition by design) as designation_partition from emp;

Output:



```
SQL> Select ename,design,salary,sum(salary) over() as sum_salary,sum(salary) over(Partition by design) as designation_partition from emp;

ENAME
--------------------------------------------------
DESIGN                                                 SALARY SUM_SALARY
-------------------------------------------------- ---------- ----------
DESIGNATION_PARTITION
---------------------
Karan
Analyst                                                 50000     295000
               115000

Prachi
Analyst                                                 65000     295000
               115000

ENAME
--------------------------------------------------
DESIGN                                                 SALARY SUM_SALARY
-------------------------------------------------- ---------- ----------
DESIGNATION_PARTITION
---------------------

Pranjal
Developer                                               60000     295000
                60000

Falguni
Manager                                                 45000     295000

ENAME
--------------------------------------------------
DESIGN                                                 SALARY SUM_SALARY
-------------------------------------------------- ---------- ----------
DESIGNATION_PARTITION
---------------------
               120000

Darshan
Manager                                                 75000     295000
               120000
```

Practical 7 : Constraints

a. Not Null

Name varchar2(15) constraint faculty_name_nn NOT NULL

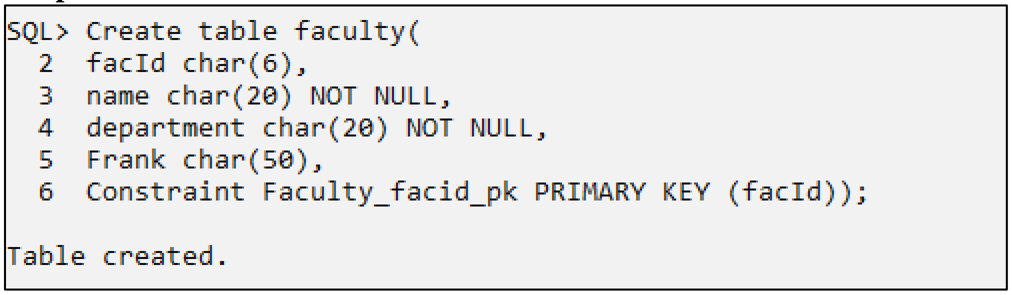
b. Unique Key

<u>Table level</u>
Constraint dept_deptname_uk UNIQUE(DeptName),

<u>Column level</u>
DeptName varchar(12) constraint dept_deptname_uk unique,

c. Primary Key

Code: Create table faculty(
facId char(6),
name char(20) NOT NULL,
department char(20) NOT NULL,
Frank char(50),
Constraint Faculty_facid_pk PRIMARY KEY (facId));
Output:



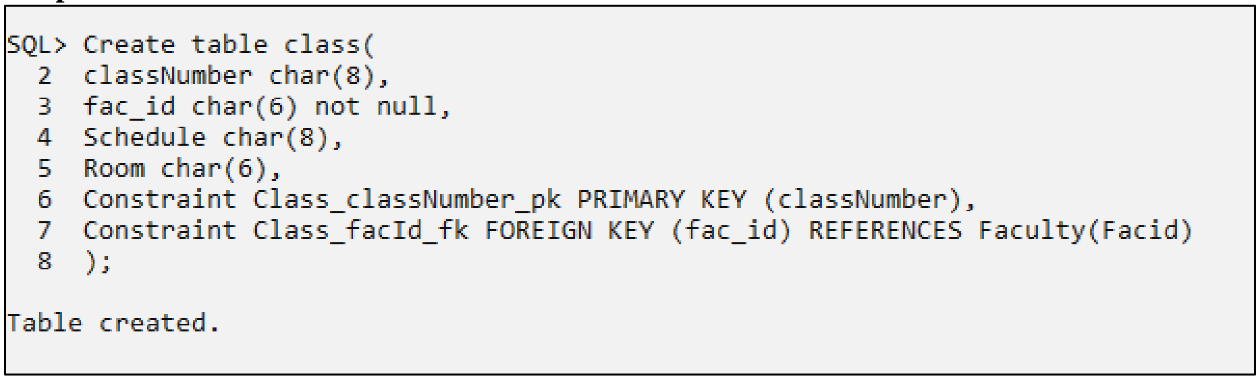
```
SQL> Create table faculty(
  2  facId char(6),
  3  name char(20) NOT NULL,
  4  department char(20) NOT NULL,
  5  Frank char(50),
  6  Constraint Faculty_facid_pk PRIMARY KEY (facId));

Table created.
```

d. Foreign Key

Code: Create table class(
classNumber char(8),
fac_id char(6) not null,

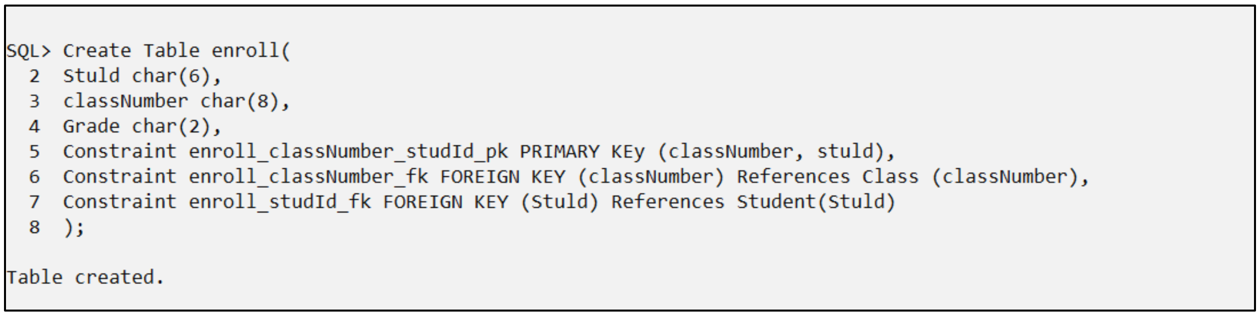
Schedule char(8),
Room char(6),
Constraint Class_classNumber_pk PRIMARY KEY (classNumber),
Constraint Class_facId_fk FOREIGN KEY (fac_id) REFERENCES Faculty(Facid)
);

Output:



```
SQL> Create table class(
classNumber char(8),
fac_id char(6) not null,
Schedule char(8),
Room char(6),
Constraint Class_classNumber_pk PRIMARY KEY (classNumber),
Constraint Class_facId_fk FOREIGN KEY (fac_id) REFERENCES Faculty(Facid)
);

Table created.
```

Code: Create Table enroll(
Stuld char(6),
classNumber char(8),
Grade char(2),
Constraint enroll_classNumber_studId_pk PRIMARY KEy (classNumber, studId),
Constraint enroll_classNumber_fk FOREIGN KEY (classNumber) References Class (classNumber),
Constraint enroll_studId_fk FOREIGN KEY (Stuld) References Student(Stuld)
);



```
SQL> Create Table enroll(
Stuld char(6),
classNumber char(8),
Grade char(2),
Constraint enroll_classNumber_studId_pk PRIMARY KEy (classNumber, stuld),
Constraint enroll_classNumber_fk FOREIGN KEY (classNumber) References Class (classNumber),
Constraint enroll_studId_fk FOREIGN KEY (Stuld) References Student(Stuld)
);

Table created.
```

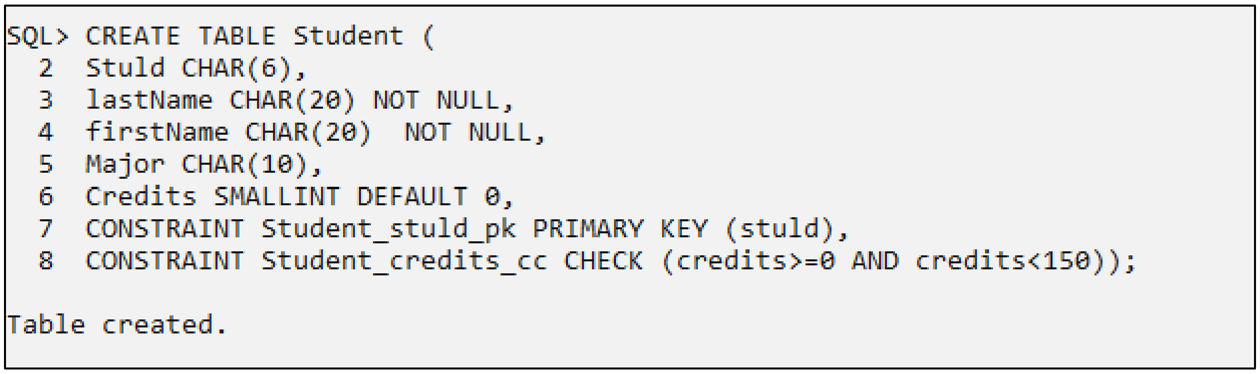
e. Check

Check constraint

Code: CREATE TABLE Student (
Stuld CHAR(6),
lastName CHAR(20) NOT NULL,
firstName CHAR(20) NOT NULL,

Major CHAR(10),
Credits SMALLINT DEFAULT 0,
CONSTRAINT Student_stuld_pk PRIMARY KEY (stuld),
CONSTRAINT Student_credits_cc CHECK (credits>=0 AND credits<150));

Output:



```
SQL> CREATE TABLE Student (
  2  Stuld CHAR(6),
  3  lastName CHAR(20) NOT NULL,
  4  firstName CHAR(20)  NOT NULL,
  5  Major CHAR(10),
  6  Credits SMALLINT DEFAULT 0,
  7  CONSTRAINT Student_stuld_pk PRIMARY KEY (stuld),
  8  CONSTRAINT Student_credits_cc CHECK (credits>=0 AND credits<150));

Table created.
```

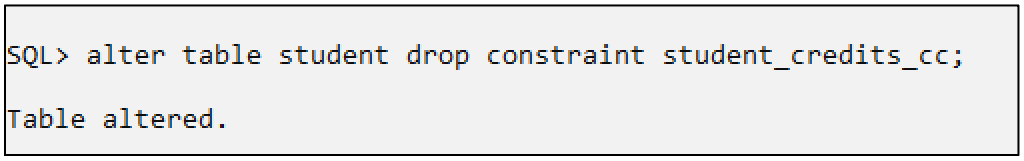
Query:
<u>Table Level</u>
DeptId Number(2) Constraint dept_deptid_ck CHECK((DeptId >= 10) and (Dept <= 99))

<u>Column Level</u>

Constraint dept_deptid_cc check((DeptId >= 10) and (Dept <= 99)),

f. Dropping a Constraint

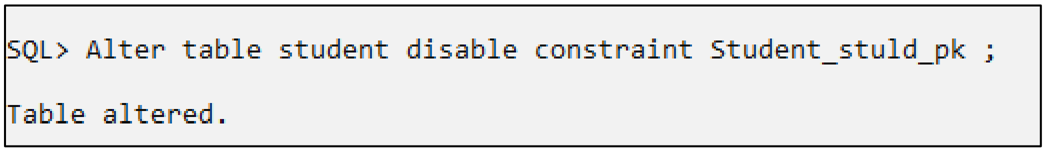
Query: Dropping constraint
Code: Alter table table_name drop constraint constraint_name;
alter table student drop constraint student_credits_cc;
Output:



```
SQL> alter table student drop constraint student_credits_cc;

Table altered.
```

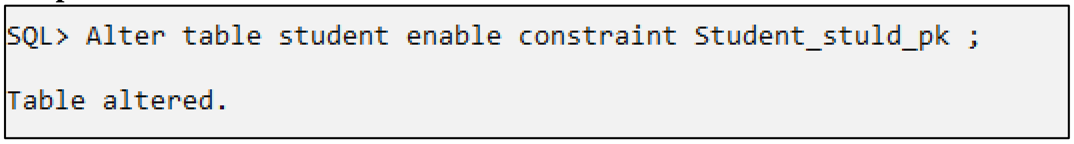
g. Enabling & Disabling

Query: Disable constraint
Code: Alter table student disable constraint Student_stuld_pk ;
Output:



```
SQL> Alter table student disable constraint Student_stuld_pk ;

Table altered.
```

Query: Enable Constraint
Code: Alter table student enable constraint Student_stuld_pk ;
Output:



```
SQL> Alter table student enable constraint Student_stuld_pk ;

Table altered.
```

Practical 8 : Joins

Syntax:

Select table1.column table2.column
From table1, table2
Where table1.column1 = table2.column2

Fetch patient id , patient name, dspec

Patient id, name - Patient
Dspec - Doctor

Example:

Select patient.pid, patient.name, docter.dspec

From patient,doctor Where patient.dId = doctor.docId

Select **p**.id, **p**.name, **d**.dspec
From patient **p**, doctor **d**
Where **p**.dID = **d**.docID

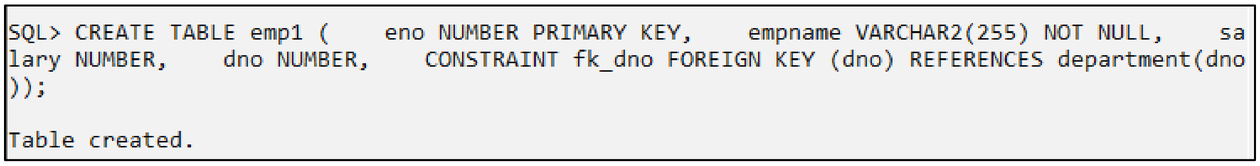
<u>**Query :** Create the following tables with necessary constraints</u>

EMP (eno, empname, salary, dno)
Department (dId, dname, locId)
Location (LocId, Location)

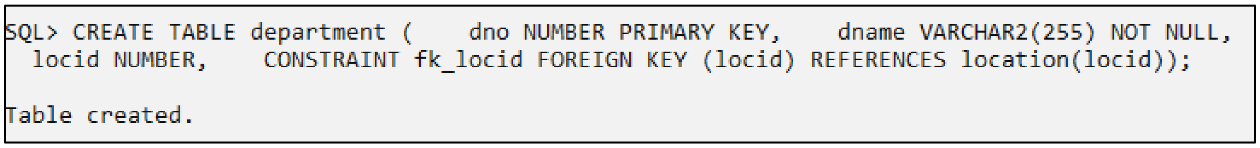
Code:

```
CREATE TABLE emp1 (
eno NUMBER PRIMARY KEY,
empname VARCHAR2(255) NOT NULL,
salary NUMBER,
dno NUMBER,
 CONSTRAINT fk_dno FOREIGN KEY (dno) REFERENCES department(dno));
```

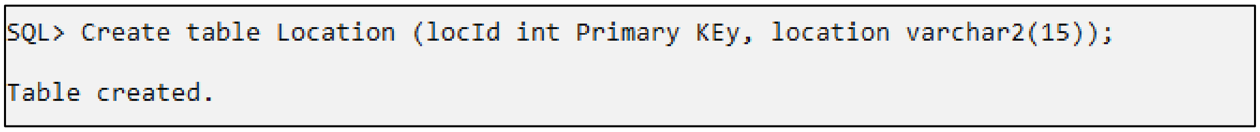
Output:


```
SQL> CREATE TABLE emp1 (    eno NUMBER PRIMARY KEY,    empname VARCHAR2(255) NOT NULL,    sa
lary NUMBER,    dno NUMBER,    CONSTRAINT fk_dno FOREIGN KEY (dno) REFERENCES department(dno
));

Table created.
```

Code:
CREATE TABLE department (
dno NUMBER PRIMARY KEY,
dname VARCHAR2(255) NOT NULL,
locid NUMBER,
CONSTRAINT fk_locid FOREIGN KEY (locid) REFERENCES location(locid));
Output:

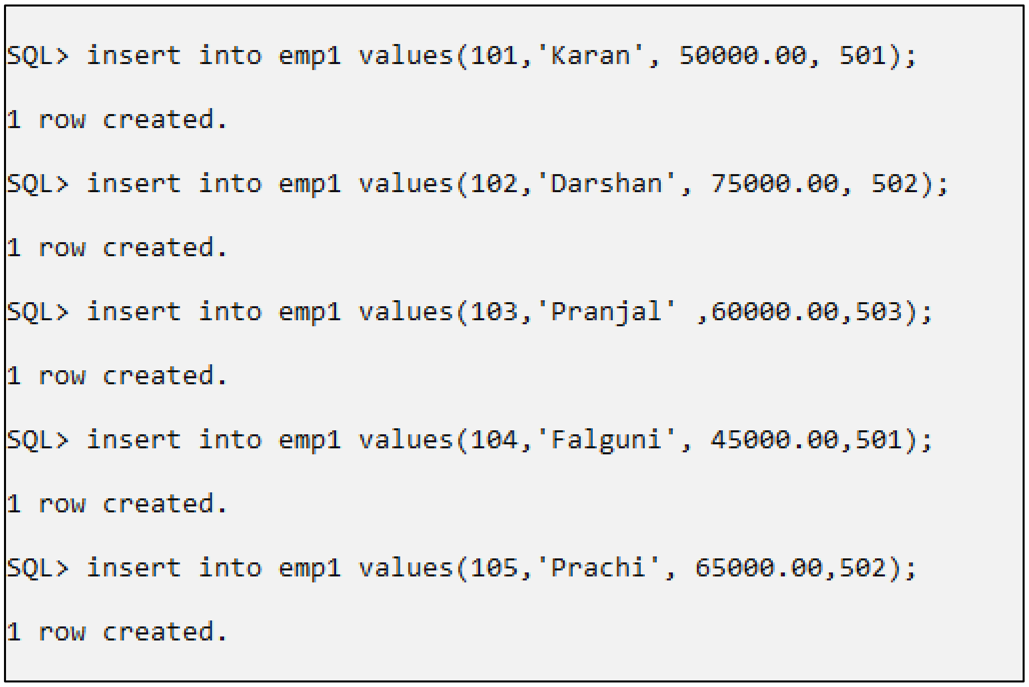

```
SQL> CREATE TABLE department (    dno NUMBER PRIMARY KEY,    dname VARCHAR2(255) NOT NULL,
  locid NUMBER,    CONSTRAINT fk_locid FOREIGN KEY (locid) REFERENCES location(locid));

Table created.
```

Code:
Create table Location (locId int primary key, location varchar2(15));
Output:


```
SQL> Create table Location (locId int Primary KEy, location varchar2(15));

Table created.
```

Insert 5 records in each table
Code:

insert into emp1 values(101,'Karan', 50000.00, 501);
insert into emp1 values(102,'Darshan', 75000.00, 502);
insert into emp1 values(103,'Pranjal' ,60000.00,503);
insert into emp1 values(104,'Falguni', 45000.00,501);
insert into emp1 values(105,'Prachi', 65000.00,502);
Output:

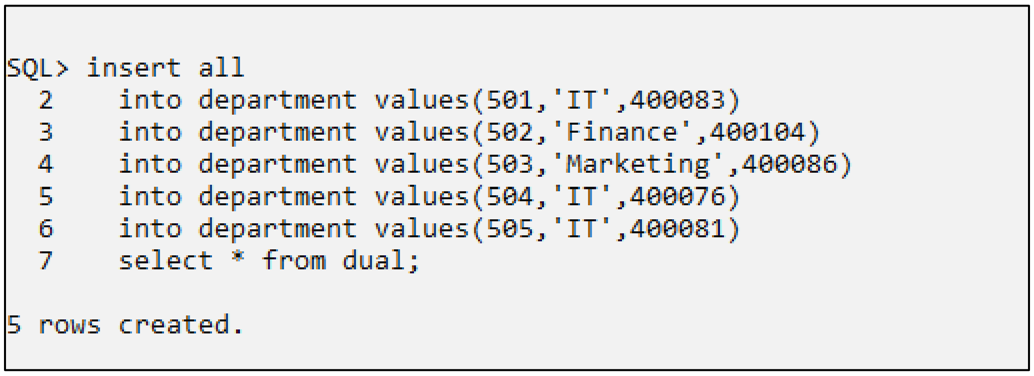


```
SQL> insert into emp1 values(101,'Karan', 50000.00, 501);

1 row created.

SQL> insert into emp1 values(102,'Darshan', 75000.00, 502);

1 row created.

SQL> insert into emp1 values(103,'Pranjal' ,60000.00,503);

1 row created.

SQL> insert into emp1 values(104,'Falguni', 45000.00,501);

1 row created.

SQL> insert into emp1 values(105,'Prachi', 65000.00,502);

1 row created.
```

Code:

```
insert all
  into department values(501,'IT',400083)
  into department values(502,'Finance',400104)
  into department values(503,'Marketing',400086)
  into department values(504,'IT',400076)
  into department values(505,'IT',400081)
  select * from dual;
```

Output:



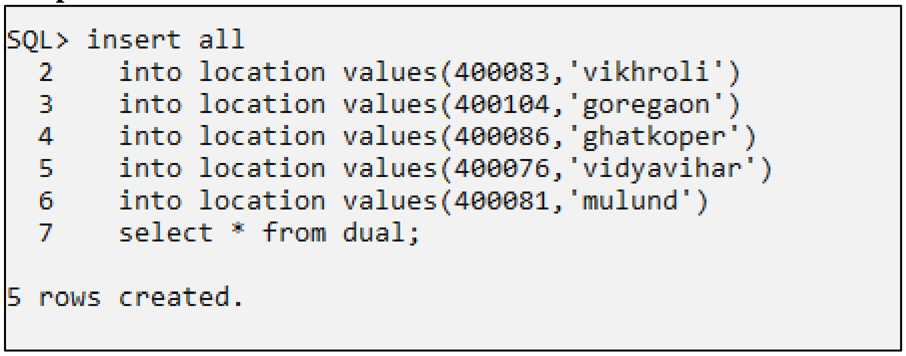
```
SQL> insert all
  2    into department values(501,'IT',400083)
  3    into department values(502,'Finance',400104)
  4    into department values(503,'Marketing',400086)
  5    into department values(504,'IT',400076)
  6    into department values(505,'IT',400081)
  7    select * from dual;

5 rows created.
```

Code:

```
insert all
  into location values(400083,'vikhroli')
  into location values(400104,'goregaon')
  into location values(400086,'ghatkoper')
  into location values(400076,'vidyavihar')
  into location values(400081,'mulund')
```

select * from dual;

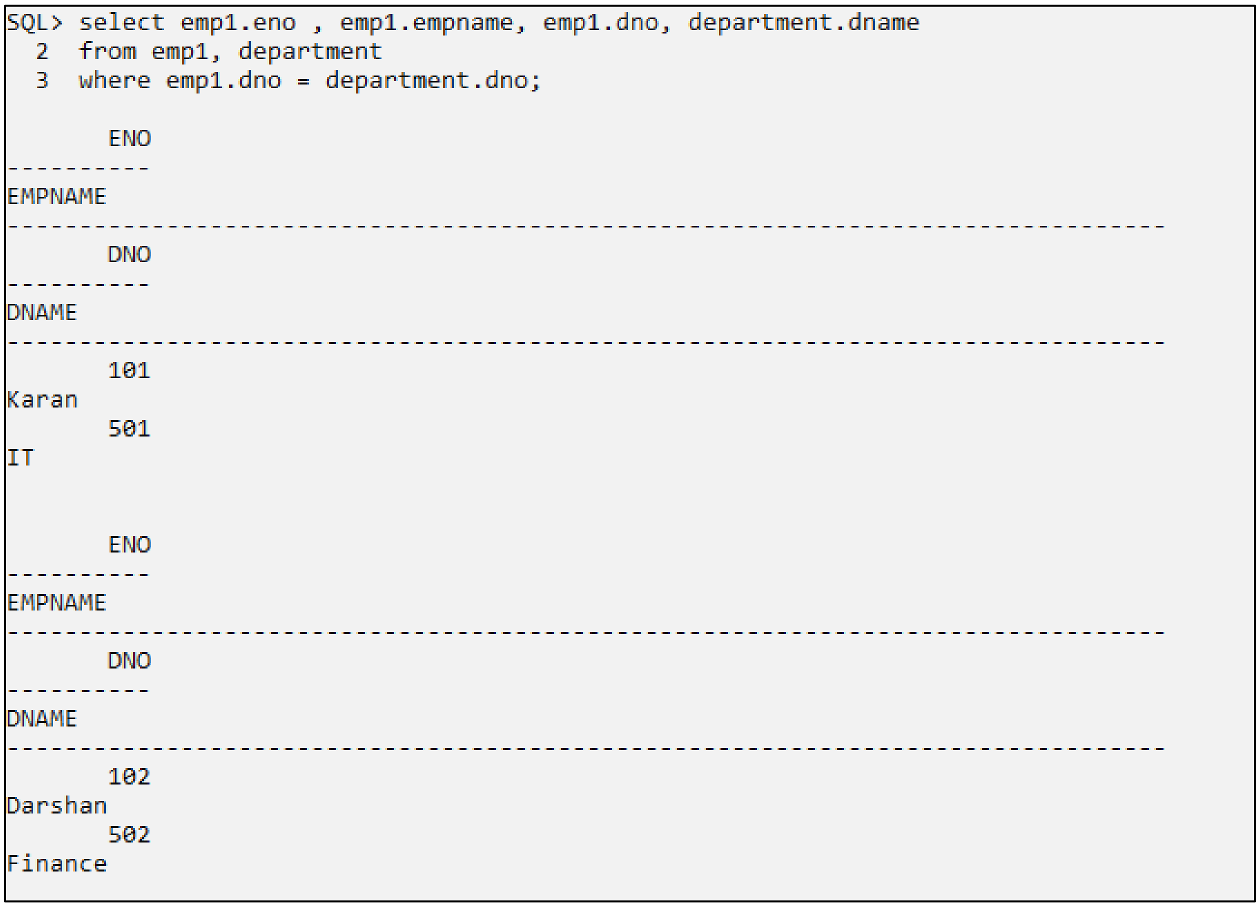
Output:



```
SQL> insert all
  2    into location values(400083,'vikhroli')
  3    into location values(400104,'goregaon')
  4    into location values(400086,'ghatkoper')
  5    into location values(400076,'vidyavihar')
  6    into location values(400081,'mulund')
  7    select * from dual;

5 rows created.
```

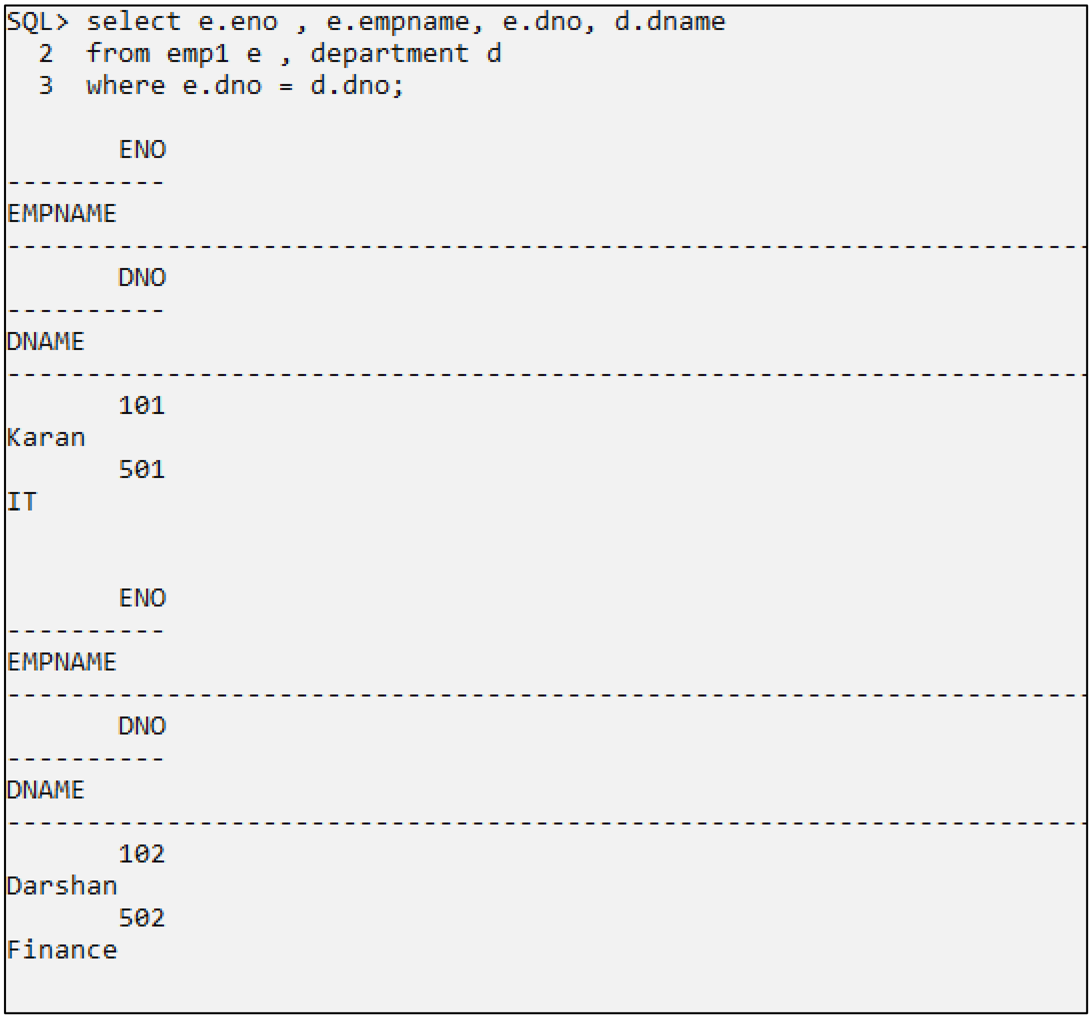
a. Equijoins

Code:
select emp1.eno , emp1.empname, emp1.dno, department.dname
from emp1, department
where emp1.dno = department.dno;
Output:


```
SQL> select emp1.eno , emp1.empname, emp1.dno, department.dname
  2  from emp1, department
  3  where emp1.dno = department.dno;

       ENO
----------
EMPNAME
--------------------------------------------------------------------------------
       DNO
----------
DNAME
--------------------------------------------------------------------------------
       101
Karan
       501
IT


       ENO
----------
EMPNAME
--------------------------------------------------------------------------------
       DNO
----------
DNAME
--------------------------------------------------------------------------------
       102
Darshan
       502
Finance
```
Code:

select e.eno , e.empname, e.dno, d.dname
from emp1 e , department d
where e.dno = d.dno;

Output:

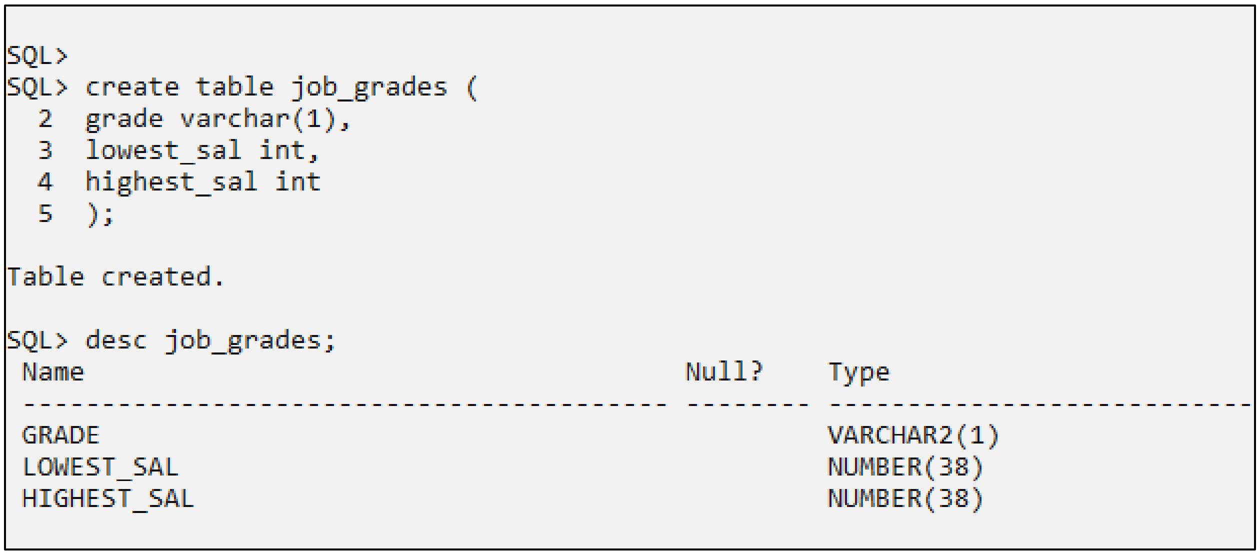


```
SQL> select e.eno , e.empname, e.dno, d.dname
  2  from emp1 e , department d
  3  where e.dno = d.dno;

       ENO
----------
EMPNAME
--------------------------------------------------------------------
       DNO
----------
DNAME
--------------------------------------------------------------------
       101
Karan
       501
IT


       ENO
----------
EMPNAME
--------------------------------------------------------------------
       DNO
----------
DNAME
--------------------------------------------------------------------
       102
Darshan
       502
Finance
```

Code:

```
 create table job_grades (
 grade varchar(1),
 lowest_sal int,
 highest_sal int );
desc job_grades;
```

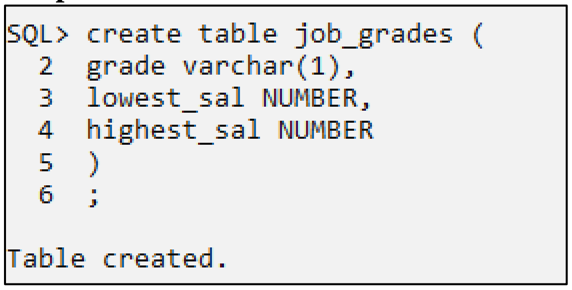
Output:


```
SQL>
SQL> create table job_grades (
  2  grade varchar(1),
  3  lowest_sal int,
  4  highest_sal int
  5  );

Table created.

SQL> desc job_grades;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 GRADE                                              VARCHAR2(1)
 LOWEST_SAL                                         NUMBER(38)
 HIGHEST_SAL                                        NUMBER(38)
```

Code:

create table job_grades (
grade varchar(1),
lowest_sal NUMBER,
highest_sal NUMBER
);

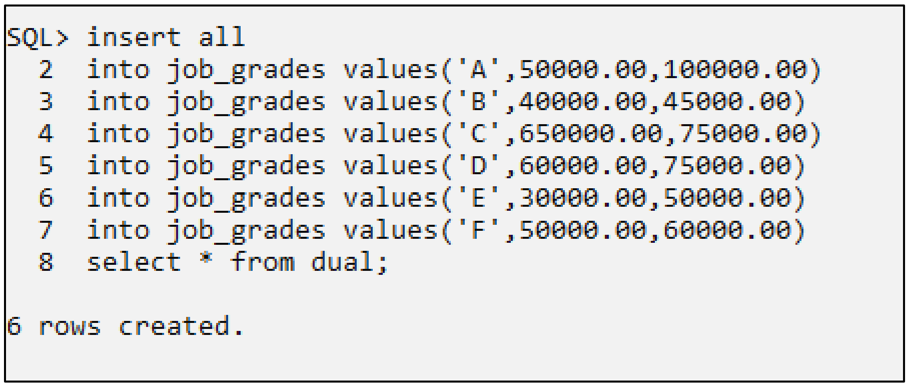
Output:


```
SQL> create table job_grades (
  2  grade varchar(1),
  3  lowest_sal NUMBER,
  4  highest_sal NUMBER
  5  )
  6  ;

Table created.
```

Code:

insert all
into job_grades values('A',50000.00,100000.00)
into job_grades values('B',40000.00,45000.00)
into job_grades values('C',650000.00,75000.00)
into job_grades values('D',60000.00,75000.00)
into job_grades values('E',30000.00,50000.00)
into job_grades values('F',50000.00,60000.00)
select * from dual;

Output:

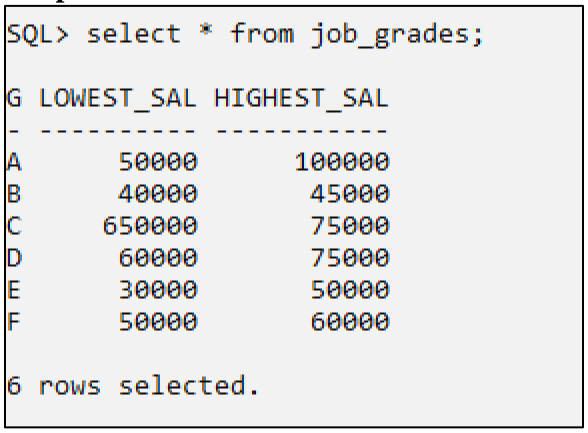


```
SQL> insert all
  2  into job_grades values('A',50000.00,100000.00)
  3  into job_grades values('B',40000.00,45000.00)
  4  into job_grades values('C',650000.00,75000.00)
  5  into job_grades values('D',60000.00,75000.00)
  6  into job_grades values('E',30000.00,50000.00)
  7  into job_grades values('F',50000.00,60000.00)
  8  select * from dual;

6 rows created.
```

Code:

Desc job_grades;

Output:

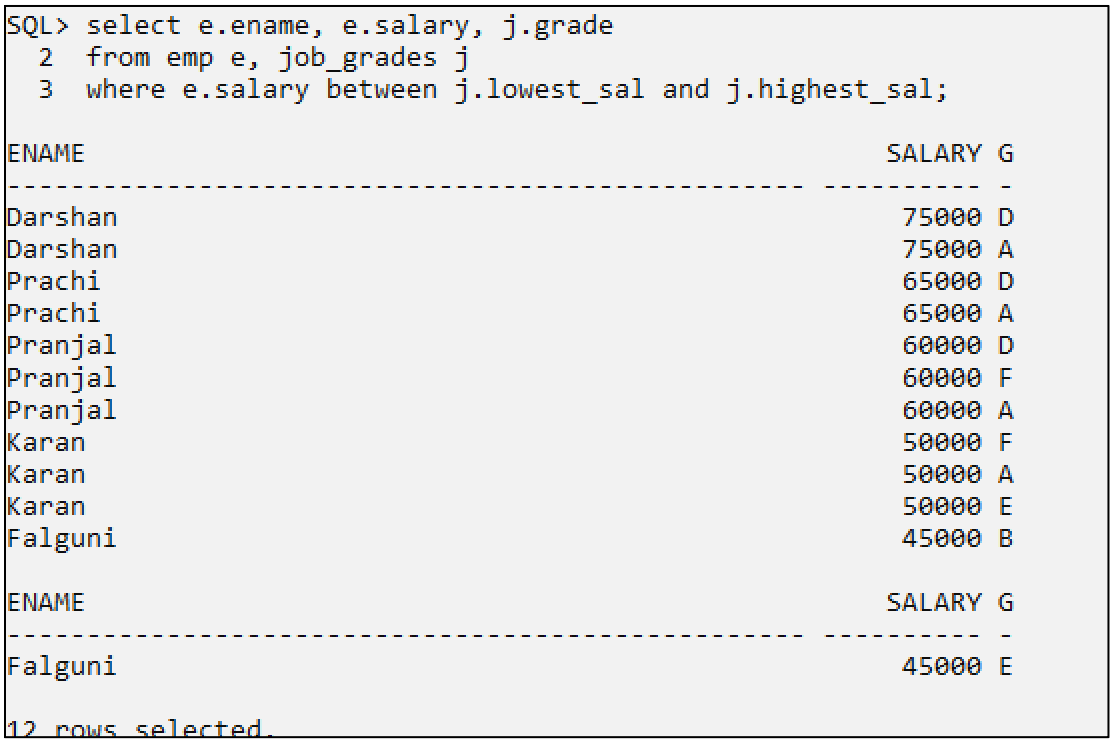


```
SQL> select * from job_grades;

G LOWEST_SAL HIGHEST_SAL
- ---------- -----------
A      50000      100000
B      40000       45000
C     650000       75000
D      60000       75000
E      30000       50000
F      50000       60000

6 rows selected.
```

Code:

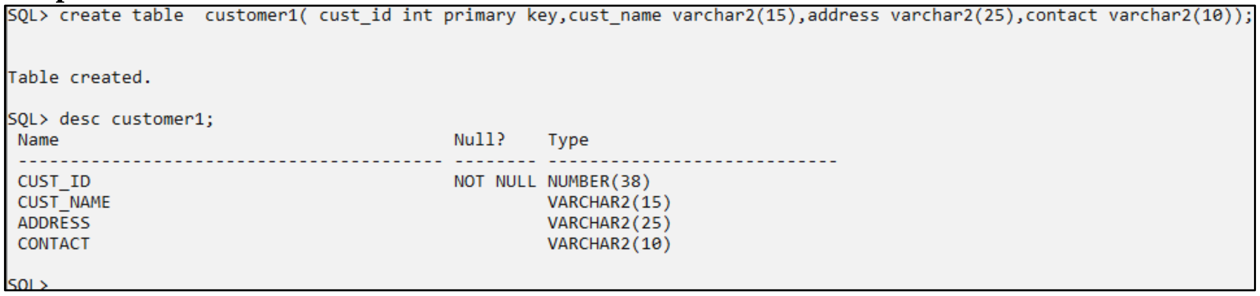
select e.ename, e.salary, j.grade
from emp e, job_grades j
where e.salary between j.lowest_sal and j.highest_sal;

Output:



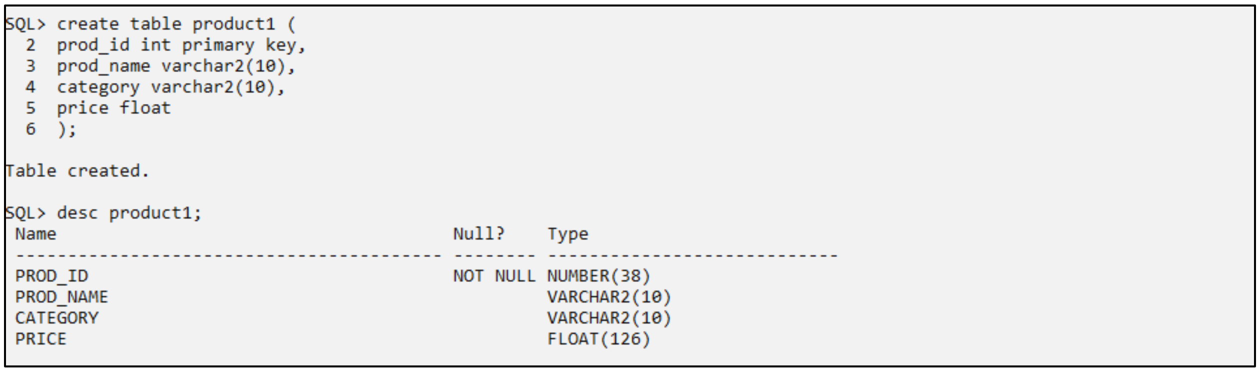
```
SQL> select e.ename, e.salary, j.grade
  2  from emp e, job_grades j
  3  where e.salary between j.lowest_sal and j.highest_sal;

ENAME                                                  SALARY G
-------------------------------------------------- ---------- -
Darshan                                                 75000 D
Darshan                                                 75000 A
Prachi                                                  65000 D
Prachi                                                  65000 A
Pranjal                                                 60000 D
Pranjal                                                 60000 F
Pranjal                                                 60000 A
Karan                                                   50000 F
Karan                                                   50000 A
Karan                                                   50000 E
Falguni                                                 45000 B

ENAME                                                  SALARY G
-------------------------------------------------- ---------- -
Falguni                                                 45000 E

12 rows selected.
```

Query : Create four table and insert data
Code:
create table customer1(cust_id int primary key
,cust_name varchar2(15)
,address varchar2(25)
,contact varchar2(10)
);
Output:



```
SQL> create table  customer1( cust_id int primary key,cust_name varchar2(15),address varchar2(25),contact varchar2(10));


Table created.

SQL> desc customer1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUST_ID                                   NOT NULL NUMBER(38)
 CUST_NAME                                          VARCHAR2(15)
 ADDRESS                                            VARCHAR2(25)
 CONTACT                                            VARCHAR2(10)

SQL>
```

Code:
create table product1 (
prod_id int primary key,
prod_name varchar2(10),
category varchar2(10),
price float
);
Output:

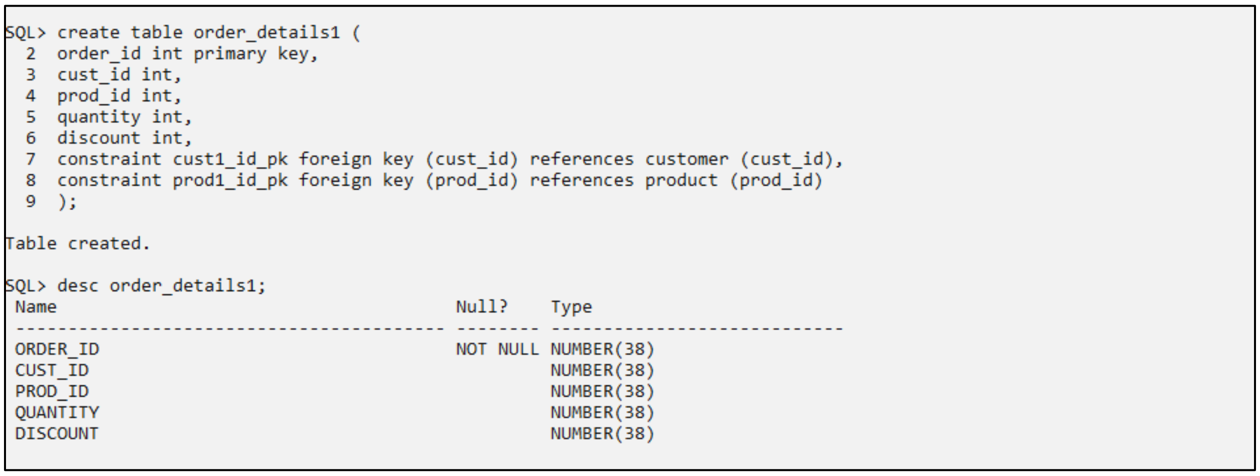


```
SQL> create table product1 (
  2  prod_id int primary key,
  3  prod_name varchar2(10),
  4  category varchar2(10),
  5  price float
  6  );

Table created.

SQL> desc product1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PROD_ID                                   NOT NULL NUMBER(38)
 PROD_NAME                                          VARCHAR2(10)
 CATEGORY                                           VARCHAR2(10)
 PRICE                                              FLOAT(126)
```

Code:

create table order_details1 (
order_id int primary key,
cust_id int,
prod_id int,
quantity int,
discount int,
constraint cust1_id_pk foreign key (cust_id) references customer (cust_id),
constraint prod1_id_pk foreign key (prod_id) references product (prod_id)
);

Output:

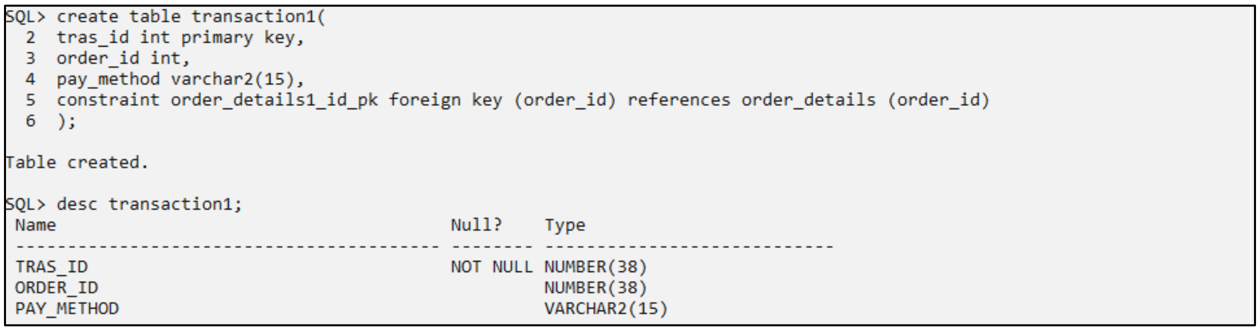


```
SQL> create table order_details1 (
  2  order_id int primary key,
  3  cust_id int,
  4  prod_id int,
  5  quantity int,
  6  discount int,
  7  constraint cust1_id_pk foreign key (cust_id) references customer (cust_id),
  8  constraint prod1_id_pk foreign key (prod_id) references product (prod_id)
  9  );

Table created.

SQL> desc order_details1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORDER_ID                                  NOT NULL NUMBER(38)
 CUST_ID                                            NUMBER(38)
 PROD_ID                                            NUMBER(38)
 QUANTITY                                           NUMBER(38)
 DISCOUNT                                           NUMBER(38)
```

Code:

create table transaction1(
tras_id int primary key,
order_id int,
pay_method varchar2(15),
constraint order_details1_id_pk foreign key (order_id) references order_details (order_id)
);

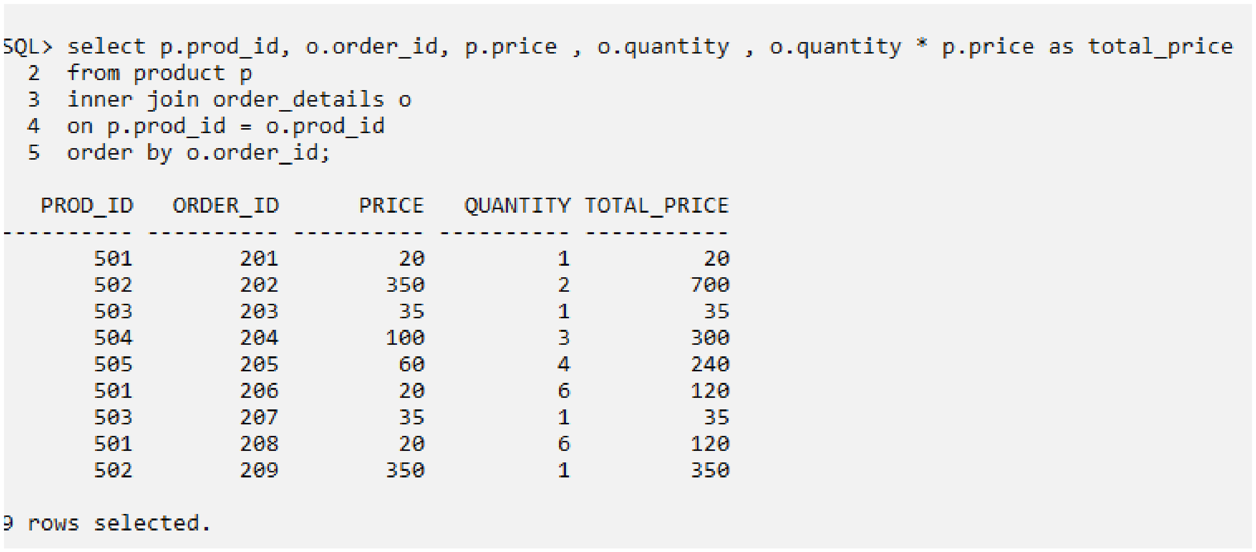
Output:


```
SQL> create table transaction1(
  2  tras_id int primary key,
  3  order_id int,
  4  pay_method varchar2(15),
  5  constraint order_details1_id_pk foreign key (order_id) references order_details (order_id)
  6  );

Table created.

SQL> desc transaction1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 TRAS_ID                                   NOT NULL NUMBER(38)
 ORDER_ID                                           NUMBER(38)
 PAY_METHOD                                         VARCHAR2(15)
```

b. Non-Equi Joins
c. Joining Three Tables

Select col_name from table1 join table 2
On table1.column_name = table2.column_name
OR
Select col_name
From table1
Inner join table2
On table1.column_name = table2.column_name

Query: write a query to list the order details Product name, unit price , quantity & price
Code:select p.prod_id, o.order_id, p.price , o.quantity , o.quantity * p.price as total_price
from product p
inner join order_details o
on p.prod_id = o.prod_id
order by o.order_id;
Output:

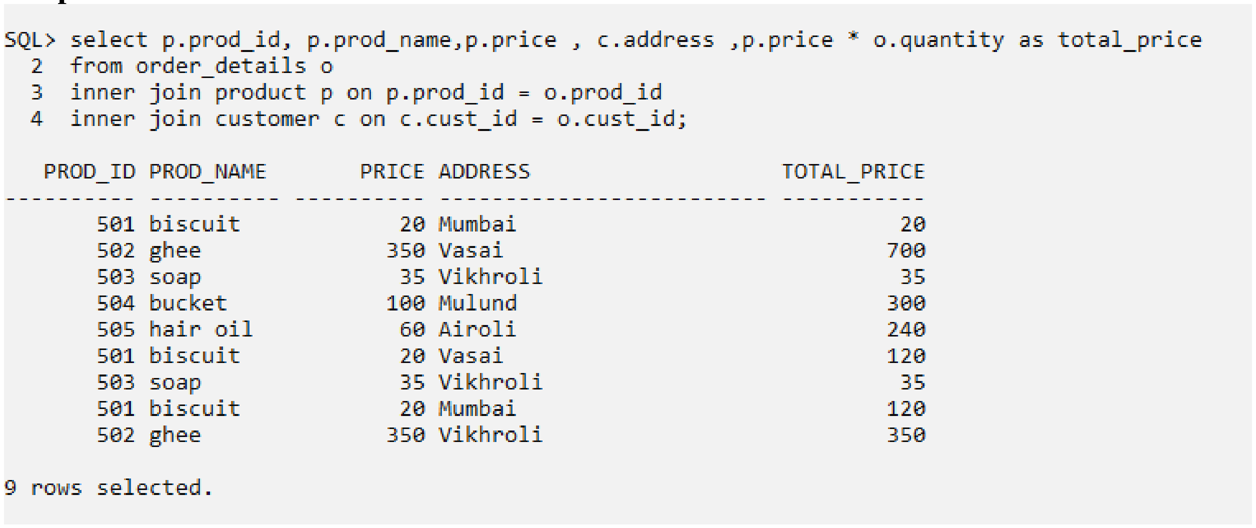


```
SQL> select p.prod_id, o.order_id, p.price , o.quantity , o.quantity * p.price as total_price
     from product p
     inner join order_details o
     on p.prod_id = o.prod_id
     order by o.order_id;

   PROD_ID   ORDER_ID      PRICE   QUANTITY TOTAL_PRICE
---------- ---------- ---------- ---------- -----------
       501        201         20          1          20
       502        202        350          2         700
       503        203         35          1          35
       504        204        100          3         300
       505        205         60          4         240
       501        206         20          6         120
       503        207         35          1          35
       501        208         20          6         120
       502        209        350          1         350

9 rows selected.
```

Query: List productid , unit price and address

select p.prod_id, p.prod_name,p.price , c.cust_name ,c.address ,p.price * o.quantity as total_price
from order_details o
inner join product p on p.prod_id = o.prod_id
inner join customer c on c.cust_id = o.cust_id;

Output:

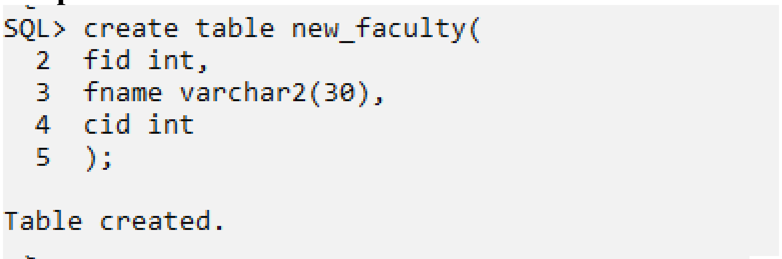


```
SQL> select p.prod_id, p.prod_name,p.price , c.address ,p.price * o.quantity as total_price
     from order_details o
     inner join product p on p.prod_id = o.prod_id
     inner join customer c on c.cust_id = o.cust_id;

   PROD_ID PROD_NAME       PRICE ADDRESS                   TOTAL_PRICE
---------- ---------- ---------- ------------------------- -----------
       501 biscuit            20 Mumbai                             20
       502 ghee              350 Vasai                             700
       503 soap               35 Vikhroli                           35
       504 bucket            100 Mulund                            300
       505 hair oil           60 Airoli                            240
       501 biscuit            20 Vasai                             120
       503 soap               35 Vikhroli                           35
       501 biscuit            20 Mumbai                            120
       502 ghee              350 Vikhroli                          350

9 rows selected.
```

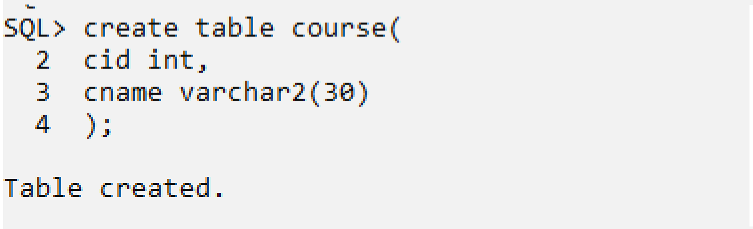
Query: Create 2 tables and insert records
create table new_faculty(
fid int,
fname varchar2(30),
cid int
);

```
create table course(
cid int,
cname varchar2(30)
);
insert all
into new_faculty values(1,'DST',101)
into new_faculty values(2,'IT',102)
into new_faculty values(3,'Finance',103)
select * from dual;
insert all
into course values(101,'Web Tech')
into course values(102,'Python')
select * from dual;
```

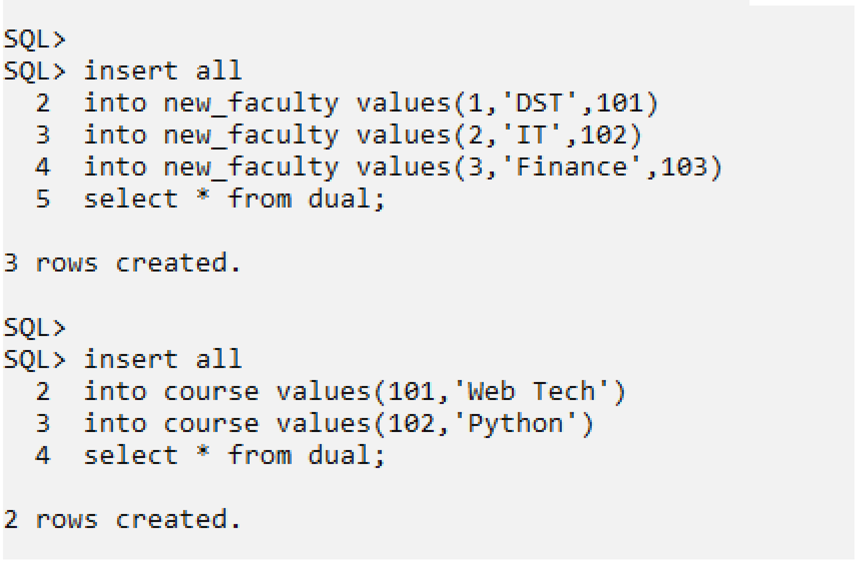
Output:



```
SQL> create table new_faculty(
  2  fid int,
  3  fname varchar2(30),
  4  cid int
  5  );

Table created.
```



```
SQL> create table course(
  2  cid int,
  3  cname varchar2(30)
  4  );

Table created.
```

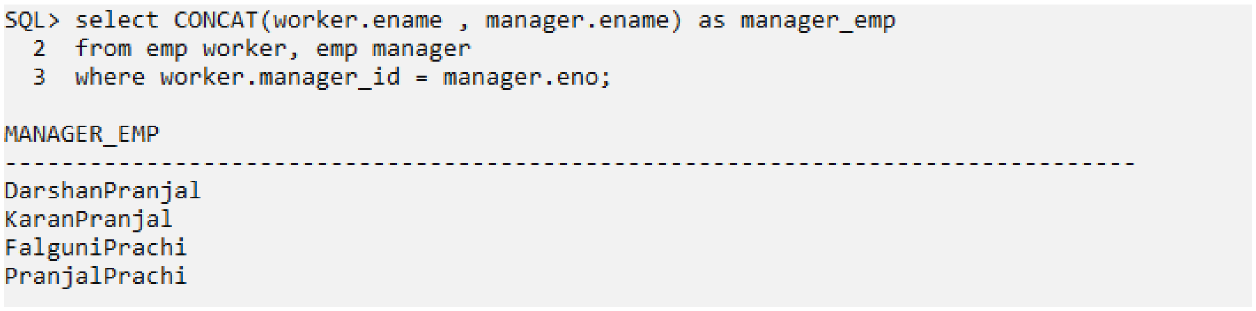


```
SQL>
SQL> insert all
  2  into new_faculty values(1,'DST',101)
  3  into new_faculty values(2,'IT',102)
  4  into new_faculty values(3,'Finance',103)
  5  select * from dual;

3 rows created.

SQL>
SQL> insert all
  2  into course values(101,'Web Tech')
  3  into course values(102,'Python')
  4  select * from dual;

2 rows created.
```

d. Self Joins

SELF JOIN

select CONCAT(worker.ename , manager.ename) as manager_emp
from emp worker, emp manager
where worker.manager_id = manager.eno;

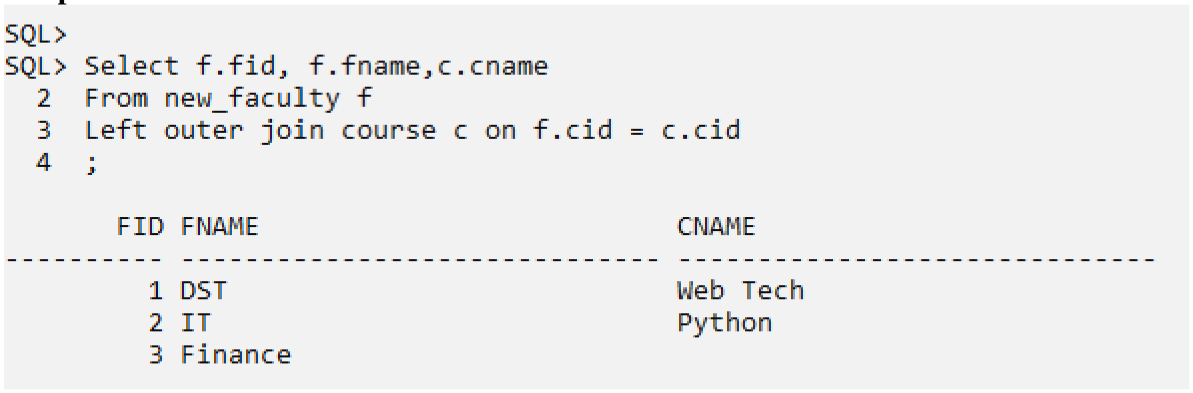


```
SQL> select CONCAT(worker.ename , manager.ename) as manager_emp
  2  from emp worker, emp manager
  3  where worker.manager_id = manager.eno;

MANAGER_EMP
--------------------------------------------------------------------------------
DarshanPranjal
KaranPranjal
FalguniPrachi
PranjalPrachi
```

e. Left Outer Joins

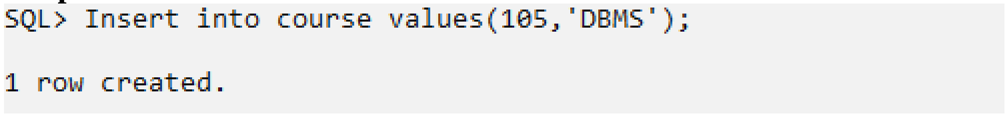
Syntax
select col_name
from table1
left join table 2
On table1.col = table2.col;

Select f.fid, f.fname,c.cname
From new_faculty f
Left outer join course c on f.cid = c.cid ;
Output:



```
SQL>
SQL> Select f.fid, f.fname,c.cname
  2  From new_faculty f
  3  Left outer join course c on f.cid = c.cid
  4  ;

       FID FNAME                          CNAME
---------- ------------------------------ ------------------------------
         1 DST                            Web Tech
         2 IT                             Python
         3 Finance
```

Query: Add One record to course
Insert into course values(105,'DBMS');
Output:

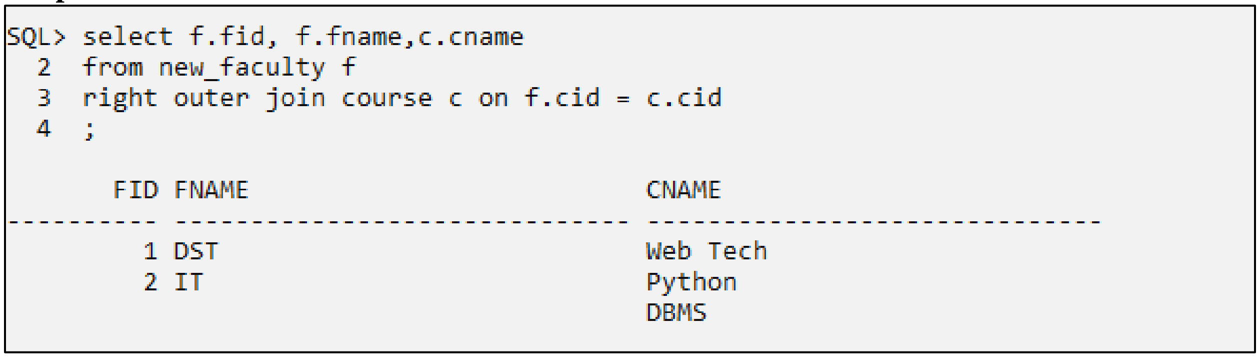


```
SQL> Insert into course values(105,'DBMS');

1 row created.
```

f. Right Outer Joins

Syntax
select col_name
from table1
Right join table 2 or right outer join table 2
On table1.col = table2.col;

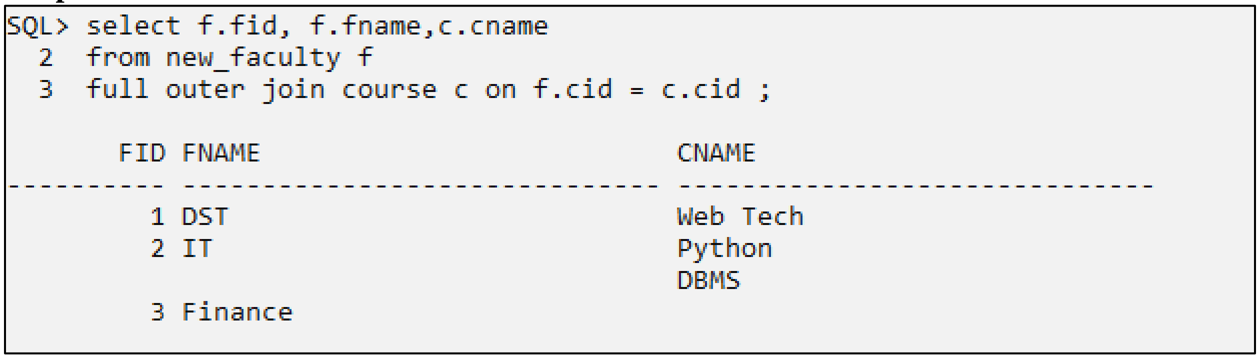
Code:
select f.fid, f.fname,c.cname
from new_faculty f
right outer join course c on f.cid = c.cid ;
Output:



```
SQL> select f.fid, f.fname,c.cname
  2  from new_faculty f
  3  right outer join course c on f.cid = c.cid
  4  ;

       FID FNAME                          CNAME
---------- ------------------------------ ------------------------------
         1 DST                            Web Tech
         2 IT                             Python
                                          DBMS
```

g. Full Outer Joins

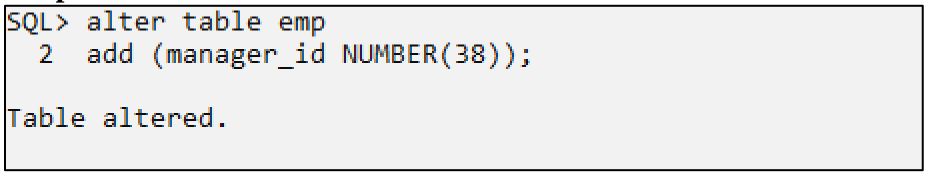
Code:
select f.fid, f.fname,c.cname
from new_faculty f
full outer join course c on f.cid = c.cid ;
Output:



```
SQL> select f.fid, f.fname,c.cname
  2  from new_faculty f
  3  full outer join course c on f.cid = c.cid ;

       FID FNAME                          CNAME
---------- ------------------------------ ------------------------------
         1 DST                            Web Tech
         2 IT                             Python
                                          DBMS

         3 Finance
```

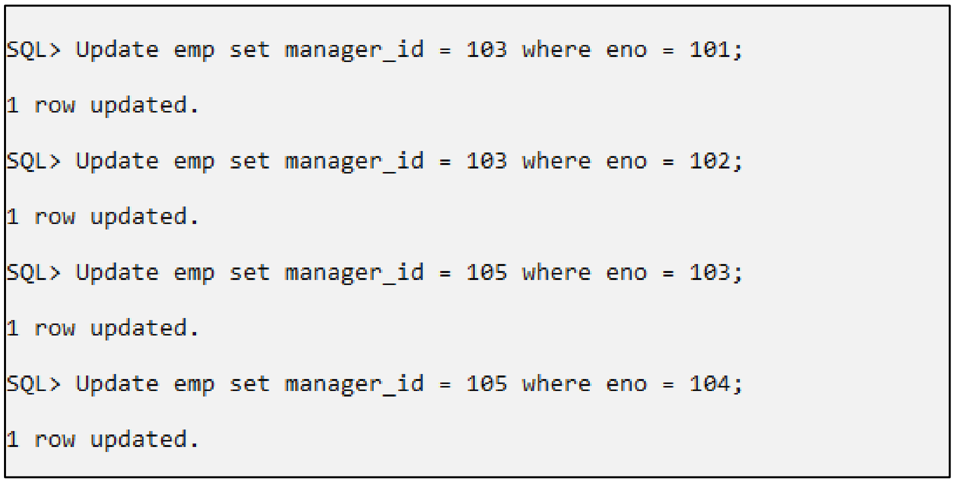
Query: Add column manager id to emp details
Code:
alter table emp add column(manager_id NUMBER(38));

Output:



```
SQL> alter table emp
  2  add (manager_id NUMBER(38));

Table altered.
```

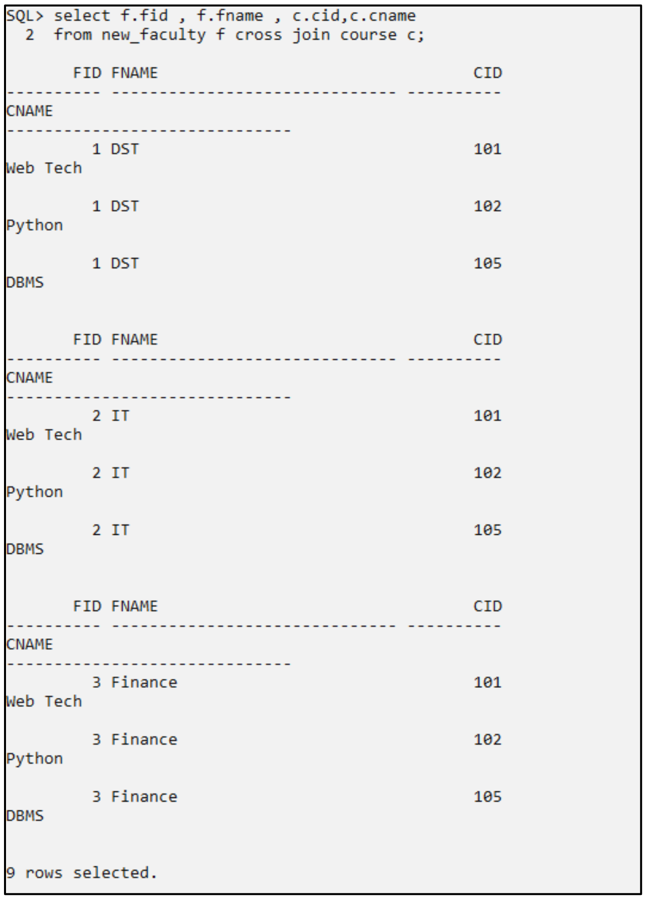
Query: Update manager id in employee table
Code:
Update emp set manager_id = 103 where eno = 101;
Update emp set manager_id = 103 where eno = 102;
Update emp set manager_id = 105 where eno = 103;
Update emp set manager_id = 105 where eno = 104;



```
SQL> Update emp set manager_id = 103 where eno = 101;

1 row updated.

SQL> Update emp set manager_id = 103 where eno = 102;

1 row updated.

SQL> Update emp set manager_id = 105 where eno = 103;

1 row updated.

SQL> Update emp set manager_id = 105 where eno = 104;

1 row updated.
```

h. Cross Joins

Code:
select f.fid , f.fname , c.cid,c.cname
from new_faculty f cross join course c;
Output:



```
SQL> select f.fid , f.fname , c.cid,c.cname
  2  from new_faculty f cross join course c;

       FID FNAME                                 CID
---------- ----------------------------- ----------
CNAME
-----------------------------
         1 DST                                   101
Web Tech

         1 DST                                   102
Python

         1 DST                                   105
DBMS


       FID FNAME                                 CID
---------- ----------------------------- ----------
CNAME
-----------------------------
         2 IT                                    101
Web Tech

         2 IT                                    102
Python

         2 IT                                    105
DBMS


       FID FNAME                                 CID
---------- ----------------------------- ----------
CNAME
-----------------------------
         3 Finance                               101
Web Tech

         3 Finance                               102
Python

         3 Finance                               105
DBMS


9 rows selected.
```

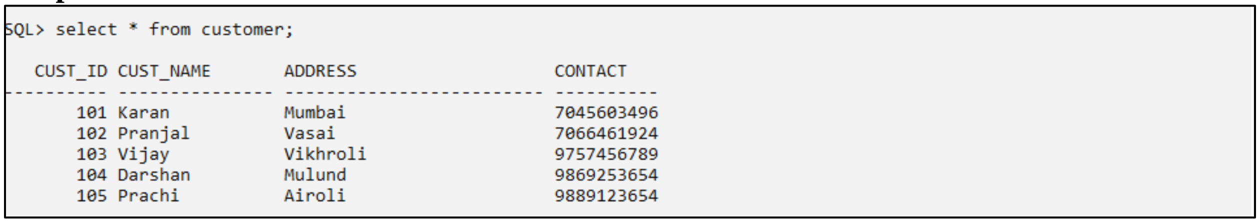
Exercise

Query: Fetch all details from all four tables

Code:

Select * from customer;

Output:

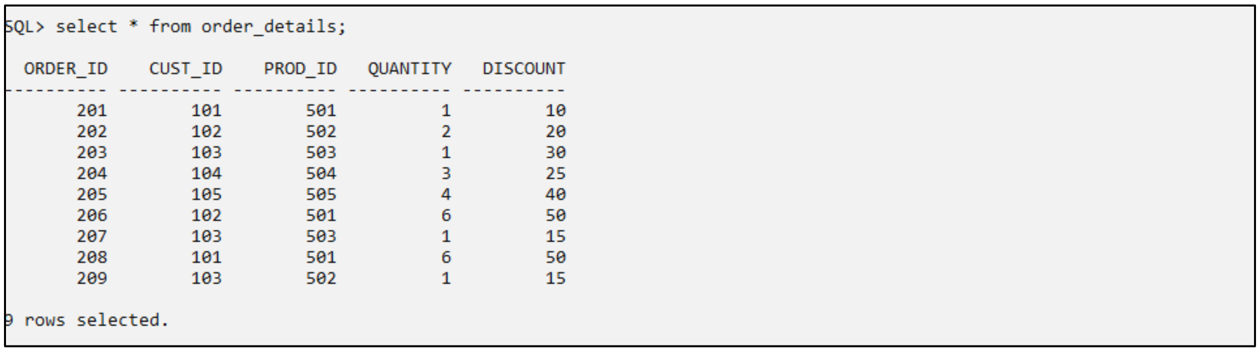


```
SQL> select * from customer;

   CUST_ID CUST_NAME       ADDRESS                   CONTACT
---------- --------------- ------------------------- ----------
       101 Karan           Mumbai                    7045603496
       102 Pranjal         Vasai                     7066461924
       103 Vijay           Vikhroli                  9757456789
       104 Darshan         Mulund                    9869253654
       105 Prachi          Airoli                    9889123654
```

Code:

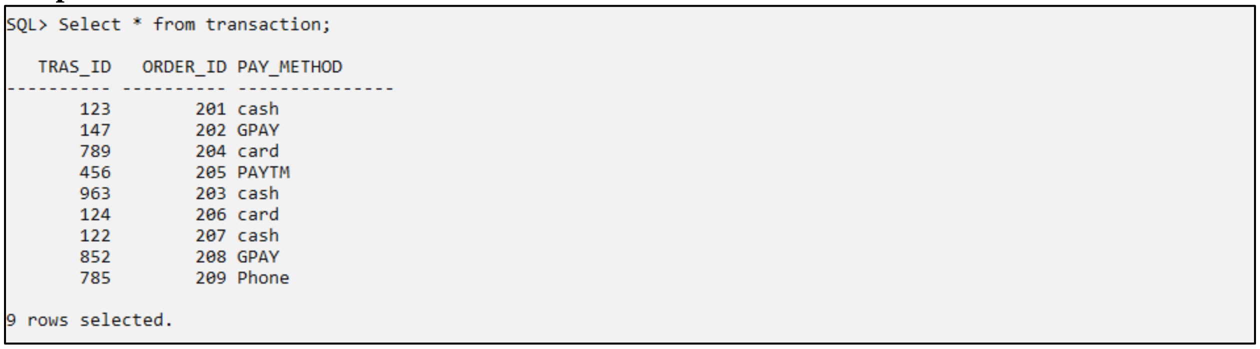
Select * from order_details;

Output:



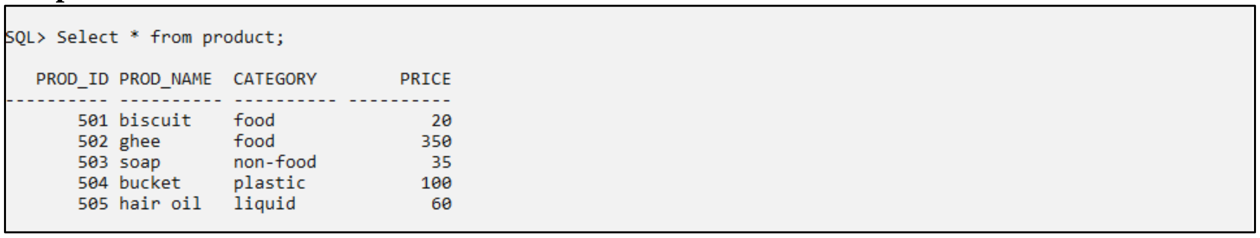
```
SQL> select * from order_details;

  ORDER_ID    CUST_ID    PROD_ID   QUANTITY   DISCOUNT
---------- ---------- ---------- ---------- ----------
       201        101        501          1         10
       202        102        502          2         20
       203        103        503          1         30
       204        104        504          3         25
       205        105        505          4         40
       206        102        501          6         50
       207        103        503          1         15
       208        101        501          6         50
       209        103        502          1         15

9 rows selected.
```

Code:
Select * from transaction;
Output:



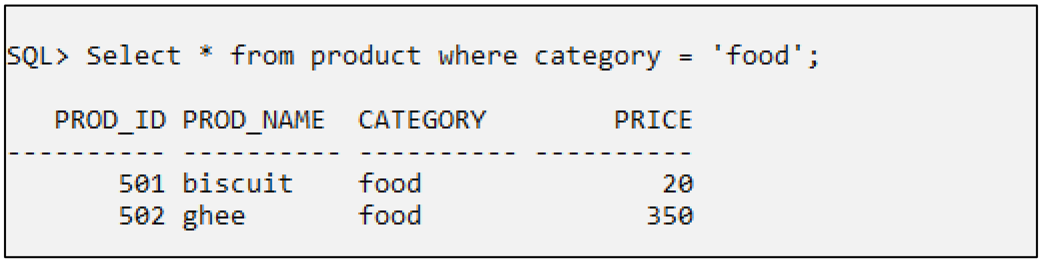
```
SQL> Select * from transaction;

   TRAS_ID   ORDER_ID PAY_METHOD
---------- ---------- ---------------
       123        201 cash
       147        202 GPAY
       789        204 card
       456        205 PAYTM
       963        203 cash
       124        206 card
       122        207 cash
       852        208 GPAY
       785        209 Phone

9 rows selected.
```

Code:
Select * from products;
Output:



```
SQL> Select * from product;

   PROD_ID PROD_NAME  CATEGORY        PRICE
---------- ---------- ---------- ----------
       501 biscuit    food               20
       502 ghee       food              350
       503 soap       non-food           35
       504 bucket     plastic           100
       505 hair oil   liquid             60
```

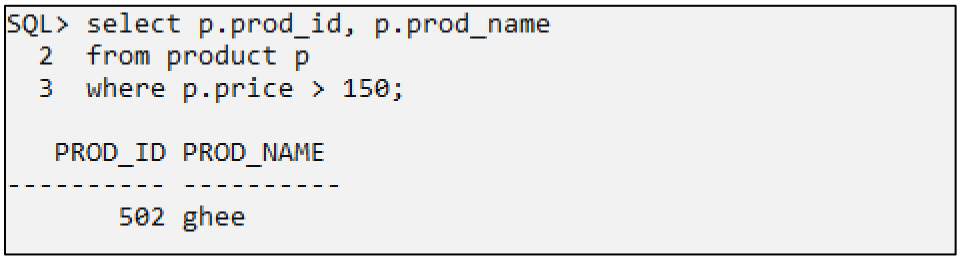
Query: Get product details in food category
Code:
select * from product where category = 'food';
Output:



```
SQL> Select * from product where category = 'food';

   PROD_ID PROD_NAME  CATEGORY        PRICE
---------- ---------- ---------- ----------
       501 biscuit    food               20
       502 ghee       food              350
```

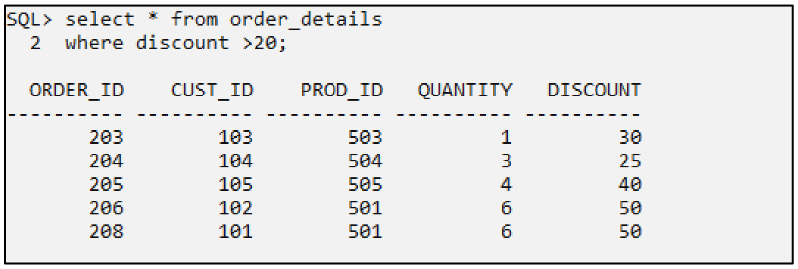
Query: Get id and name of product where price is greater than 150

Code:
select p.prod_id, p.prod_name
from product p
where p.price > 150;
Output:



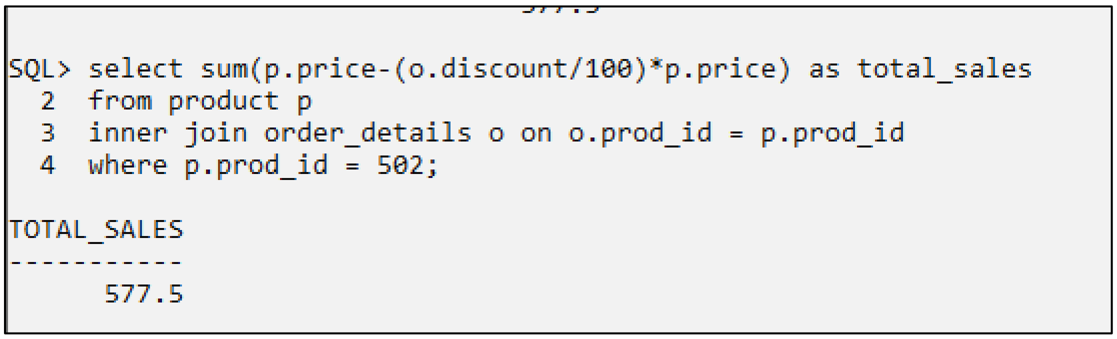
```
SQL> select p.prod_id, p.prod_name
  2  from product p
  3  where p.price > 150;

   PROD_ID PROD_NAME
---------- ----------
       502 ghee
```

Query: order details with discount more than 50%
Code:
select * from order_details where discount >20;
Output:



```
SQL> select * from order_details
  2  where discount >20;

  ORDER_ID    CUST_ID    PROD_ID   QUANTITY   DISCOUNT
---------- ---------- ---------- ---------- ----------
       203        103        503          1         30
       204        104        504          3         25
       205        105        505          4         40
       206        102        501          6         50
       208        101        501          6         50
```

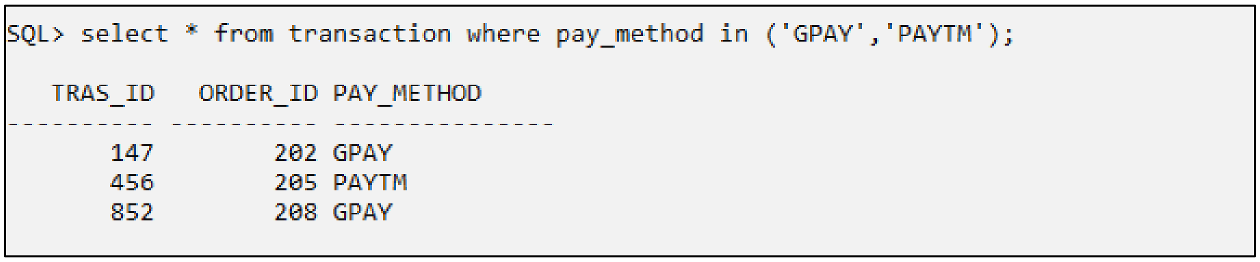
Query: get the total sales amount for product 501
Code:
select sum(p.price-(o.discount/100)*p.price)
from product p
inner join order_details o on o.prod_id = p.prod_id
where p.prod_id = 502;
Output:



```
SQL> select sum(p.price-(o.discount/100)*p.price) as total_sales
  2  from product p
  3  inner join order_details o on o.prod_id = p.prod_id
  4  where p.prod_id = 502;

TOTAL_SALES
-----------
      577.5
```

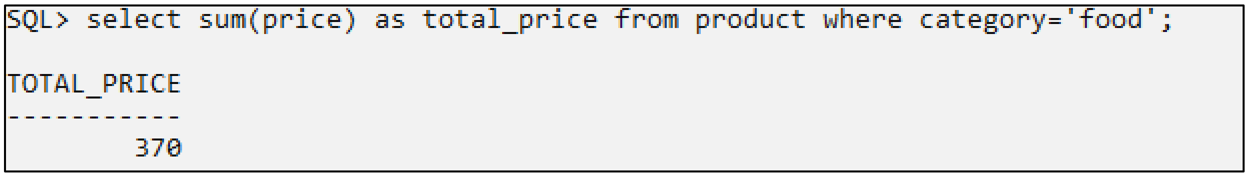
Query: Get the transaction details with UPI payment

Code:
select * from transaction where pay_method in ('GPAY','PAYTM');
Output:



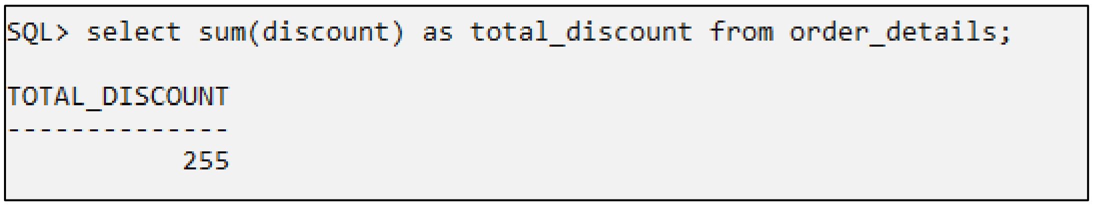
```
SQL> select * from transaction where pay_method in ('GPAY','PAYTM');

   TRAS_ID   ORDER_ID PAY_METHOD
---------- ---------- ---------------
       147        202 GPAY
       456        205 PAYTM
       852        208 GPAY
```

Query: get the total price of products in Food category
Code:
select sum(price) as total_price from product where category='food';
Output:



```
SQL> select sum(price) as total_price from product where category='food';

TOTAL_PRICE
-----------
        370
```

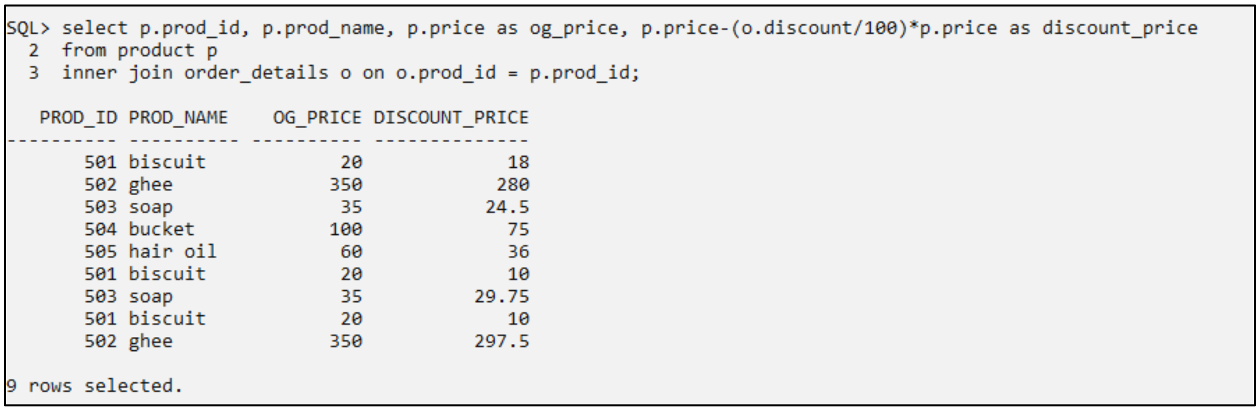
Query: get the total discount given in all orders
Code:
select sum(discount) as total_discount from order_details;
Output:



```
SQL> select sum(discount) as total_discount from order_details;

TOTAL_DISCOUNT
--------------
           255
```

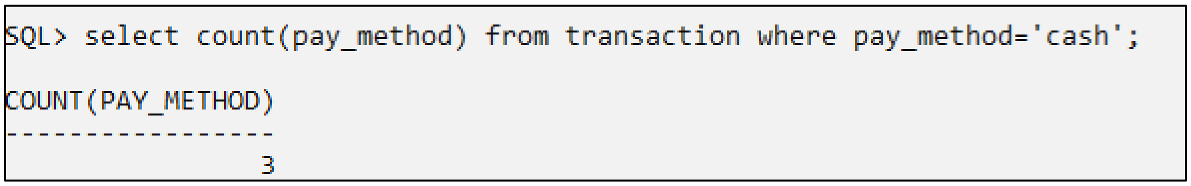
Query: get the product wise total discounted price

Code:
select p.prod_id, p.prod_name, p.price as og_price, p.price-(o.discount/100)*p.price as discount_price
from product p
inner join order_details o on o.prod_id = p.prod_id;
Output:



```
SQL> select p.prod_id, p.prod_name, p.price as og_price, p.price-(o.discount/100)*p.price as discount_price
  2  from product p
  3  inner join order_details o on o.prod_id = p.prod_id;

   PROD_ID PROD_NAME    OG_PRICE DISCOUNT_PRICE
---------- ---------- ---------- --------------
       501 biscuit            20             18
       502 ghee              350            280
       503 soap               35           24.5
       504 bucket            100             75
       505 hair oil           60             36
       501 biscuit            20             10
       503 soap               35          29.75
       501 biscuit            20             10
       502 ghee              350          297.5

9 rows selected.
```

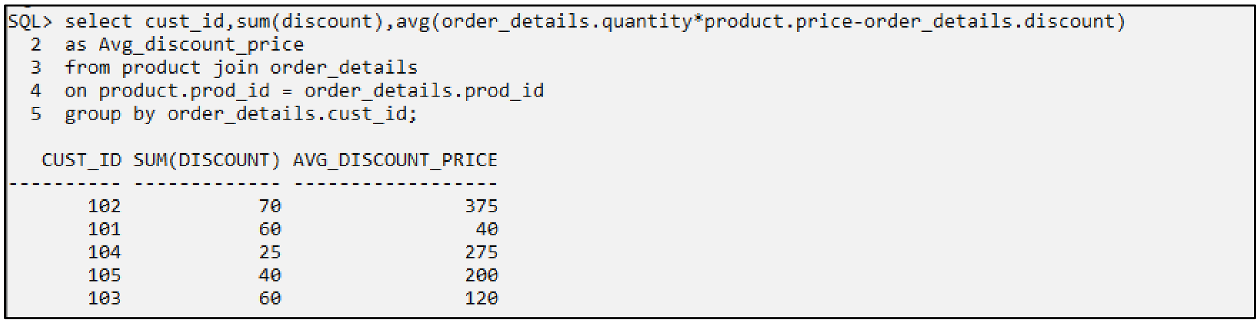
Query: How many transaction used cash payment
Code:
select count(pay_method) from transcation where pay_method='cash';
Output:



```
SQL> select count(pay_method) from transaction where pay_method='cash';

COUNT(PAY_METHOD)
-----------------
                3
```

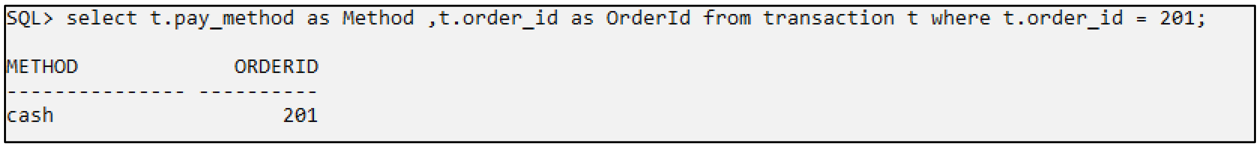
Query: Get the prod id with total sales amount exceeding 500
Code:
select p.prod_id, sum(p.price-(o.discount/100)*p.price) as ts
from product p
inner join order_details o
on o.prod_id = p.prod_id
where ts >= 500;

Query: Get the average discounted price for each customer
Code:
select cust_id,sum(discount),avg(order_details.quantity*product.price-order_details.discount) as Avg_discount_price
from product join order_details
on product.prod_id = order_details.prod_id
group by order_details.cust_id;
Output:



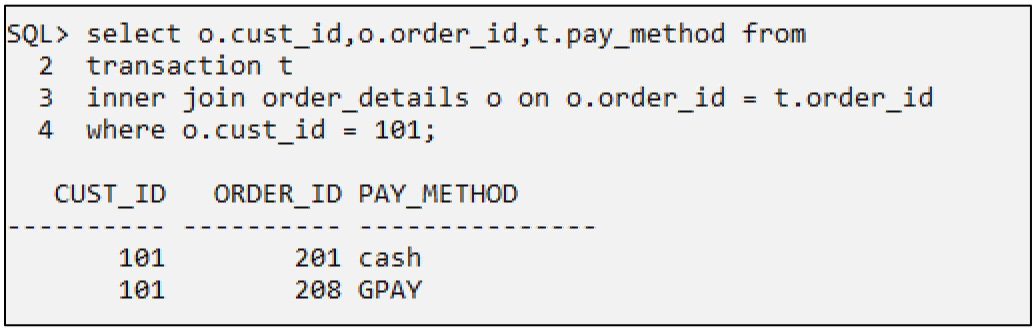
```
SQL> select cust_id,sum(discount),avg(order_details.quantity*product.price-order_details.discount)
     as Avg_discount_price
     from product join order_details
     on product.prod_id = order_details.prod_id
     group by order_details.cust_id;

   CUST_ID SUM(DISCOUNT) AVG_DISCOUNT_PRICE
---------- ------------- ------------------
       102            70                375
       101            60                 40
       104            25                275
       105            40                200
       103            60                120
```

Query: get the transaction method method for order 201
Code:
select t.pay_method as Method ,t.order_id as OrderId from transaction t where t.order_id = 201;
Output:



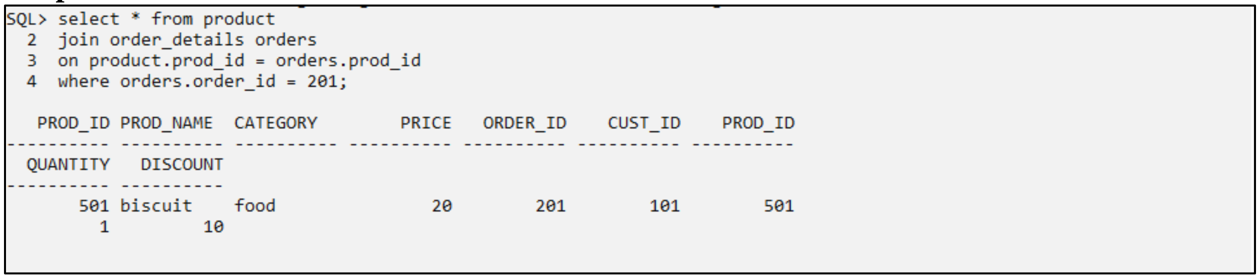
```
SQL> select t.pay_method as Method ,t.order_id as OrderId from transaction t where t.order_id = 201;

METHOD             ORDERID
--------------- ----------
cash                   201
```

Query: Get the transaction method used by customer 201
Code:
select o.cust_id,o.order_id,t.pay_method from
transaction t
inner join order_details o on o.order_id = t.order_id
where o.cust_id = 101;



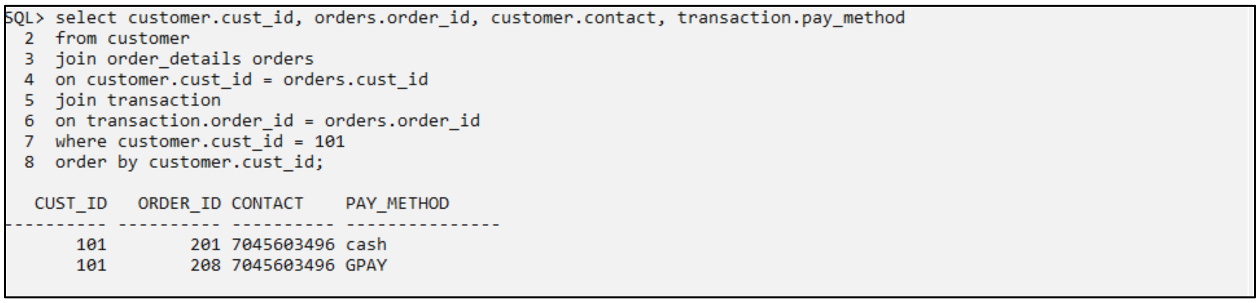
```
SQL> select o.cust_id,o.order_id,t.pay_method from
     transaction t
     inner join order_details o on o.order_id = t.order_id
     where o.cust_id = 101;

   CUST_ID   ORDER_ID PAY_METHOD
---------- ---------- ---------------
       101        201 cash
       101        208 GPAY
```

Query: Get the product details of an order id 201.
Output:



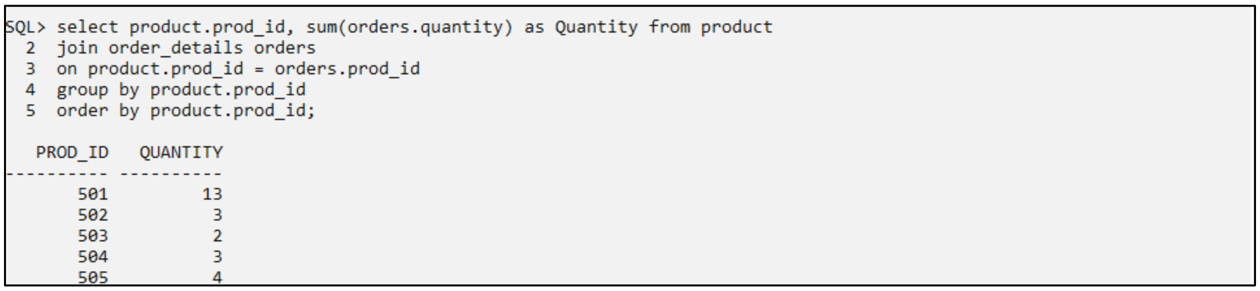
```
SQL> select * from product
     join order_details orders
     on product.prod_id = orders.prod_id
     where orders.order_id = 201;

   PROD_ID PROD_NAME  CATEGORY        PRICE   ORDER_ID    CUST_ID    PROD_ID
---------- ---------- ---------- ---------- ---------- ---------- ----------
  QUANTITY   DISCOUNT
---------- ----------
       501 biscuit    food               20        201        101        501
         1         10
```

Query: Get the customer contact and payment method for an order with id 101.
Code:
select customer.cust_id, orders.order_id, customer.contact, transaction.pay_method
from customer
join order_details orders
on customer.cust_id = orders.cust_id
join transaction
on transaction.order_id = orders.order_id
where customer.cust_id = 101
order by customer.cust_id;
Output:



```
SQL> select customer.cust_id, orders.order_id, customer.contact, transaction.pay_method
  2  from customer
  3  join order_details orders
  4  on customer.cust_id = orders.cust_id
  5  join transaction
  6  on transaction.order_id = orders.order_id
  7  where customer.cust_id = 101
  8  order by customer.cust_id;

   CUST_ID   ORDER_ID CONTACT    PAY_METHOD
---------- ---------- ---------- ---------------
       101        201 7045603496 cash
       101        208 7045603496 GPAY
```

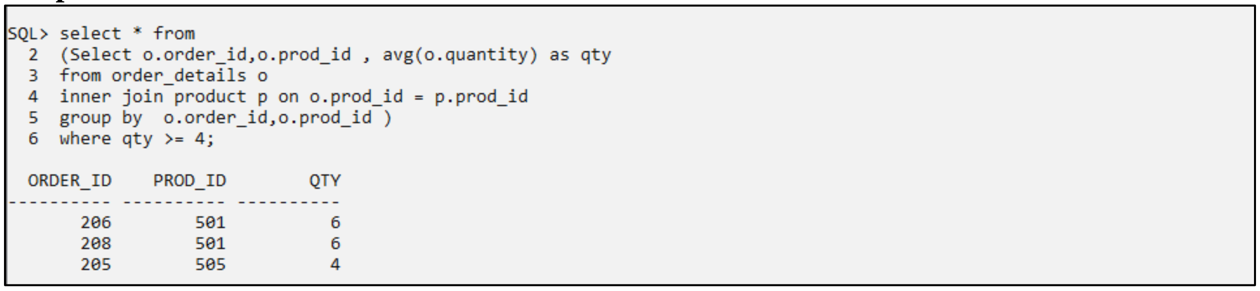
Query: Get the product wise total quantity in all orders
Code:
select product.prod_id, sum(orders.quantity) as Quantity from product
join order_details orders
on product.prod_id = orders.prod_id
group by product.prod_id
order by product.prod_id;
Output:



```
SQL> select product.prod_id, sum(orders.quantity) as Quantity from product
  2  join order_details orders
  3  on product.prod_id = orders.prod_id
  4  group by product.prod_id
  5  order by product.prod_id;

   PROD_ID   QUANTITY
---------- ----------
       501         13
       502          3
       503          2
       504          3
       505          4
```

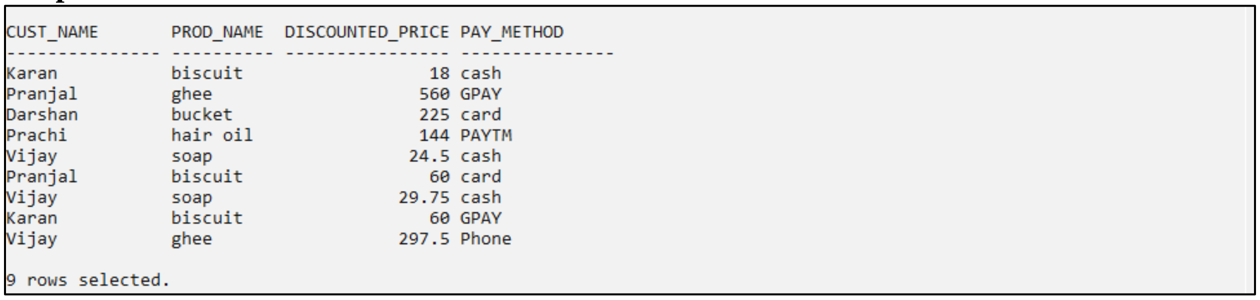
Query: Get the order id with average quantity of products more than 4.
Code:
select * from
(Select o.order_id,o.prod_id , avg(o.quantity) as qty
from order_details o
inner join product p on o.prod_id = p.prod_id
group by o.order_id,o.prod_id)

where qty >= 4;
Output:



```
SQL> select * from
  (Select o.order_id,o.prod_id , avg(o.quantity) as qty
  from order_details o
  inner join product p on o.prod_id = p.prod_id
  group by  o.order_id,o.prod_id )
  where qty >= 4;

  ORDER_ID    PROD_ID        QTY
---------- ---------- ----------
       206        501          6
       208        501          6
       205        505          4
```

Query: Get the customer name,product name,discounted price,payment_method for an order id 201.
Code:
select customer.cust_name, product.prod_name,
(orders.quantity * product.price -(orders.discount / 100)*orders.quantity * product.price) as discounted_price, pay_method from order_details orders
join customer
on orders.cust_id = customer.cust_id
join product
on orders.prod_id = product.prod_id
join transaction
on transaction.order_id = orders.order_id
where orders.order_id = 201;
Output:



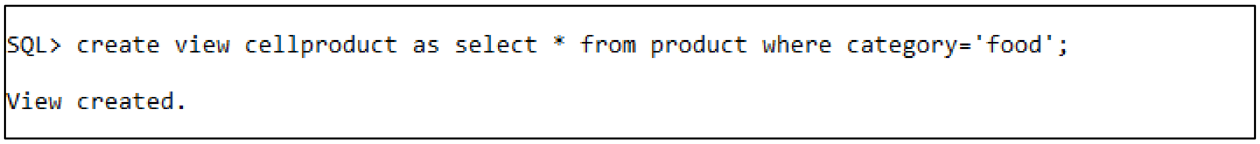
```
CUST_NAME       PROD_NAME  DISCOUNTED_PRICE PAY_METHOD
--------------- ---------- ---------------- ---------------
Karan           biscuit                  18 cash
Pranjal         ghee                    560 GPAY
Darshan         bucket                  225 card
Prachi          hair oil                144 PAYTM
Vijay           soap                   24.5 cash
Pranjal         biscuit                  60 card
Vijay           soap                  29.75 cash
Karan           biscuit                  60 GPAY
Vijay           ghee                  297.5 Phone

9 rows selected.
```

Practical 9 : Sequence, View, Index, Synonyms, Set Operations

Views:

Syntax:
Create View Syntax
CREATE [OR REPLACE] [FORCE] [NOFORCE]
VIEW <view-name>
[(column alias name...)]
AS <Query> [with] [check option]
[Read Only] [Constraint];
Code:

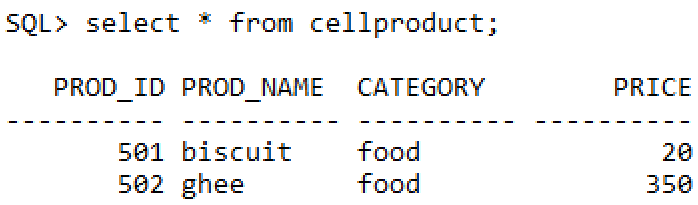
create view cellproduct as select * from product where category='food'
Output:



```
SQL> create view cellproduct as select * from product where category='food';

View created.
```

Query: Selecting from view cellproduct
Code:

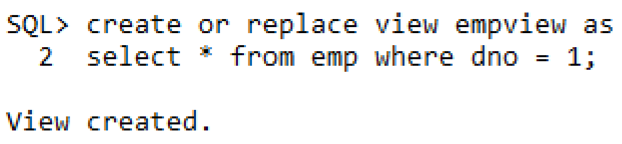
select * from cellproduct;
Output:



```
SQL> select * from cellproduct;

   PROD_ID PROD_NAME  CATEGORY        PRICE
---------- ---------- ---------- ----------
       501 biscuit    food               20
       502 ghee       food              350
```

Query:create view with specific dept no 103
Code:

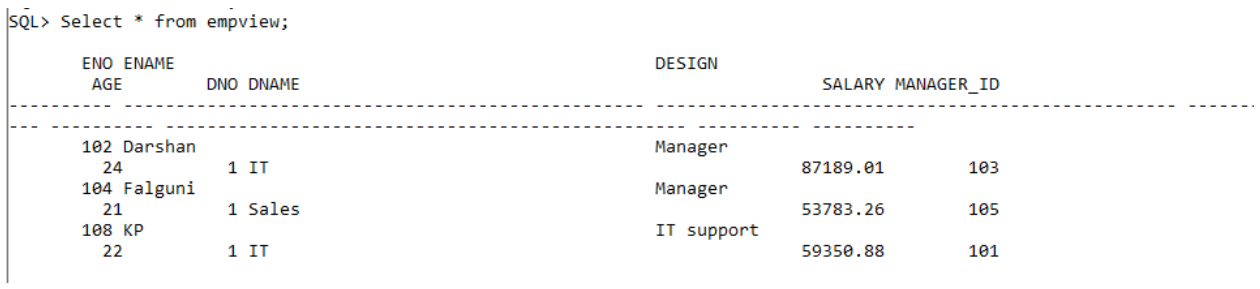
create or replace view empview as
select * from emp where dno = 1;
Output:



```
SQL> create or replace view empview as
  2  select * from emp where dno = 1;

View created.
```

Code:

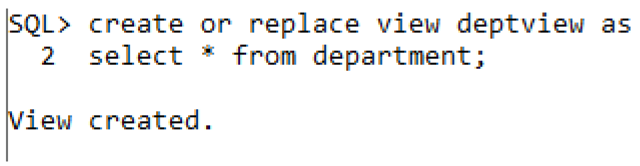
Select * from empview;
Output:



```
SQL> Select * from empview;

       ENO ENAME                                              DESIGN
        AGE        DNO DNAME                                                    SALARY MANAGER_ID
---------- -------------------------------------------------- -------------------------------------------------- ------
--- ---------- -------------------------------------------------- ---------- ----------
       102 Darshan                                            Manager
         24          1 IT                                                  87189.01        103
       104 Falguni                                            Manager
         21          1 Sales                                               53783.26        105
       108 KP                                                 IT support
         22          1 IT                                                  59350.88        101
```

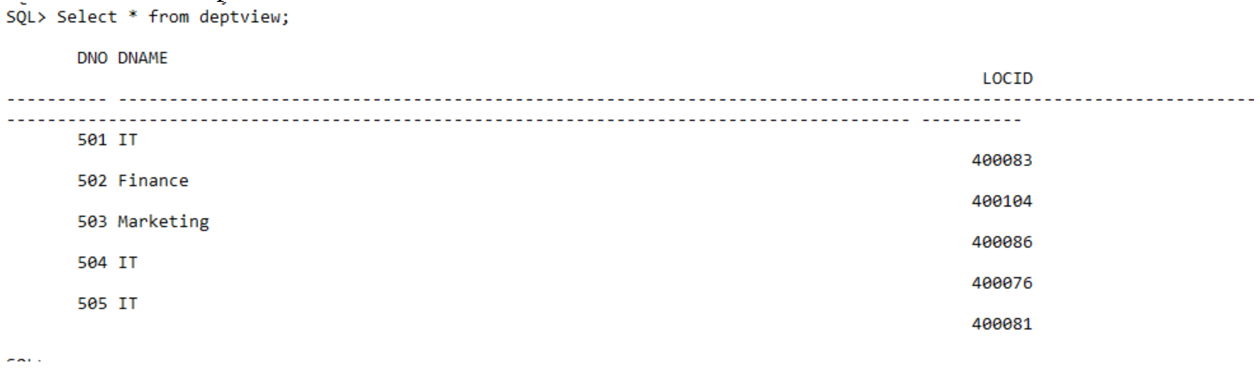
Query: create view select all from department
Code:

create or replace view deptview as
select * from department;
Output:



```
SQL> create or replace view deptview as
  2  select * from department;

View created.
```

Select * from deptview;



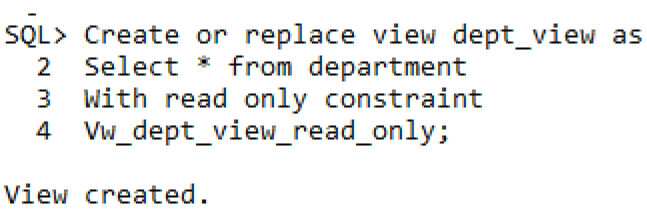
```
SQL> Select * from deptview;

       DNO DNAME
                                                                                                    LOCID
---------- ----------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------- ----------
       501 IT
                                                                                                   400083
       502 Finance
                                                                                                   400104
       503 Marketing
                                                                                                   400086
       504 IT
                                                                                                   400076
       505 IT
                                                                                                   400081
```

Query:create view with read constraint

Code:

Create or replace view dept_view as
Select * from department
With read only constraint
Vw_dept_view_read_only;

Output:

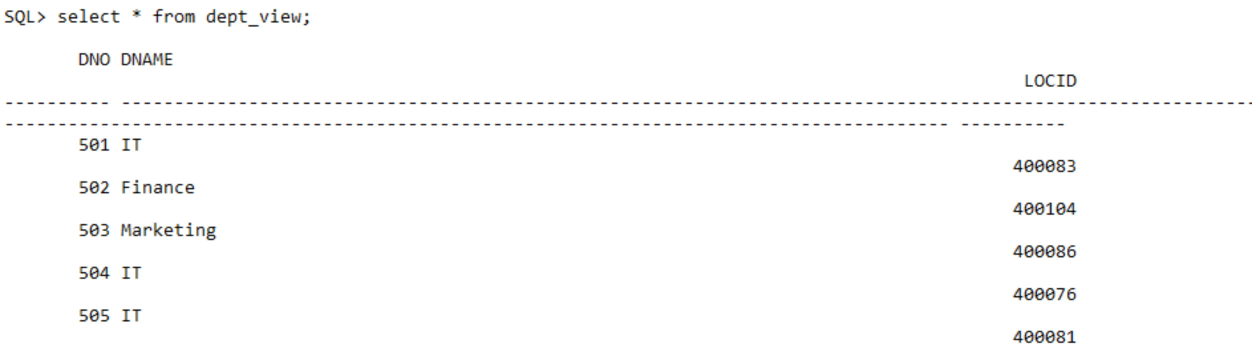


```
SQL> Create or replace view dept_view as
  2  Select * from department
  3  With read only constraint
  4  Vw_dept_view_read_only;

View created.
```

Code:

select * from dept_view;

Ouptut:



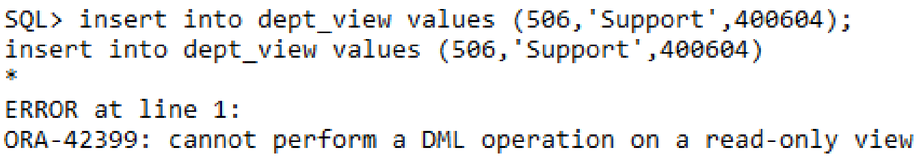
```
SQL> select * from dept_view;

       DNO DNAME
                                                                                           LOCID
---------- ----------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------- ----------
       501 IT
                                                                                          400083
       502 Finance
                                                                                          400104
       503 Marketing
                                                                                          400086
       504 IT
                                                                                          400076
       505 IT
                                                                                          400081
```

Query. Try inserting in read only view

Code:

insert into dept_view values (506,'Support',400604);

Output:



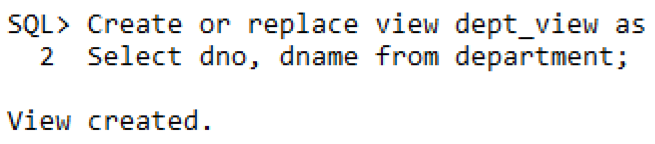
```
SQL> insert into dept_view values (506,'Support',400604);
insert into dept_view values (506,'Support',400604)
*
ERROR at line 1:
ORA-42399: cannot perform a DML operation on a read-only view
```

Query: Change View

Code:

Create or replace view dept_view as
Select dno, dname from department;

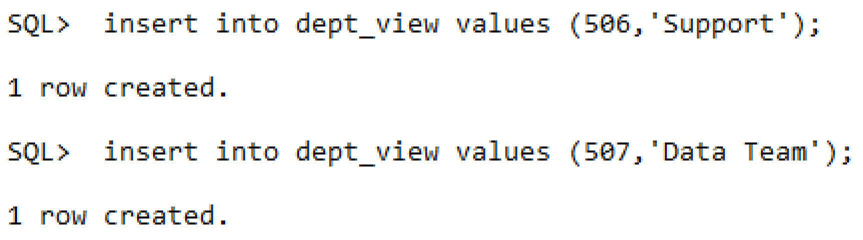
Output:



```
SQL> Create or replace view dept_view as
  2  Select dno, dname from department;

View created.
```

Code:

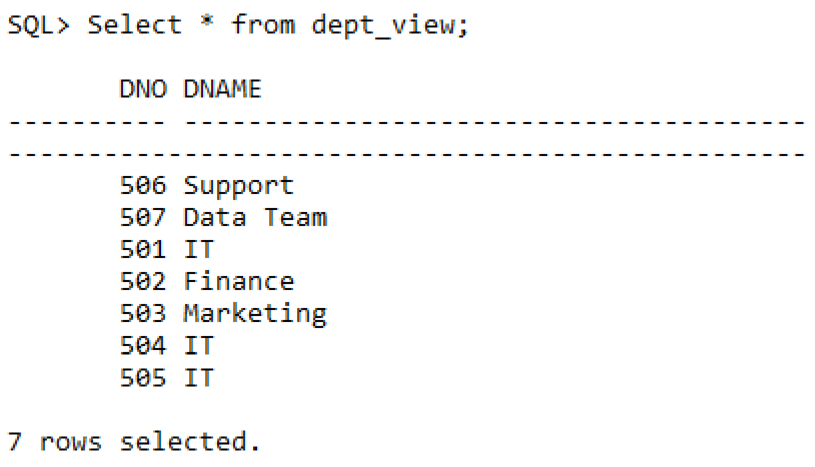
 insert into dept_view values (506,'Support');
 insert into dept_view values (507,'Data Team');
Output:



```
SQL>  insert into dept_view values (506,'Support');

1 row created.

SQL>  insert into dept_view values (507,'Data Team');

1 row created.
```

Code:

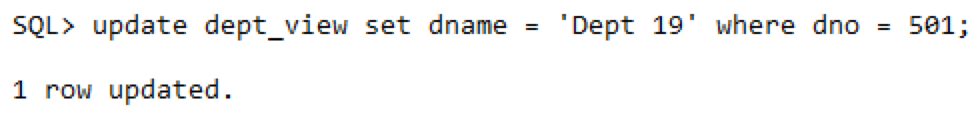
Select * from dept_view;
Output:



```
SQL> Select * from dept_view;

       DNO DNAME
---------- ---------------------------------------
--------------------------------------------------
       506 Support
       507 Data Team
       501 IT
       502 Finance
       503 Marketing
       504 IT
       505 IT

7 rows selected.
```

Query: Update View table
Code:

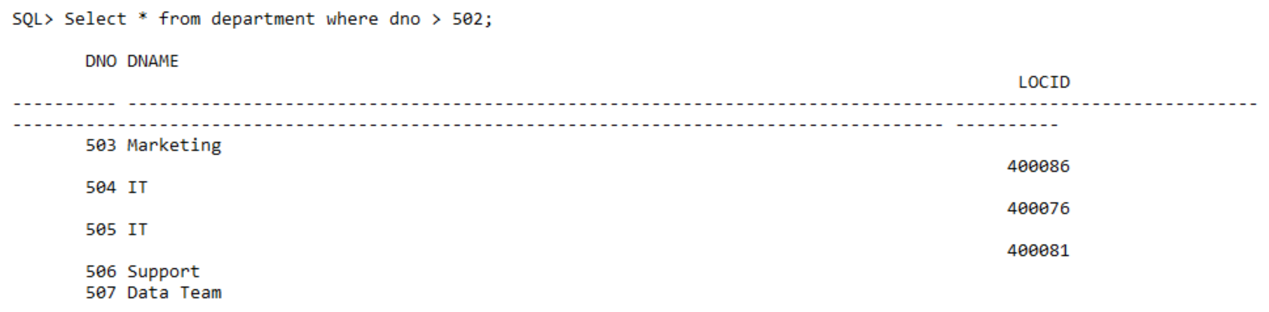
 update dept_view set dname = 'Dept 19' where dno = 501;
Output:



```
SQL> update dept_view set dname = 'Dept 19' where dno = 501;

1 row updated.
```

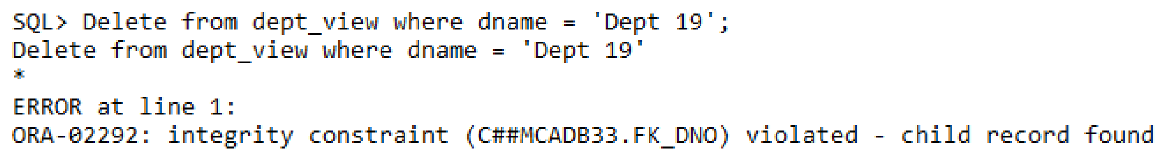
Code:

Select * from department where dno > 502;
Output:



```
SQL> Select * from department where dno > 502;

       DNO DNAME
                                                                                                    LOCID
---------- ----------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------- ----------
       503 Marketing
                                                                                                   400086
       504 IT
                                                                                                   400076
       505 IT
                                                                                                   400081
       506 Support
       507 Data Team
```

<u>Delete View</u>
<u>(same constraint are also create in view)</u>
Delete from dept_view where dname = 'Dept 19';



```
SQL> Delete from dept_view where dname = 'Dept 19';
Delete from dept_view where dname = 'Dept 19'
*
ERROR at line 1:
ORA-02292: integrity constraint (C##MCADB33.FK_DNO) violated - child record found
```

<u>**SYNONYMS**</u>

A synonym is an alias, that is, form of shorthand used to simplify the task of referencing a database object

Creating Synonyms
The general form of the create synonym is

CREATE [PUBLIC] SYNONYM SYNONYM_NAME FOR OBJECT_NAME;

Dropping Synonyms:

DROP SYNONYM SYNONYM_NAME;

DROP SYNONYM SYNNN;

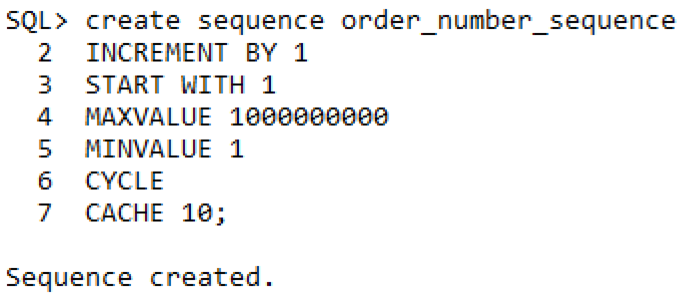
SEQUENCES

Syntax:

```
CREATE SEQUENCE <SEQUENCE_NAME>
[INCREMENT BY <NUMBER>]
[START WITH <START VALUE NUMBER>]
[MAXVALUE <MAXIMUM VALUE NUMBER>]
[NOMAXVALUE]
[MINVALUE <MINIMUM VALUE NUMBER>]
[CYCLE]
[NOCYCLE]
[CACHE <NUMBER OF SEQUENCE VALUE TO CACHE>]
[NOCACHE]
[ORDER]
[NOORDER];
```

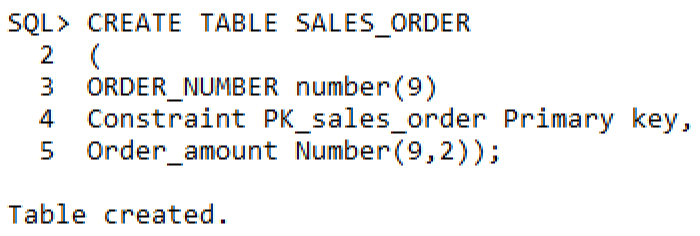
Example

```
create sequence order_number_sequence
INCREMENT BY 1
START WITH 1
MAXVALUE 1000000000
MINVALUE 1
CYCLE
CACHE 10;
```



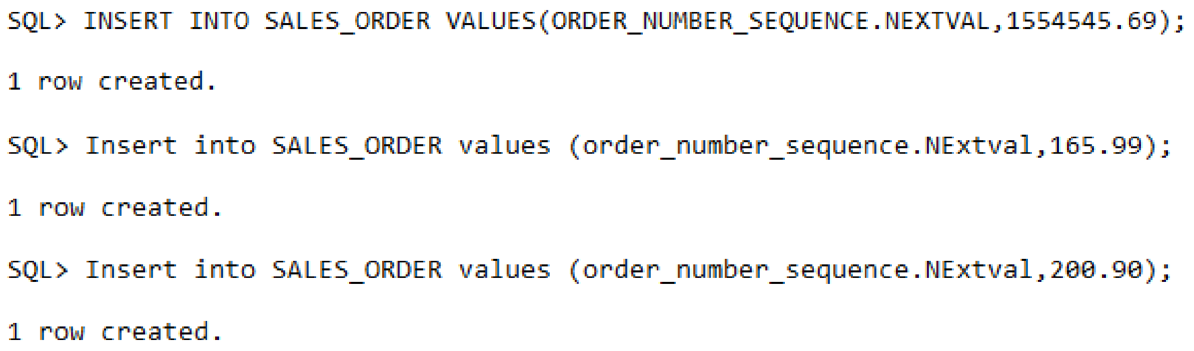
```
SQL> create sequence order_number_sequence
  2  INCREMENT BY 1
  3  START WITH 1
  4  MAXVALUE 1000000000
  5  MINVALUE 1
  6  CYCLE
  7  CACHE 10;

Sequence created.
```

```
CREATE TABLE SALES_ORDER
(
ORDER_NUMBER number(9)
Constraint PK_sales_order Primary key,
Order_amount Number(9,2));
```



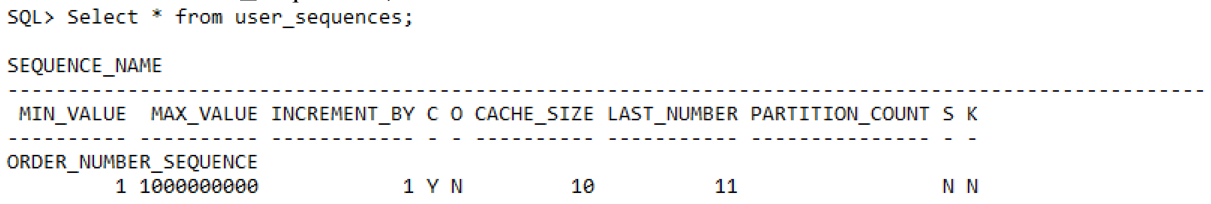
```
SQL> CREATE TABLE SALES_ORDER
  2  (
  3  ORDER_NUMBER number(9)
  4  Constraint PK_sales_order Primary key,
  5  Order_amount Number(9,2));

Table created.
```

Insert into SALES_ORDER values (order_number_sequence.NExtval,155.69);
Insert into SALES_ORDER values (order_number_sequence.NExtval,165.99);
Insert into SALES_ORDER values (order_number_sequence.NExtval,200.90);



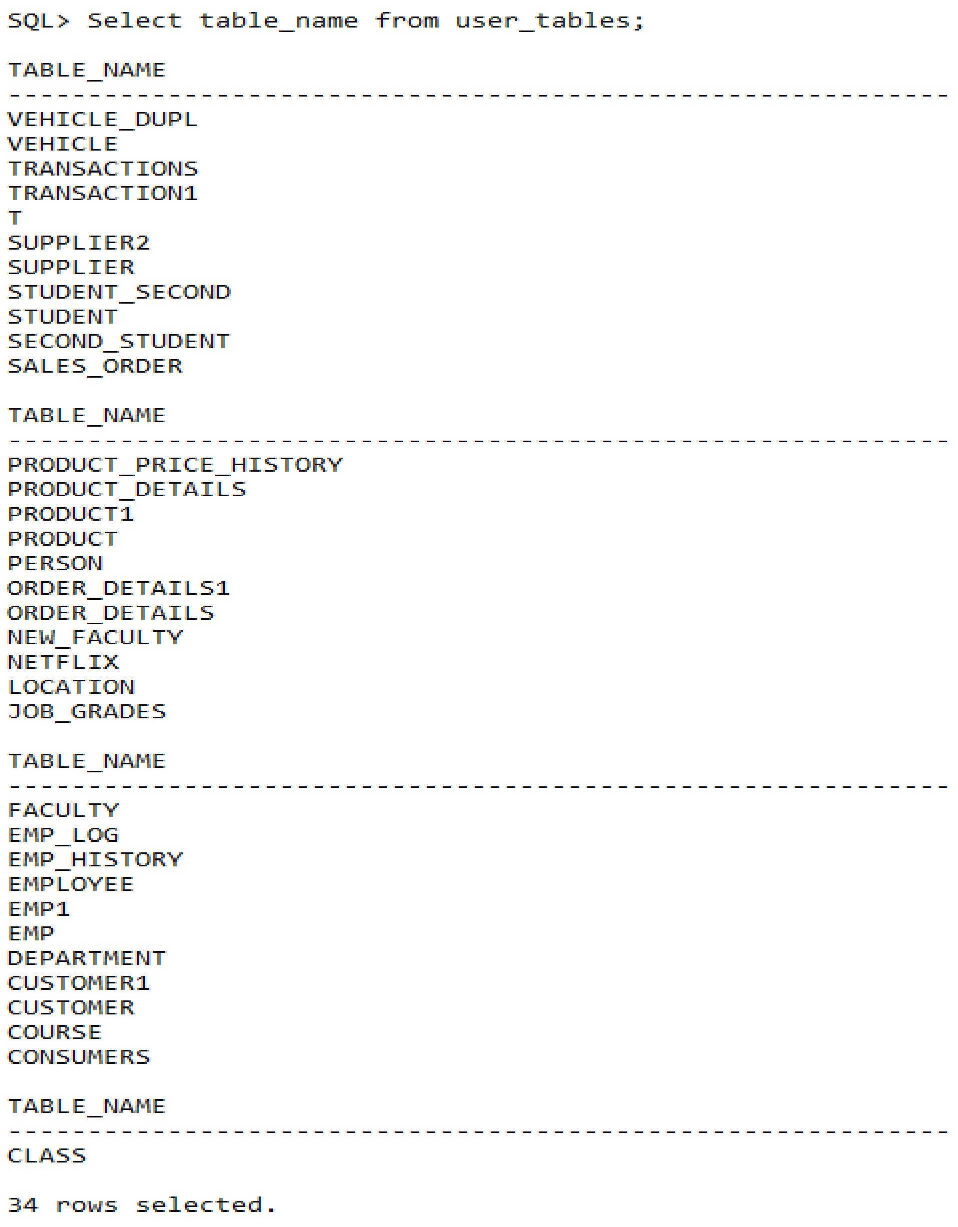
```
SQL> INSERT INTO SALES_ORDER VALUES(ORDER_NUMBER_SEQUENCE.NEXTVAL,1554545.69);

1 row created.

SQL> Insert into SALES_ORDER values (order_number_sequence.NExtval,165.99);

1 row created.

SQL> Insert into SALES_ORDER values (order_number_sequence.NExtval,200.90);

1 row created.
```

Select * from user_sequences;



```
SQL> Select * from user_sequences;

SEQUENCE_NAME
--------------------------------------------------------------------------------------------------
 MIN_VALUE  MAX_VALUE INCREMENT_BY C O CACHE_SIZE LAST_NUMBER PARTITION_COUNT S K
---------- ---------- ------------ - - ---------- ----------- --------------- - -
ORDER_NUMBER_SEQUENCE
         1 1000000000            1 Y N         10          11                 N N
```

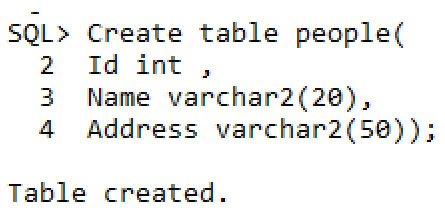
Select table_name from user_tables;



```
SQL> Select table_name from user_tables;

TABLE_NAME
-----------------------------------------------------------
VEHICLE_DUPL
VEHICLE
TRANSACTIONS
TRANSACTION1
T
SUPPLIER2
SUPPLIER
STUDENT_SECOND
STUDENT
SECOND_STUDENT
SALES_ORDER

TABLE_NAME
-----------------------------------------------------------
PRODUCT_PRICE_HISTORY
PRODUCT_DETAILS
PRODUCT1
PRODUCT
PERSON
ORDER_DETAILS1
ORDER_DETAILS
NEW_FACULTY
NETFLIX
LOCATION
JOB_GRADES

TABLE_NAME
-----------------------------------------------------------
FACULTY
EMP_LOG
EMP_HISTORY
EMPLOYEE
EMP1
EMP
DEPARTMENT
CUSTOMER1
CUSTOMER
COURSE
CONSUMERS

TABLE_NAME
-----------------------------------------------------------
CLASS

34 rows selected.
```

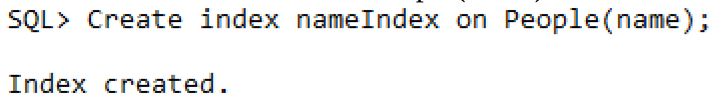
<u>Index</u>

```
Create table people(
Id int ,
Name varchar2(20),
Address varchar2(50));
```



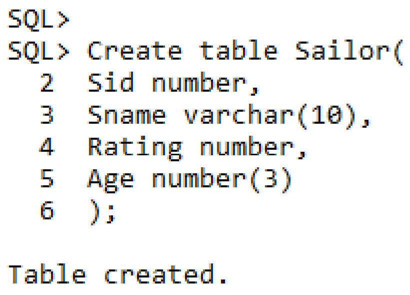
```
SQL> Create table people(
  2  Id int ,
  3  Name varchar2(20),
  4  Address varchar2(50));

Table created.
```

<u>Syntax:</u>
Create index nameIndex on People(name)



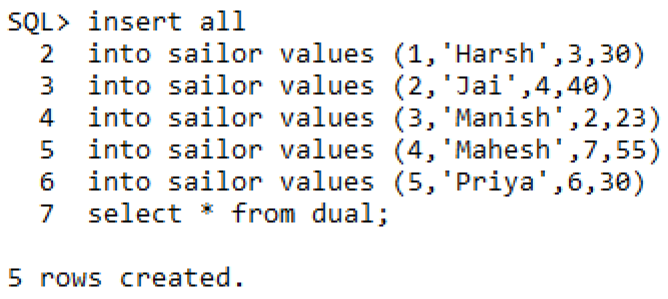
```
SQL> Create index nameIndex on People(name);

Index created.
```

Create table Sailor(
Sid number,
Sname varchar(10),
Rating number,
Age number(3)
);



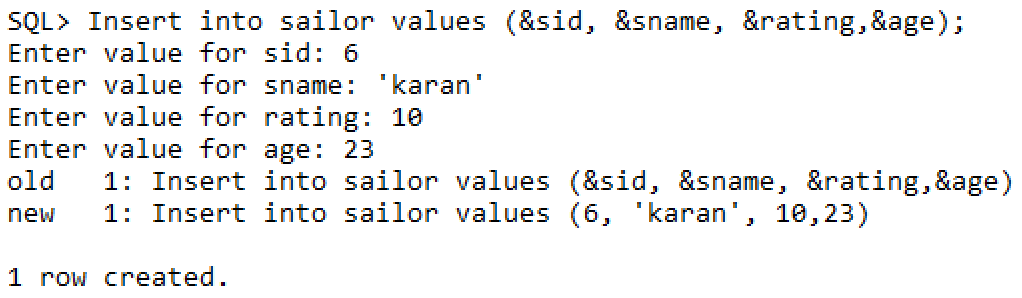
```
SQL>
SQL> Create table Sailor(
  2  Sid number,
  3  Sname varchar(10),
  4  Rating number,
  5  Age number(3)
  6  );

Table created.
```

insert all
into sailor values (1,'Harsh',3,30)
into sailor values (2,'Jai',4,40)
into sailor values (3,'Manish',2,23)
into sailor values (4,'Mahesh',7,55)
into sailor values (5,'Priya',6,30)
select * from dual;



```
SQL> insert all
  2  into sailor values (1,'Harsh',3,30)
  3  into sailor values (2,'Jai',4,40)
  4  into sailor values (3,'Manish',2,23)
  5  into sailor values (4,'Mahesh',7,55)
  6  into sailor values (5,'Priya',6,30)
  7  select * from dual;

5 rows created.
```

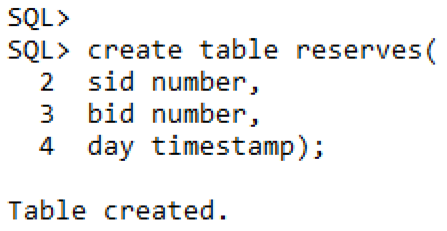
Insert into sailors values (&sid, &sname, &rating,&age);



```
SQL> Insert into sailor values (&sid, &sname, &rating,&age);
Enter value for sid: 6
Enter value for sname: 'karan'
Enter value for rating: 10
Enter value for age: 23
old   1: Insert into sailor values (&sid, &sname, &rating,&age)
new   1: Insert into sailor values (6, 'karan', 10,23)

1 row created.
```

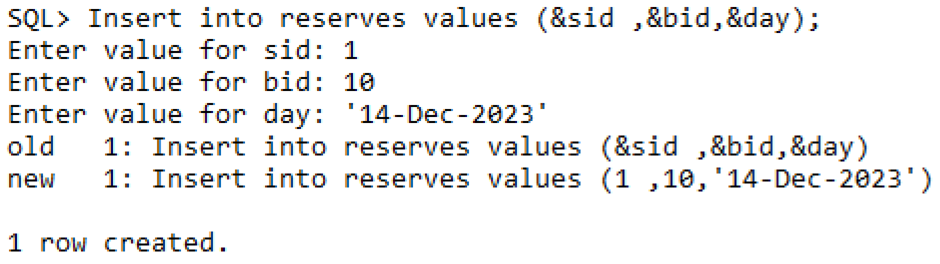
<u>Q. create table reserves and insert data</u>

create table reserves(
sid number,
bid number,
day timestamp);



```
SQL>
SQL> create table reserves(
  2  sid number,
  3  bid number,
  4  day timestamp);

Table created.
```

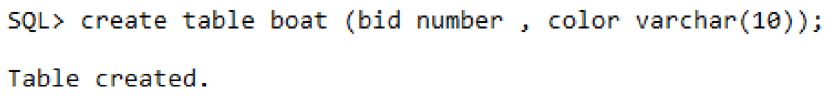
Insert into reserves values (&sid ,&bid,&day);



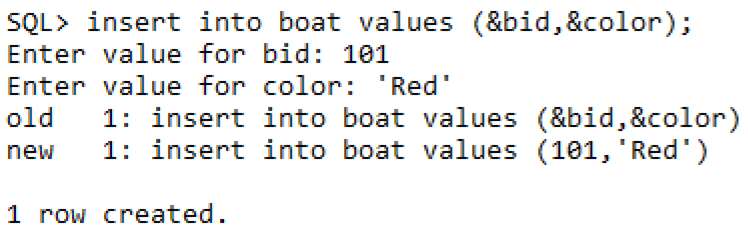
```
SQL> Insert into reserves values (&sid ,&bid,&day);
Enter value for sid: 1
Enter value for bid: 10
Enter value for day: '14-Dec-2023'
old   1: Insert into reserves values (&sid ,&bid,&day)
new   1: Insert into reserves values (1 ,10,'14-Dec-2023')

1 row created.
```

<u>Q. create table boat</u>

create table boat (bid number , color varchar(10));

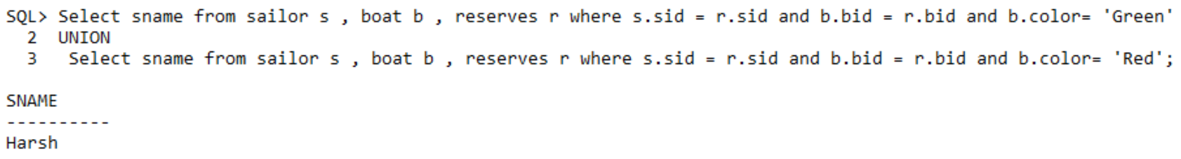


```
SQL> create table boat (bid number , color varchar(10));

Table created.
```



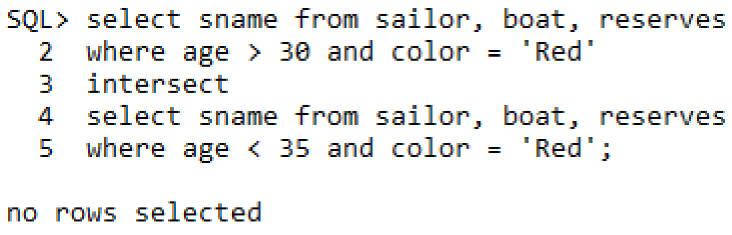
```
SQL> insert into boat values (&bid,&color);
Enter value for bid: 101
Enter value for color: 'Red'
old   1: insert into boat values (&bid,&color)
new   1: insert into boat values (101,'Red')

1 row created.
```

Select sname from sailor s , boat b , reserves r where s.sid = r.sid and b.bid = r.bid and b.color= ‘Green’ UNION Select sname from sailor s , boat b , reserves r where s.sid = r.sid and b.bid = r.bid and b.color= ‘Green’;



```
SQL> Select sname from sailor s , boat b , reserves r where s.sid = r.sid and b.bid = r.bid and b.color= 'Green'
  2  UNION
  3   Select sname from sailor s , boat b , reserves r where s.sid = r.sid and b.bid = r.bid and b.color= 'Red';

SNAME
----------
Harsh
```

select sname from sailor, boat, reserves
where age > 30 and color = 'Blue'
intersect
select sname from sailor, boat, reserves
where age < 35 and color = 'Red';



```
SQL> select sname from sailor, boat, reserves
  2  where age > 30 and color = 'Red'
  3  intersect
  4  select sname from sailor, boat, reserves
  5  where age < 35 and color = 'Red';

no rows selected
```

select sname from sailor, boat, reserves
where color = 'Blue'
minus
select sname from sailor, boat, reserves
where age between 30 and 35;

```
SQL> select sname from sailor, boat, reserves
  2  where color = 'Blue'
  3  minus
  4  select sname from sailor, boat, reserves
  5  where age between 30 and 35;

SNAME
----------
Jai
Mahesh
Manish
karan
```

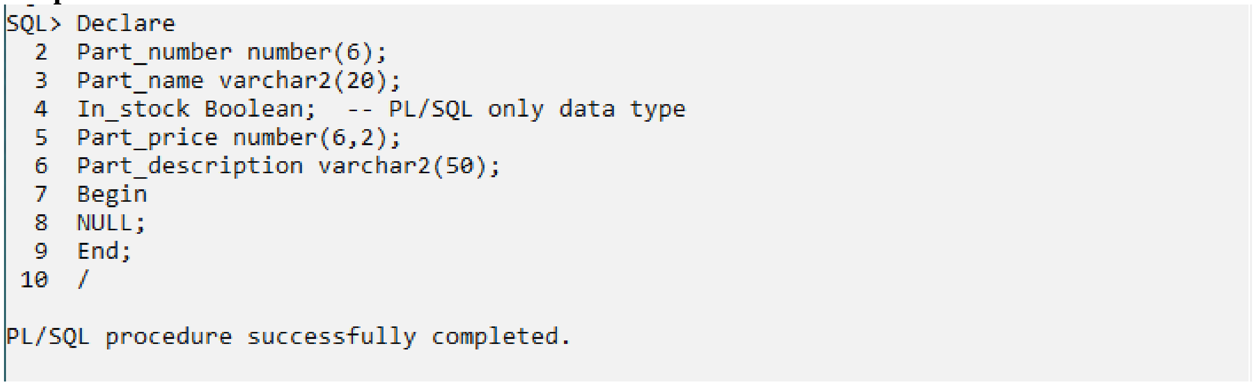
Practical 10 : PL/SQL Practical Programming

a. Variables, Identifiers

Code:

```
Declare
        Part_number number(6);
        Part_name varchar2(20);
        In_stock Boolean;  -– PL/SQL only data type
        Part_price number(6,2);
        Part_description varchar2(50);
Begin
        NULL;
End;
/
```

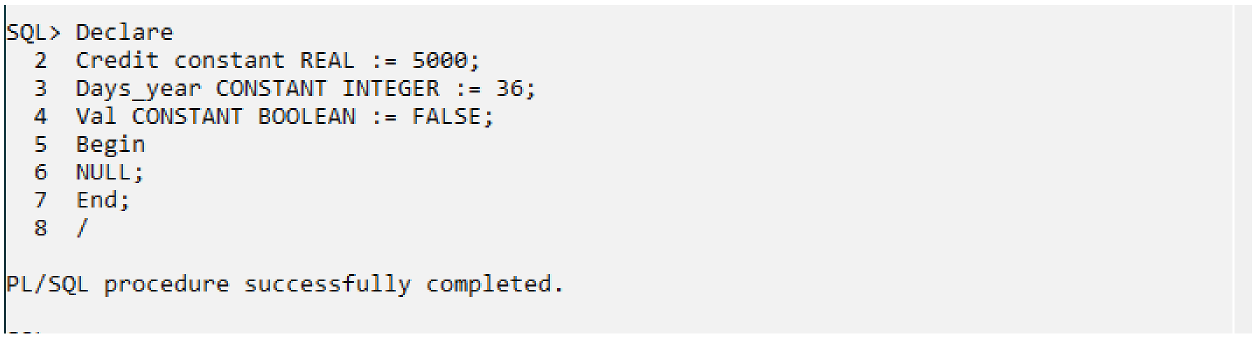
Output:



```
SQL> Declare
  2  Part_number number(6);
  3  Part_name varchar2(20);
  4  In_stock Boolean;  -- PL/SQL only data type
  5  Part_price number(6,2);
  6  Part_description varchar2(50);
  7  Begin
  8  NULL;
  9  End;
 10  /

PL/SQL procedure successfully completed.
```

```
Declare
        Credit constant REAL := 5000;
        Days_year CONSTANT INTEGER := 36;
        Val CONSTANT BOOLEAN := FALSE;
Begin
        NULL;
End;
/
```

Output:



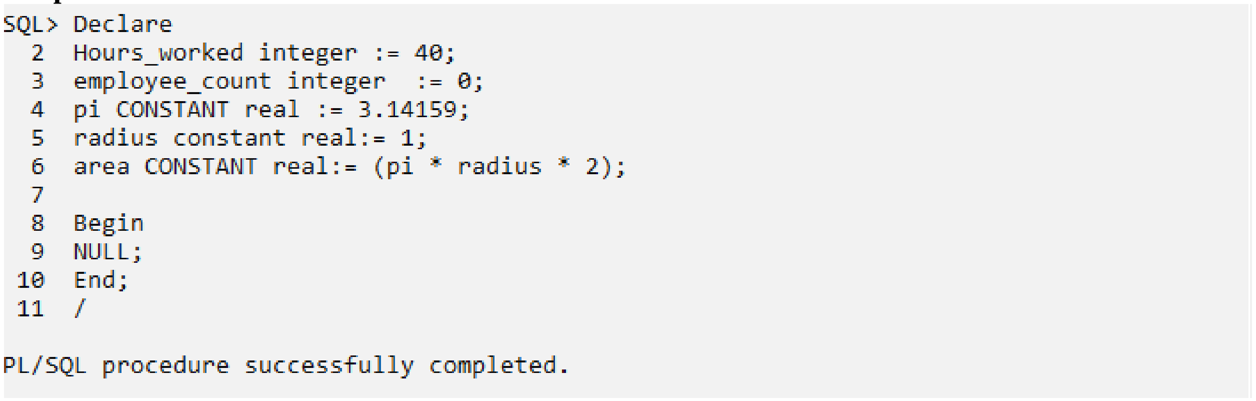
```
SQL> Declare
  2  Credit constant REAL := 5000;
  3  Days_year CONSTANT INTEGER := 36;
  4  Val CONSTANT BOOLEAN := FALSE;
  5  Begin
  6  NULL;
  7  End;
  8  /

PL/SQL procedure successfully completed.
```

b. Comment

Code:

```
Declare

        Hours_worked integer := 40;
        employee_count integer  := 0;
        pi CONSTANT real := 3.14159;
        radius constant real:= 1;
        area CONSTANT real:= (pi * radius * 2);

Begin
        NULL;
End;
/
```

Output:



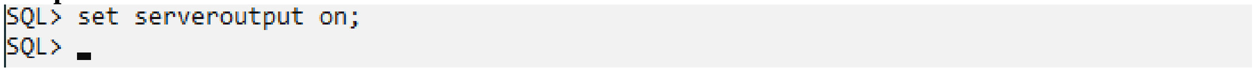
```
SQL> Declare
     Hours_worked integer := 40;
     employee_count integer  := 0;
     pi CONSTANT real := 3.14159;
     radius constant real:= 1;
     area CONSTANT real:= (pi * radius * 2);

     Begin
     NULL;
     End;
     /

PL/SQL procedure successfully completed.
```

Code:

```
set serveroutput on;
```

To display output

Output:

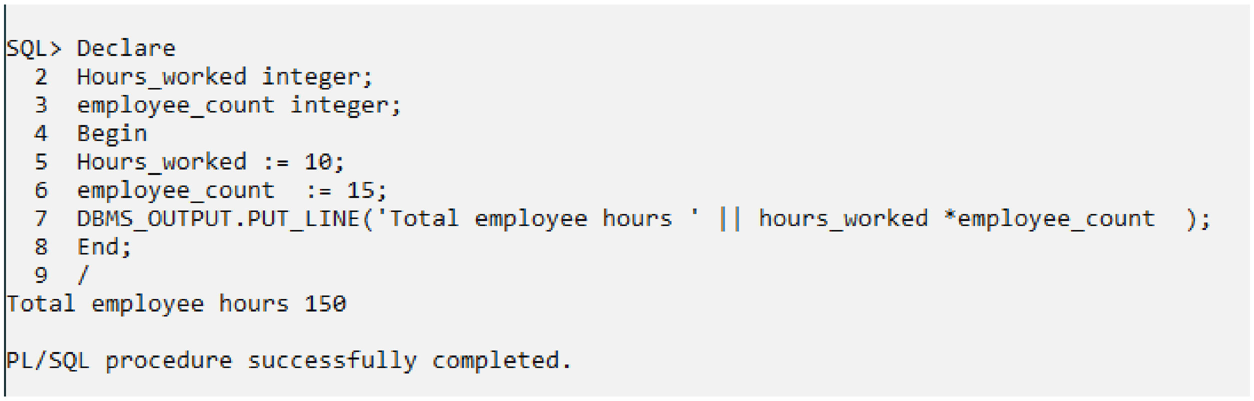


```
SQL> set serveroutput on;
SQL>
```

Code:

```
Declare
        Hours_worked integer;
        employee_count integer;
Begin
        Hours_worked := 10;
        employee_count  := 15;
                DBMS_OUTPUT.PUT_LINE('Total employee hours ' || hours_worked
*employee_count  );
End;
/
```

Output:



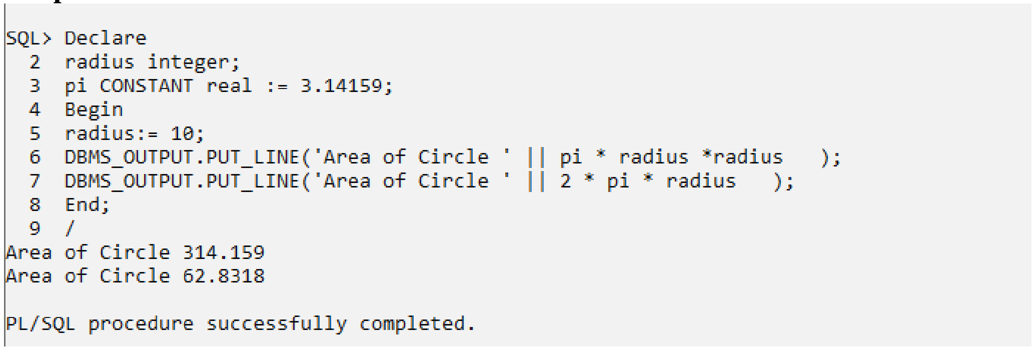
```
SQL> Declare
Hours_worked integer;
employee_count integer;
Begin
Hours_worked := 10;
employee_count  := 15;
DBMS_OUTPUT.PUT_LINE('Total employee hours ' || hours_worked *employee_count  );
End;
/
Total employee hours 150

PL/SQL procedure successfully completed.
```

c. PL/SQL Block

Query: Write the PL/SQL block
Find the area and circumference of circle
Code:

```
Declare
        radius integer;
        pi CONSTANT real := 3.14159;
Begin
        radius:= 10;
        DBMS_OUTPUT.PUT_LINE('Area of Circle ' || pi * radius *radius   );
        DBMS_OUTPUT.PUT_LINE('Area of Circle ' || 2 * pi * radius   );
End;
/
```

Output:

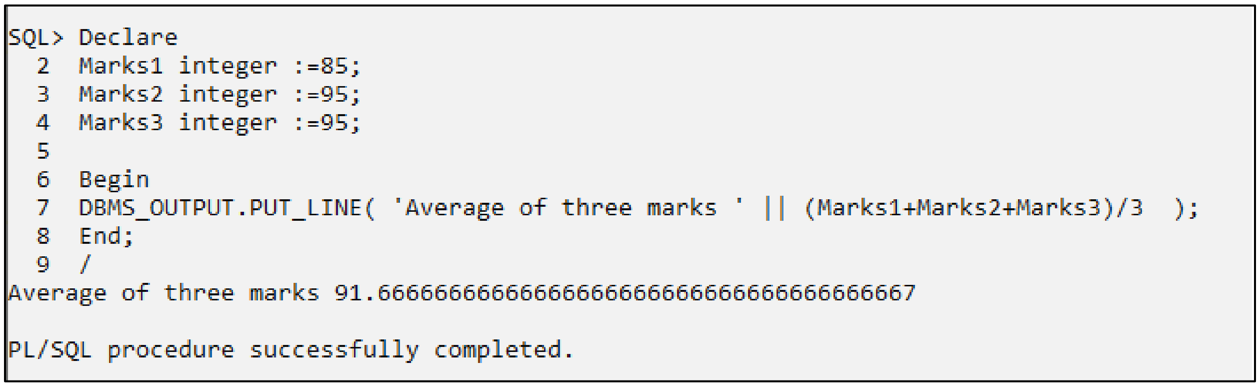


```
SQL> Declare
radius integer;
pi CONSTANT real := 3.14159;
Begin
radius:= 10;
DBMS_OUTPUT.PUT_LINE('Area of Circle ' || pi * radius *radius   );
DBMS_OUTPUT.PUT_LINE('Area of Circle ' || 2 * pi * radius   );
End;
/
Area of Circle 314.159
Area of Circle 62.8318

PL/SQL procedure successfully completed.
```

Query: Average of three marks
Code:

```
Declare
        Marks1 integer :=85;
        Marks2 integer :=95;
        Marks3 integer :=95;

Begin
        DBMS_OUTPUT.PUT_LINE( 'Average of three marks ' || (Marks1+Marks2+Marks3)/3  );
End;
/
```

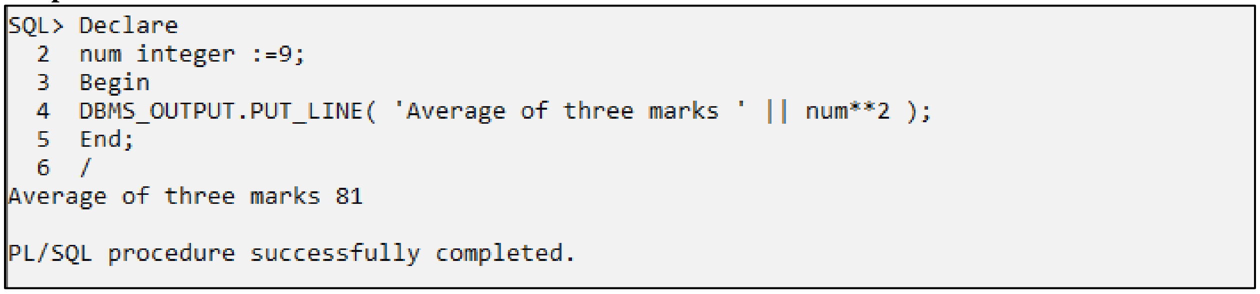
Output:



```
SQL> Declare
Marks1 integer :=85;
Marks2 integer :=95;
Marks3 integer :=95;

Begin
DBMS_OUTPUT.PUT_LINE( 'Average of three marks ' || (Marks1+Marks2+Marks3)/3  );
End;
/
Average of three marks 91.66666666666666666666666666666666666667

PL/SQL procedure successfully completed.
```

d. Structure

Query: Square of a Number
Code:
Declare
    num integer :=9;
Begin
    DBMS_OUTPUT.PUT_LINE('Average of three marks ' || num**2);
End;
/
Output:



```
SQL> Declare
num integer :=9;
Begin
DBMS_OUTPUT.PUT_LINE( 'Average of three marks ' || num**2 );
End;
/
Average of three marks 81

PL/SQL procedure successfully completed.
```

Practical 11 : Control Statements

a. Conditional Statements

- Conditional selection statements
  IF
- The Loop statements
  Loop
  For Loop
  While Loop
- Sequentail control statements
  GOTO which goews to a specified statement, and
  Null, which does nothing

b. Simple IF Statements

Syntax:

```
IF condition
Then
        Statements
End if;
```

c. Compound IF Statements

Syntax:

```
IF condition
Then
        Statements
Else
        Else statements
End if;
```

d. IF-THEN-ELSE Statements

Syntax:

```
IF condition
Then
        Statements
Elsif
        Elsif statements
Else
        Else statements
End if;
```

Example:

```
IF new_balance < minimum_balance THEN
  Overdrawn i=TRUE,
ELSE
  Overdrawn i = FALSE,
END IF;
```

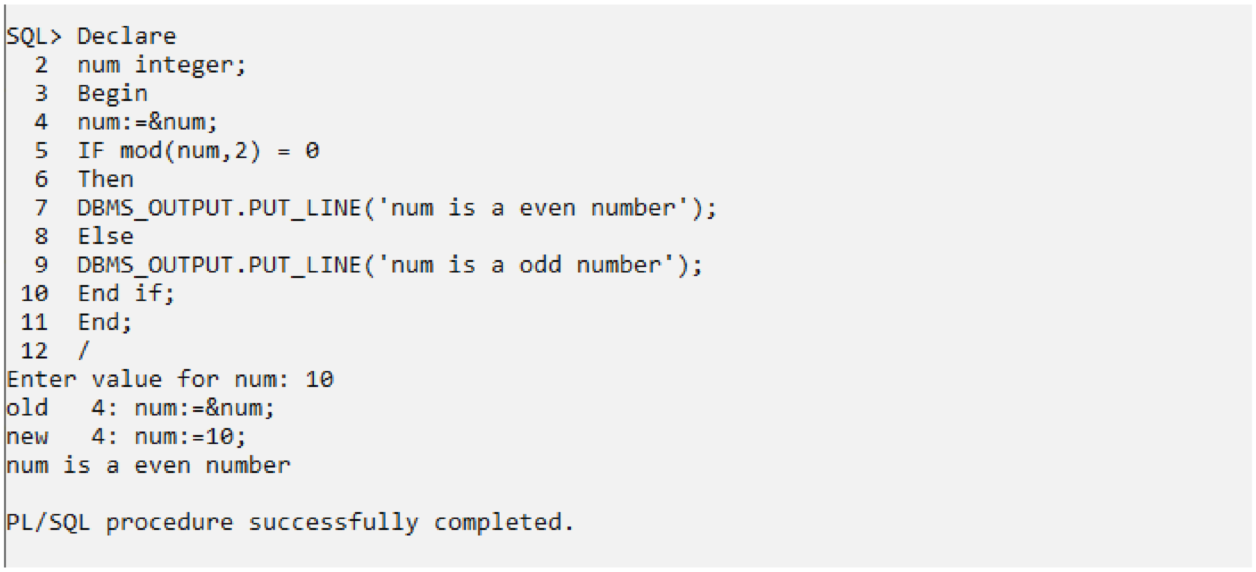
```
IF sales > (quota + 200) THEN
   Bonus := (sales-quota)/4,
ELSE
   Bonus :=50,
END IF;
```

Query: PLSQL block to check whether a number is even or odd

Code:

```
Declare
        num integer;
Begin
        num:=&num;
        IF mod(num,2) = 0
        Then
                DBMS_OUTPUT.PUT_LINE('num is a even number');
        Else
                DBMS_OUTPUT.PUT_LINE('num is a odd number');
        End if;
End;
/
```

Output:



```
SQL> Declare
  2  num integer;
  3  Begin
  4  num:=&num;
  5  IF mod(num,2) = 0
  6  Then
  7  DBMS_OUTPUT.PUT_LINE('num is a even number');
  8  Else
  9  DBMS_OUTPUT.PUT_LINE('num is a odd number');
 10  End if;
 11  End;
 12  /
Enter value for num: 10
old   4: num:=&num;
new   4: num:=10;
num is a even number

PL/SQL procedure successfully completed.
```

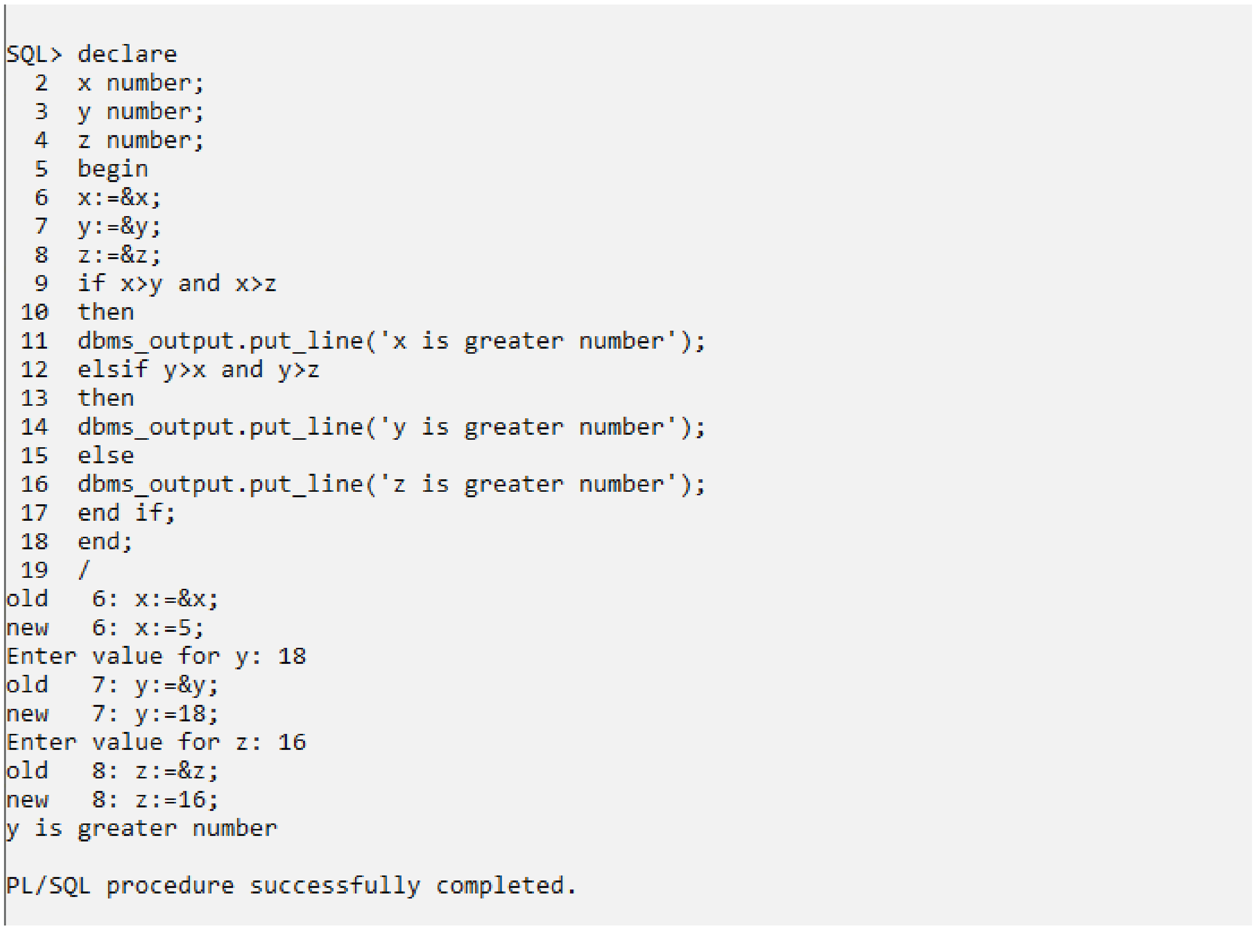
Query: PLSQL block to find the largest number of three number

Code:

```
declare
        x number;
        y number;
        z number;
begin
```

```
        x:=&x;
        y:=&y;
        z:=&z;
        if x>y and x>z
                then
                dbms_output.put_line('x is greater number');
        elsif y>x and y>z
                then
                dbms_output.put_line('y is greater number');
        else
                dbms_output.put_line('z is greater number');
        end if;
end;
/
```

Output:



```
SQL> declare
  2  x number;
  3  y number;
  4  z number;
  5  begin
  6  x:=&x;
  7  y:=&y;
  8  z:=&z;
  9  if x>y and x>z
 10  then
 11  dbms_output.put_line('x is greater number');
 12  elsif y>x and y>z
 13  then
 14  dbms_output.put_line('y is greater number');
 15  else
 16  dbms_output.put_line('z is greater number');
 17  end if;
 18  end;
 19  /
old   6: x:=&x;
new   6: x:=5;
Enter value for y: 18
old   7: y:=&y;
new   7: y:=18;
Enter value for z: 16
old   8: z:=&z;
new   8: z:=16;
y is greater number

PL/SQL procedure successfully completed.
```

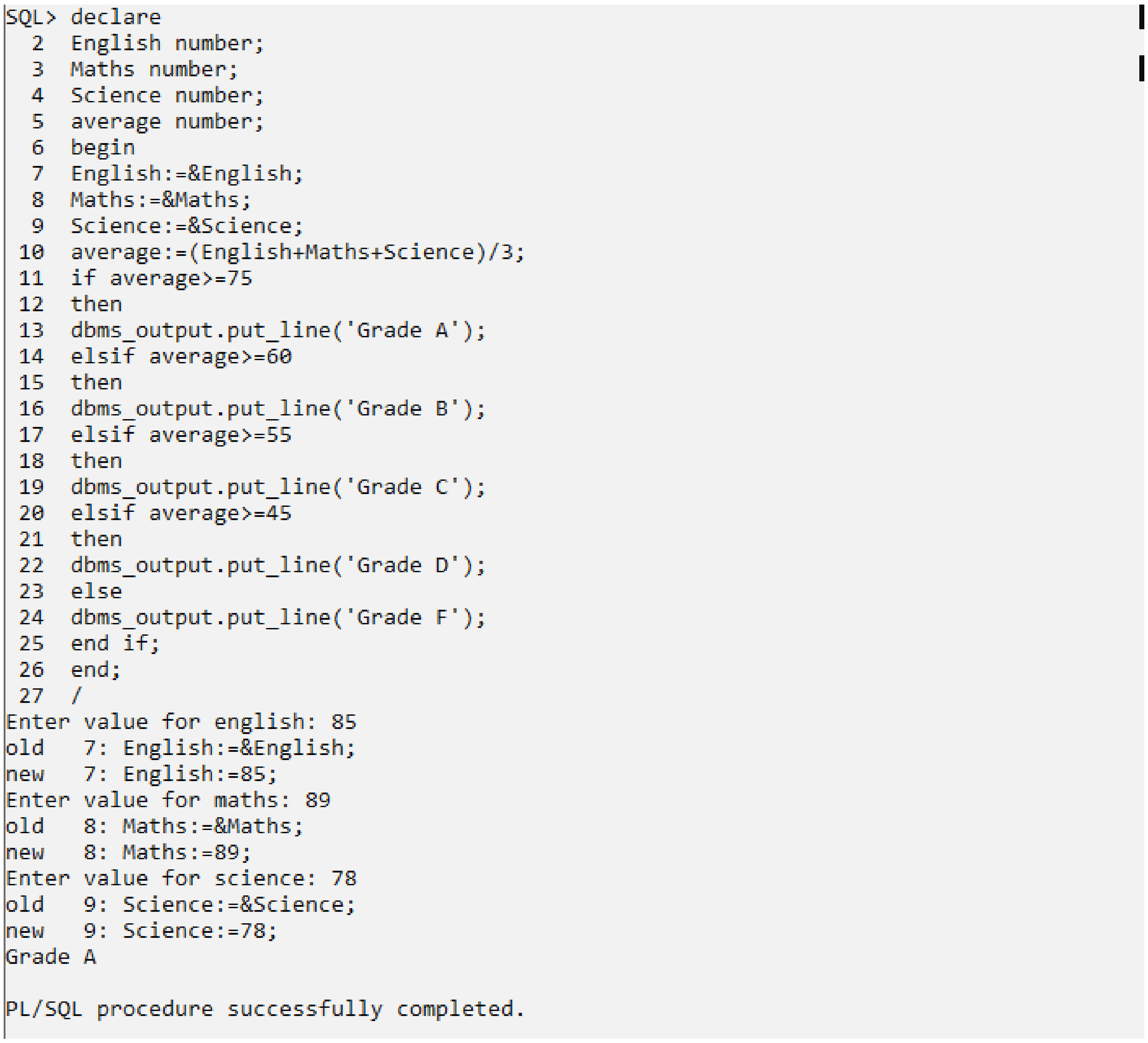
Query: PLSQL block to accept marks of 3 subject and find the average marks and do the grading

Code:

```
declare
        English number;
        Maths number;
        Science number;
        average number;
```

```
begin
        English:=&English;
        Maths:=&Maths;
        Science:=&Science;
        average:=(English+Maths+Science)/3;
        if average>=75
        then
                dbms_output.put_line('Grade A');
        elsif average>=60
        then
                dbms_output.put_line('Grade B');
        elsif average>=55
        then
                dbms_output.put_line('Grade C');
        elsif average>=45
        then
                dbms_output.put_line('Grade D');
        else
                dbms_output.put_line('Grade F');
        end if;
end;
/
```

Output:



```
SQL> declare
  2  English number;
  3  Maths number;
  4  Science number;
  5  average number;
  6  begin
  7  English:=&English;
  8  Maths:=&Maths;
  9  Science:=&Science;
 10  average:=(English+Maths+Science)/3;
 11  if average>=75
 12  then
 13  dbms_output.put_line('Grade A');
 14  elsif average>=60
 15  then
 16  dbms_output.put_line('Grade B');
 17  elsif average>=55
 18  then
 19  dbms_output.put_line('Grade C');
 20  elsif average>=45
 21  then
 22  dbms_output.put_line('Grade D');
 23  else
 24  dbms_output.put_line('Grade F');
 25  end if;
 26  end;
 27  /
Enter value for english: 85
old   7: English:=&English;
new   7: English:=85;
Enter value for maths: 89
old   8: Maths:=&Maths;
new   8: Maths:=89;
Enter value for science: 78
old   9: Science:=&Science;
new   9: Science:=78;
Grade A

PL/SQL procedure successfully completed.
```

Practical 12 : Loop

a. Basic Loop

Syntax:

```
Loop
[program statements]
IF condition then
        EXIT;
END IF;
        [additional  program statements]
END LOOP
```

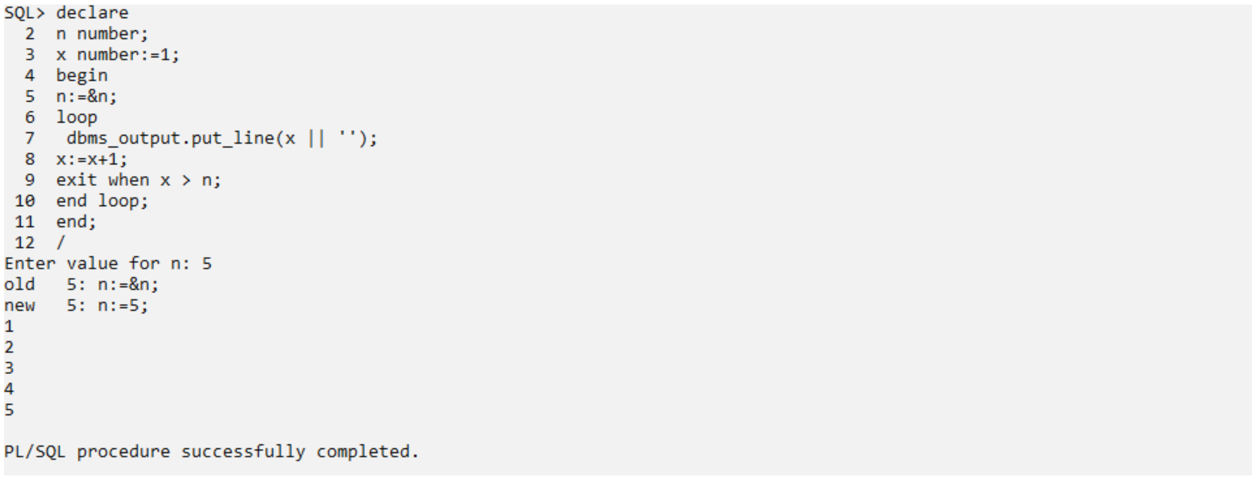
```
LOOP
        Program statements
EXIT WHEN condition;
END LOOP;
```

Query: PLSQL Block for basic loop

Code:

```
declare
        n number;
        x number:=1;
begin
        n:=&n;
        loop
                dbms_output.put_line(x || '');
                x:=x+1;
                exit when x > n;
        end loop;
end;
/
```

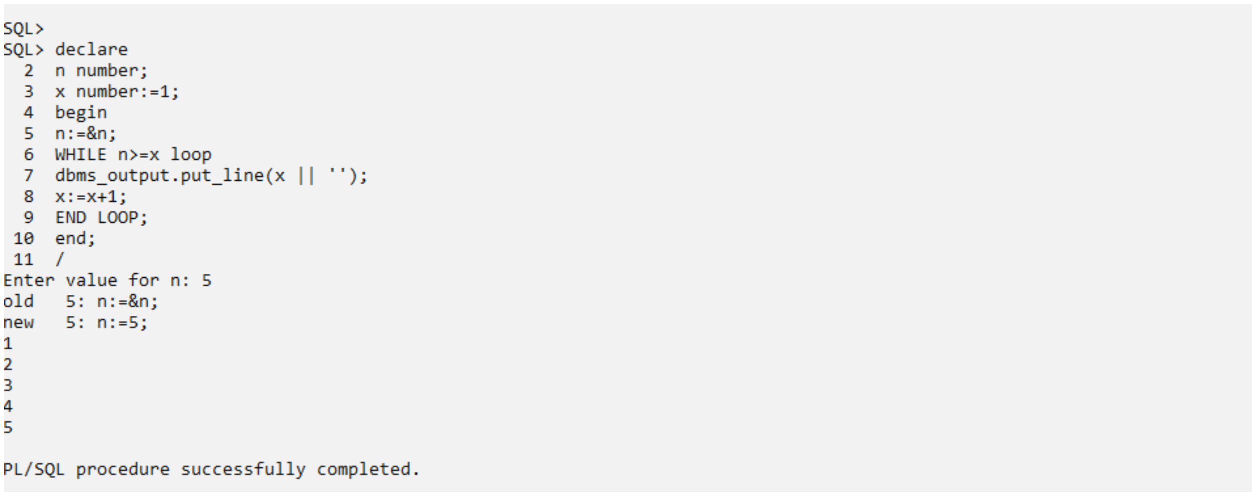
Output:


```
SQL> declare
n number;
x number:=1;
begin
n:=&n;
loop
 dbms_output.put_line(x || '');
x:=x+1;
exit when x > n;
end loop;
end;
/
Enter value for n: 5
old   5: n:=&n;
new   5: n:=5;
1
2
3
4
5

PL/SQL procedure successfully completed.
```

b. WHILE Loop

Syntax:
```
WHILE condition loop
        statements
END LOOP
```

Query: PLSQL Block for while loop
Code:
```
declare
        n number;
        x number:=1;
begin
        n:=&n;
WHILE n>=x loop
        dbms_output.put_line(x || '');
                x:=x+1;
END LOOP;
end;
/
```
Output:

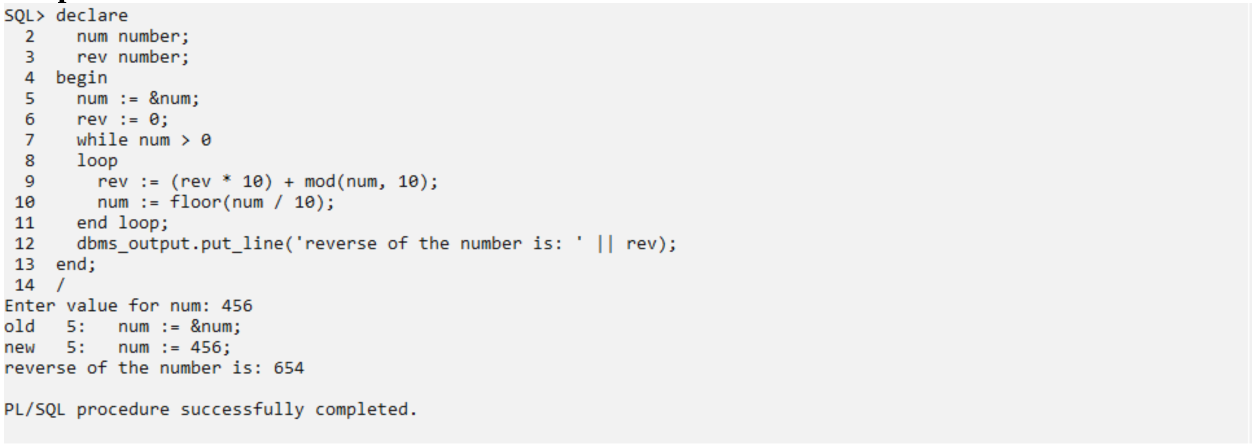


```
SQL>
SQL> declare
  2  n number;
  3  x number:=1;
  4  begin
  5  n:=&n;
  6  WHILE n>=x loop
  7  dbms_output.put_line(x || '');
  8  x:=x+1;
  9  END LOOP;
 10  end;
 11  /
Enter value for n: 5
old   5: n:=&n;
new   5: n:=5;
1
2
3
4
5

PL/SQL procedure successfully completed.
```

Query: PLSQL Block for reverse a number
Code:

```
declare
  num number;
  rev number;
begin
  num := &num;
  rev := 0;
  while num > 0
  loop
    rev := (rev * 10) + mod(num, 10);
    num := floor(num / 10);
  end loop;
  dbms_output.put_line('reverse of the number is: ' || rev);
end;
/
```

Output:



```
SQL> declare
  2    num number;
  3    rev number;
  4  begin
  5    num := &num;
  6    rev := 0;
  7    while num > 0
  8    loop
  9      rev := (rev * 10) + mod(num, 10);
 10      num := floor(num / 10);
 11    end loop;
 12    dbms_output.put_line('reverse of the number is: ' || rev);
 13  end;
 14  /
Enter value for num: 456
old   5:   num := &num;
new   5:   num := 456;
reverse of the number is: 654

PL/SQL procedure successfully completed.
```

c. FOR Loop

Syntax:

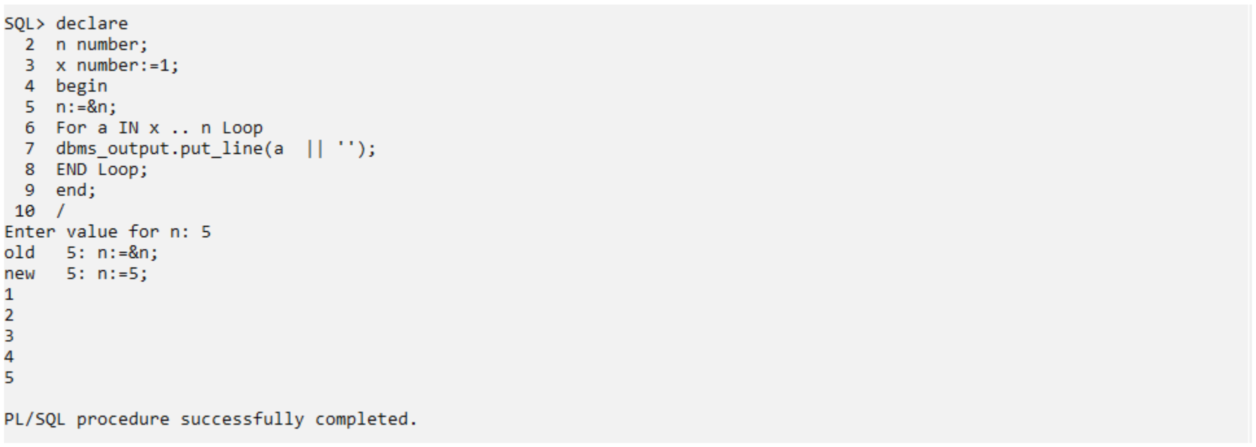
```
For counter_variable IN start_value .. end_value
Loop
        Program statements
END Loop;


For counter_variable IN REVERSE start_value .. end_value
Loop
        Program statements
END Loop;
/
```

Query: PLSQL Block using FOR loop

Code:

```
declare
        n number;
        x number:=1;
begin
        n:=&n;
For a IN x .. n Loop
        dbms_output.put_line(a  || '');
END Loop;
end;
/
```

Output:



```
SQL> declare
  2  n number;
  3  x number:=1;
  4  begin
  5  n:=&n;
  6  For a IN x .. n Loop
  7  dbms_output.put_line(a  || '');
  8  END Loop;
  9  end;
 10  /
Enter value for n: 5
old   5: n:=&n;
new   5: n:=5;
1
2
3
4
5

PL/SQL procedure successfully completed.
```

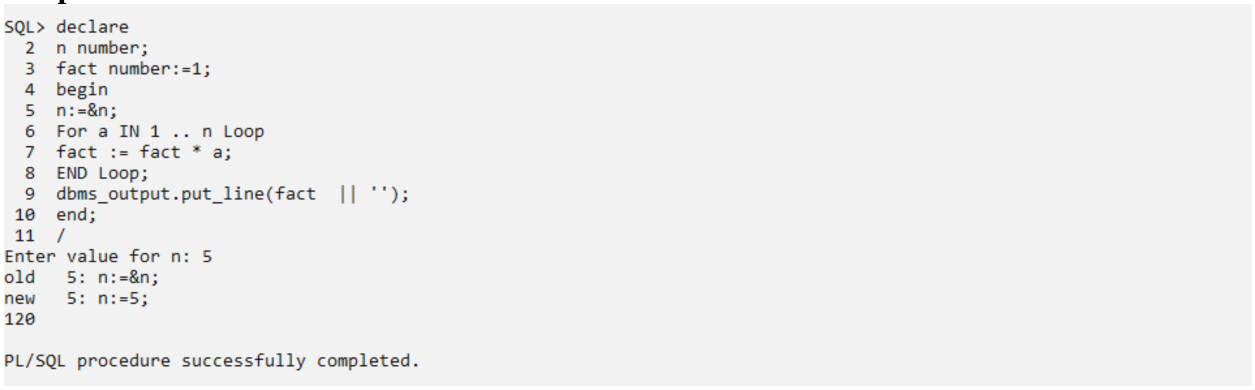
Query: PLSQL Block for factorial of a number

Code:

```
declare
```

```
        n number;
        fact number:=1;
begin
        n:=&n;
For a IN 1 .. n Loop
        fact := fact * a;
END Loop;
dbms_output.put_line(fact  || '');
end;
/
```

Output:



```
SQL> declare
  2  n number;
  3  fact number:=1;
  4  begin
  5  n:=&n;
  6  For a IN 1 .. n Loop
  7  fact := fact * a;
  8  END Loop;
  9  dbms_output.put_line(fact  || '');
 10  end;
 11  /
Enter value for n: 5
old   5: n:=&n;
new   5: n:=5;
120

PL/SQL procedure successfully completed.
```

Query: to find the even number between 1 and 50 in reverse order

Code:

```
declare
        n number;
begin
        n:=&n;
        For a IN REVERSE 1 .. n Loop
        If mod(a,2) = 0 then
                dbms_output.put_line( a  || '');
        End if;
END Loop;
end;
/
```

Output:

```
SQL> declare
  2  n number;
  3  begin
  4  n:=&n;
  5  For a IN REVERSE 1 .. n Loop
  6  If mod(a,2) = 0 then
  7  dbms_output.put_line( a  || '');
  8  End if;
  9  END Loop;
 10
 11  end;
 12  /
Enter value for n: 6
old   4: n:=&n;
new   4: n:=6;
6
4
2

PL/SQL procedure successfully completed.
```

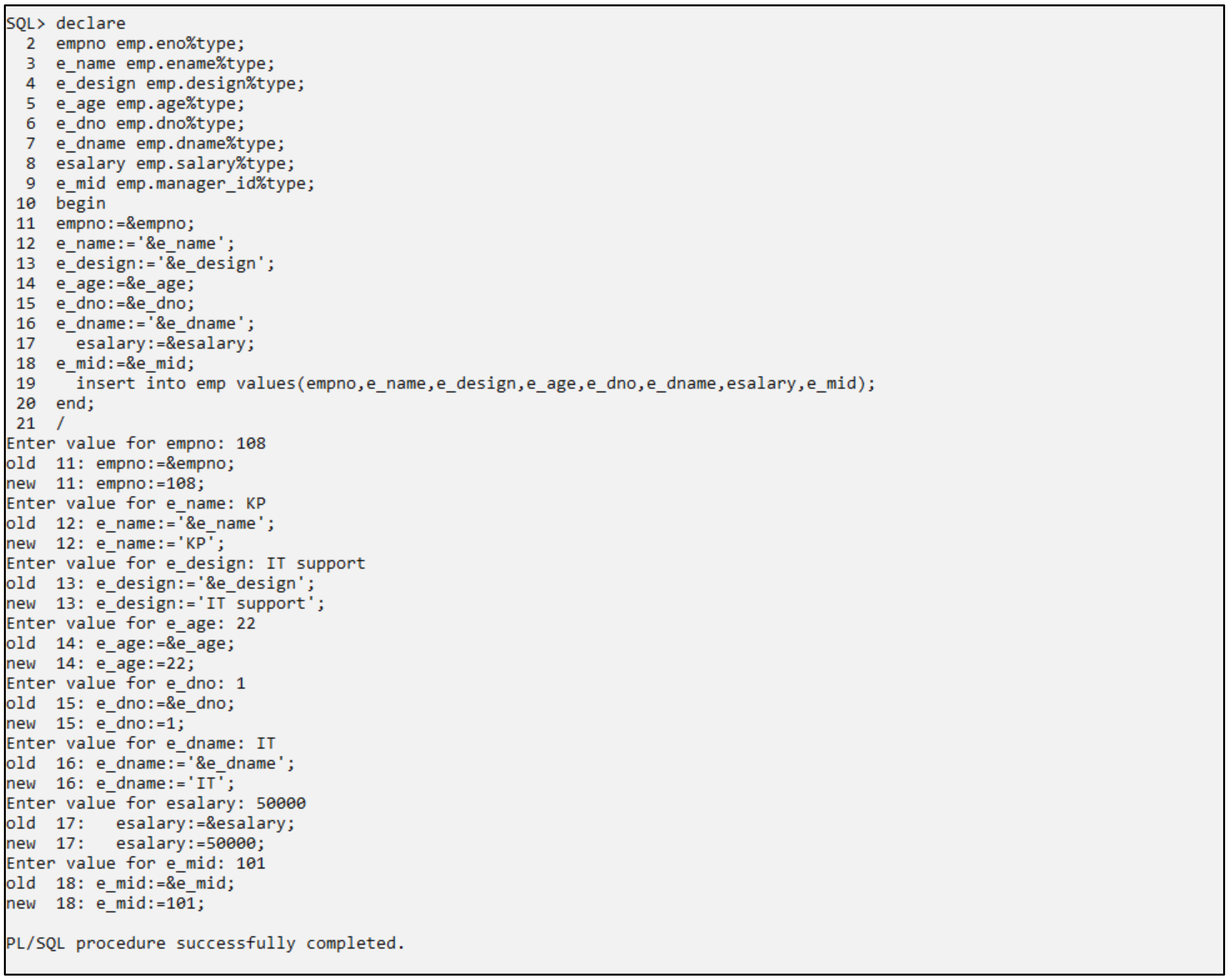
Practical 13 : DML Operations Using PL/SQL

a. Insert

Code :

```
declare
        empno emp.eno%type;
        e_name emp.ename%type;
        e_design emp.design%type;
        e_age emp.age%type;
        e_dno emp.dno%type;
        e_dname emp.dname%type;
        esalary emp.salary%type;
        e_mid emp.manager_id%type;
begin
        empno:=&empno;
        e_name:='&e_name';
        e_design:='&e_design';
        e_age:=&e_age;
        e_dno:=&e_dno;
        e_dname:='&e_dname';
  esalary:=&esalary;
        e_mid:=&e_mid;
  insert into emp values(empno,e_name,e_design,e_age,e_dno,e_dname,esalary,e_mid);
end;
/
```

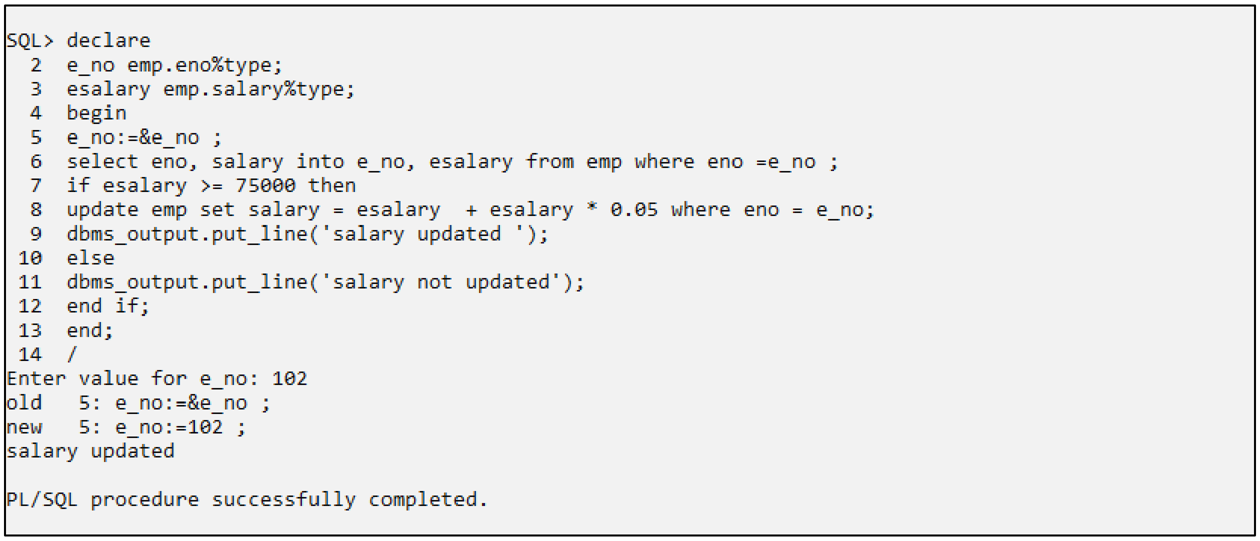
Output:



```
SQL> declare
  2  empno emp.eno%type;
  3  e_name emp.ename%type;
  4  e_design emp.design%type;
  5  e_age emp.age%type;
  6  e_dno emp.dno%type;
  7  e_dname emp.dname%type;
  8  esalary emp.salary%type;
  9  e_mid emp.manager_id%type;
 10  begin
 11  empno:=&empno;
 12  e_name:='&e_name';
 13  e_design:='&e_design';
 14  e_age:=&e_age;
 15  e_dno:=&e_dno;
 16  e_dname:='&e_dname';
 17    esalary:=&esalary;
 18  e_mid:=&e_mid;
 19    insert into emp values(empno,e_name,e_design,e_age,e_dno,e_dname,esalary,e_mid);
 20  end;
 21  /
Enter value for empno: 108
old  11: empno:=&empno;
new  11: empno:=108;
Enter value for e_name: KP
old  12: e_name:='&e_name';
new  12: e_name:='KP';
Enter value for e_design: IT support
old  13: e_design:='&e_design';
new  13: e_design:='IT support';
Enter value for e_age: 22
old  14: e_age:=&e_age;
new  14: e_age:=22;
Enter value for e_dno: 1
old  15: e_dno:=&e_dno;
new  15: e_dno:=1;
Enter value for e_dname: IT
old  16: e_dname:='&e_dname';
new  16: e_dname:='IT';
Enter value for esalary: 50000
old  17:   esalary:=&esalary;
new  17:   esalary:=50000;
Enter value for e_mid: 101
old  18: e_mid:=&e_mid;
new  18: e_mid:=101;

PL/SQL procedure successfully completed.
```

b. Update

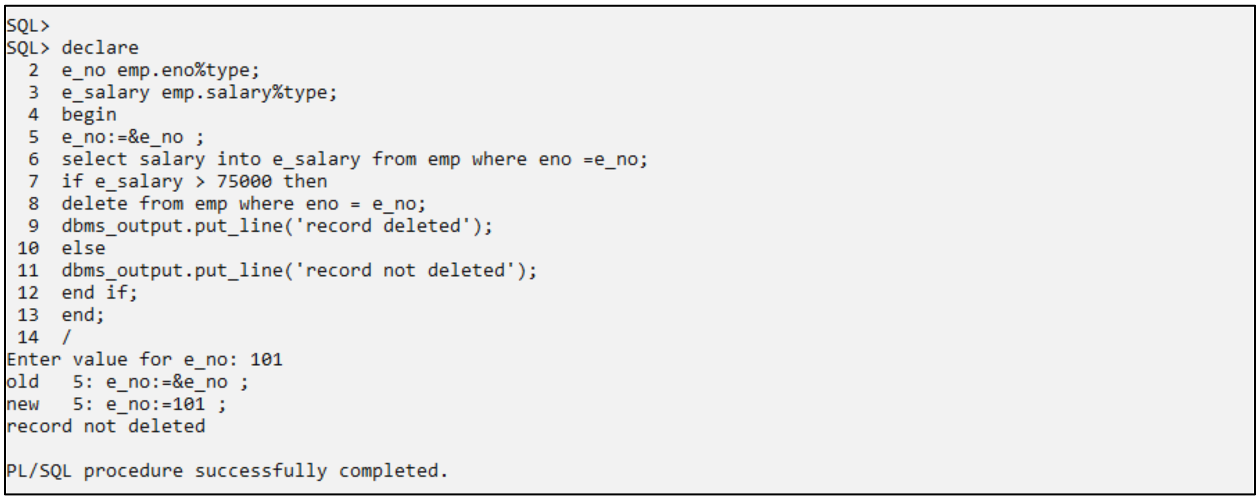
```
declare
        e_no emp.eno%type;
        esalary emp.salary%type;
begin
        e_no:=&e_no ;
        select eno, salary into e_no, esalary from emp where eno =e_no ;
        if esalary >= 75000 then
                update emp set salary = esalary  + esalary * 0.05 where eno = e_no;
                dbms_output.put_line('salary updated ');
        else
                dbms_output.put_line('salary not updated');
        end if;
end;
/
```



```
SQL> declare
  2  e_no emp.eno%type;
  3  esalary emp.salary%type;
  4  begin
  5  e_no:=&e_no ;
  6  select eno, salary into e_no, esalary from emp where eno =e_no ;
  7  if esalary >= 75000 then
  8  update emp set salary = esalary  + esalary * 0.05 where eno = e_no;
  9  dbms_output.put_line('salary updated ');
 10  else
 11  dbms_output.put_line('salary not updated');
 12  end if;
 13  end;
 14  /
Enter value for e_no: 102
old   5: e_no:=&e_no ;
new   5: e_no:=102 ;
salary updated

PL/SQL procedure successfully completed.
```

c. Delete

```
declare
        e_no emp.eno%type;
        e_salary emp.salary%type;
begin
        e_no:=&e_no ;
        select salary into e_salary from emp where eno =e_no;
        if e_salary > 75000 then
                delete from emp where eno = e_no;
                dbms_output.put_line('record deleted');
        else
                dbms_output.put_line('record not deleted');
        end if;
end;
/
```

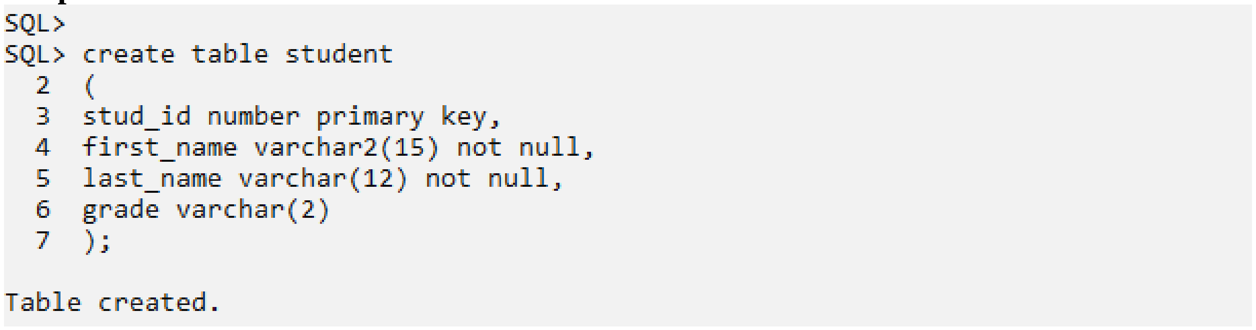


```
SQL>
SQL> declare
  2  e_no emp.eno%type;
  3  e_salary emp.salary%type;
  4  begin
  5  e_no:=&e_no ;
  6  select salary into e_salary from emp where eno =e_no;
  7  if e_salary > 75000 then
  8  delete from emp where eno = e_no;
  9  dbms_output.put_line('record deleted');
 10  else
 11  dbms_output.put_line('record not deleted');
 12  end if;
 13  end;
 14  /
Enter value for e_no: 101
old   5: e_no:=&e_no ;
new   5: e_no:=101 ;
record not deleted

PL/SQL procedure successfully completed.
```

d. Merge

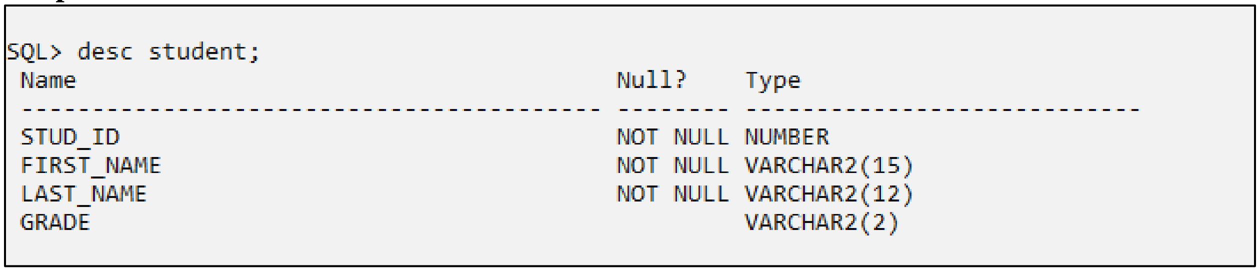
Query: Create two table to learn merge

Code:create table student
(
stud_id number primary key,
first_name varchar2(15) not null,
last_name varchar(12) not null,
grade varchar(2)
);
Output:



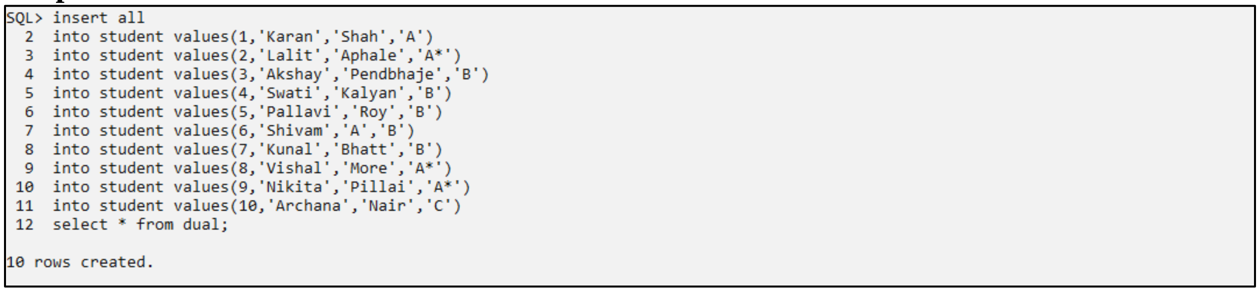
```
SQL>
SQL> create table student
  2  (
  3  stud_id number primary key,
  4  first_name varchar2(15) not null,
  5  last_name varchar(12) not null,
  6  grade varchar(2)
  7  );

Table created.
```

Code:desc student;
Output:



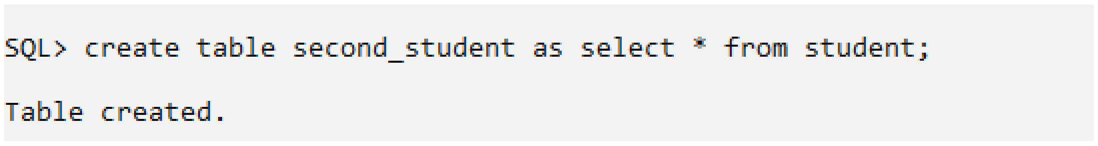
```
SQL> desc student;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUD_ID                                   NOT NULL NUMBER
 FIRST_NAME                                NOT NULL VARCHAR2(15)
 LAST_NAME                                 NOT NULL VARCHAR2(12)
 GRADE                                              VARCHAR2(2)
```

Code: Insert 10 records into student
Output:



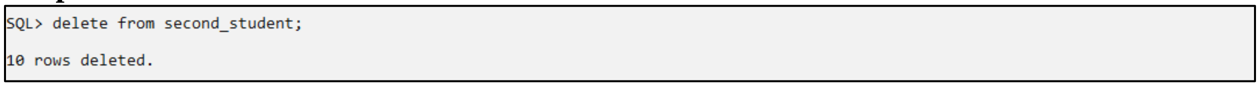
```
SQL> insert all
  2  into student values(1,'Karan','Shah','A')
  3  into student values(2,'Lalit','Aphale','A*')
  4  into student values(3,'Akshay','Pendbhaje','B')
  5  into student values(4,'Swati','Kalyan','B')
  6  into student values(5,'Pallavi','Roy','B')
  7  into student values(6,'Shivam','A','B')
  8  into student values(7,'Kunal','Bhatt','B')
  9  into student values(8,'Vishal','More','A*')
 10  into student values(9,'Nikita','Pillai','A*')
 11  into student values(10,'Archana','Nair','C')
 12  select * from dual;

10 rows created.
```

Code: create table second_student as select * from student;
Output:



```
SQL> create table second_student as select * from student;

Table created.
```

Code: delete from second_student;
Output:

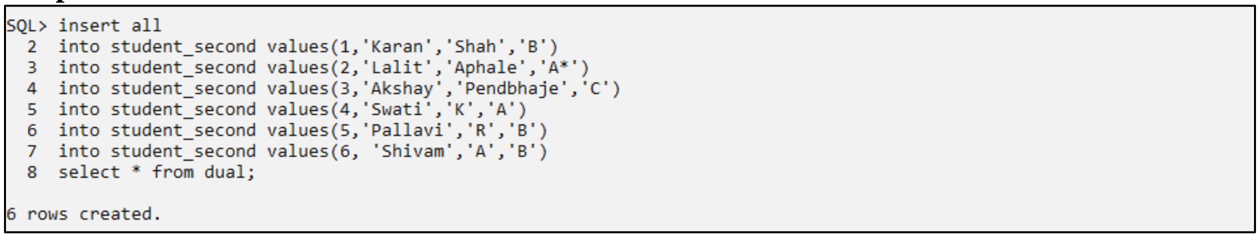


```
SQL> delete from second_student;

10 rows deleted.
```

Code:

insert all

into student_second values(1,'Karan','Shah','B')
into student_second values(2,'Lalit','Aphale','A*')
into student_second values(3,'Akshay','Pendbhaje','C')
into student_second values(4,'Swati','K','A')
into student_second values(5,'Pallavi','R','B')
into student_second values(6, 'Shivam','A','B')
select * from dual;
Output:

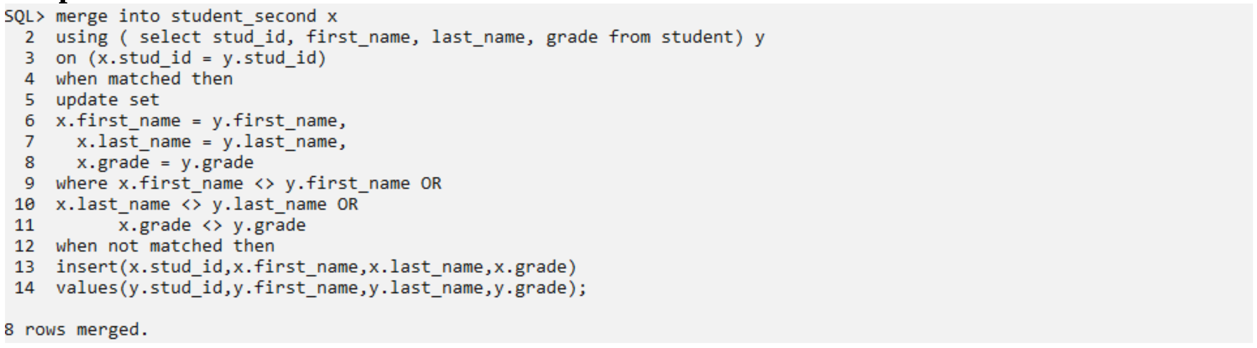


```
SQL> insert all
  2  into student_second values(1,'Karan','Shah','B')
  3  into student_second values(2,'Lalit','Aphale','A*')
  4  into student_second values(3,'Akshay','Pendbhaje','C')
  5  into student_second values(4,'Swati','K','A')
  6  into student_second values(5,'Pallavi','R','B')
  7  into student_second values(6, 'Shivam','A','B')
  8  select * from dual;

6 rows created.
```

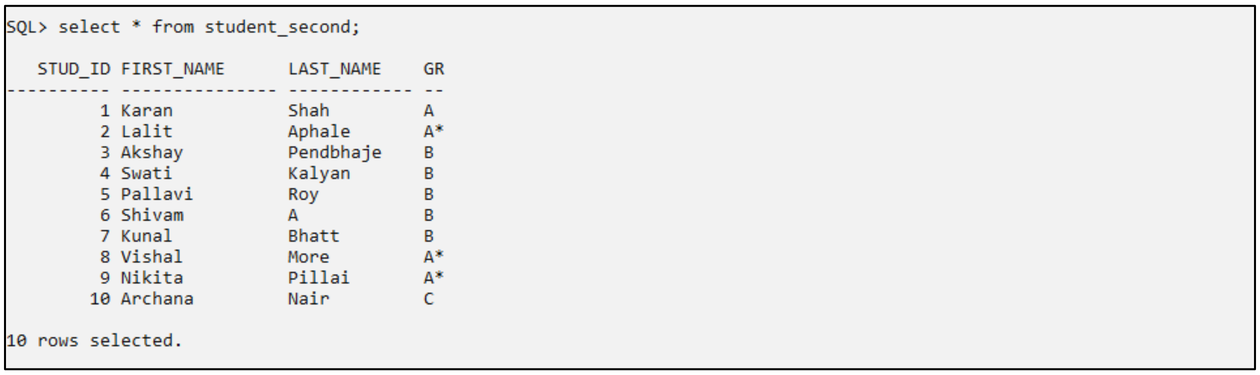
Code:

```
merge into student_second x
using ( select stud_id, first_name, last_name, grade from student) y
on (x.stud_id = y.stud_id)
when matched then
update set
            x.first_name = y.first_name,
  x.last_name = y.last_name,
  x.grade = y.grade
where x.first_name <> y.first_name OR
                                    x.last_name <> y.last_name OR
      x.grade <> y.grade
when not matched then
insert(x.stud_id,x.first_name,x.last_name,x.grade)
values(y.stud_id,y.first_name,y.last_name,y.grade);
```

Output:



```
SQL> merge into student_second x
     using ( select stud_id, first_name, last_name, grade from student) y
     on (x.stud_id = y.stud_id)
     when matched then
     update set
     x.first_name = y.first_name,
       x.last_name = y.last_name,
       x.grade = y.grade
     where x.first_name <> y.first_name OR
     x.last_name <> y.last_name OR
           x.grade <> y.grade
     when not matched then
     insert(x.stud_id,x.first_name,x.last_name,x.grade)
     values(y.stud_id,y.first_name,y.last_name,y.grade);

8 rows merged.
```



```
SQL> select * from student_second;

   STUD_ID FIRST_NAME      LAST_NAME    GR
---------- --------------- ------------ --
         1 Karan           Shah         A
         2 Lalit           Aphale       A*
         3 Akshay          Pendbhaje    B
         4 Swati           Kalyan       B
         5 Pallavi         Roy          B
         6 Shivam          A            B
         7 Kunal           Bhatt        B
         8 Vishal          More         A*
         9 Nikita          Pillai       A*
        10 Archana         Nair         C

10 rows selected.
```

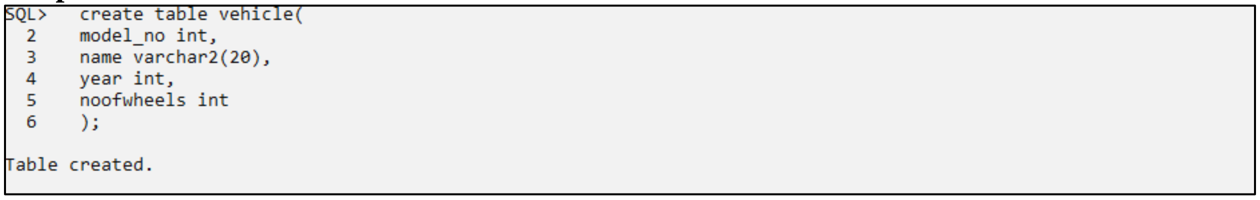
Query:
Create the following relations
VEHICLE (model_no,name,year,noofwheels)
VEHICLE_DUPL(model_no,name,year,noofwheels)
Enter few similar and few varying data into table

Merge the tables and get the accurate data in vehicle

Code:

```
create table vehicle(
model_no varchar2(20),
year int,
noofwheels int
);
```

Output:

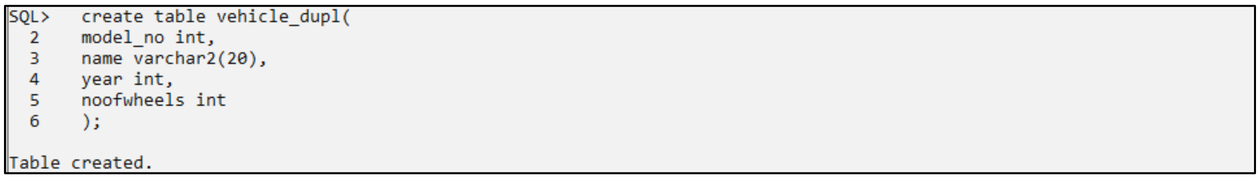


```
SQL>   create table vehicle(
       model_no int,
       name varchar2(20),
       year int,
       noofwheels int
       );

Table created.
```

Code:

```
create table vehicle_dupl(
model_no int,
name varchar2(20),
year int,
noofwheels int
);
```

Output:

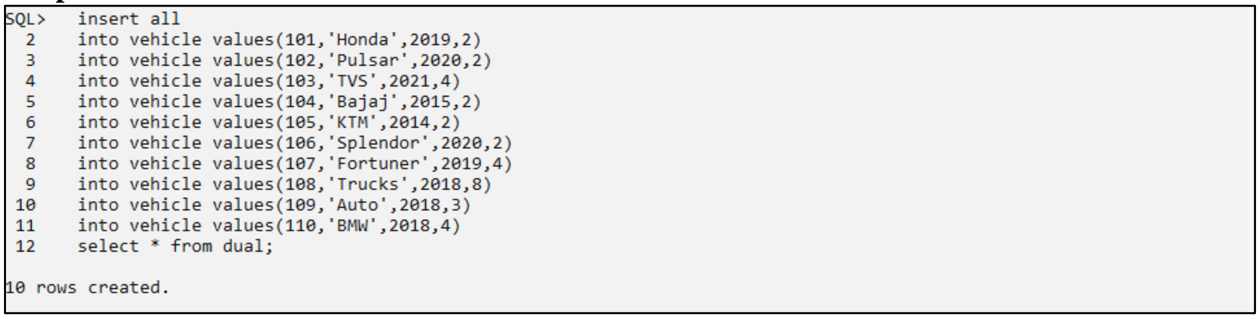


```
SQL>   create table vehicle_dupl(
  2    model_no int,
  3    name varchar2(20),
  4    year int,
  5    noofwheels int
  6    );

Table created.
```

Code:

```
insert all
into vehicle values(101,'Honda',2019,2)
into vehicle values(102,'Pulsar',2020,2)
into vehicle values(103,'TVS',2021,4)
into vehicle values(104,'Bajaj',2015,2)
into vehicle values(105,'KTM',2014,2)
into vehicle values(106,'Splendor',2020,2)
into vehicle values(107,'Fortuner',2019,4)
into vehicle values(108,'Trucks',2018,8)
into vehicle values(109,'Auto',2018,3)
into vehicle values(110,'BMW',2018,4)
select * from dual;
```

Output:



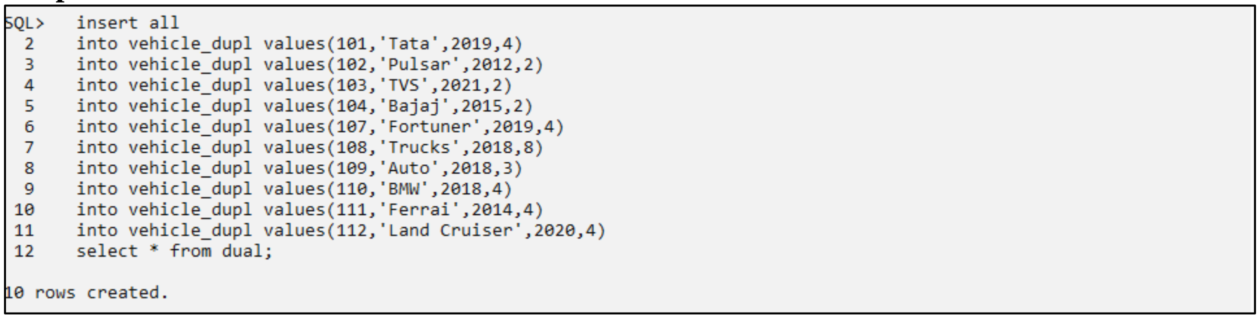
```
SQL>   insert all
  2    into vehicle values(101,'Honda',2019,2)
  3    into vehicle values(102,'Pulsar',2020,2)
  4    into vehicle values(103,'TVS',2021,4)
  5    into vehicle values(104,'Bajaj',2015,2)
  6    into vehicle values(105,'KTM',2014,2)
  7    into vehicle values(106,'Splendor',2020,2)
  8    into vehicle values(107,'Fortuner',2019,4)
  9    into vehicle values(108,'Trucks',2018,8)
 10    into vehicle values(109,'Auto',2018,3)
 11    into vehicle values(110,'BMW',2018,4)
 12    select * from dual;

10 rows created.
```

Code:

```
insert all
into vehicle_dupl values(101,'Tata',2019,4)
into vehicle_dupl values(102,'Pulsar',2012,2)
into vehicle_dupl values(103,'TVS',2021,2)
into vehicle_dupl values(104,'Bajaj',2015,2)
into vehicle_dupl values(107,'Fortuner',2019,4)
into vehicle_dupl values(108,'Trucks',2018,8)
```

```
  into vehicle_dupl values(109,'Auto',2018,3)
  into vehicle_dupl values(110,'BMW',2018,4)
  into vehicle_dupl values(111,'Ferrai',2014,4)
  into vehicle_dupl values(112,'Land Cruiser',2020,4)
  select * from dual;
```

Output:

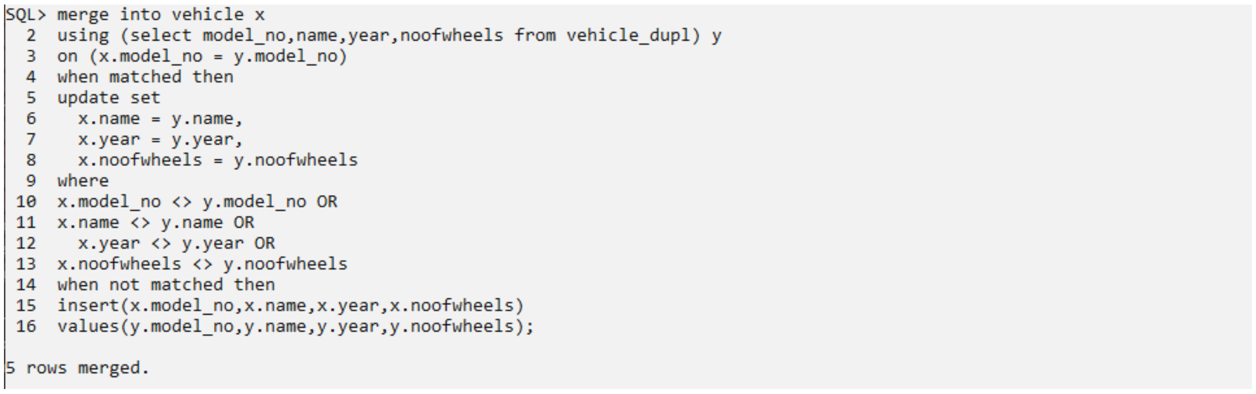


```
SQL>   insert all
       into vehicle_dupl values(101,'Tata',2019,4)
       into vehicle_dupl values(102,'Pulsar',2012,2)
       into vehicle_dupl values(103,'TVS',2021,2)
       into vehicle_dupl values(104,'Bajaj',2015,2)
       into vehicle_dupl values(107,'Fortuner',2019,4)
       into vehicle_dupl values(108,'Trucks',2018,8)
       into vehicle_dupl values(109,'Auto',2018,3)
       into vehicle_dupl values(110,'BMW',2018,4)
       into vehicle_dupl values(111,'Ferrai',2014,4)
       into vehicle_dupl values(112,'Land Cruiser',2020,4)
       select * from dual;

10 rows created.
```

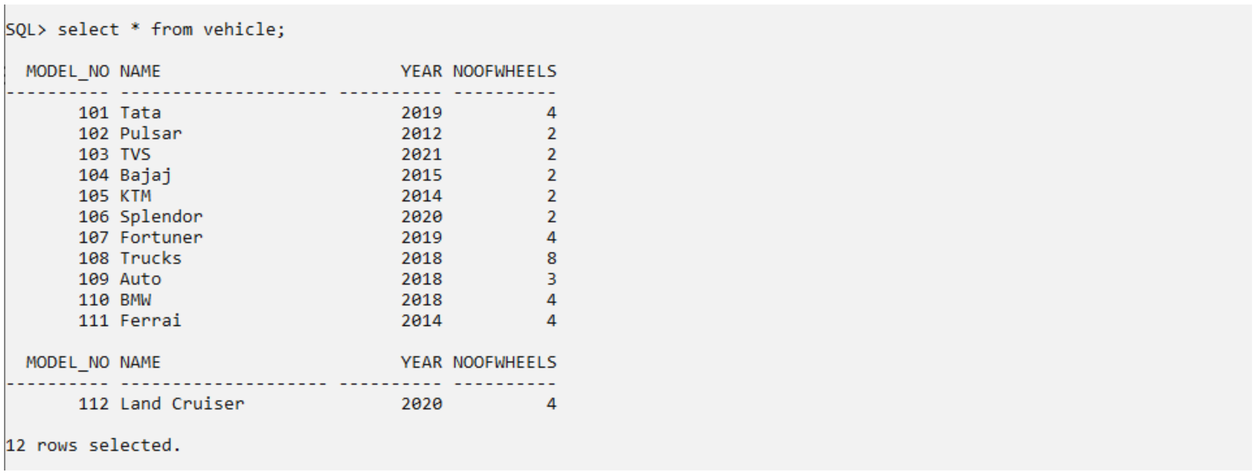
Code:

```
merge into vehicle x
using (select model_no,name,year,noofwheels from vehicle_dupl) y
on (x.model_no = y.model_no)
when matched then
update set
  x.name = y.name,
  x.year = y.year,
  x.noofwheels = y.noofwheels
where
            x.model_no <> y.model_no     OR
            x.name <> y.name OR
  x.year <> y.year OR
            x.noofwheels <> y.noofwheels
when not matched then
insert(x.model_no,x.name,x.year,x.noofwheels)
values(y.model_no,y.name,y.year,y.noofwheels);
```

Ouptut



```
SQL> merge into vehicle x
     using (select model_no,name,year,noofwheels from vehicle_dupl) y
     on (x.model_no = y.model_no)
     when matched then
     update set
       x.name = y.name,
       x.year = y.year,
       x.noofwheels = y.noofwheels
     where
     x.model_no <> y.model_no OR
     x.name <> y.name OR
       x.year <> y.year OR
     x.noofwheels <> y.noofwheels
     when not matched then
     insert(x.model_no,x.name,x.year,x.noofwheels)
     values(y.model_no,y.name,y.year,y.noofwheels);

5 rows merged.
```



```
SQL> select * from vehicle;

  MODEL_NO NAME                       YEAR NOOFWHEELS
---------- -------------------- ---------- ----------
       101 Tata                       2019          4
       102 Pulsar                     2012          2
       103 TVS                        2021          2
       104 Bajaj                      2015          2
       105 KTM                        2014          2
       106 Splendor                   2020          2
       107 Fortuner                   2019          4
       108 Trucks                     2018          8
       109 Auto                       2018          3
       110 BMW                        2018          4
       111 Ferrai                     2014          4

  MODEL_NO NAME                       YEAR NOOFWHEELS
---------- -------------------- ---------- ----------
       112 Land Cruiser               2020          4

12 rows selected.
```

Practical 14 : Exceptions

a. Exception Handling

Syntax:

```
Declare
Declaration section
Begin
        Exception Section
Exception
        When ex_name then error-handling-statement
        When ex_name2 then error-handling-statement
        When others then error-handling-statement
End;
```

When an exception is raised , Oracle searches for an appropriate exception handler in the exception section

b. Types of Exceptions

<u>Named system Exceptions</u>

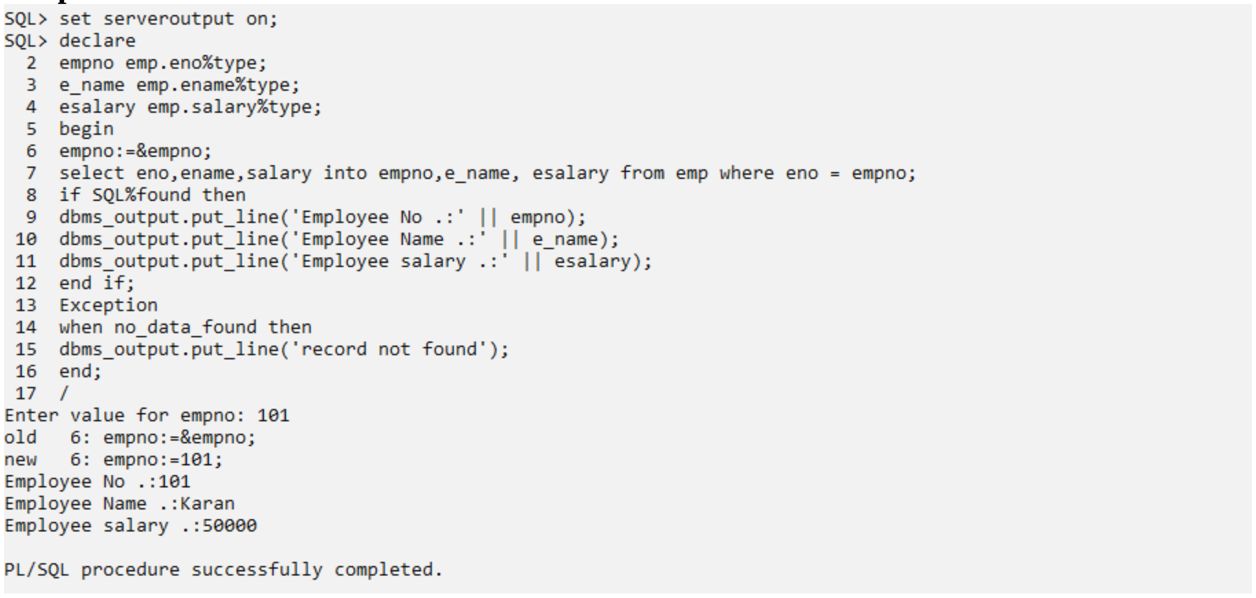
Syntax:

```
Begin
        Execution section
Exception
        When no_data_found then
        dbms_output.put_line(‘A select … into did not return any row’);
END;
```

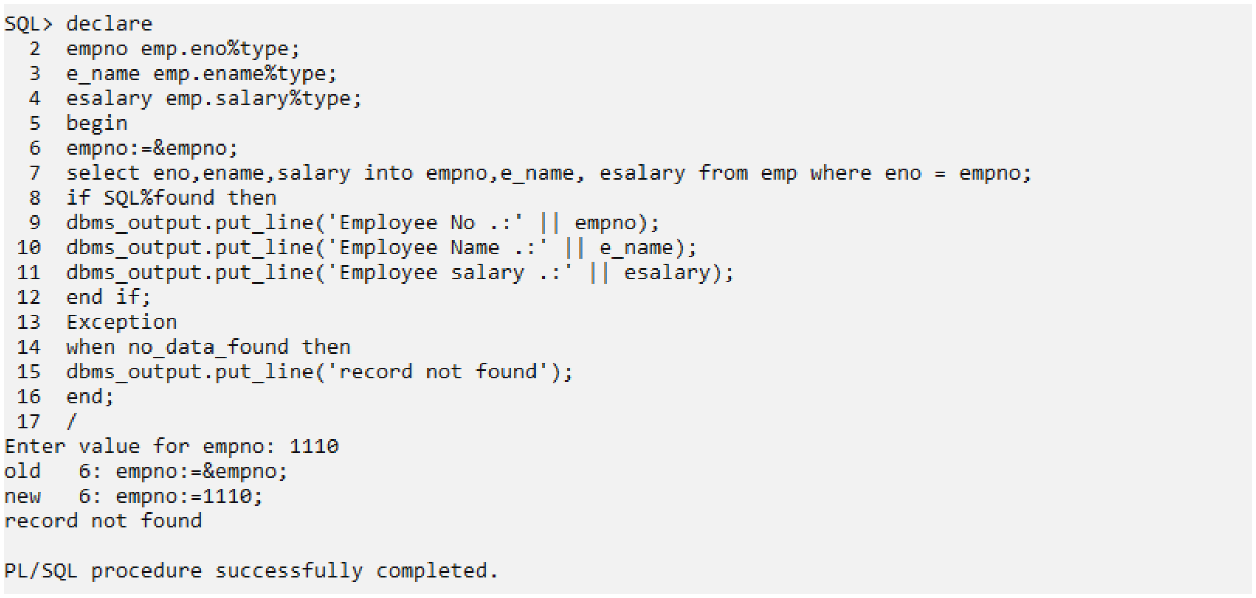
Code:

```
declare
        empno emp.eno%type;
        e_name emp.ename%type;
        esalary emp.salary%type;
begin
        empno:=&empno;
        select eno,ename,salary into empno,e_name, esalary from emp where eno = empno;
        if SQL%found then
                dbms_output.put_line('Employee No .:' || empno);
                dbms_output.put_line('Employee Name .:' || e_name);
                dbms_output.put_line('Employee salary .:' || esalary);
        end if;
Exception
when no_data_found then
                dbms_output.put_line('record not found');
end;
/
```

Output:



```
SQL> set serveroutput on;
SQL> declare
  2  empno emp.eno%type;
  3  e_name emp.ename%type;
  4  esalary emp.salary%type;
  5  begin
  6  empno:=&empno;
  7  select eno,ename,salary into empno,e_name, esalary from emp where eno = empno;
  8  if SQL%found then
  9  dbms_output.put_line('Employee No .:' || empno);
 10  dbms_output.put_line('Employee Name .:' || e_name);
 11  dbms_output.put_line('Employee salary .:' || esalary);
 12  end if;
 13  Exception
 14  when no_data_found then
 15  dbms_output.put_line('record not found');
 16  end;
 17  /
Enter value for empno: 101
old   6: empno:=&empno;
new   6: empno:=101;
Employee No .:101
Employee Name .:Karan
Employee salary .:50000

PL/SQL procedure successfully completed.
```



```
SQL> declare
  2  empno emp.eno%type;
  3  e_name emp.ename%type;
  4  esalary emp.salary%type;
  5  begin
  6  empno:=&empno;
  7  select eno,ename,salary into empno,e_name, esalary from emp where eno = empno;
  8  if SQL%found then
  9  dbms_output.put_line('Employee No .:' || empno);
 10  dbms_output.put_line('Employee Name .:' || e_name);
 11  dbms_output.put_line('Employee salary .:' || esalary);
 12  end if;
 13  Exception
 14  when no_data_found then
 15  dbms_output.put_line('record not found');
 16  end;
 17  /
Enter value for empno: 1110
old   6: empno:=&empno;
new   6: empno:=1110;
record not found

PL/SQL procedure successfully completed.
```

<u>Unnamed System Exceptions</u>

Those system exception for which oracle does not provide a name is known as unnamed system exception

Syntax:

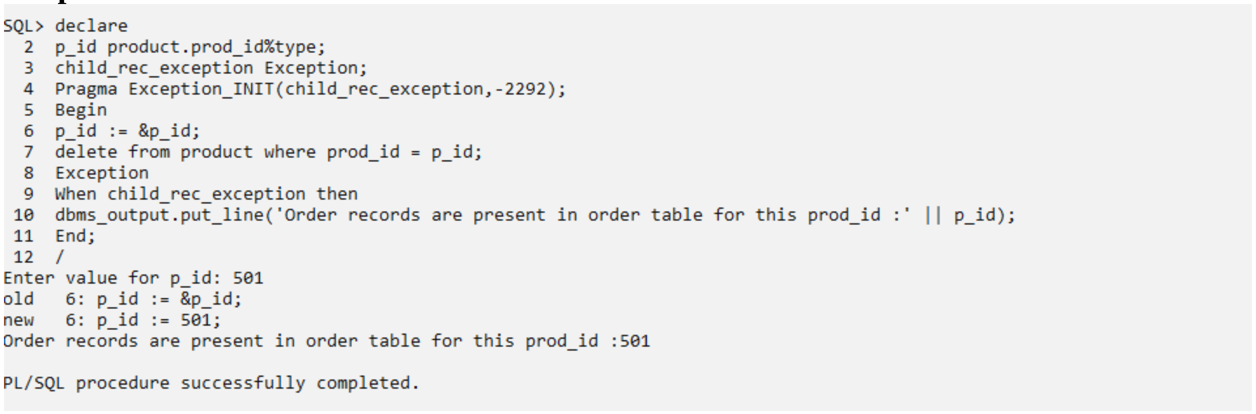
```
Declare
        exception _name Exception;
Pragma
        Exception_init(exception, err_code);
Begin
        Execution section
        Exception when exception_name then handle the exception
End;
```

Pragma Exception_init : this directive binds a user defined exception to a particular error number.

Code:

```
declare
        p_id product.prod_id%type;
        child_rec_exception Exception;
        Pragma Exception_INIT(child_rec_exception,-2292);
Begin
        p_id := &p_id;
        delete from product where prod_id = p_id;
Exception
        When child_rec_exception then
        dbms_output.put_line('Order records are present in order table for this prod_id :' || p_id);
End;
/
;
/
```

Output:



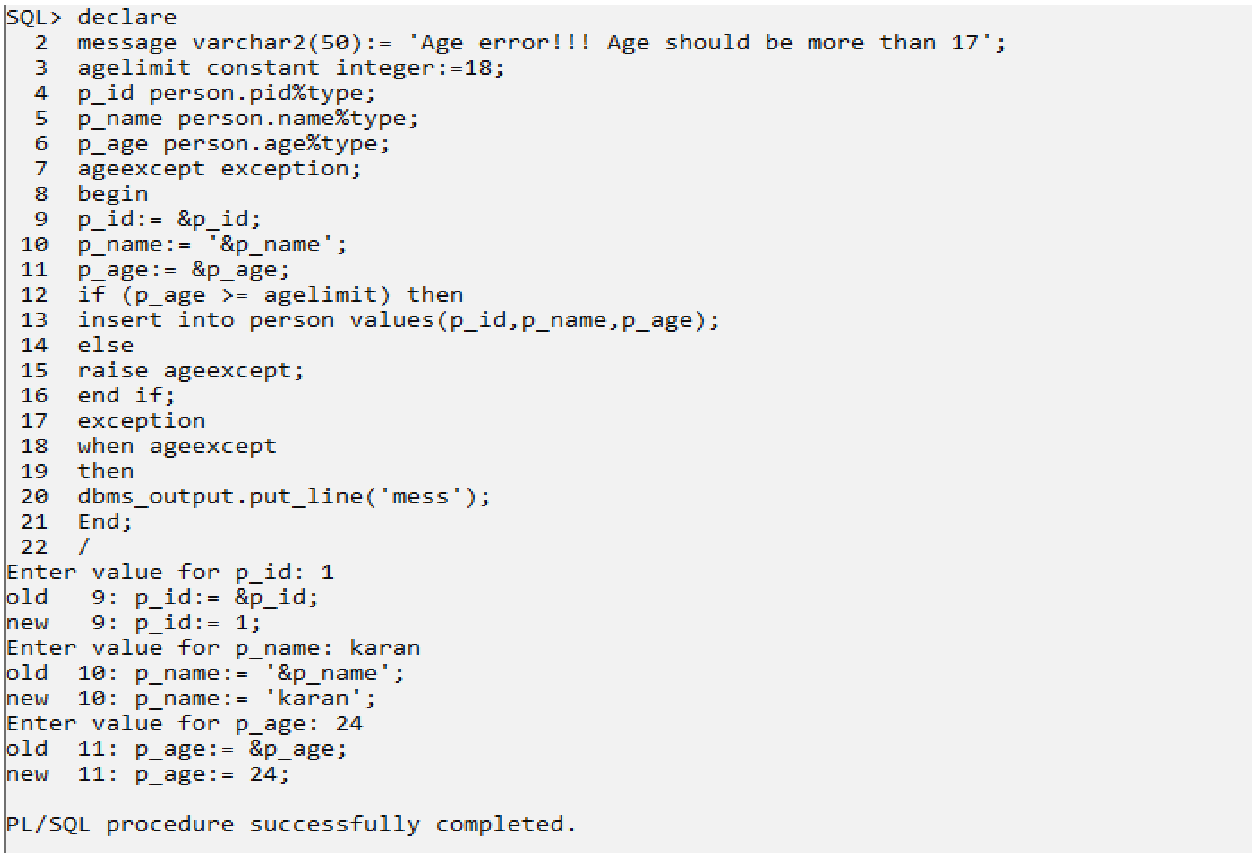
```
SQL> declare
  2  p_id product.prod_id%type;
  3  child_rec_exception Exception;
  4  Pragma Exception_INIT(child_rec_exception,-2292);
  5  Begin
  6  p_id := &p_id;
  7  delete from product where prod_id = p_id;
  8  Exception
  9  When child_rec_exception then
 10  dbms_output.put_line('Order records are present in order table for this prod_id :' || p_id);
 11  End;
 12  /
Enter value for p_id: 501
old   6: p_id := &p_id;
new   6: p_id := 501;
Order records are present in order table for this prod_id :501

PL/SQL procedure successfully completed.
```

<u>User defined Exception</u>

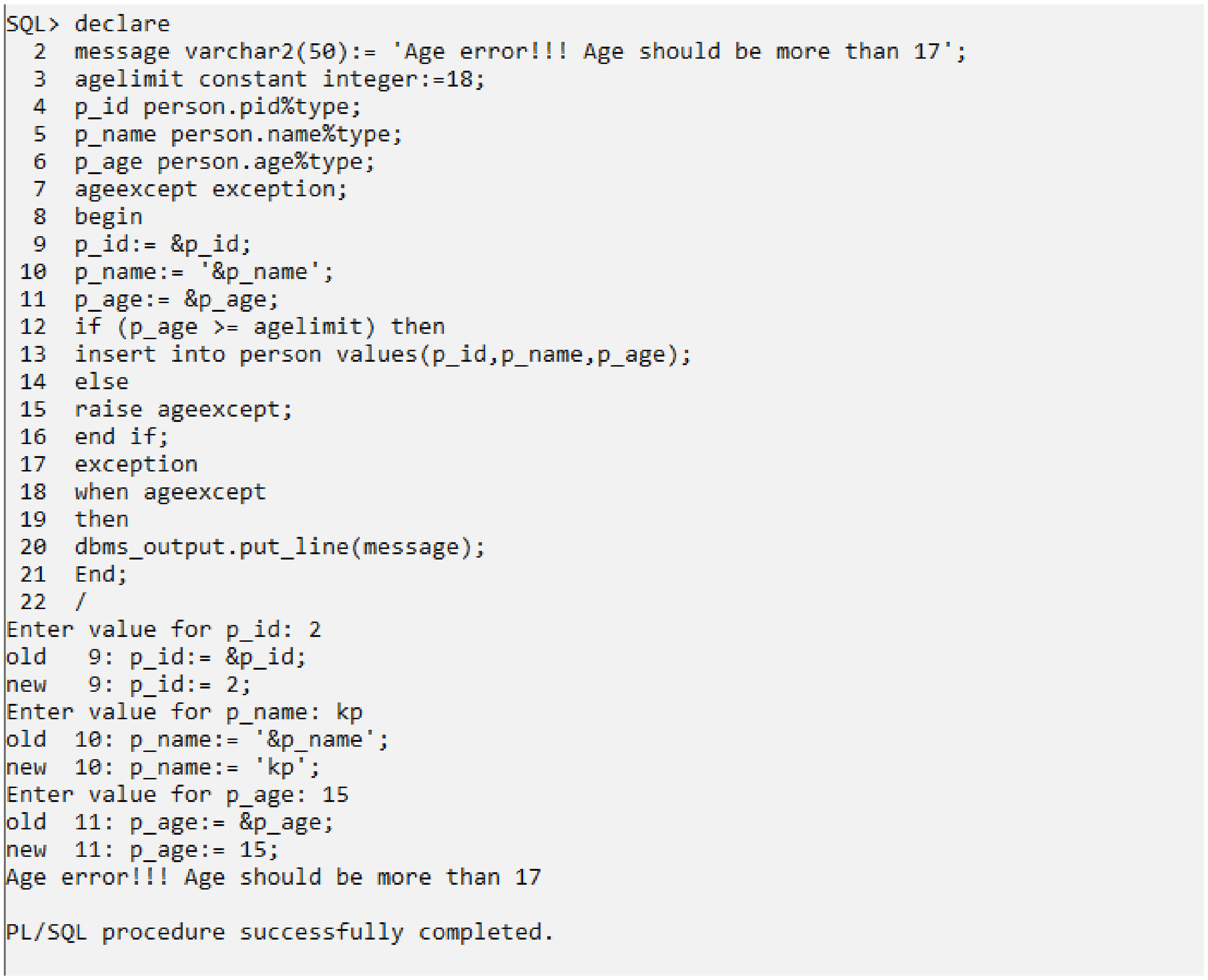
Code:

```
declare
        message varchar2(50):= 'Age error!!! Age should be more than 17';
        agelimit constant integer:=18;
        p_id person.pid%type;
        p_name person.name%type;
        p_age person.age%type;
        ageexcept exception;
begin
        p_id:= &p_id;
        p_name:= '&p_name';
        p_age:= &p_age;
        if (p_age >= agelimit) then
                insert into person values(p_id,p_name,p_age);
        else
        raise ageexcept;
        end if;
        exception
        when ageexcept
        then
        dbms_output.put_line(message);
End;
/
```

Output:



```
SQL> declare
  2  message varchar2(50):= 'Age error!!! Age should be more than 17';
  3  agelimit constant integer:=18;
  4  p_id person.pid%type;
  5  p_name person.name%type;
  6  p_age person.age%type;
  7  ageexcept exception;
  8  begin
  9  p_id:= &p_id;
 10  p_name:= '&p_name';
 11  p_age:= &p_age;
 12  if (p_age >= agelimit) then
 13  insert into person values(p_id,p_name,p_age);
 14  else
 15  raise ageexcept;
 16  end if;
 17  exception
 18  when ageexcept
 19  then
 20  dbms_output.put_line('mess');
 21  End;
 22  /
Enter value for p_id: 1
old   9: p_id:= &p_id;
new   9: p_id:= 1;
Enter value for p_name: karan
old  10: p_name:= '&p_name';
new  10: p_name:= 'karan';
Enter value for p_age: 24
old  11: p_age:= &p_age;
new  11: p_age:= 24;

PL/SQL procedure successfully completed.
```



```
SQL> declare
  2  message varchar2(50):= 'Age error!!! Age should be more than 17';
  3  agelimit constant integer:=18;
  4  p_id person.pid%type;
  5  p_name person.name%type;
  6  p_age person.age%type;
  7  ageexcept exception;
  8  begin
  9  p_id:= &p_id;
 10  p_name:= '&p_name';
 11  p_age:= &p_age;
 12  if (p_age >= agelimit) then
 13  insert into person values(p_id,p_name,p_age);
 14  else
 15  raise ageexcept;
 16  end if;
 17  exception
 18  when ageexcept
 19  then
 20  dbms_output.put_line(message);
 21  End;
 22  /
Enter value for p_id: 2
old   9: p_id:= &p_id;
new   9: p_id:= 2;
Enter value for p_name: kp
old  10: p_name:= '&p_name';
new  10: p_name:= 'kp';
Enter value for p_age: 15
old  11: p_age:= &p_age;
new  11: p_age:= 15;
Age error!!! Age should be more than 17

PL/SQL procedure successfully completed.
```

Practical 15 : Cursor

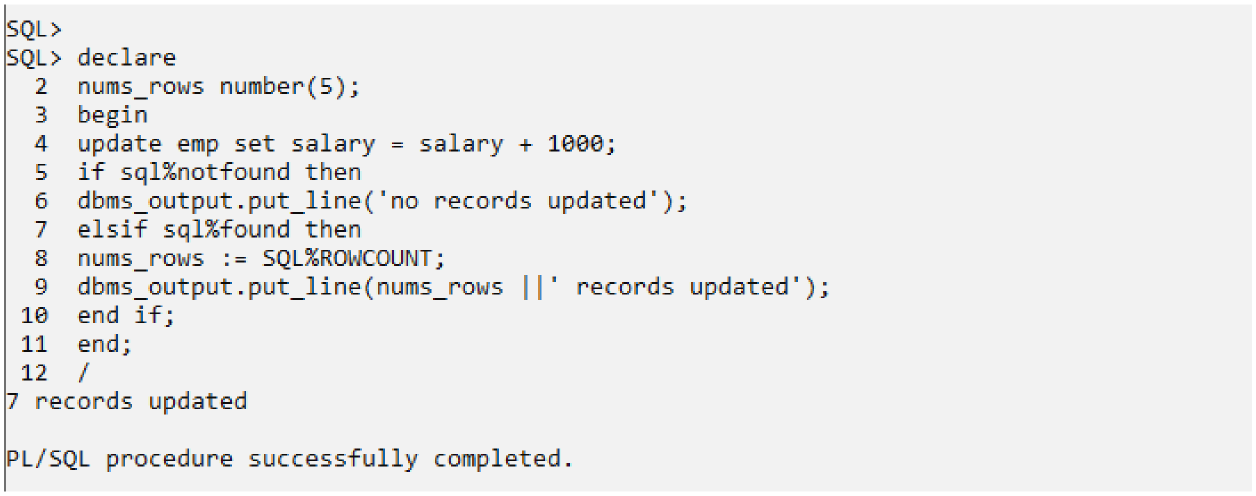
Cursor is a Temporary Memory or Temporary Work Station. It is Allocated by Database Server at the Time of Performing DML operations on the Table by the User. Cursors are used to store Database Tables.

a. Implicit Cursor

Query: Increase the salary of employees by Rs 1000
Code:
(Implicit cursors)
Syntax:

```
declare
        nums_rows number(5);
begin
update emp set salary = salary + 1000;
if sql%notfound then
dbms_output.put_line('no records updated');
elsif sql%found then
nums_rows := SQL%ROWCOUNT;
dbms_output.put_line(nums_rows ||' records updated');
end if;
end;
/
```

Output:



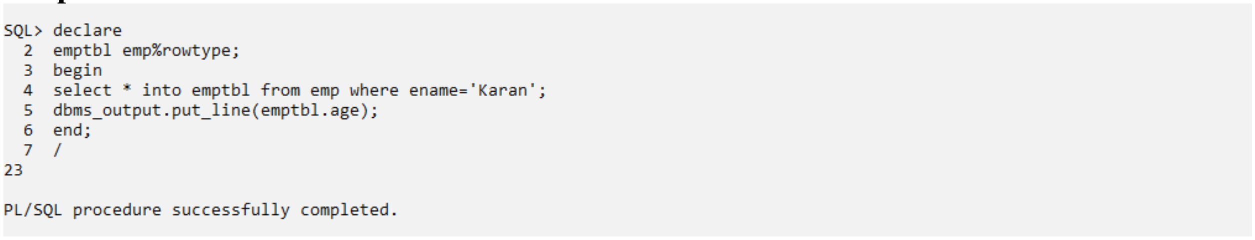
```
SQL>
SQL> declare
  2  nums_rows number(5);
  3  begin
  4  update emp set salary = salary + 1000;
  5  if sql%notfound then
  6  dbms_output.put_line('no records updated');
  7  elsif sql%found then
  8  nums_rows := SQL%ROWCOUNT;
  9  dbms_output.put_line(nums_rows ||' records updated');
 10  end if;
 11  end;
 12  /
7 records updated

PL/SQL procedure successfully completed.
```

Code:

```
declare
emptbl emp%rowtype;
begin
select * into emptbl from emp where ename='Karan';
dbms_output.put_line(emptbl.age);
```

end;
/

Output:

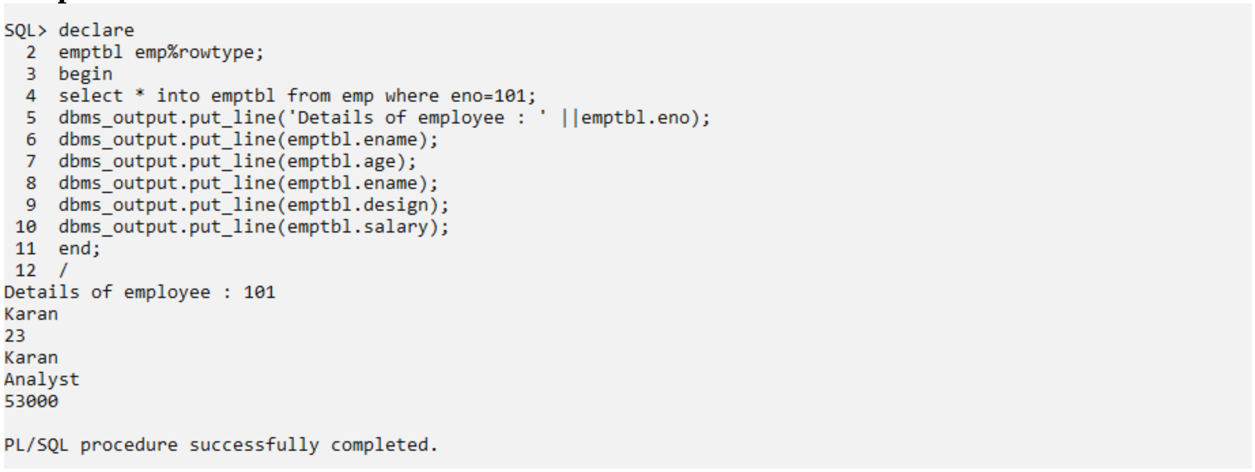


```
SQL> declare
emptbl emp%rowtype;
begin
select * into emptbl from emp where ename='Karan';
dbms_output.put_line(emptbl.age);
end;
/
23

PL/SQL procedure successfully completed.
```

Code:

```
declare
emptbl emp%rowtype;
begin
select * into emptbl from emp where eno=101;
dbms_output.put_line('Details of employee : ' ||emptbl.eno);
dbms_output.put_line(emptbl.ename);
dbms_output.put_line(emptbl.age);
dbms_output.put_line(emptbl.ename);
dbms_output.put_line(emptbl.design);
dbms_output.put_line(emptbl.salary);
end;
/
```

Output:



```
SQL> declare
emptbl emp%rowtype;
begin
select * into emptbl from emp where eno=101;
dbms_output.put_line('Details of employee : ' ||emptbl.eno);
dbms_output.put_line(emptbl.ename);
dbms_output.put_line(emptbl.age);
dbms_output.put_line(emptbl.ename);
dbms_output.put_line(emptbl.design);
dbms_output.put_line(emptbl.salary);
end;
/
Details of employee : 101
Karan
23
Karan
Analyst
53000

PL/SQL procedure successfully completed.
```

b. Explicit Cursor

Syntax: CURSOR cursor_name is Select_statement

<u>Four steps in using an explicit cursor</u>

1. Declare :
2. Open
3. Fetch
4. Close

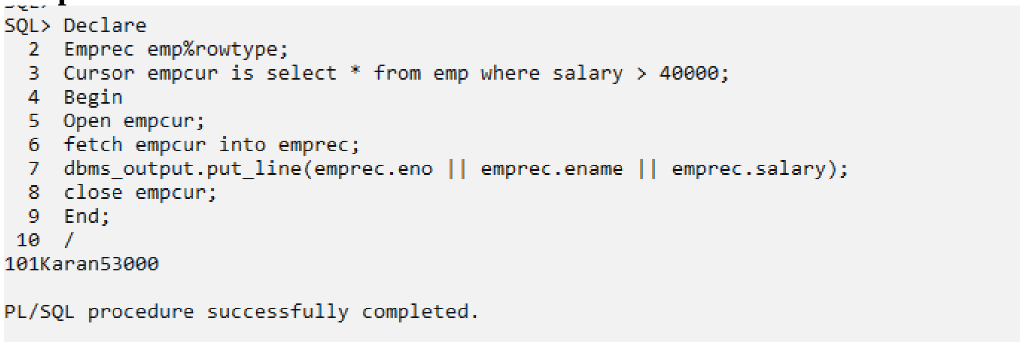
Syntax:

```
Declare

        Variables
        Records
        Create a cursor
Begin
        Open cursor;
        Fetch cursor;
        Process the statements
Close cursor
End;
```

Query: Display empno, fname and salary using cursor (one row only)
Code:

```
Declare
        Emprec emp%rowtype;
        Cursor empcur is select * from emp where salary > 40000;
Begin
        Open empcur;
        fetch empcur into emprec;
                dbms_output.put_line(emprec.eno || emprec.ename || emprec.salary);
        close empcur;
End;
/
```

Output:

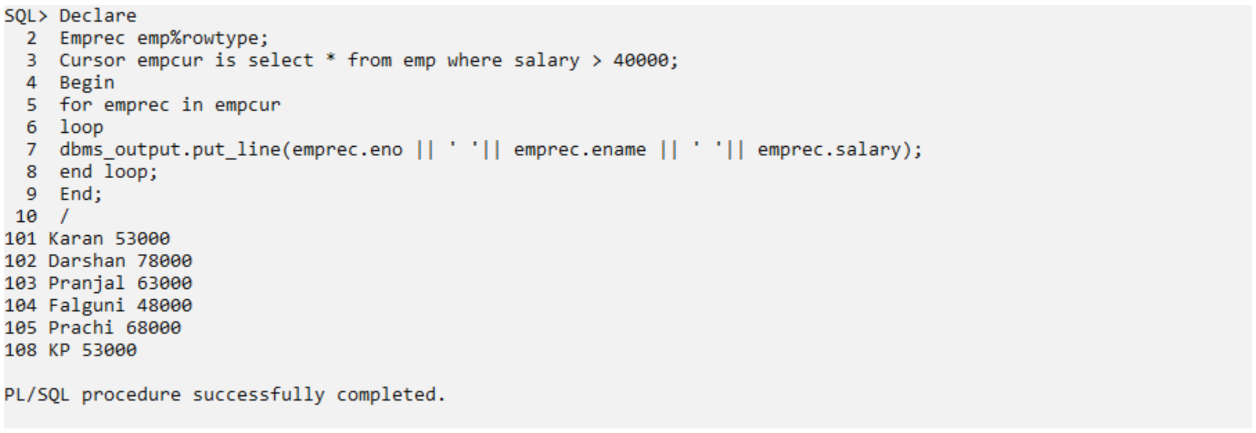


```
SQL> Declare
  2  Emprec emp%rowtype;
  3  Cursor empcur is select * from emp where salary > 40000;
  4  Begin
  5  Open empcur;
  6  fetch empcur into emprec;
  7  dbms_output.put_line(emprec.eno || emprec.ename || emprec.salary);
  8  close empcur;
  9  End;
 10  /
101Karan53000

PL/SQL procedure successfully completed.
```

<u>(multiple row)</u>
Code:

```
Declare
        Emprec emp%rowtype;
        Cursor empcur is select * from emp where salary > 40000;
Begin
        for emprec in empcur
        loop
                dbms_output.put_line(emprec.eno || ' '|| emprec.ename || ' '|| emprec.salary);
        end loop;
End;
/
```

Output:



```
SQL> Declare
  Emprec emp%rowtype;
  Cursor empcur is select * from emp where salary > 40000;
  Begin
  for emprec in empcur
  loop
  dbms_output.put_line(emprec.eno || ' '|| emprec.ename || ' '|| emprec.salary);
  end loop;
  End;
  /
101 Karan 53000
102 Darshan 78000
103 Pranjal 63000
104 Falguni 48000
105 Prachi 68000
108 KP 53000

PL/SQL procedure successfully completed.
```

Query: Cursor with parameters

Cursor cursor_name(parameter_list) is cursor_query

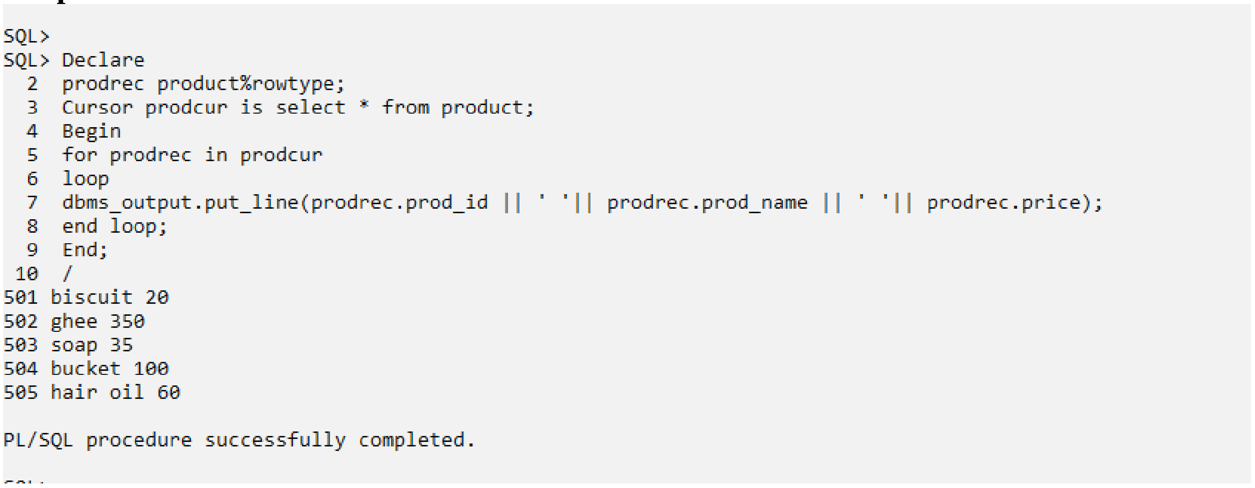
Code:

```
Declare
        Emprec emp%rowtype;
        eno_get emp.eno%type;
        Cursor empcur is select * from emp where eno = eno_get;
Begin
        eno_get:=&eno_get;
        for emprec in empcur(eno_get)
        loop
                dbms_output.put_line(emprec.eno || ' '|| emprec.ename || ' '|| emprec.salary);
        end loop;
End;
/
```

Query: write a PLSQL block to print the product details using cursor(loop)
Code:

```
Declare
        prodrec product%rowtype;
        Cursor prodcur is select * from product;
Begin
        for prodrec in prodcur
        loop
                dbms_output.put_line(prodrec.prod_id || ' '|| prodrec.prod_name || ' '||
prodrec.price);
        end loop;
End;
/
```

Output:



```
SQL>
SQL> Declare
  prodrec product%rowtype;
  Cursor prodcur is select * from product;
  Begin
  for prodrec in prodcur
  loop
  dbms_output.put_line(prodrec.prod_id || ' '|| prodrec.prod_name || ' '|| prodrec.price);
  end loop;
  End;
  /
501 biscuit 20
502 ghee 350
503 soap 35
504 bucket 100
505 hair oil 60

PL/SQL procedure successfully completed.
```

Practical 16 : Records

Records are composite datatypes which means it is a combination of different scaler types like char, varchar, number etc.
Each scalar data types in the record holds a value.
A record can be visualized as a row of data.

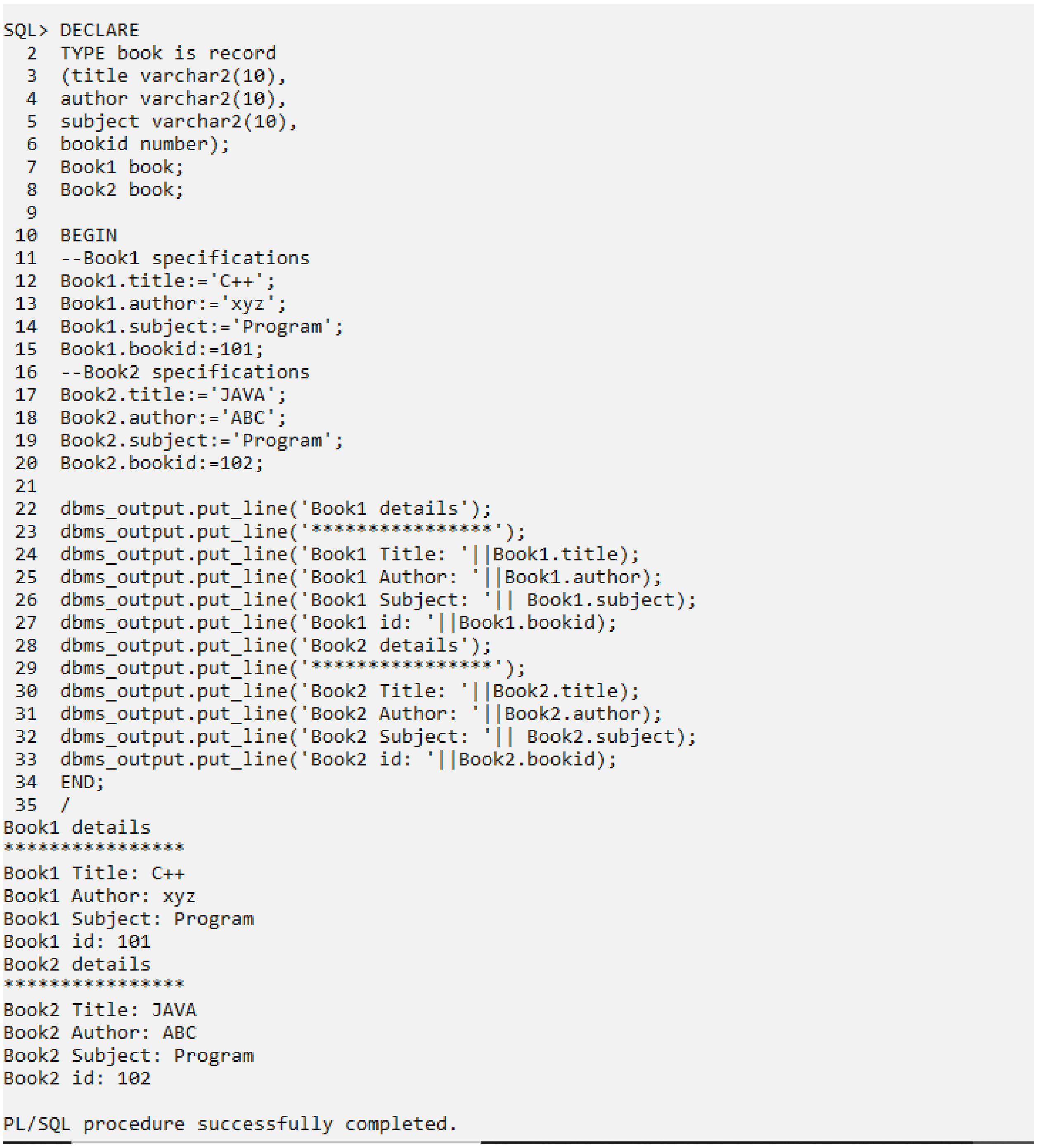
SYNTAX

```
Type_record_type_name IS RECORD
(first_col_name_column_datatype,
Second_col_name column_datatype,...);
```

Code:

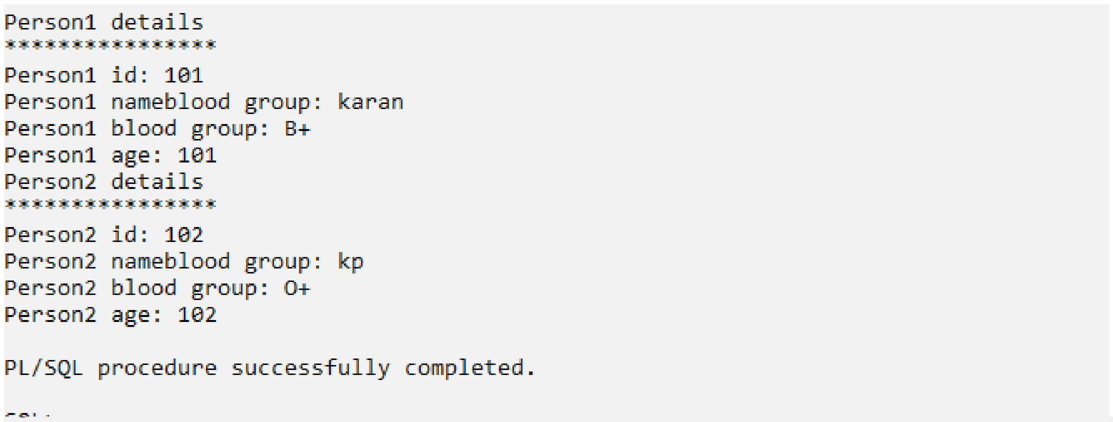
```
DECLARE
TYPE book is record
(title varchar2(10),
author varchar2(10),
subject varchar2(10),
bookid number);
Book1 book;
Book2 book;

BEGIN
--Book1 specifications
Book1.title:='C++';
Book1.author:='xyz';
Book1.subject:='Program';
Book1.bookid:=101;
--Book2 specifications
Book2.title:='JAVA';
Book2.author:='ABC';
Book2.subject:='Program';
Book2.bookid:=102;

dbms_output.put_line('Book1 details');
dbms_output.put_line('****************');
dbms_output.put_line('Book1 Title: '||Book1.title);
dbms_output.put_line('Book1 Author: '||Book1.author);
dbms_output.put_line('Book1 Subject: '|| Book1.subject);
dbms_output.put_line('Book1 id: '||Book1.bookid);
dbms_output.put_line('Book2 details');
dbms_output.put_line('****************');
dbms_output.put_line('Book2 Title: '||Book2.title);
dbms_output.put_line('Book2 Author: '||Book2.author);
dbms_output.put_line('Book2 Subject: '|| Book2.subject);
dbms_output.put_line('Book2 id: '||Book2.bookid);
END;
/
```

Output:

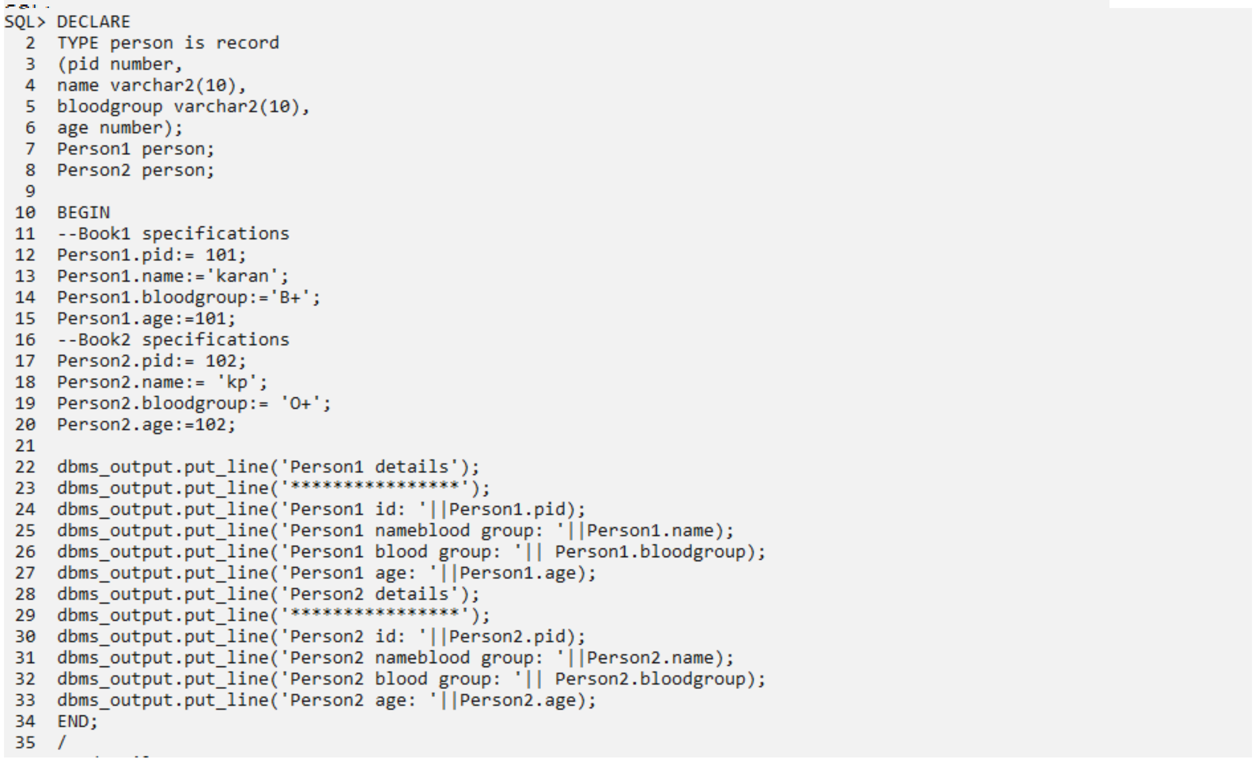


```
SQL> DECLARE
TYPE book is record
(title varchar2(10),
author varchar2(10),
subject varchar2(10),
bookid number);
Book1 book;
Book2 book;

BEGIN
--Book1 specifications
Book1.title:='C++';
Book1.author:='xyz';
Book1.subject:='Program';
Book1.bookid:=101;
--Book2 specifications
Book2.title:='JAVA';
Book2.author:='ABC';
Book2.subject:='Program';
Book2.bookid:=102;

dbms_output.put_line('Book1 details');
dbms_output.put_line('****************');
dbms_output.put_line('Book1 Title: '||Book1.title);
dbms_output.put_line('Book1 Author: '||Book1.author);
dbms_output.put_line('Book1 Subject: '|| Book1.subject);
dbms_output.put_line('Book1 id: '||Book1.bookid);
dbms_output.put_line('Book2 details');
dbms_output.put_line('****************');
dbms_output.put_line('Book2 Title: '||Book2.title);
dbms_output.put_line('Book2 Author: '||Book2.author);
dbms_output.put_line('Book2 Subject: '|| Book2.subject);
dbms_output.put_line('Book2 id: '||Book2.bookid);
END;
/
Book1 details
****************
Book1 Title: C++
Book1 Author: xyz
Book1 Subject: Program
Book1 id: 101
Book2 details
****************
Book2 Title: JAVA
Book2 Author: ABC
Book2 Subject: Program
Book2 id: 102

PL/SQL procedure successfully completed.
```

Query: Create a record to store person details pid, name , bloodgroup and age
Code:
```
DECLARE
TYPE person is record
(pid number,
name varchar2(10),
bloodgroup varchar2(10),
age number);
Person1 person;
Person2 person;
BEGIN
Person1.pid:= 101;
Person1.name:='karan';
Person1.bloodgroup:='B+';
Person1.age:=101;
Person2.pid:= 102;
Person2.name:= 'kp';
Person2.bloodgroup:= 'O+';
Person2.age:=102;
dbms_output.put_line('Person1 details');
dbms_output.put_line('****************');
dbms_output.put_line('Person1 id: '||Person1.pid);
dbms_output.put_line('Person1 nameblood group: '||Person1.name);
dbms_output.put_line('Person1 blood group: '|| Person1.bloodgroup);
dbms_output.put_line('Person1 age: '||Person1.age);
dbms_output.put_line('Person2 details');
dbms_output.put_line('****************');
dbms_output.put_line('Person2 id: '||Person2.pid);
dbms_output.put_line('Person2 nameblood group: '||Person2.name);
dbms_output.put_line('Person2 blood group: '|| Person2.bloodgroup);
dbms_output.put_line('Person2 age: '||Person2.age);
END;
/
```
Output:



```
Person1 details
****************
Person1 id: 101
Person1 nameblood group: karan
Person1 blood group: B+
Person1 age: 101
Person2 details
****************
Person2 id: 102
Person2 nameblood group: kp
Person2 blood group: O+
Person2 age: 102

PL/SQL procedure successfully completed.
```



```
SQL> DECLARE
TYPE person is record
(pid number,
name varchar2(10),
bloodgroup varchar2(10),
age number);
Person1 person;
Person2 person;

BEGIN
--Book1 specifications
Person1.pid:= 101;
Person1.name:='karan';
Person1.bloodgroup:='B+';
Person1.age:=101;
--Book2 specifications
Person2.pid:= 102;
Person2.name:= 'kp';
Person2.bloodgroup:= 'O+';
Person2.age:=102;

dbms_output.put_line('Person1 details');
dbms_output.put_line('****************');
dbms_output.put_line('Person1 id: '||Person1.pid);
dbms_output.put_line('Person1 nameblood group: '||Person1.name);
dbms_output.put_line('Person1 blood group: '|| Person1.bloodgroup);
dbms_output.put_line('Person1 age: '||Person1.age);
dbms_output.put_line('Person2 details');
dbms_output.put_line('****************');
dbms_output.put_line('Person2 id: '||Person2.pid);
dbms_output.put_line('Person2 nameblood group: '||Person2.name);
dbms_output.put_line('Person2 blood group: '|| Person2.bloodgroup);
dbms_output.put_line('Person2 age: '||Person2.age);
END;
/
```

Practical 17: Triggers

A trigger is a pl/sql block structure which is fired when a DML statements like insert, delete, update is executed on a database table

a. Trigger

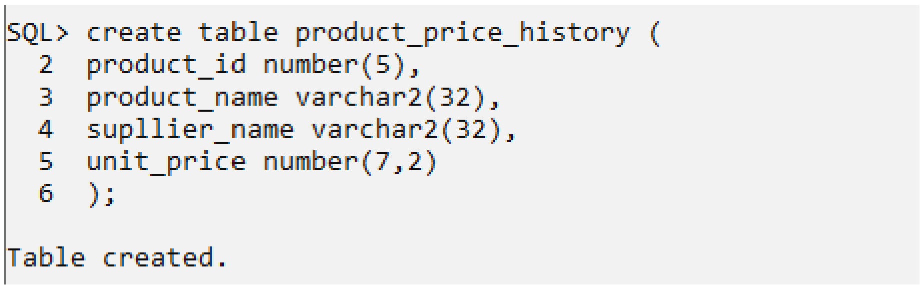
Syntax for creating a Trigger:

Syntax:
CREATE[OR REPLACE] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF}
{INSERT [OR] | UPDATE[OR] | DELETE}
[OF col_name] ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
BEGIN
–sql statements
END;

Code:

```
create table product_price_history (
product_id number(5),
product_name varchar2(32),
supllier_name varchar2(32),
unit_price number(7,2));
```

Output:

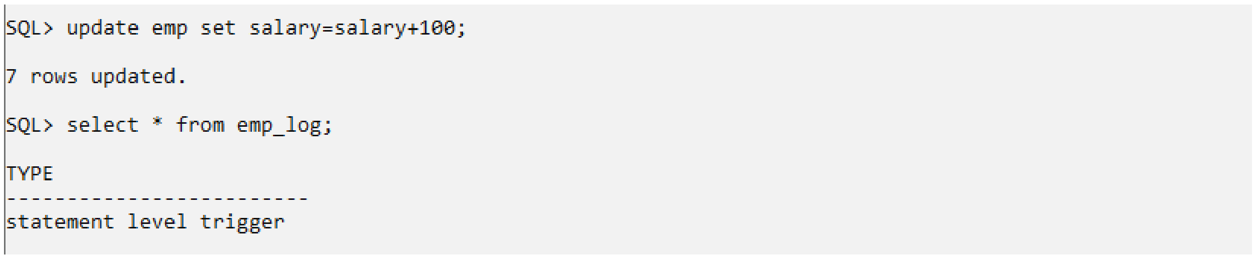


```
SQL> create table product_price_history (
  2  product_id number(5),
  3  product_name varchar2(32),
  4  supllier_name varchar2(32),
  5  unit_price number(7,2)
  6  );

Table created.
```

Code:

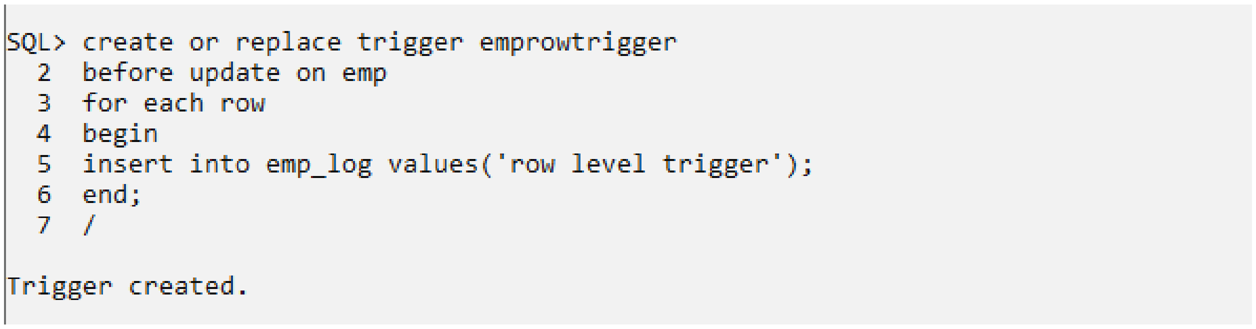
```
update emp set salary=salary+100;
select * from emp_log;
select * from emp_log;
```

Output:



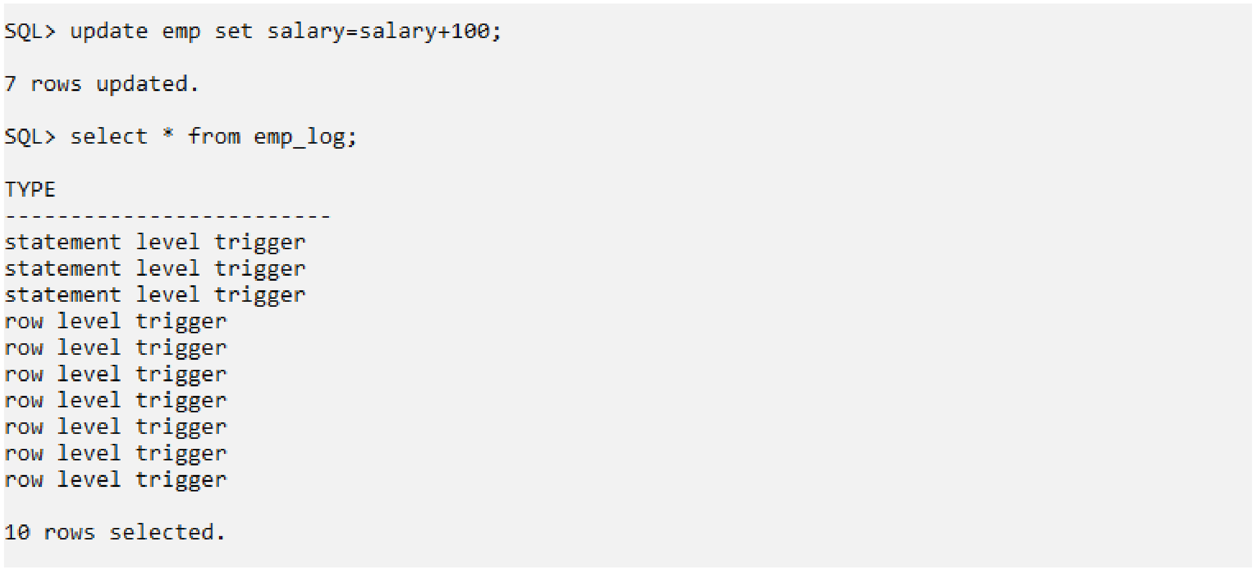
```
SQL> update emp set salary=salary+100;

7 rows updated.

SQL> select * from emp_log;

TYPE
-------------------------
statement level trigger
```

Code:
create or replace trigger emprowtrigger
before update on emp
for each row
begin
insert into emp_log values('row level trigger');
end;
Output:



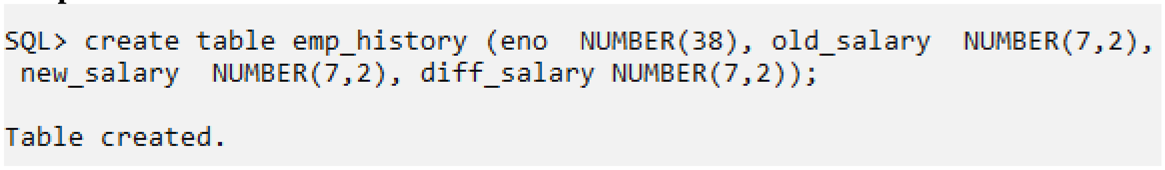
```
SQL> create or replace trigger emprowtrigger
  2  before update on emp
  3  for each row
  4  begin
  5  insert into emp_log values('row level trigger');
  6  end;
  7  /

Trigger created.
```

Code:
update emp set salary=salary+100;
select * from emp_log;
Output:



```
SQL> update emp set salary=salary+100;

7 rows updated.

SQL> select * from emp_log;

TYPE
-------------------------
statement level trigger
statement level trigger
statement level trigger
row level trigger
row level trigger
row level trigger
row level trigger
row level trigger
row level trigger
row level trigger

10 rows selected.
```

Code:
create table emp_history (eno NUMBER(38), old_salary NUMBER(7,2), new_salary NUMBER(7,2), diff_salary NUMBER(7,2));
Output:



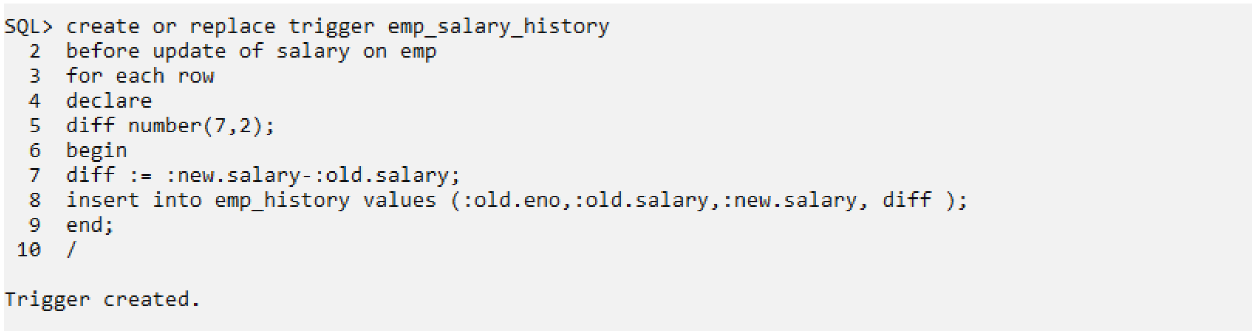
```
SQL> create table emp_history (eno  NUMBER(38), old_salary  NUMBER(7,2),
 new_salary  NUMBER(7,2), diff_salary NUMBER(7,2));

Table created.
```

b. Row Level Trigger
c. Statement Level Trigger

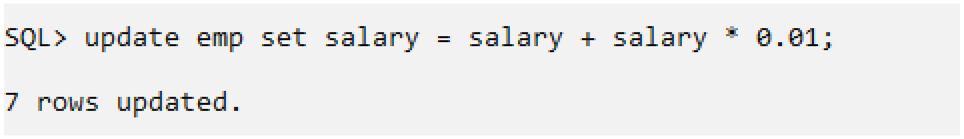
Code:

```
create or replace trigger emp_salary_history
before update of salary on emp
for each row
declare
diff number(7,2);
begin
diff := :new.salary-:old.salary;
insert into emp_history values (:old.eno,:old.salary,:new.salary, diff );
end;
/
```

Output:



```
SQL> create or replace trigger emp_salary_history
before update of salary on emp
for each row
declare
diff number(7,2);
begin
diff := :new.salary-:old.salary;
insert into emp_history values (:old.eno,:old.salary,:new.salary, diff );
end;
/

Trigger created.
```



```
SQL> update emp set salary = salary + salary * 0.01;

7 rows updated.
```

```
SQL> select * from emp_history;

       ENO OLD_SALARY NEW_SALARY DIFF_SALARY
---------- ---------- ---------- -----------
       101      53300      53833         533
       102      78300      79083         783
       103      63300      63933         633
       104      48300      48783         483
       105      68300      68983         683
       107      13300      13433         133
       108      53300      53833         533

7 rows selected.
```

Practical 18 : Functions

A subprogram is a return a value. Procedure : perform statement without return value

Syntax:

```
Create [OR REPLACE] Function_name [parameters]
Return return_datatype;
Is
Declaration_section
Begin
        Execution_section
        Return return_var;
Exception
        Execution section
        Return return_var
END;
```

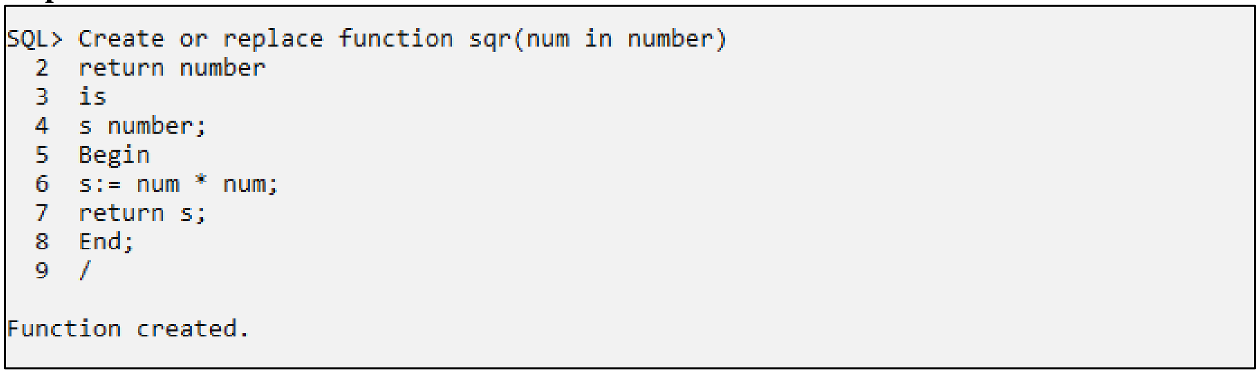
Drop function function_name;

a. Create Function

Code:

```
Create or replace function sqr(num in number) return number
Is
S number;
Begin
s:= num * num;
return s;
End;
/
```

Ouptut:



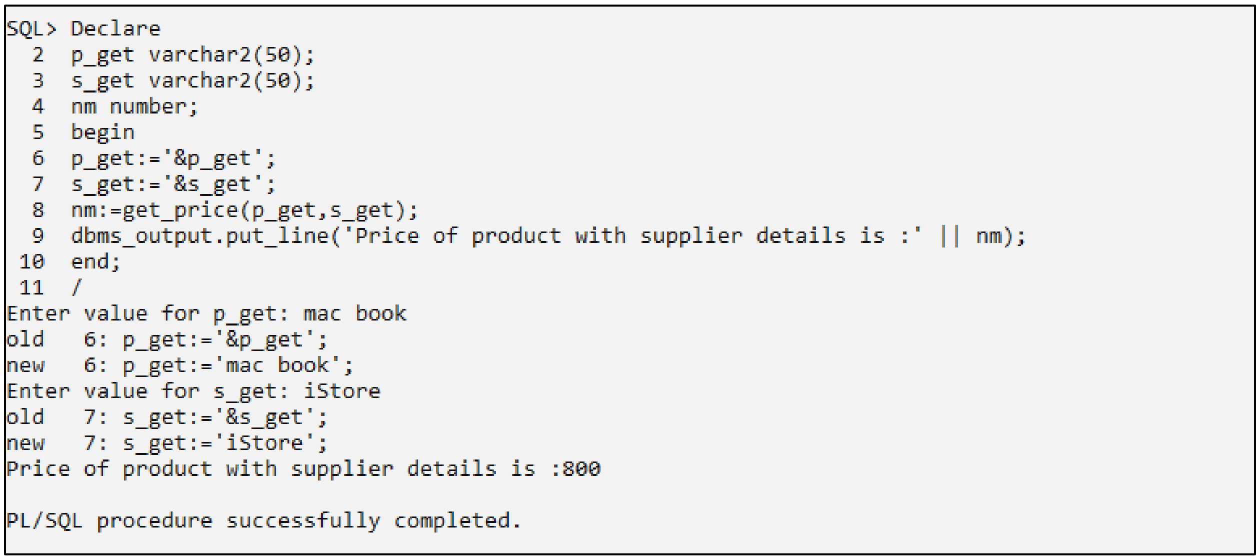
```
SQL> Create or replace function sqr(num in number)
  2  return number
  3  is
  4  s number;
  5  Begin
  6  s:= num * num;
  7  return s;
  8  End;
  9  /

Function created.
```

b. Function with Arguments

Query: PL/SQL block to call function
Code:

```
Declare
        p_get varchar2(50);
        s_get varchar2(50);
        nm number;
begin
        p_get:='&p_get';
        s_get:='&s_get';
        nm:=get_price(p_get,s_get);
        dbms_output.put_line('Price of product with supplier details is :' || nm);
end;
/
```

Output:



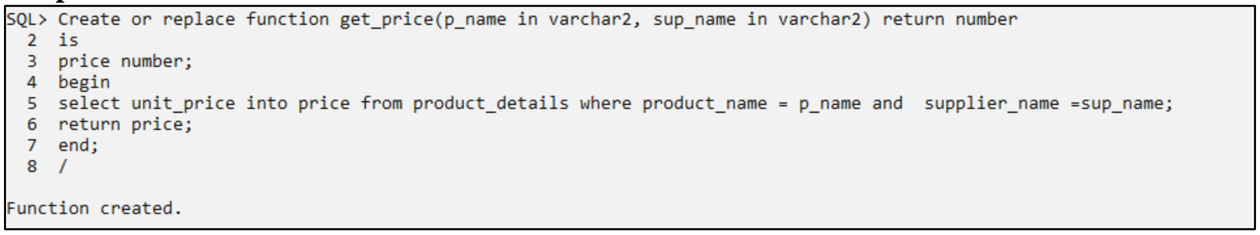
```
SQL> Declare
  2  p_get varchar2(50);
  3  s_get varchar2(50);
  4  nm number;
  5  begin
  6  p_get:='&p_get';
  7  s_get:='&s_get';
  8  nm:=get_price(p_get,s_get);
  9  dbms_output.put_line('Price of product with supplier details is :' || nm);
 10  end;
 11  /
Enter value for p_get: mac book
old   6: p_get:='&p_get';
new   6: p_get:='mac book';
Enter value for s_get: iStore
old   7: s_get:='&s_get';
new   7: s_get:='iStore';
Price of product with supplier details is :800

PL/SQL procedure successfully completed.
```

c. Executing Function

Query. create a function to accept product name and supplier name and display the unit price
Code:

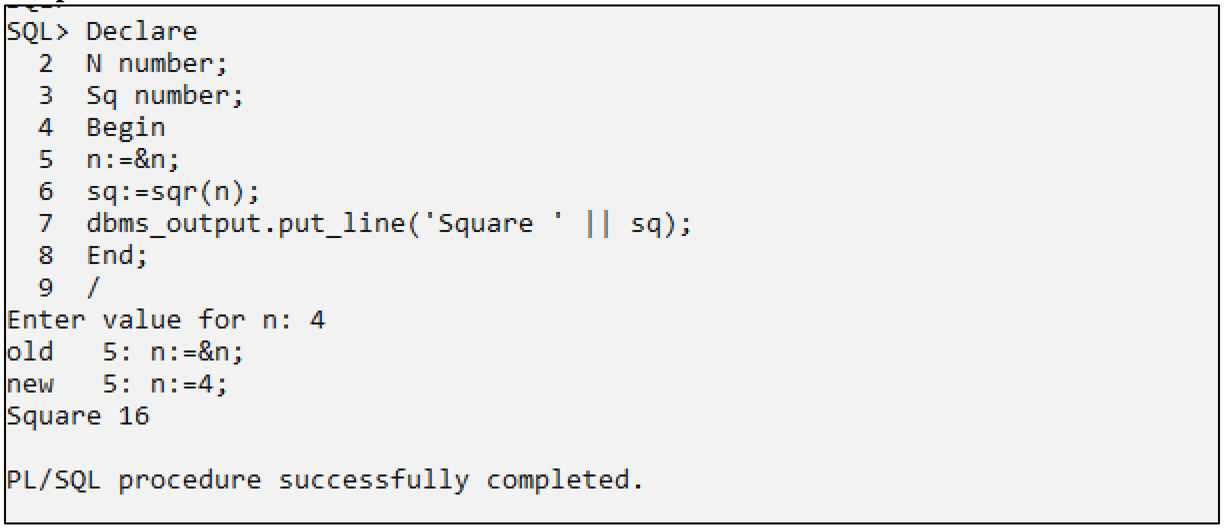
```
Create or replace function get_price(p_name in varchar2, sup_name in varchar2) return number
is
price number;
begin
select unit_price into price from product_details where product_name = p_name
and  supplier_name =sup_name;
return price;
```

end;
/
Output:



```
SQL> Create or replace function get_price(p_name in varchar2, sup_name in varchar2) return number
  2  is
  3  price number;
  4  begin
  5  select unit_price into price from product_details where product_name = p_name and  supplier_name =sup_name;
  6  return price;
  7  end;
  8  /

Function created.
```

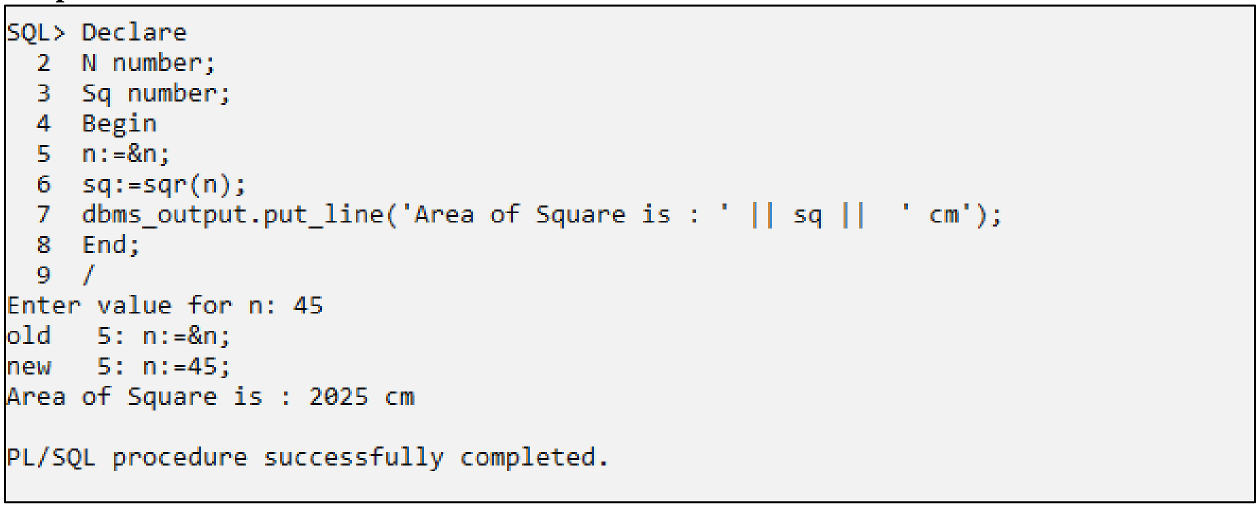
Code :
Declare
N number;
Sq number;
Begin
n:=&n;
sq:=sqr(n);
dbms_output.put_line('Square ' || sq);
End;
/
Output:



```
SQL> Declare
  2  N number;
  3  Sq number;
  4  Begin
  5  n:=&n;
  6  sq:=sqr(n);
  7  dbms_output.put_line('Square ' || sq);
  8  End;
  9  /
Enter value for n: 4
old   5: n:=&n;
new   5: n:=4;
Square 16

PL/SQL procedure successfully completed.
```

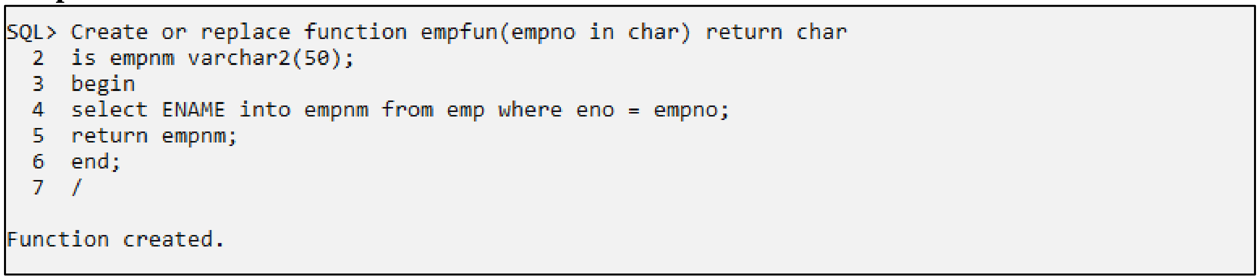
Query: write a function to accept the side of square and find area.
Code:
Declare
N number;
Sq number;
Begin
n:=&n;
sq:=sqr(n);
dbms_output.put_line('Area of Square is : ' || sq || ' cm');
End;

/
Output:



```
SQL> Declare
  2  N number;
  3  Sq number;
  4  Begin
  5  n:=&n;
  6  sq:=sqr(n);
  7  dbms_output.put_line('Area of Square is : ' || sq ||  ' cm');
  8  End;
  9  /
Enter value for n: 45
old   5: n:=&n;
new   5: n:=45;
Area of Square is : 2025 cm

PL/SQL procedure successfully completed.
```

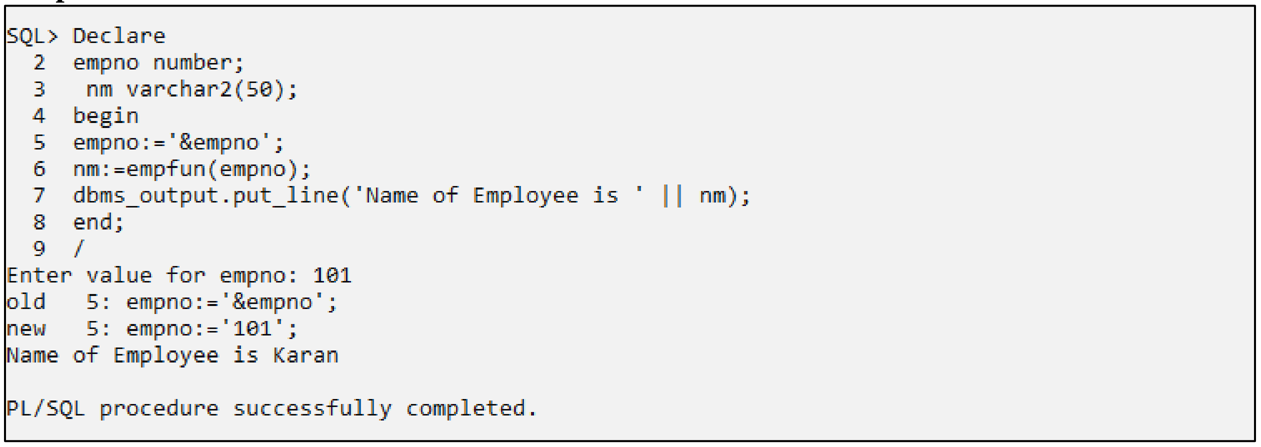
Query: Create a function to accept empno as a parameter and return empname
Code:
```
Create or replace function empfun(empno in char) return char
is empnm varchar2(50);
begin
select ENAME into empnm from emp where eno = empno;
return empnm;
end;
/
```
Output:



```
SQL> Create or replace function empfun(empno in char) return char
  2  is empnm varchar2(50);
  3  begin
  4  select ENAME into empnm from emp where eno = empno;
  5  return empnm;
  6  end;
  7  /

Function created.
```

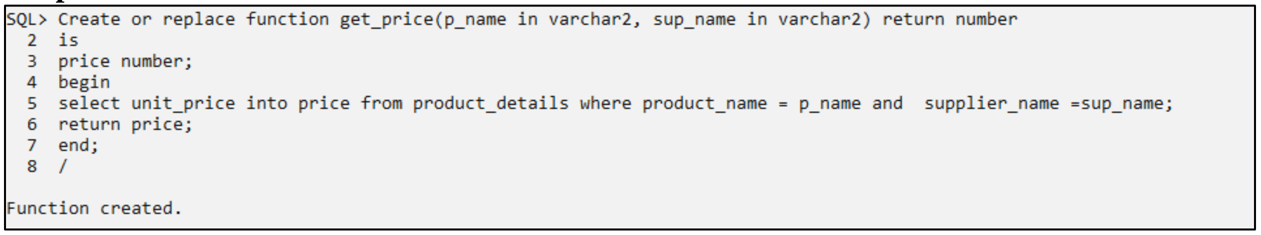
Query: PL/SQL block to call function
Code:
```
Declare
        empno number;
        nm varchar2(50);
begin
        empno:='&empno';
        nm:=empfun(empno);
        dbms_output.put_line('Name of Employee is ' || nm);
end;
/
```

Output:



```
SQL> Declare
  2  empno number;
  3   nm varchar2(50);
  4  begin
  5  empno:='&empno';
  6  nm:=empfun(empno);
  7  dbms_output.put_line('Name of Employee is ' || nm);
  8  end;
  9  /
Enter value for empno: 101
old   5: empno:='&empno';
new   5: empno:='101';
Name of Employee is Karan

PL/SQL procedure successfully completed.
```

Query: create a function to accept product name and supplier name and display the unit price
Code:
Create or replace function get_price(p_name in varchar2, sup_name in varchar2) return number
is
price number;
begin
select unit_price into price from product_details where product_name = p_name
and supplier_name =sup_name;
return price;
end;
/
Output:



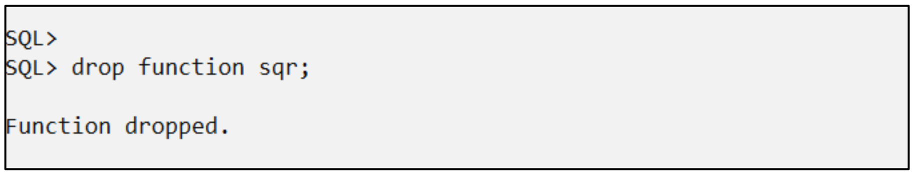
```
SQL> Create or replace function get_price(p_name in varchar2, sup_name in varchar2) return number
  2  is
  3  price number;
  4  begin
  5  select unit_price into price from product_details where product_name = p_name and  supplier_name =sup_name;
  6  return price;
  7  end;
  8  /

Function created.
```

d. Dropping Function

Code:

drop function sqr;

Output:



```
SQL>
SQL> drop function sqr;

Function dropped.
```

Practical 19 : Procedure

Syntax:
CREATE [OR REPLACE] PROCEDURE procedure_name
[(parameter_name [IN | OUT | IN OUT] type [,....])]
{IS | AS}
BEGIN
<PROCEDURE_BODY>
END PROCEDURE_NAME;

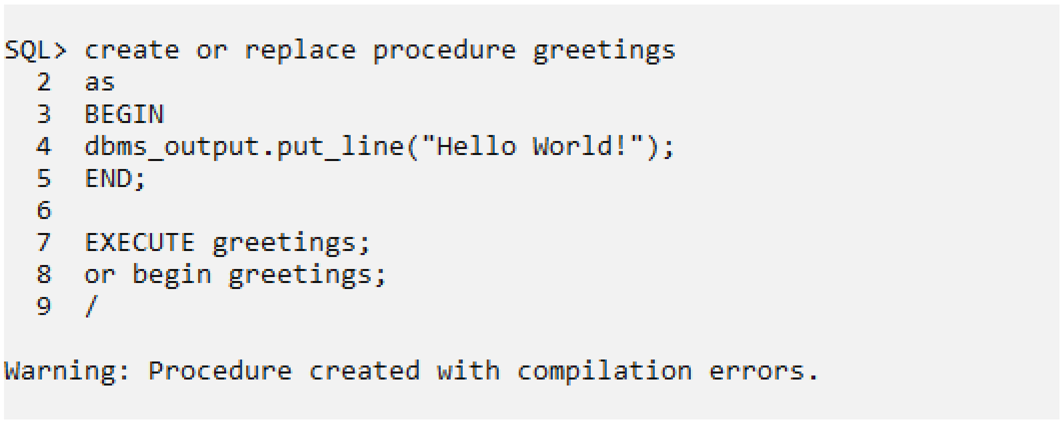
Procedure-name sepcifices the name of the procedure
[OR REPLACE]

a. Executing Procedures

Query: Simple store procedure
Code:

```
create or replace procedure greetings()
as
BEGIN
        dbms_output.put_line("Hello World!");
END;

EXECUTE greetings;
or begin greetings;
/
```

Output:



```
SQL> create or replace procedure greetings
  2  as
  3  BEGIN
  4  dbms_output.put_line("Hello World!");
  5  END;
  6
  7  EXECUTE greetings;
  8  or begin greetings;
  9  /

Warning: Procedure created with compilation errors.
```

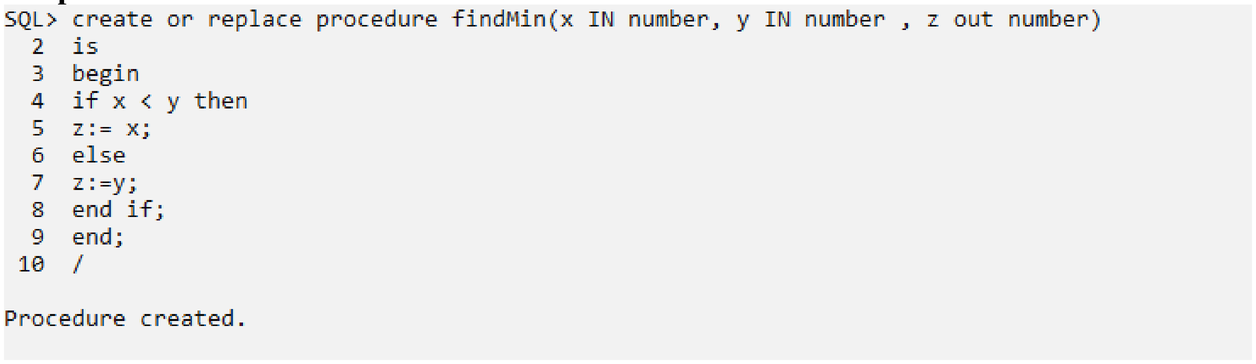
b. Procedure with Parameters

Query. Stored Procedure to find minimum number

Code:

```
create or replace procedure findMin(x IN number, y IN number , z out number)
is
begin
        if x < y then
                z:= x;
        else
                z:=y;
        end if;
end;
/
```

Output:

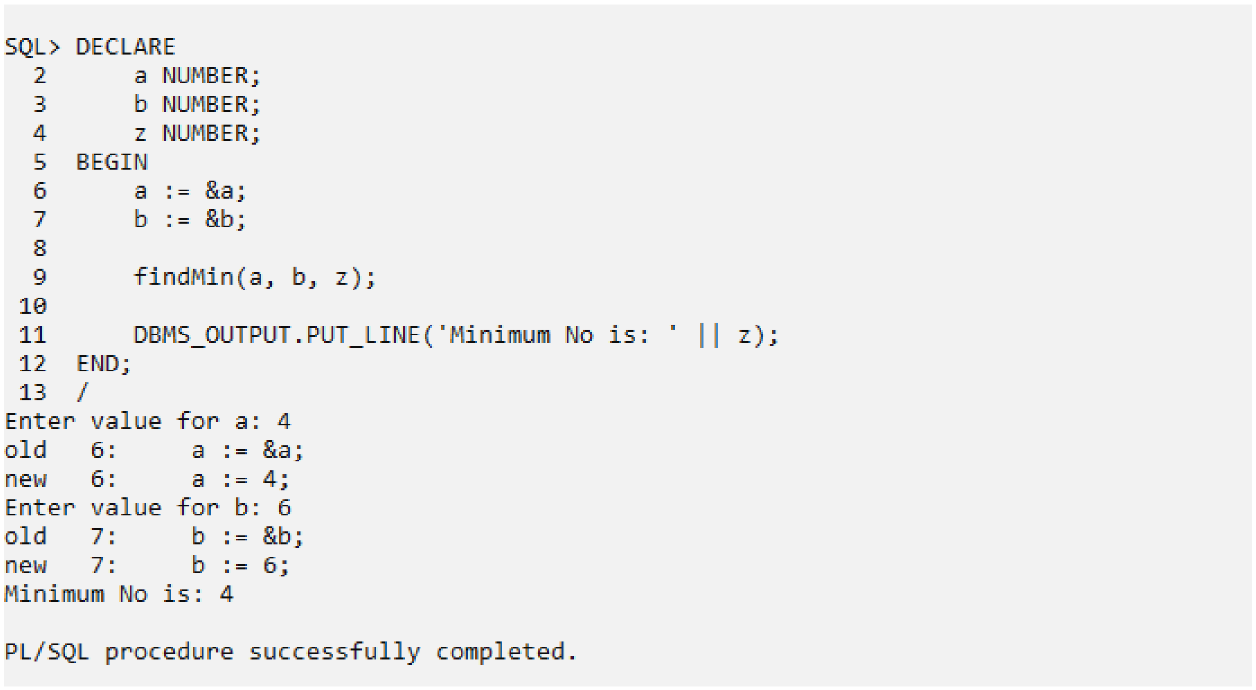


```
SQL> create or replace procedure findMin(x IN number, y IN number , z out number)
is
begin
if x < y then
z:= x;
else
z:=y;
end if;
end;
/

Procedure created.
```

Code:

```
declare
        a number;
        b number;
        z number;
begin
        a:=&a;
        b:=&b;
         findMin(a,b,z);
        dbms_output.put_line("Minimum No is :" || z);
end;
/
```

Output:



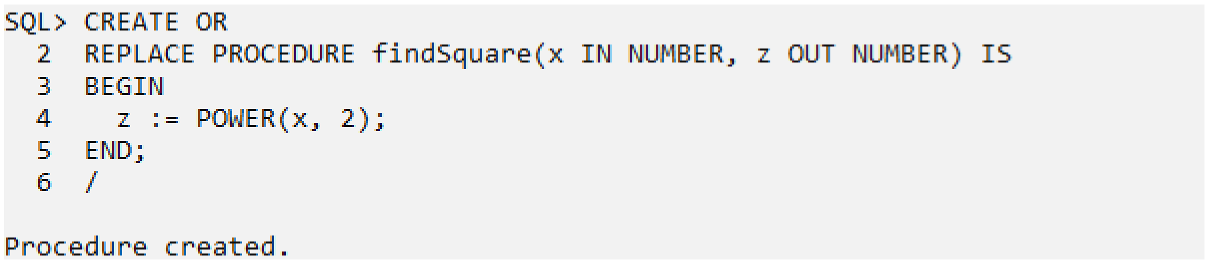
```
SQL> DECLARE
  2      a NUMBER;
  3      b NUMBER;
  4      z NUMBER;
  5  BEGIN
  6      a := &a;
  7      b := &b;
  8
  9      findMin(a, b, z);
 10
 11      DBMS_OUTPUT.PUT_LINE('Minimum No is: ' || z);
 12  END;
 13  /
Enter value for a: 4
old   6:     a := &a;
new   6:     a := 4;
Enter value for b: 6
old   7:     b := &b;
new   7:     b := 6;
Minimum No is: 4

PL/SQL procedure successfully completed.
```

Query: Stored Procedure to find square of 10

Code:

```
CREATE OR
REPLACE PROCEDURE findSquare(x IN NUMBER, z OUT NUMBER) IS
BEGIN
  z := POWER(x,
END;
/
```

Output:



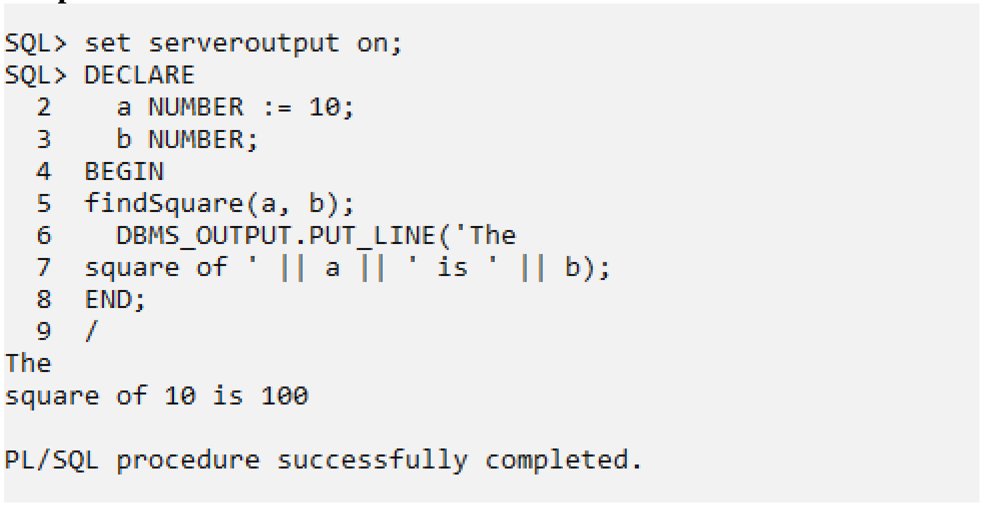
```
SQL> CREATE OR
  2  REPLACE PROCEDURE findSquare(x IN NUMBER, z OUT NUMBER) IS
  3  BEGIN
  4    z := POWER(x, 2);
  5  END;
  6  /

Procedure created.
```

Code:

```
DECLARE
  a NUMBER := 10;
  b NUMBER;
BEGIN
```

```
        findSquare(a, b);
  DBMS_OUTPUT.PUT_LINE('The
square of ' || a || ' is ' || b);
END;
/
```

Output:



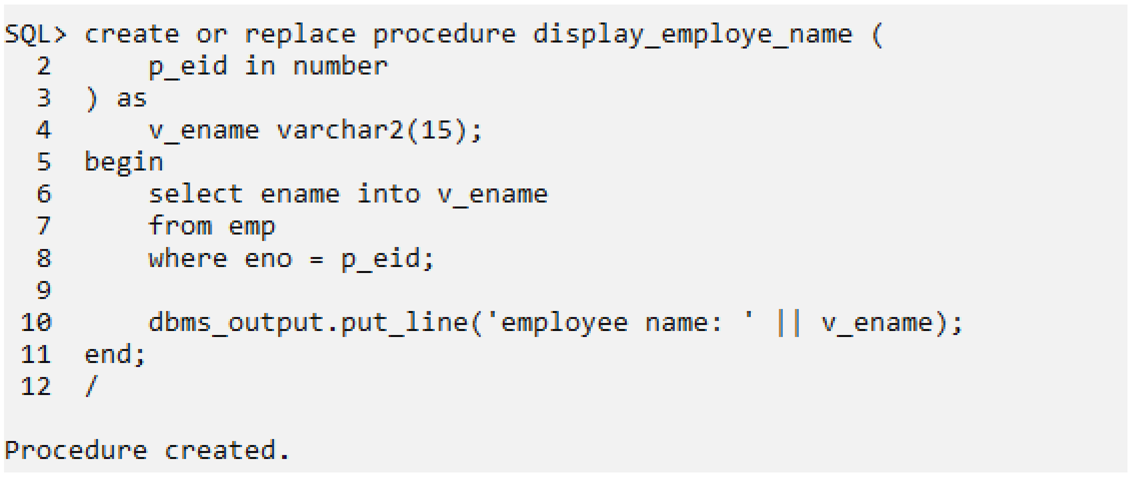
```
SQL> set serveroutput on;
SQL> DECLARE
  2    a NUMBER := 10;
  3    b NUMBER;
  4  BEGIN
  5  findSquare(a, b);
  6    DBMS_OUTPUT.PUT_LINE('The
  7  square of ' || a || ' is ' || b);
  8  END;
  9  /
The
square of 10 is 100

PL/SQL procedure successfully completed.
```

Query :Procedure to accept employee number from user and display the name

Code:

```
create or replace procedure display_employe_name (
    p_eid in number
) as
    v_ename varchar2(15);
begin
    select ename into v_ename
    from emp
    where eno = p_eid;

    dbms_output.put_line('employee name: ' || v_ename);
end;
/
```

Output:

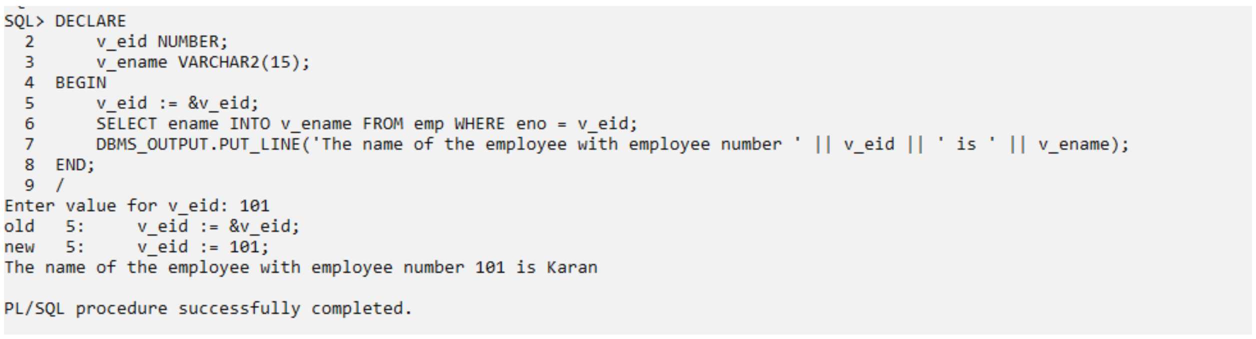


```
SQL> create or replace procedure display_employe_name (
  2      p_eid in number
  3  ) as
  4      v_ename varchar2(15);
  5  begin
  6      select ename into v_ename
  7      from emp
  8      where eno = p_eid;
  9
 10      dbms_output.put_line('employee name: ' || v_ename);
 11  end;
 12  /

Procedure created.
```

Code:

```
DECLARE
   v_eid NUMBER;
   v_ename VARCHAR2(15);
BEGIN
   v_eid := &v_eid;
   SELECT ename INTO v_ename FROM emp WHERE eno = v_eid;
   DBMS_OUTPUT.PUT_LINE('The name of the employee with employee number ' || v_eid || '
is ' || v_ename);
END;
/
```

Output:

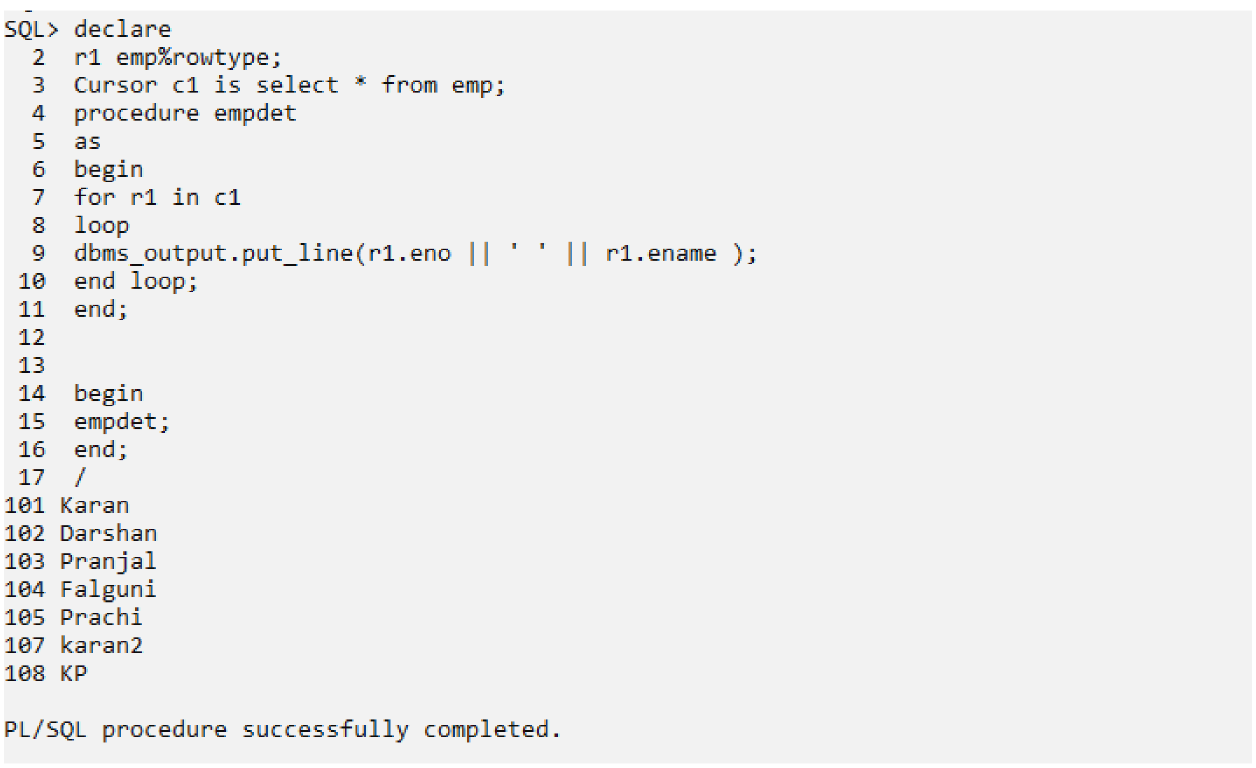


```
SQL> DECLARE
  2      v_eid NUMBER;
  3      v_ename VARCHAR2(15);
  4  BEGIN
  5      v_eid := &v_eid;
  6      SELECT ename INTO v_ename FROM emp WHERE eno = v_eid;
  7      DBMS_OUTPUT.PUT_LINE('The name of the employee with employee number ' || v_eid || ' is ' || v_ename);
  8  END;
  9  /
Enter value for v_eid: 101
old   5:     v_eid := &v_eid;
new   5:     v_eid := 101;
The name of the employee with employee number 101 is Karan

PL/SQL procedure successfully completed.
```

Code:

```
declare
r1 emp%rowtype;
Cursor c1 is select * from emp;
procedure empdet
as
begin
        for r1 in c1
        loop
                dbms_output.put_line(r1.eno || ' ' || r1.ename );
        end loop;
end;


begin
        empdet;
end;
/
```

Output:



```
SQL> declare
r1 emp%rowtype;
Cursor c1 is select * from emp;
procedure empdet
as
begin
for r1 in c1
loop
dbms_output.put_line(r1.eno || ' ' || r1.ename );
end loop;
end;


begin
empdet;
end;
/
101 Karan
102 Darshan
103 Pranjal
104 Falguni
105 Prachi
107 karan2
108 KP

PL/SQL procedure successfully completed.
```

Practical 20 : Packages

Packages are schema objects that groups logically related PL/SQL types, variables, and subprograms.

Syntax:

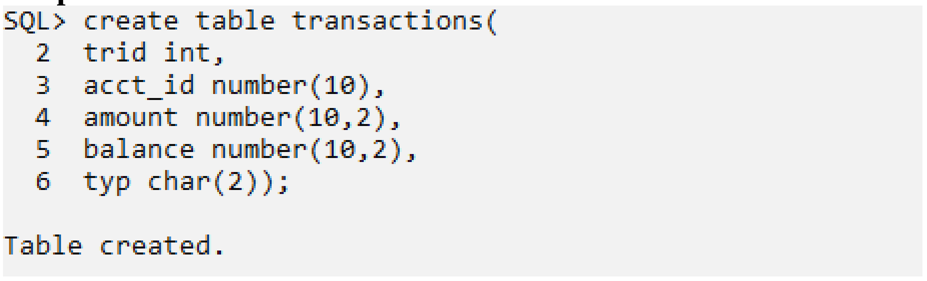
```
CREATE OR REPLACE PACKAGE package_name AS
   TYPE emp_type IS RECORD (
      emp_id NUMBER,
      emp_name VARCHAR2(100)
   );
   pi CONSTANT NUMBER := 3.14159;
   PROCEDURE display_employee_info(emp_id IN NUMBER);
END package_name;
/

-- Package Body
CREATE OR REPLACE PACKAGE BODY package_name AS
   PROCEDURE display_employee_info(emp_id IN NUMBER) IS
   BEGIN
      -- Implementation logic here
      NULL;
   END display_employee_info;
END package_name;
/
```

Query: Create table transaction

Code:

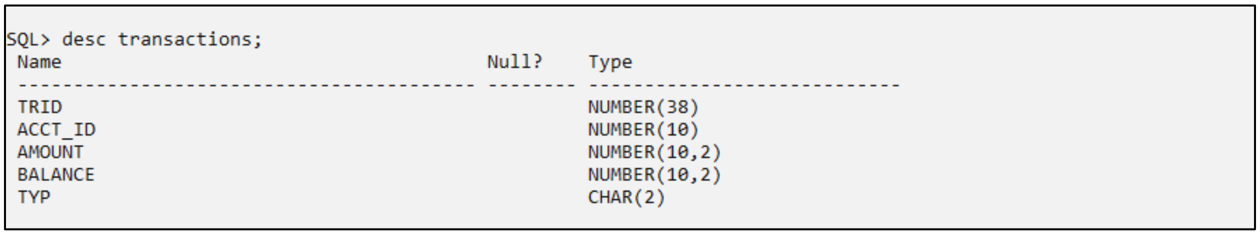
```
create table transactions(trid int, acct_id number(10),amount number(10,2), balance number(10,2),typ char(2));
insert into transactions values(12121212,9745645756,1000,100,'CR');
insert into transactions values(2323232,5634546446,300,200,'DR');
```

Output:



```
SQL> create table transactions(
  2  trid int,
  3  acct_id number(10),
  4  amount number(10,2),
  5  balance number(10,2),
  6  typ char(2));

Table created.
```

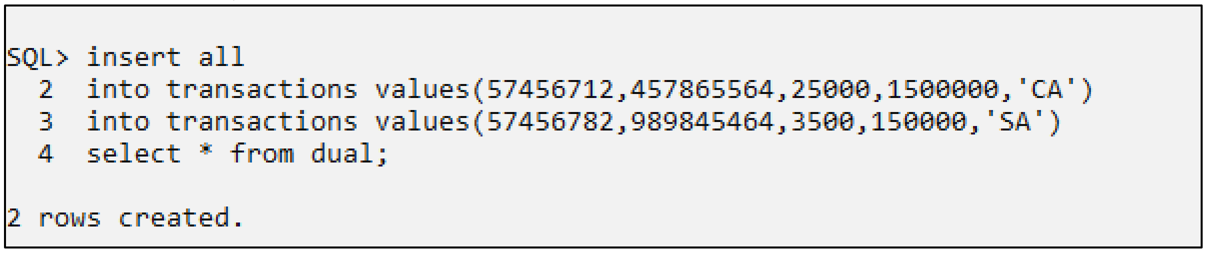
Code:



```
SQL> desc transactions;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 TRID                                               NUMBER(38)
 ACCT_ID                                            NUMBER(10)
 AMOUNT                                             NUMBER(10,2)
 BALANCE                                            NUMBER(10,2)
 TYP                                                CHAR(2)
```

Code:

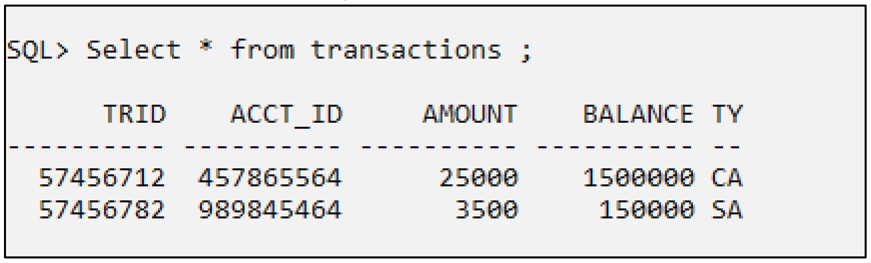
insert all
into transactions values(57456712,457865564,25000,1500000,'CA')
into transactions values(57456782,989845464,3500,150000,'SA')
select * from dual;



```
SQL> insert all
  2  into transactions values(57456712,457865564,25000,1500000,'CA')
  3  into transactions values(57456782,989845464,3500,150000,'SA')
  4  select * from dual;

2 rows created.
```

Code:

Select * from transactions ;



```
SQL> Select * from transactions ;

      TRID    ACCT_ID     AMOUNT    BALANCE TY
---------- ---------- ---------- ---------- --
  57456712  457865564      25000    1500000 CA
  57456782  989845464       3500     150000 SA
```

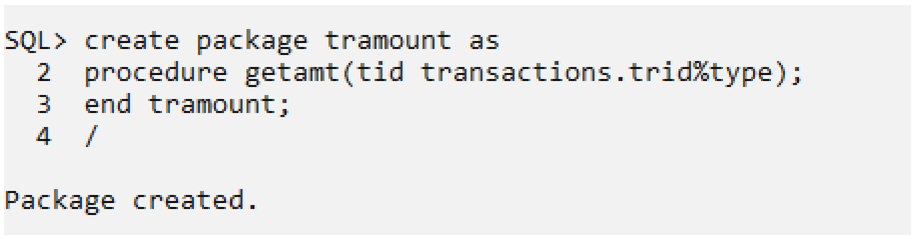
a. Creating Package

b. Package Body
c. Dropping Package

Code:

create or replace package tramount as
procedure getamt(tid transactions.trid%type);
end tramount;
/

Output:

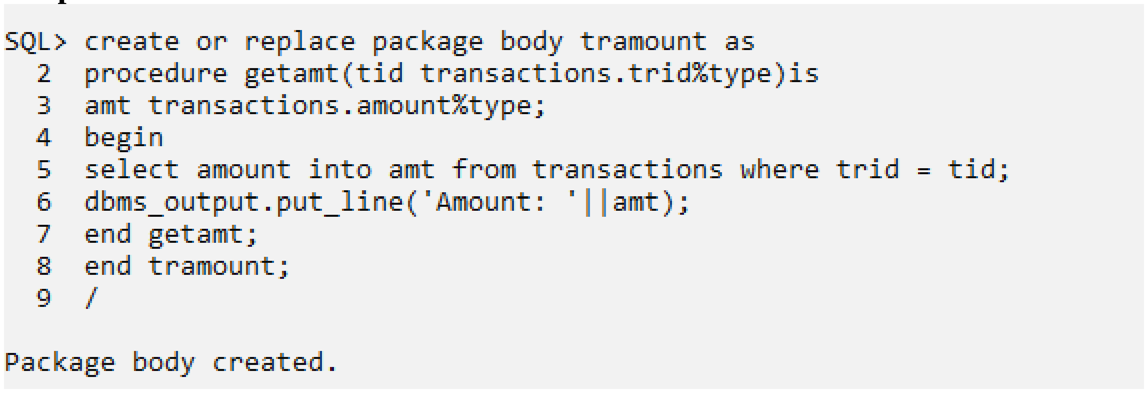


```
SQL> create package tramount as
procedure getamt(tid transactions.trid%type);
end tramount;
/

Package created.
```

Code:

```
create or replace package body tramount as
procedure getamt(tid transactions.trid%type)is
amt transactions.amount%type;
begin
select amount into amt from transactions where trid = tid;
dbms_output.put_line('Amount: '||amt);
end getamt;
end tramount;
/
```

Output:



```
SQL> create or replace package body tramount as
procedure getamt(tid transactions.trid%type)is
amt transactions.amount%type;
begin
select amount into amt from transactions where trid = tid;
dbms_output.put_line('Amount: '||amt);
end getamt;
end tramount;
/

Package body created.
```

Code:

```
declare
id transactions.trid%type:=&id;
begin
tramount.getamt(id);
end;
```

Output:

```
SQL> declare
  2  id transactions.trid%type:=&id;
  3  begin
  4  tramount.getamt(id);
  5  end;
  6  /
Enter value for id: 57456712
old   2: id transactions.trid%type:=&id;
new   2: id transactions.trid%type:=57456712;
Amount: 25000

PL/SQL procedure successfully completed.
```