DBMS JOURNAL

Name: Darshan K Panchal
Roll No: 32
Course: FY-MCA
Semester : I
Subject : Database Application

Practical no-1Study of DDL Commands

a. **Create**

Syntax:

Create table table_name(

Col1 datatype column constraint,

Col2 datatype column constraint,

Col2 datatype column constraint,

Description:

To create table command defines each column of the table uniquely.

1. Create table for Customer table

SQL> create table customers

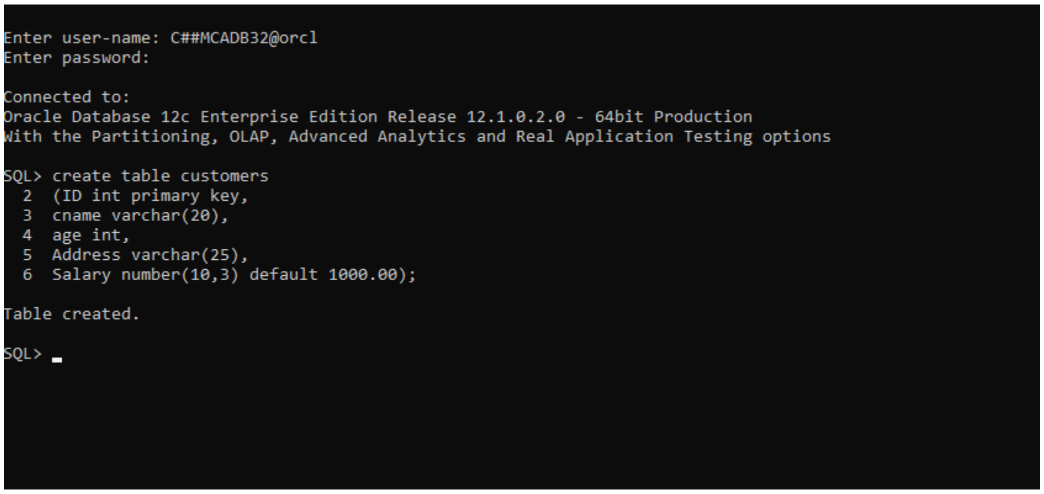
(ID int primary key,

cname varchar(20),

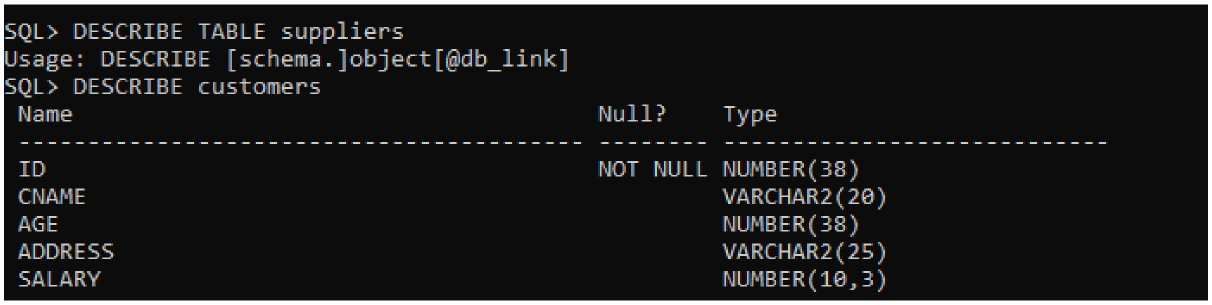
age int,

Address varchar(25),

Salary number(10,3) default 1000.00);



```
Enter user-name: C##MCADB32@orcl
Enter password:

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options

SQL> create table customers
  2  (ID int primary key,
  3  cname varchar(20),
  4  age int,
  5  Address varchar(25),
  6  Salary number(10,3) default 1000.00);

Table created.

SQL>
```



```
SQL> DESCRIBE TABLE suppliers
Usage: DESCRIBE [schema.]object[@db_link]
SQL> DESCRIBE customers
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER(38)
 CNAME                                              VARCHAR2(20)
 AGE                                                NUMBER(38)
 ADDRESS                                            VARCHAR2(25)
 SALARY                                             NUMBER(10,3)
```

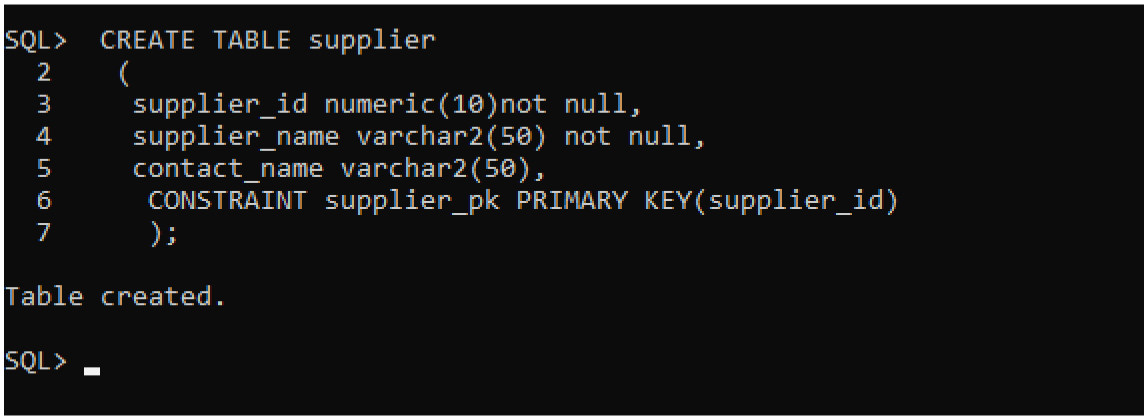
2. Create a Table for supplier using primary key

1. SQL> CREATE TABLE supplier

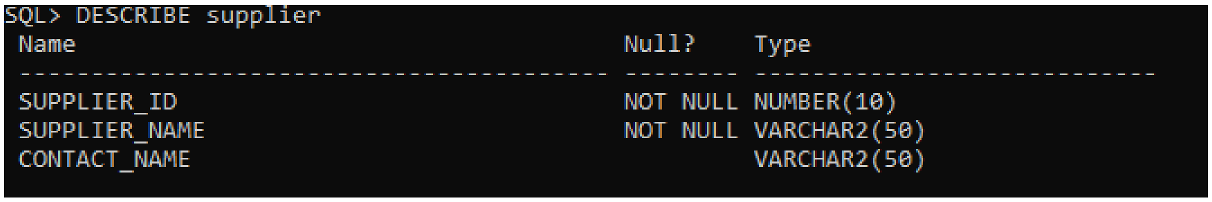
   (supplier_id numeric(10)not null,

        supplier_name varchar2(50) not null,
         contact_name varchar2(50),
         CONSTRAINT supplier_pk PRIMARY KEY(supplier_id)
           );

   Table created.



```
SQL>  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)
     );

Table created.

SQL>
```



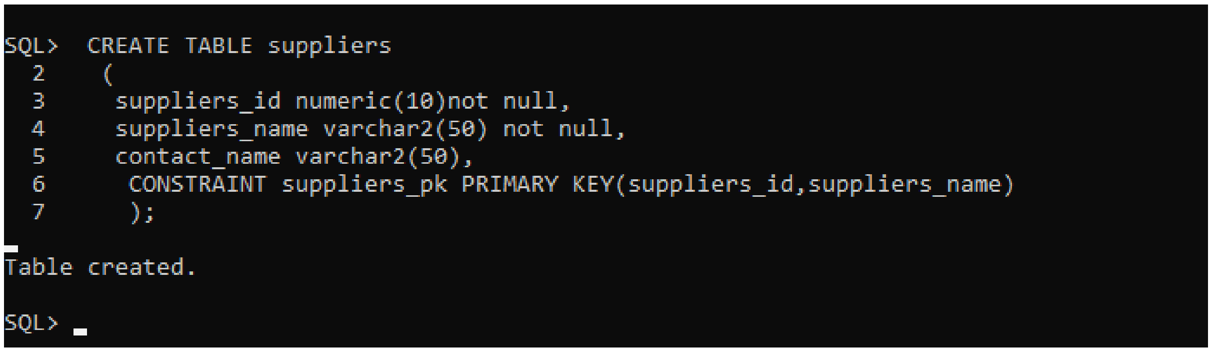
```
SQL> DESCRIBE supplier
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SUPPLIER_ID                               NOT NULL NUMBER(10)
 SUPPLIER_NAME                             NOT NULL VARCHAR2(50)
 CONTACT_NAME                                       VARCHAR2(50)
```

2. Create a supplier table with 2 primary key

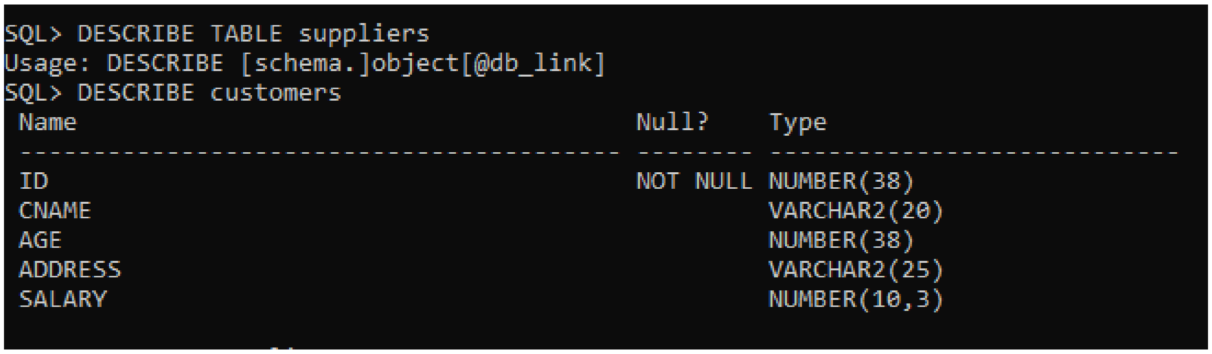
3. SQL> CREATE TABLE suppliers

```
 (
suppliers_id numeric(10)not null,
suppliers_name varchar2(50) not null,
contact_name varchar2(50),
CONSTRAINT suppliers_pk PRIMARY KEY(suppliers_id,suppliers_name)
 );
```

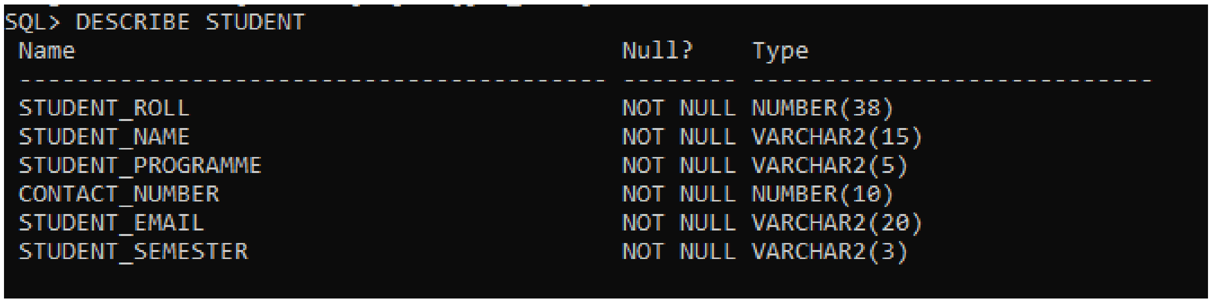
Table created.



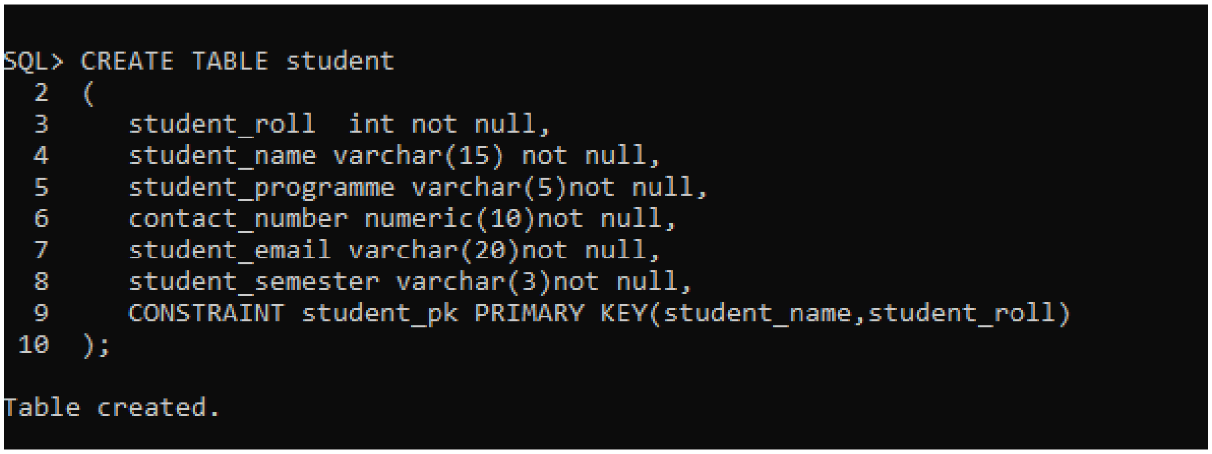
```
SQL>  CREATE TABLE suppliers
  2    (
  3     suppliers_id numeric(10)not null,
  4     suppliers_name varchar2(50) not null,
  5     contact_name varchar2(50),
  6      CONSTRAINT suppliers_pk PRIMARY KEY(suppliers_id,suppliers_name)
  7      );

Table created.

SQL>
```



```
SQL> DESCRIBE TABLE suppliers
Usage: DESCRIBE [schema.]object[@db_link]
SQL> DESCRIBE customers
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER(38)
 CNAME                                              VARCHAR2(20)
 AGE                                                NUMBER(38)
 ADDRESS                                            VARCHAR2(25)
 SALARY                                             NUMBER(10,3)
```

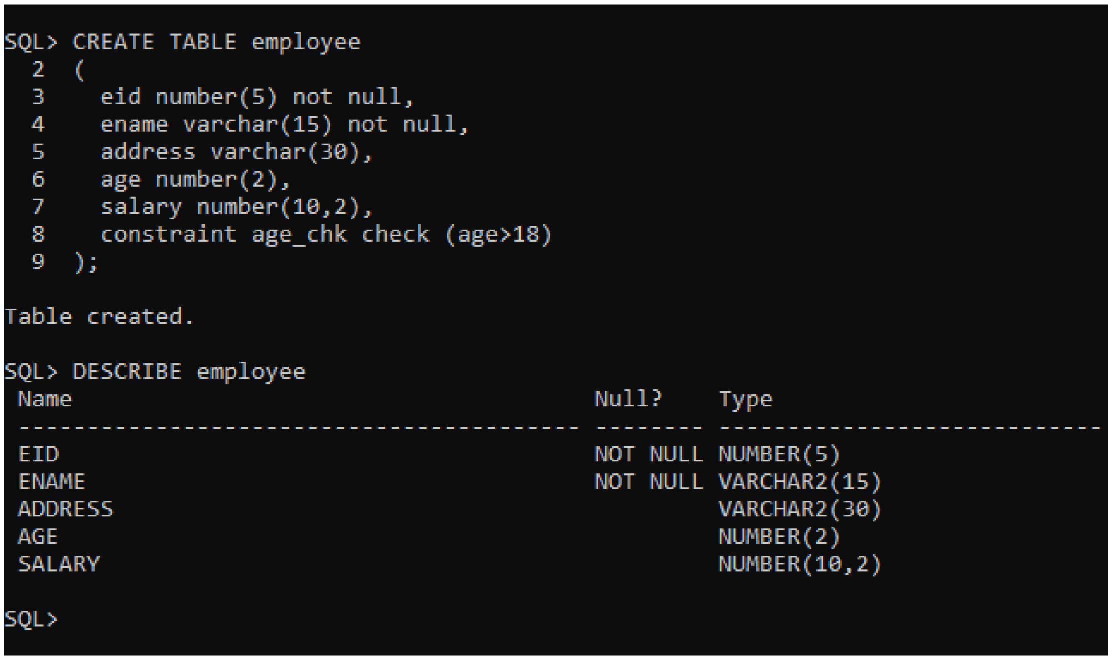


```
SQL> DESCRIBE STUDENT
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENT_ROLL                              NOT NULL NUMBER(38)
 STUDENT_NAME                              NOT NULL VARCHAR2(15)
 STUDENT_PROGRAMME                         NOT NULL VARCHAR2(5)
 CONTACT_NUMBER                            NOT NULL NUMBER(10)
 STUDENT_EMAIL                             NOT NULL VARCHAR2(20)
 STUDENT_SEMESTER                          NOT NULL VARCHAR2(3)
```

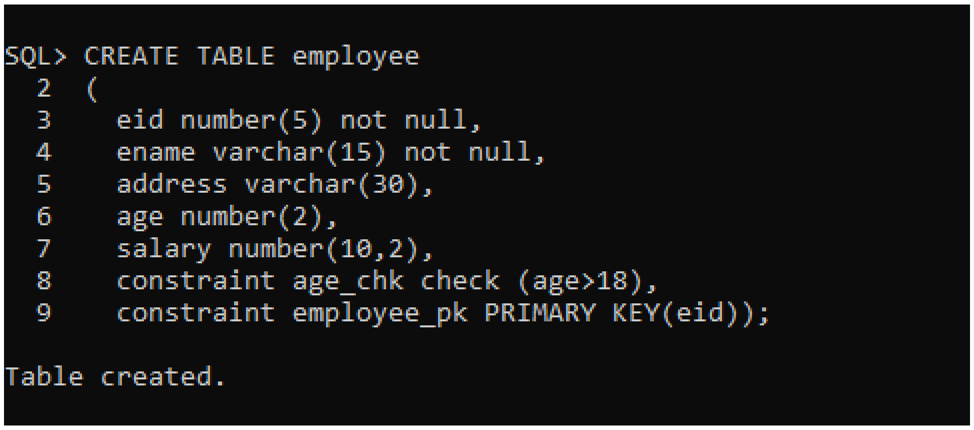


```
SQL> CREATE TABLE student
  (
     student_roll  int not null,
     student_name varchar(15) not null,
     student_programme varchar(5)not null,
     contact_number numeric(10)not null,
     student_email varchar(20)not null,
     student_semester varchar(3)not null,
     CONSTRAINT student_pk PRIMARY KEY(student_name,student_roll)
  );

Table created.
```

4. Create table using check constraints



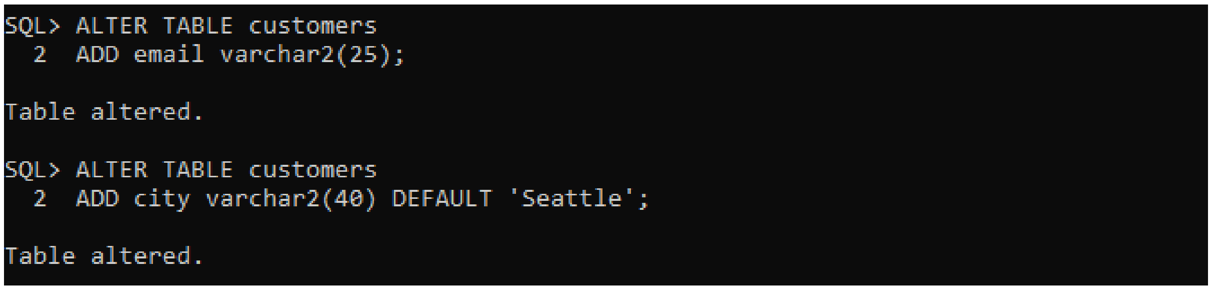
```
SQL> CREATE TABLE employee
  (
    eid number(5) not null,
    ename varchar(15) not null,
    address varchar(30),
    age number(2),
    salary number(10,2),
    constraint age_chk check (age>18)
  );

Table created.

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

SQL>
```



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

Table created.
```

b.. ALTER TABLE

1) **Alter query**
   Syntax:
   Alter table table _name and col_name data_type(n);
   Description:
   By the use of Alter table Command we can modify our existing tables.
   Code:
   Alter table customers
   Add email varchar2(25);
2) **Default value using Alter**
   Syntax:
   Alter table customers add email varchar2(25);
   Description:
   The default constraint is used to set a default value for a column.
   The default value will be added to all new records, if no other is specified.
   Command:
   Alter table employee3 and city varchar2(40) default ‘seattle’;
   Output:



```
SQL> ALTER TABLE customers
  2  ADD email varchar2(25);

Table altered.

SQL> ALTER TABLE customers
  2  ADD city varchar2(40) DEFAULT 'Seattle';

Table altered.
```

3. Modify table with in default value:

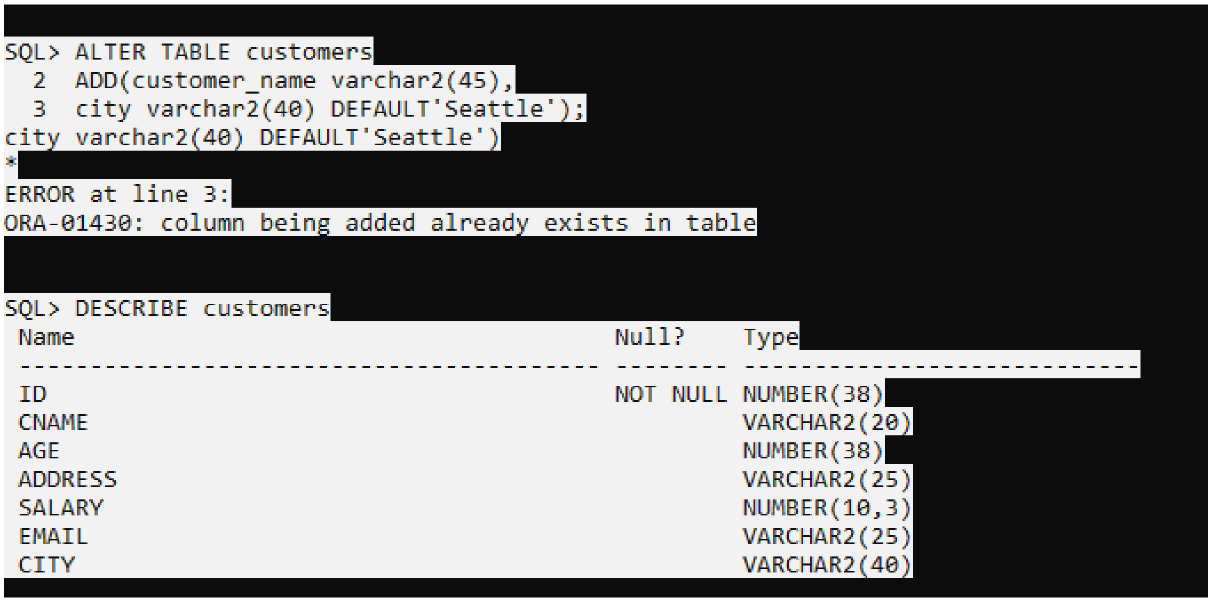
Syntax:

Alter table table_name modify col data_type ;

Alter table table_name modify column_name data_type not null

Code:

Alter table employee3 modify email varchar(20);

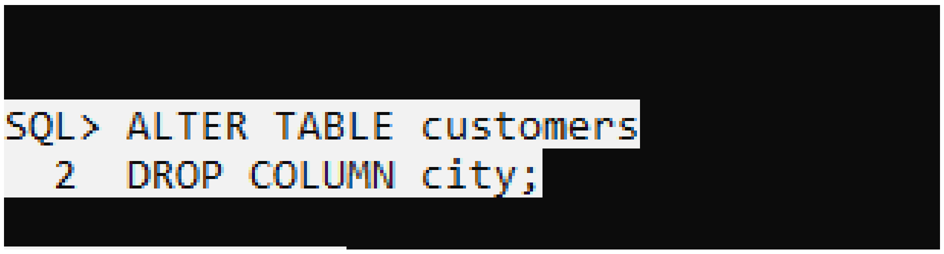


```
SQL> ALTER TABLE customers
  2  ADD(customer_name varchar2(45),
  3  city varchar2(40) DEFAULT'Seattle');
city varchar2(40) DEFAULT'Seattle')
*
ERROR at line 3:
ORA-01430: column being added already exists in table


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

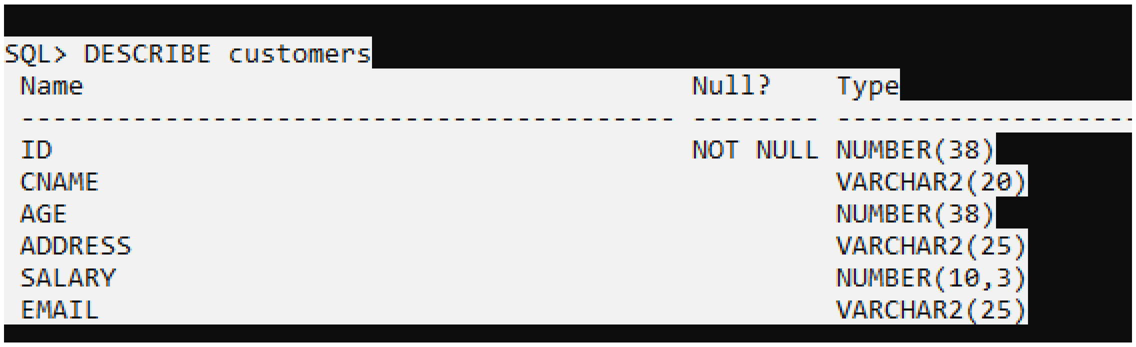
C. Drop:

1. **DROP COLUMN**
   Syntax:
   alter table customers
   drop column city;
   Description:
   This Command will drop particular column
   Code:
   ALTER TABLE customers
   DROP COLUMN city;

Output:

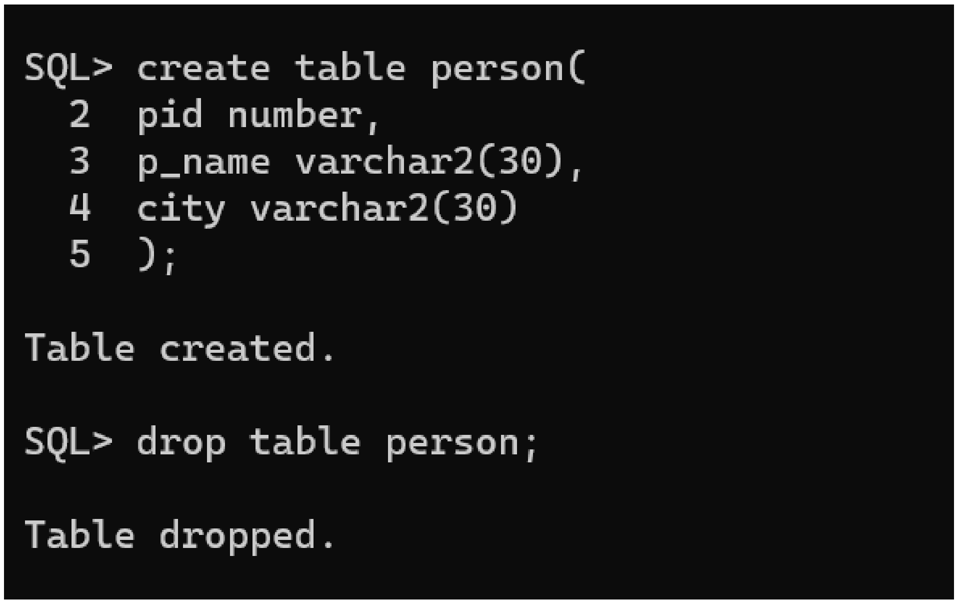


```
SQL> ALTER TABLE customers
  2  DROP COLUMN city;
```


```
SQL> DESCRIBE customers
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER(38)
 CNAME                                              VARCHAR2(20)
 AGE                                                NUMBER(38)
 ADDRESS                                            VARCHAR2(25)
 SALARY                                             NUMBER(10,3)
 EMAIL                                              VARCHAR2(25)
```

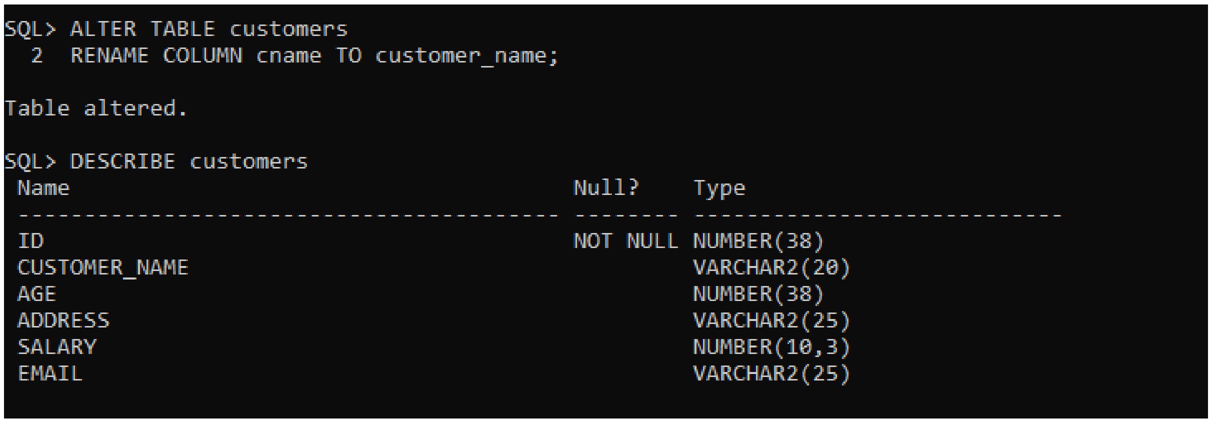
3. Drop table:
Syntax: Drop table table_name;
Code: Drop table person;
create table person(
2 pid number,
3 p_name varchar2(30),
4 city varchar2(30)
5);

Output:


```
SQL> create table person(
  2  pid number,
  3  p_name varchar2(30),
  4  city varchar2(30)
  5  );

Table created.

SQL> drop table person;

Table dropped.
```

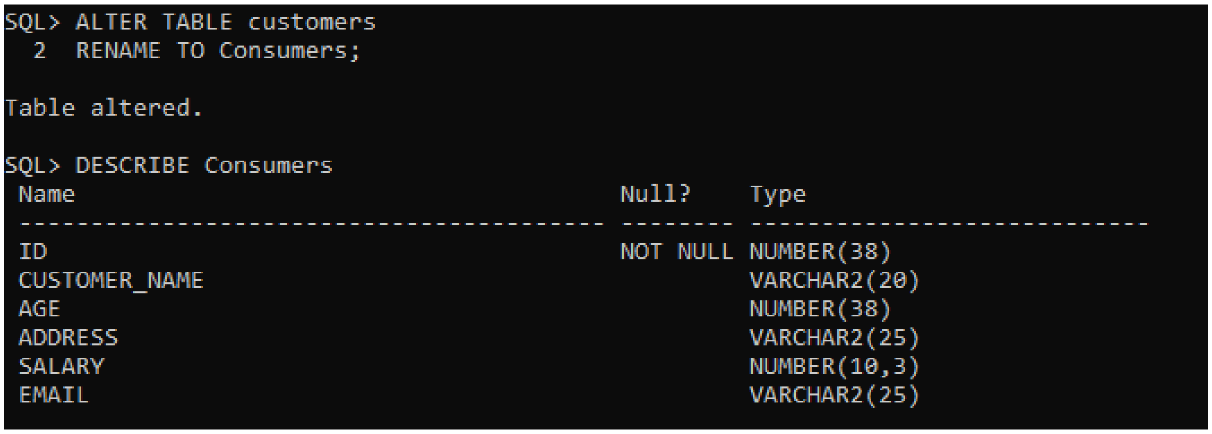
d. Rename:

1. **Rename column**
   Syntax:
   Alter table table_name rename column old_name to new_name;
   Description:
   The old column of table customers
   Alter table customers
   rename column cname to customer_name;



```
SQL> ALTER TABLE customers
  2  RENAME COLUMN cname TO customer_name;

Table altered.

SQL> DESCRIBE customers
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER(38)
 CUSTOMER_NAME                                      VARCHAR2(20)
 AGE                                                NUMBER(38)
 ADDRESS                                            VARCHAR2(25)
 SALARY                                             NUMBER(10,3)
 EMAIL                                              VARCHAR2(25)
```

2. **Rename table**
   Syntax:
   alter table customers
   rename to consumers;



```
SQL> ALTER TABLE customers
  2  RENAME TO Consumers;

Table altered.

SQL> DESCRIBE Consumers
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER(38)
 CUSTOMER_NAME                                      VARCHAR2(20)
 AGE                                                NUMBER(38)
 ADDRESS                                            VARCHAR2(25)
 SALARY                                             NUMBER(10,3)
 EMAIL                                              VARCHAR2(25)
```

Practical no-2 Study of DML Commands

1. **Data manipulation language**

A. Insert:

1. **Inserting values into columns:**

Syntax:

INSERT INTO table_name(

Column1,column2,…column)

VALUES (value1,value2…valueN);

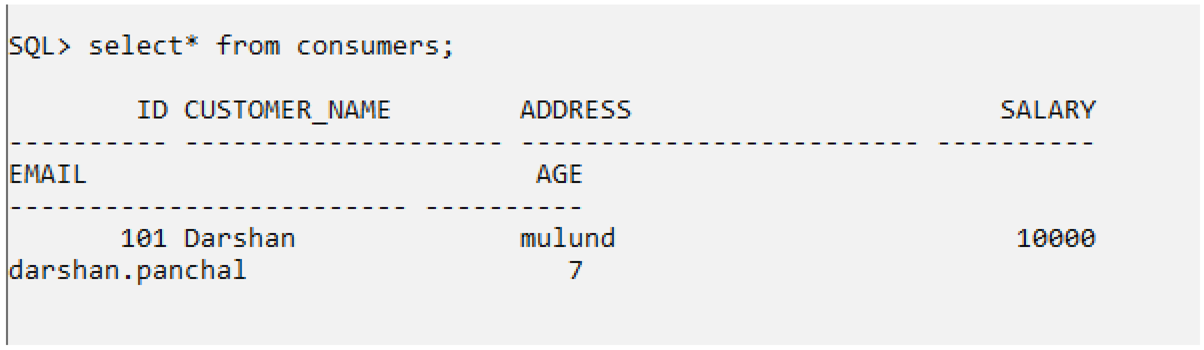
Description:

Adds value into the respective table

Code:

INSERT INTO consumers values(101,'Darshan', 21, 'Mulund', 10000);

Output:



```
SQL> select* from consumers;

        ID CUSTOMER_NAME        ADDRESS                          SALARY
---------- -------------------- ------------------------- ----------
EMAIL                            AGE
------------------------- ----------
       101 Darshan              mulund                            10000
darshan.panchal                    7
```

Inserting multiple values:

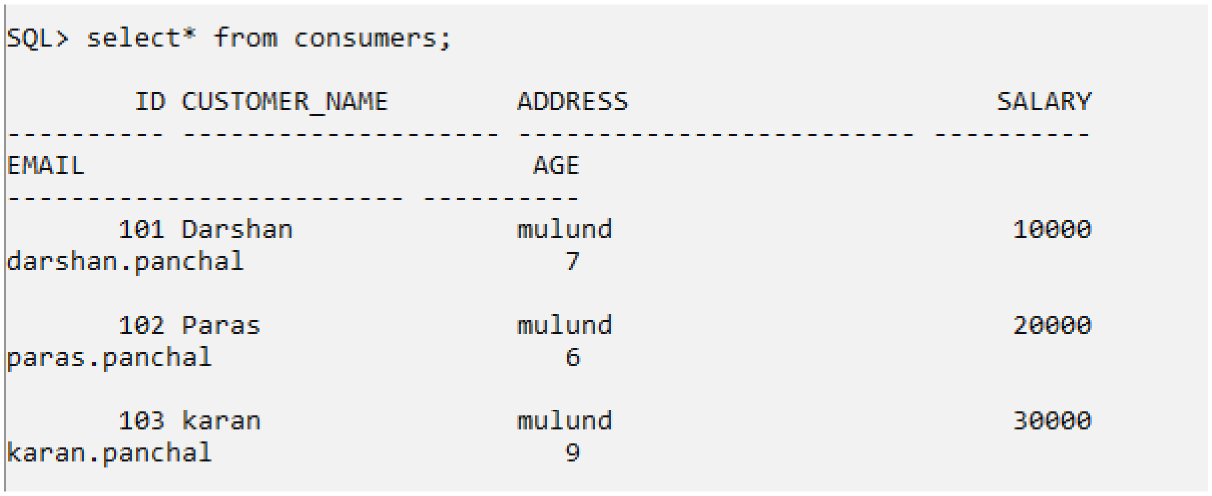
Code:

insert into consumers values(101,'Darshan','mulund',10000,'darshan.panchal',7);

insert into consumers values(102,'Paras','mulund',20000,'paras.panchal',6);

insert into consumers values(103,'karan','mulund',30000,'karan.panchal',9);

Output:



```
SQL> select* from consumers;

        ID CUSTOMER_NAME        ADDRESS                        SALARY
---------- -------------------- ------------------------- ----------
EMAIL                            AGE
------------------------- ----------
       101 Darshan              mulund                          10000
darshan.panchal                    7

       102 Paras                mulund                          20000
paras.panchal                      6

       103 karan                mulund                          30000
karan.panchal                      9
```

Code for insert all:

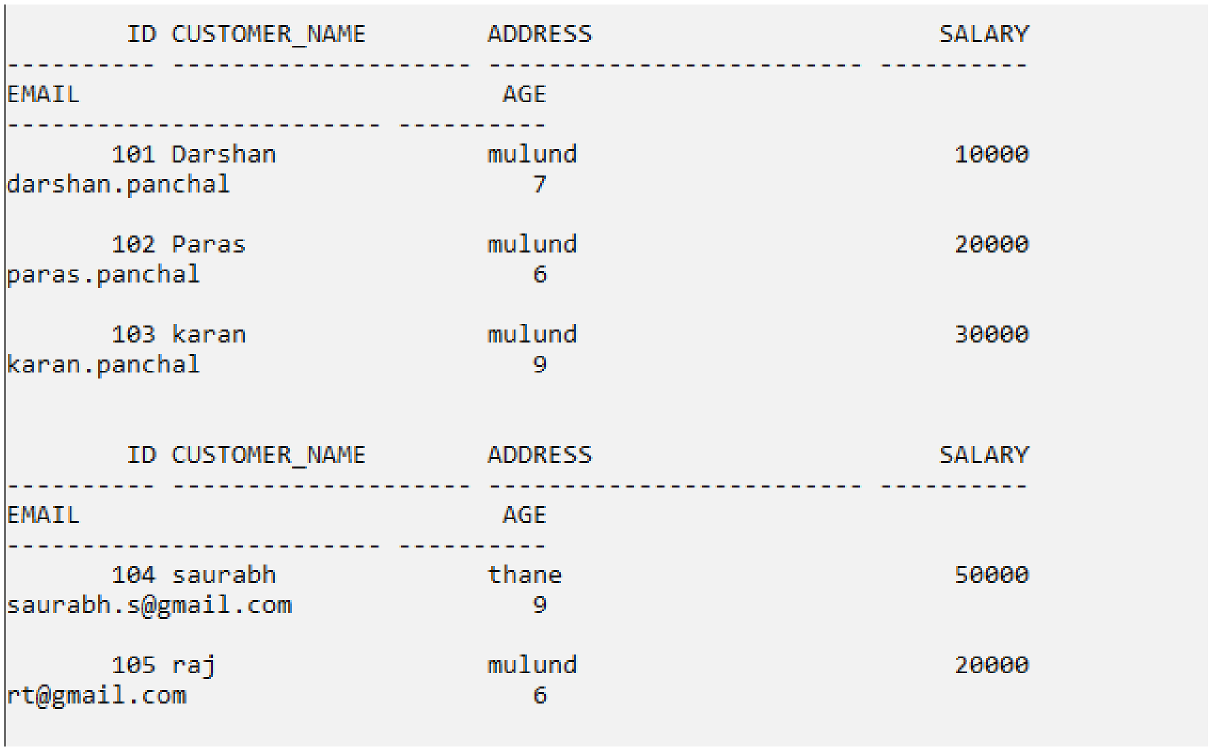
insert all

into consumers values(104,'saurabh','thane',50000,'saurabh.s@gmail.com',9)

into consumers values(105,'raj','mulund',20000,'rt@gmail.com',6)

select * from dual;

Output:



```
        ID CUSTOMER_NAME        ADDRESS                       SALARY
---------- -------------------- ------------------------- ----------
EMAIL                             AGE
------------------------- ----------
       101 Darshan              mulund                         10000
darshan.panchal                     7

       102 Paras                mulund                         20000
paras.panchal                       6

       103 karan                mulund                         30000
karan.panchal                       9


        ID CUSTOMER_NAME        ADDRESS                       SALARY
---------- -------------------- ------------------------- ----------
EMAIL                             AGE
------------------------- ----------
       104 saurabh              thane                          50000
saurabh.s@gmail.com                 9

       105 raj                  mulund                         20000
rt@gmail.com                        6
```

Code:

INSERT ALL

Into suppliers(supplier_id, supplier_name) values (1000, “Ibm”)

Into suppliers(supplier_id, supplier_name) values (2000, “Microsoft”)

select * from dual;

Insert 5 records in to new_ott table

Code:

insert all

into new_ott values('201','Nun','Movie',19102021,'Horror','S.D',6,06,255,'3.4',299,'No','Hindi','India')

into new_ott values('202','Starlight','Web show',29012023,'Sci-fic','J.R',7,10,300,3.9,199,'Yes','English','Italy')

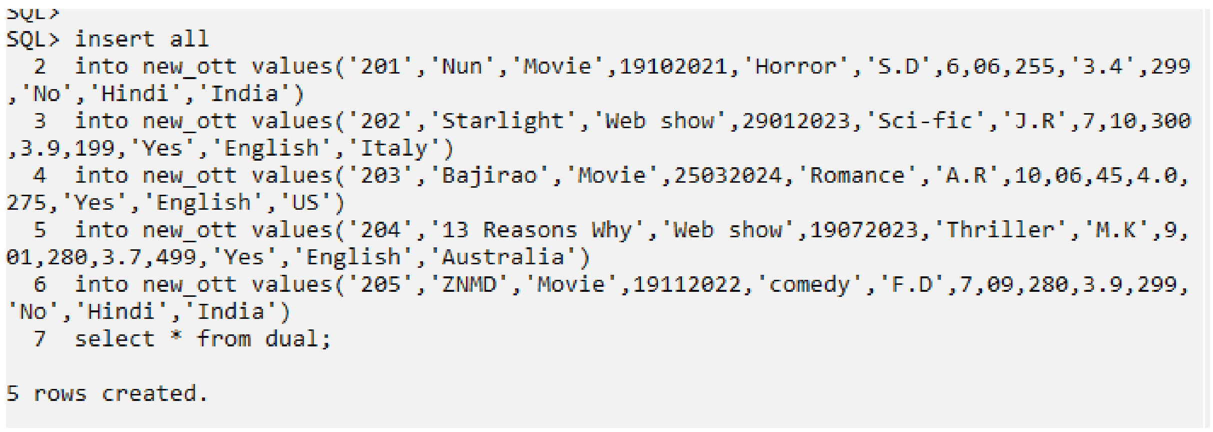
into new_ott values('203','Bajirao','Movie',25032024,'Romance','A.R',10,06,45,4.0,275,'Yes','English','US')

into new_ott values('204','13 Reasons Why','Web show',19072023,'Thriller','M.K',9,01,280,3.7,499,'Yes','English','Australia')

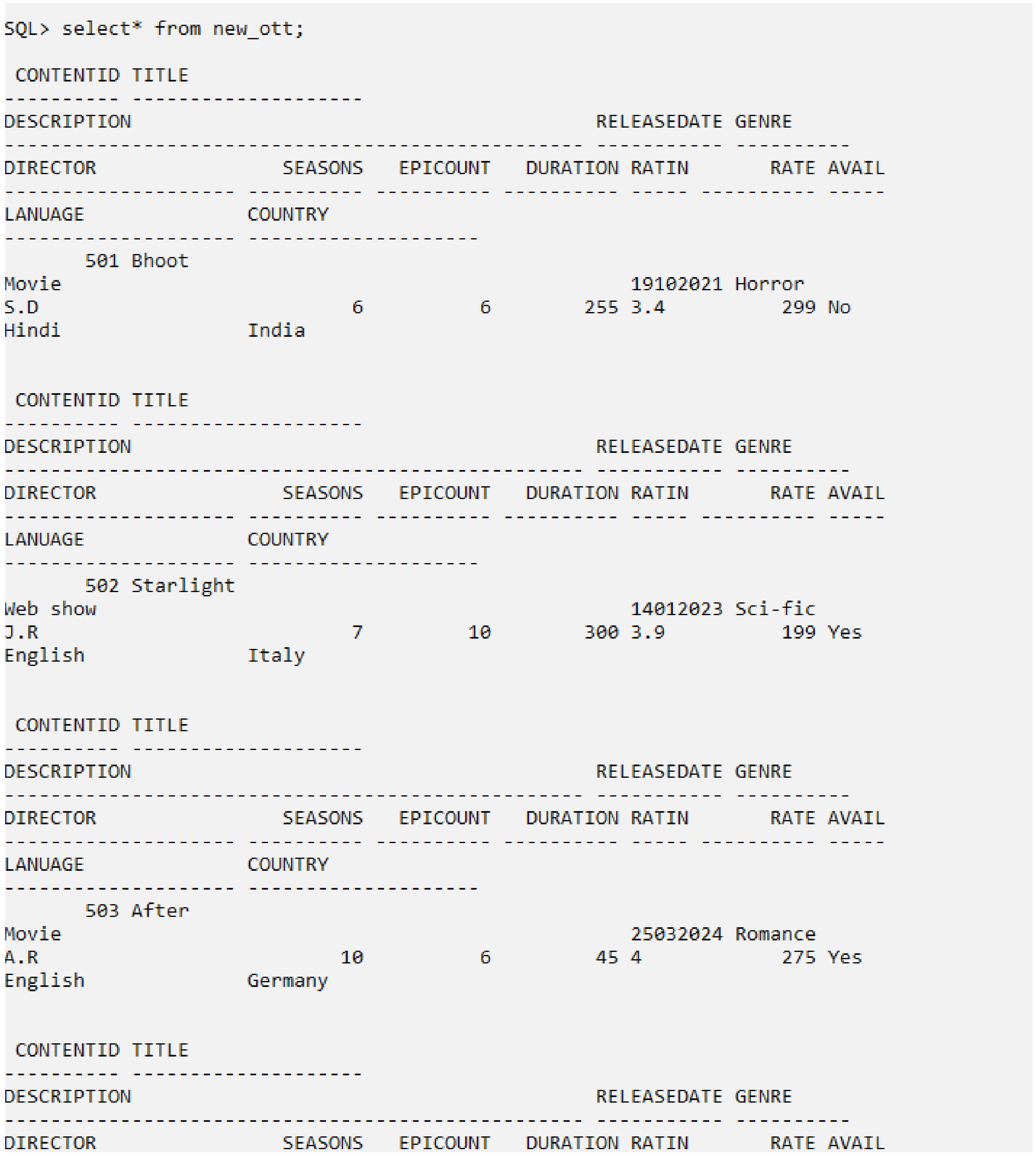
into new_ott values('205','ZNMD','Movie',19112022,'comedy','F.D',7,09,280,3.9,299,'No','Hindi','India')

select * from dual;

Output:

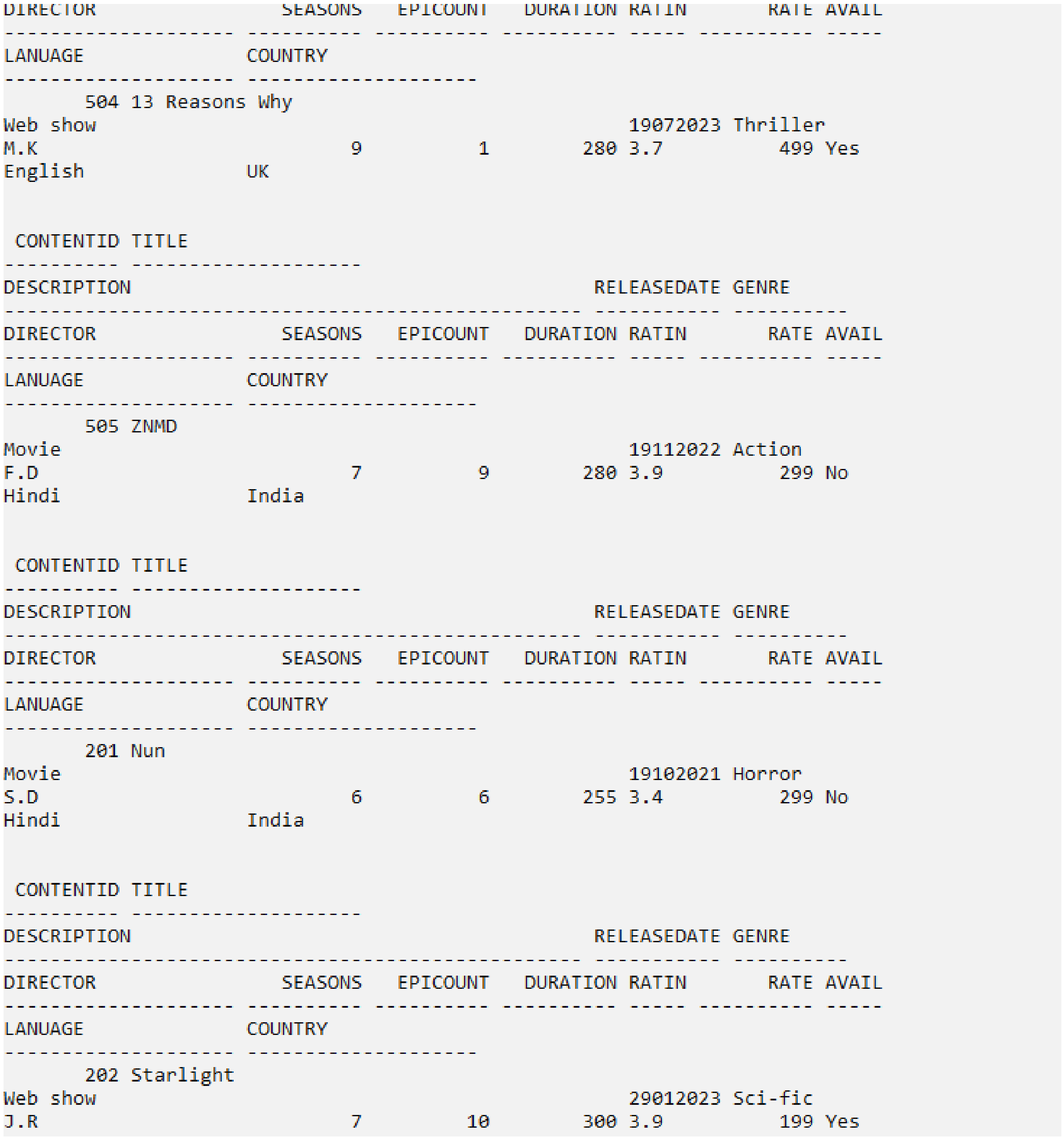


```
SQL>
SQL> insert all
  2  into new_ott values('201','Nun','Movie',19102021,'Horror','S.D',6,06,255,'3.4',299
,'No','Hindi','India')
  3  into new_ott values('202','Starlight','Web show',29012023,'Sci-fic','J.R',7,10,300
,3.9,199,'Yes','English','Italy')
  4  into new_ott values('203','Bajirao','Movie',25032024,'Romance','A.R',10,06,45,4.0,
275,'Yes','English','US')
  5  into new_ott values('204','13 Reasons Why','Web show',19072023,'Thriller','M.K',9,
01,280,3.7,499,'Yes','English','Australia')
  6  into new_ott values('205','ZNMD','Movie',19112022,'comedy','F.D',7,09,280,3.9,299,
'No','Hindi','India')
  7  select * from dual;

5 rows created.
```

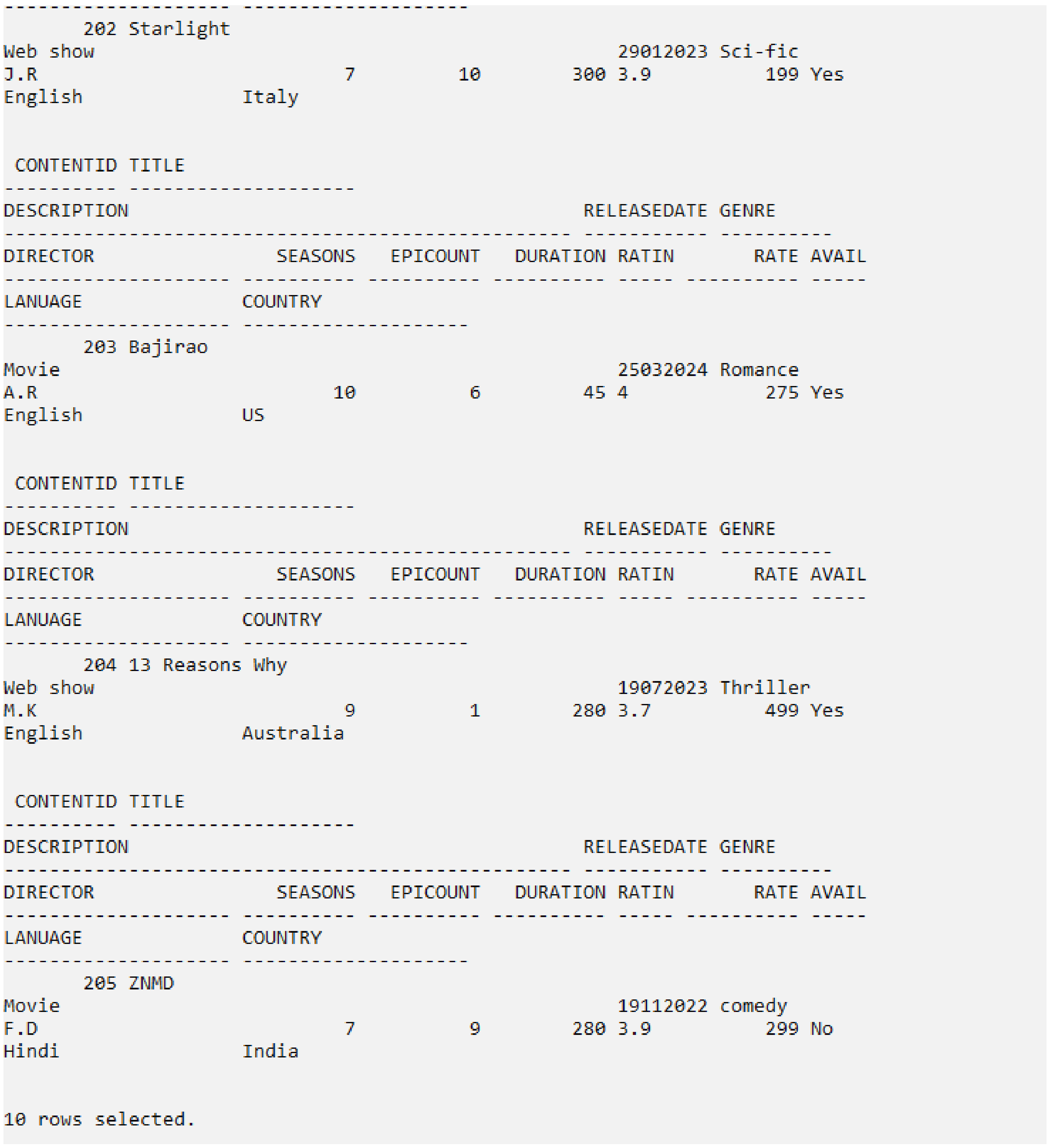


```
SQL> select* from new_ott;

 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       501 Bhoot
Movie                                                 19102021 Horror
S.D                           6          6        255 3.4          299 No
Hindi                India


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       502 Starlight
Web show                                              14012023 Sci-fic
J.R                           7         10        300 3.9          199 Yes
English              Italy


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       503 After
Movie                                                 25032024 Romance
A.R                          10          6         45 4            275 Yes
English              Germany


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
```

Output:



```
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       504 13 Reasons Why
Web show                                              19072023 Thriller
M.K                           9          1        280 3.7          499 Yes
English              UK


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       505 ZNMD
Movie                                                 19112022 Action
F.D                           7          9        280 3.9          299 No
Hindi                India


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       201 Nun
Movie                                                 19102021 Horror
S.D                           6          6        255 3.4          299 No
Hindi                India


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       202 Starlight
Web show                                              29012023 Sci-fic
J.R                           7         10        300 3.9          199 Yes
```



```
-------------------- --------------------
       202 Starlight
Web show                                              29012023 Sci-fic
J.R                           7         10        300 3.9          199 Yes
English              Italy


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       203 Bajirao
Movie                                                 25032024 Romance
A.R                          10          6         45 4            275 Yes
English              US


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       204 13 Reasons Why
Web show                                              19072023 Thriller
M.K                           9          1        280 3.7          499 Yes
English              Australia


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       205 ZNMD
Movie                                                 19112022 comedy
F.D                           7          9        280 3.9          299 No
Hindi                India


10 rows selected.
```

10 rows which are selected

***UPDATE**

*UPDATE table_name

Syntax

SET column= value1,value2=valueN

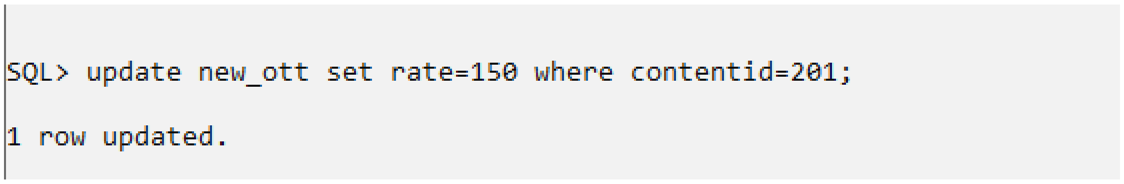
WHERE [condition];

Example:

update employee set salary=15000 where id=202;

#Update the price when content id is given:

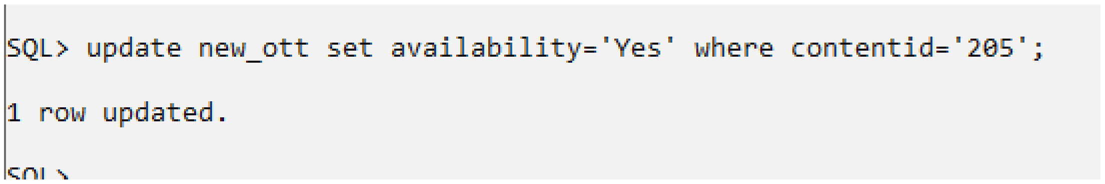
update new_ott set rate=150 where contentid=201;



```
SQL> update new_ott set rate=150 where contentid=201;

1 row updated.
```

#update change the availability for content with episode count more than 30

update new_ott set availability='Yes' where contentid='205';



```
SQL> update new_ott set availability='Yes' where contentid='205';

1 row updated.

SQL>
```

Delete commands

Syntax:

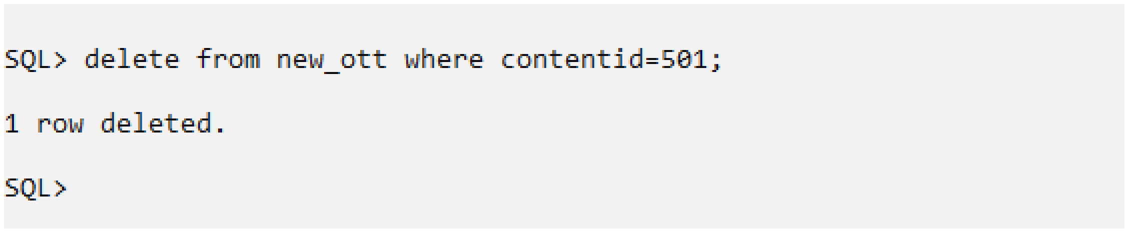
Delete From table_name

Where[condition]

DELETE from customers where id = 102;

delete from new_ott

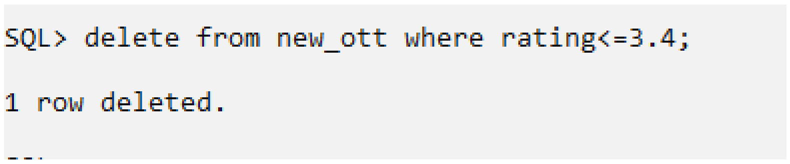
delete from new_ott where contentid=501;



```
SQL> delete from new_ott where contentid=501;

1 row deleted.

SQL>
```

#2. Delete the content with less rating:

delete from new_ott where rating<=3.4;



```
SQL> delete from new_ott where rating<=3.4;

1 row deleted.
```

Output:

Deleted the column with rating less than or equal to 3.4.

```
 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       203 Bajirao
Movie                                                 25032024 Romance
A.R                          10          6         45 4            275 Yes
English              US

 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       204 13 Reasons Why
Web show                                              19072023 Thriller
M.K                           9          1        280 3.7          499 Yes
English              Australia

 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       205 ZNMD
Movie                                                 19112022 comedy
F.D                           7          9        280 3.9          299 No
Hindi                India


8 rows selected.
```

For fetching the data in the new_ott table.

Practical No-3 Study Sql Select Statements

1. **Select**
    a. **Selecting specific columns:**

Syntax:

Select column_name from table_name:

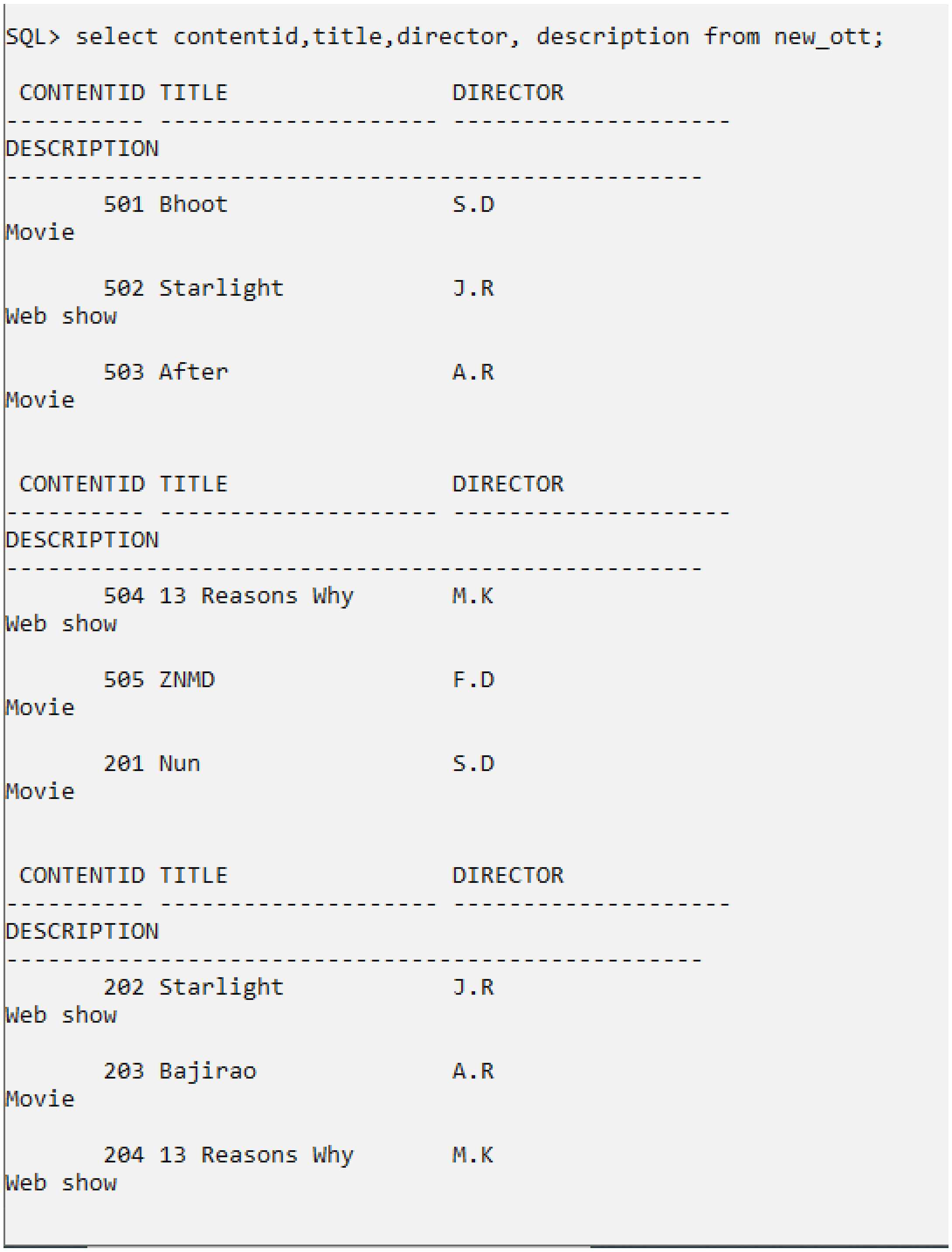
Description:

It fetches all the records of the specific column from the table

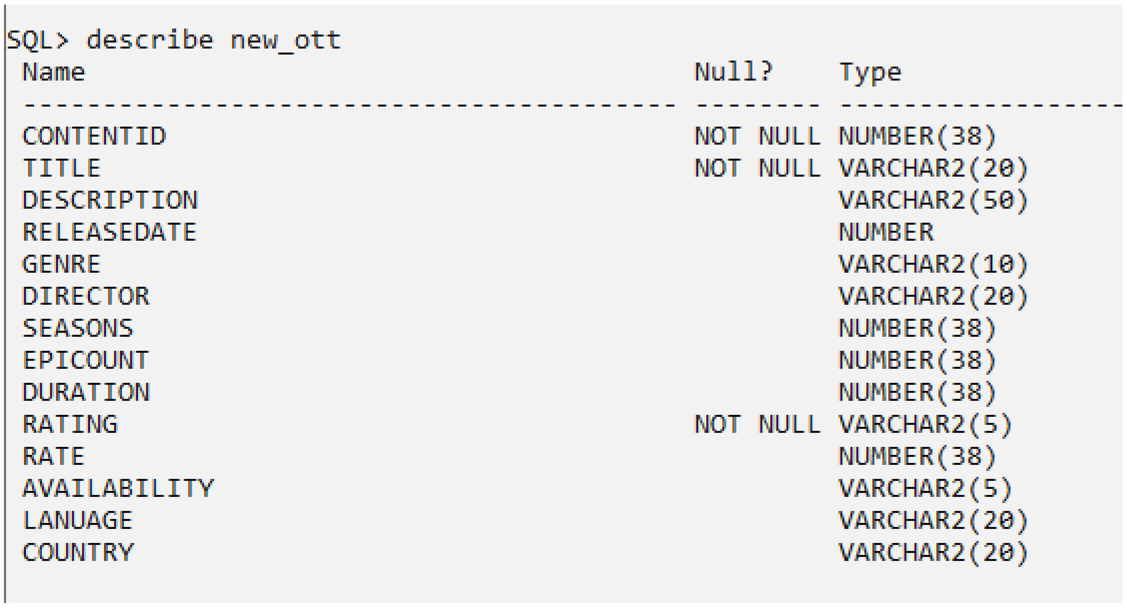
Code:

```
select content id,title,director, description from new_ott;
```

Output:



```
SQL> select contentid,title,director, description from new_ott;

 CONTENTID TITLE                DIRECTOR
---------- -------------------- --------------------
DESCRIPTION
--------------------------------------------------
       501 Bhoot                S.D
Movie

       502 Starlight            J.R
Web show

       503 After                A.R
Movie


 CONTENTID TITLE                DIRECTOR
---------- -------------------- --------------------
DESCRIPTION
--------------------------------------------------
       504 13 Reasons Why       M.K
Web show

       505 ZNMD                 F.D
Movie

       201 Nun                  S.D
Movie


 CONTENTID TITLE                DIRECTOR
---------- -------------------- --------------------
DESCRIPTION
--------------------------------------------------
       202 Starlight            J.R
Web show

       203 Bajirao              A.R
Movie

       204 13 Reasons Why       M.K
Web show
```



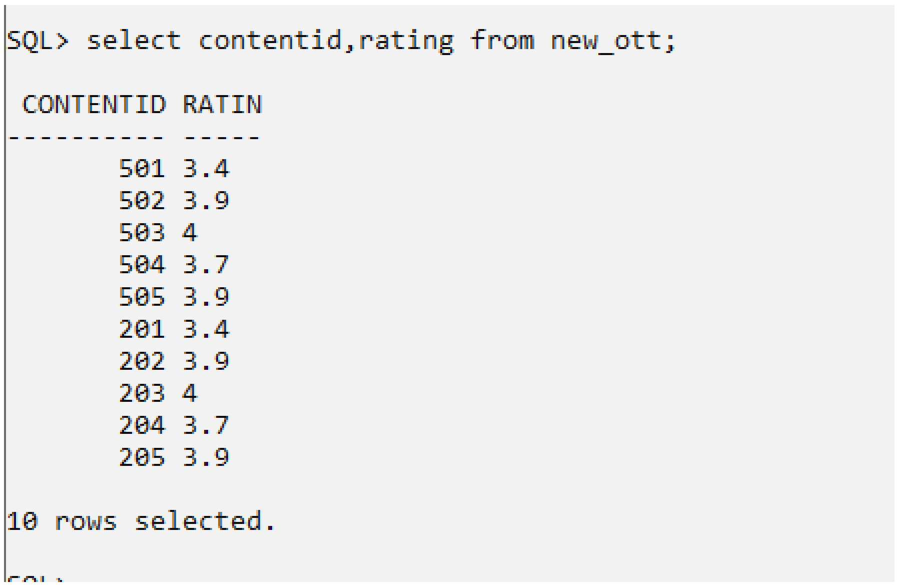
```
SQL> describe new_ott
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------
 CONTENTID                                 NOT NULL NUMBER(38)
 TITLE                                     NOT NULL VARCHAR2(20)
 DESCRIPTION                                        VARCHAR2(50)
 RELEASEDATE                                        NUMBER
 GENRE                                              VARCHAR2(10)
 DIRECTOR                                           VARCHAR2(20)
 SEASONS                                            NUMBER(38)
 EPICOUNT                                           NUMBER(38)
 DURATION                                           NUMBER(38)
 RATING                                    NOT NULL VARCHAR2(5)
 RATE                                               NUMBER(38)
 AVAILABILITY                                       VARCHAR2(5)
 LANUAGE                                            VARCHAR2(20)
 COUNTRY                                            VARCHAR2(20)
```

For fetching the data of specific column name

Code:

select contentid,rating from new_ott;

Output:



```
SQL> select contentid,rating from new_ott;

 CONTENTID RATIN
---------- -----
       501 3.4
       502 3.9
       503 4
       504 3.7
       505 3.9
       201 3.4
       202 3.9
       203 4
       204 3.7
       205 3.9

10 rows selected.
```

Selecting column using WHERE condition

Syntax

SELECT column1,column2,...,column n

FROM table_name

WHERE[condition]

Example for where command:

select*from consumers where salary < 10000;

1. **Concatenation Operator:**

Syntax:

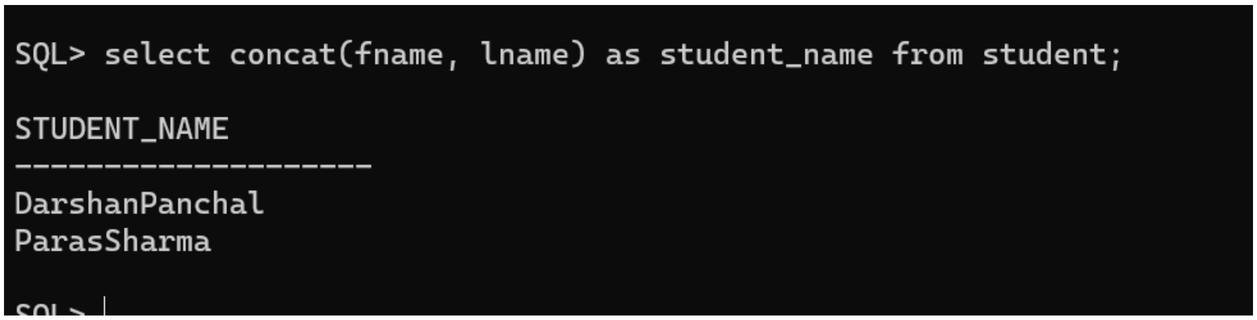
Select CONCAT(f_name,l_name)AS name from Table_name:

Description:

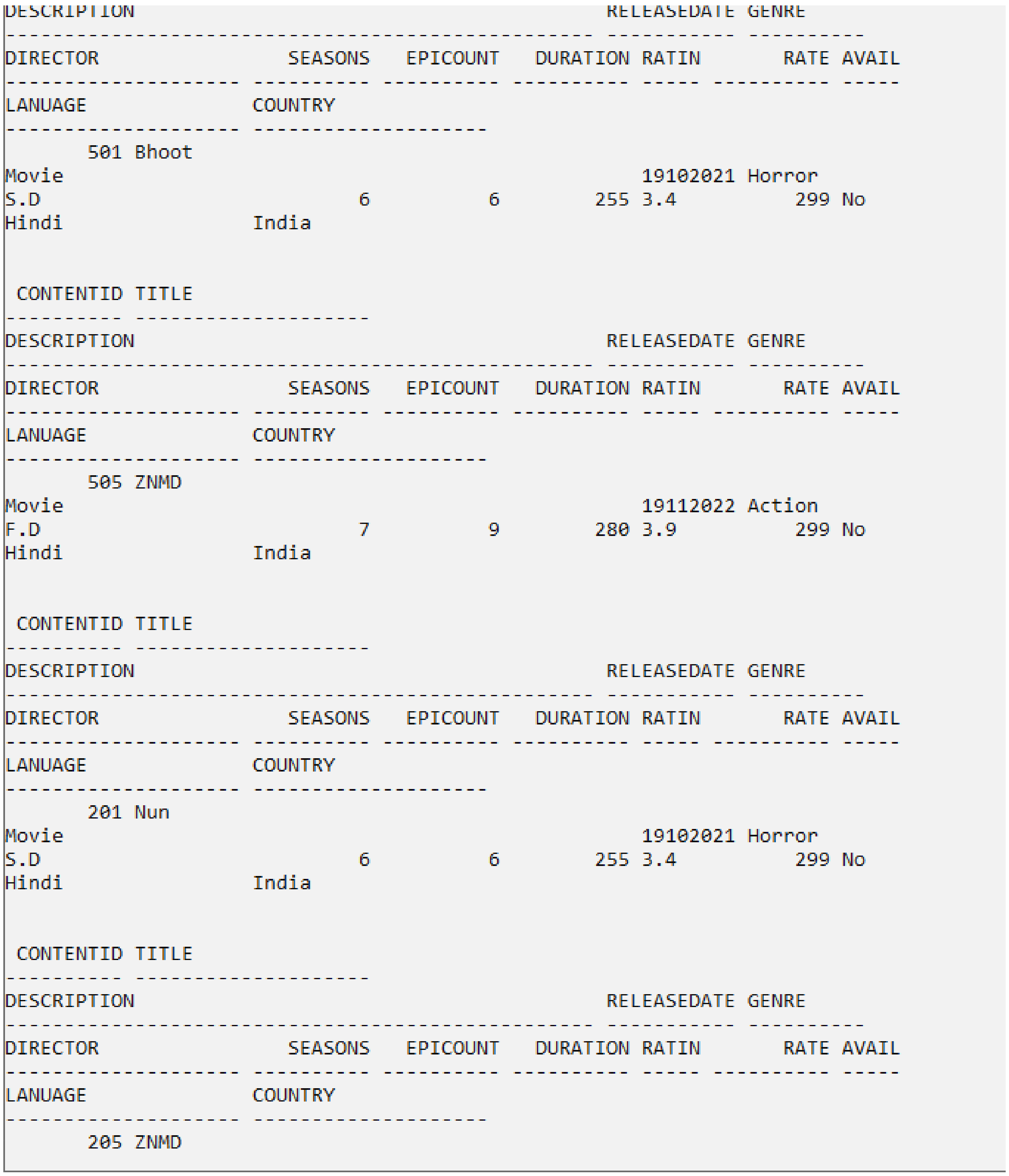
Used to concatenate strings, columns in
to k=join two column into one

Code:

Select concat(fname,lname) as as
student_name from student;

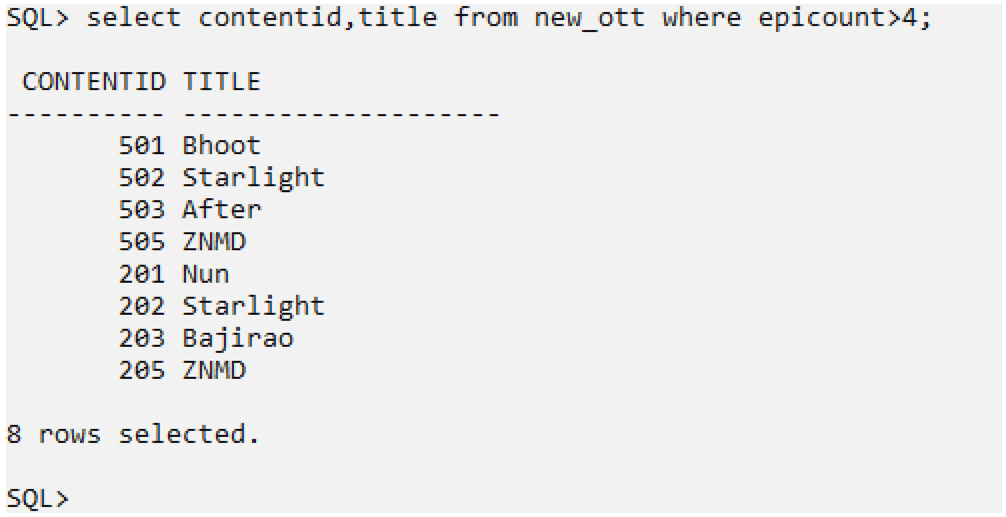


```
SQL> select concat(fname, lname) as student_name from student;

STUDENT_NAME
--------------------
DarshanPanchal
ParasSharma

SQL> 
```



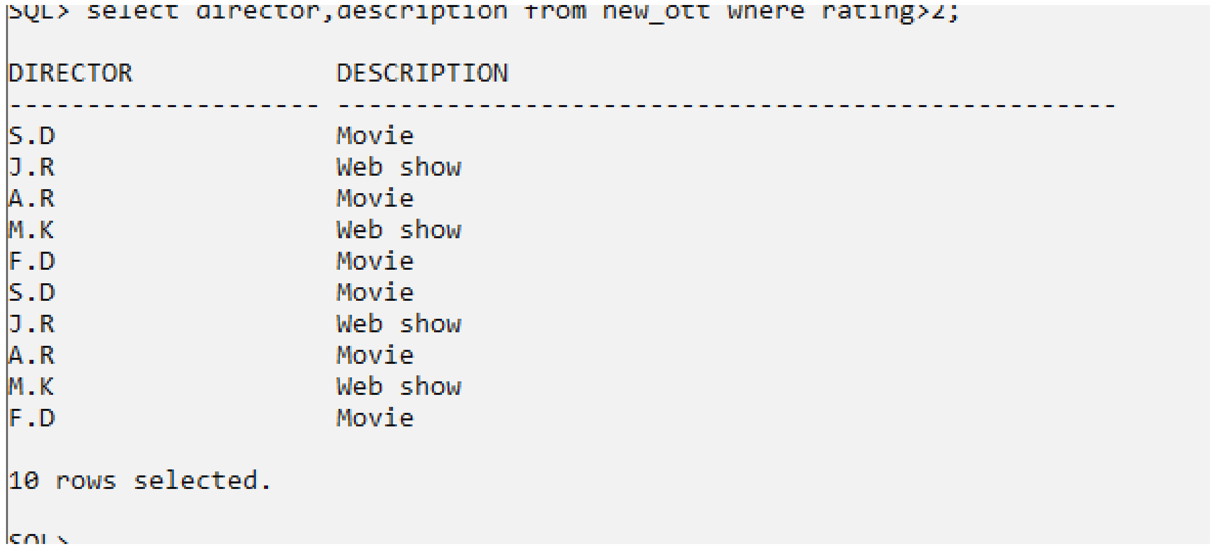
```
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       501 Bhoot
Movie                                                 19102021 Horror
S.D                           6          6        255 3.4          299 No
Hindi                India


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       505 ZNMD
Movie                                                 19112022 Action
F.D                           7          9        280 3.9          299 No
Hindi                India


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       201 Nun
Movie                                                 19102021 Horror
S.D                           6          6        255 3.4          299 No
Hindi                India


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       205 ZNMD
```

select contentid,title from new_ott where epicount>400;



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

 CONTENTID TITLE
---------- --------------------
       501 Bhoot
       502 Starlight
       503 After
       505 ZNMD
       201 Nun
       202 Starlight
       203 Bajirao
       205 ZNMD

8 rows selected.

SQL>
```

select director,description from new_ott where rating>2;

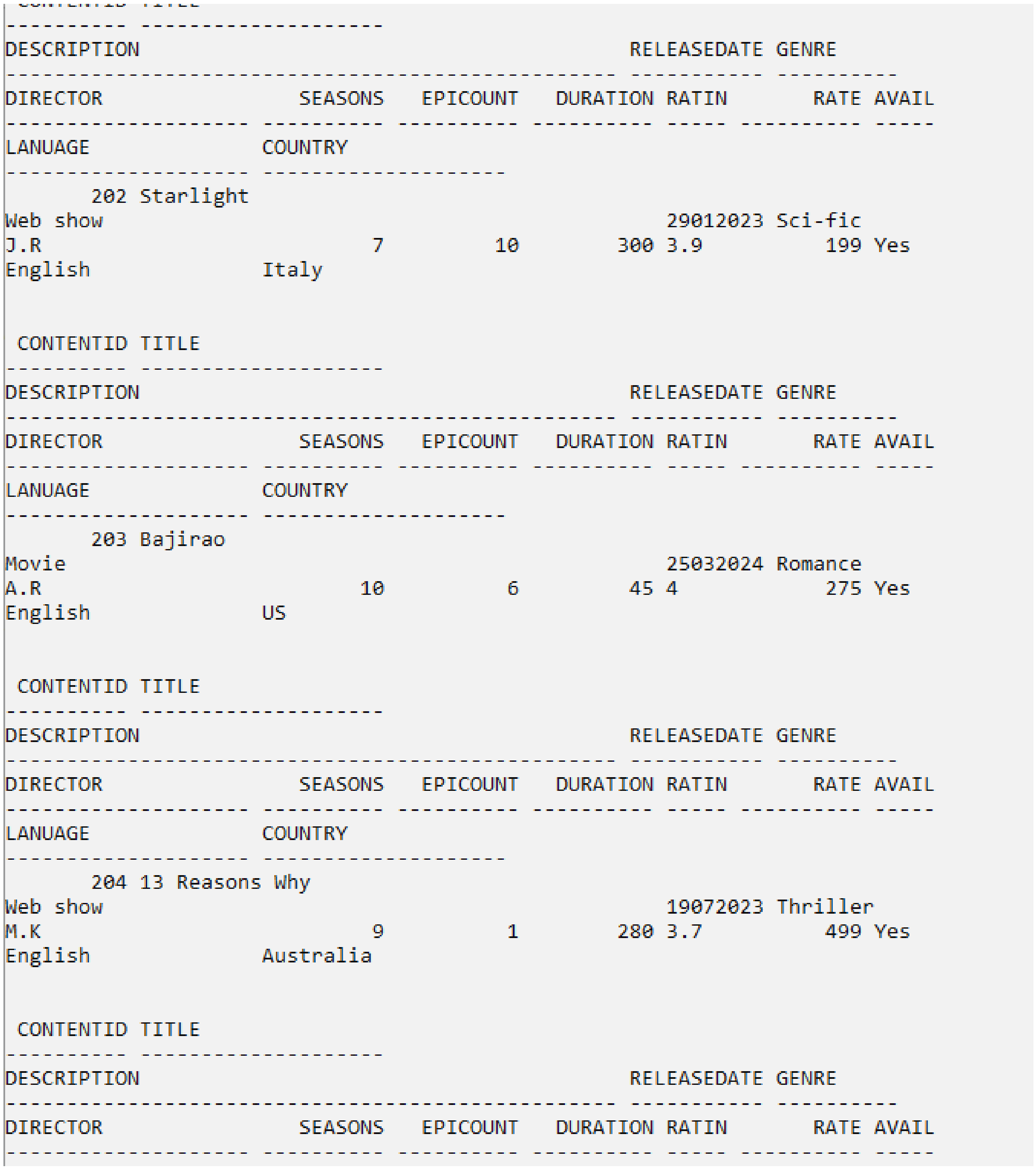


```
SQL> select director,description from new_ott where rating>2;

DIRECTOR             DESCRIPTION
-------------------- --------------------------------------------------
S.D                  Movie
J.R                  Web show
A.R                  Movie
M.K                  Web show
F.D                  Movie
S.D                  Movie
J.R                  Web show
A.R                  Movie
M.K                  Web show
F.D                  Movie

10 rows selected.

SQL>
```

C) Logical Conditions
Description:
Logical operators enable us to use more than one condition.
Code:

Select * from new_ott where genre not in ('romance');

Output:



```
 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       202 Starlight
Web show                                              29012023 Sci-fic
J.R                           7         10        300 3.9          199 Yes
English              Italy


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       203 Bajirao
Movie                                                 25032024 Romance
A.R                          10          6         45 4            275 Yes
English              US


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       204 13 Reasons Why
Web show                                              19072023 Thriller
M.K                           9          1        280 3.7          499 Yes
English              Australia


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
```

Logical Conditions using (and) or conditions

Syntax:

Select column1,column2, columnn

form table_name

where[condition] AND [condition2] And [condition N]

Example

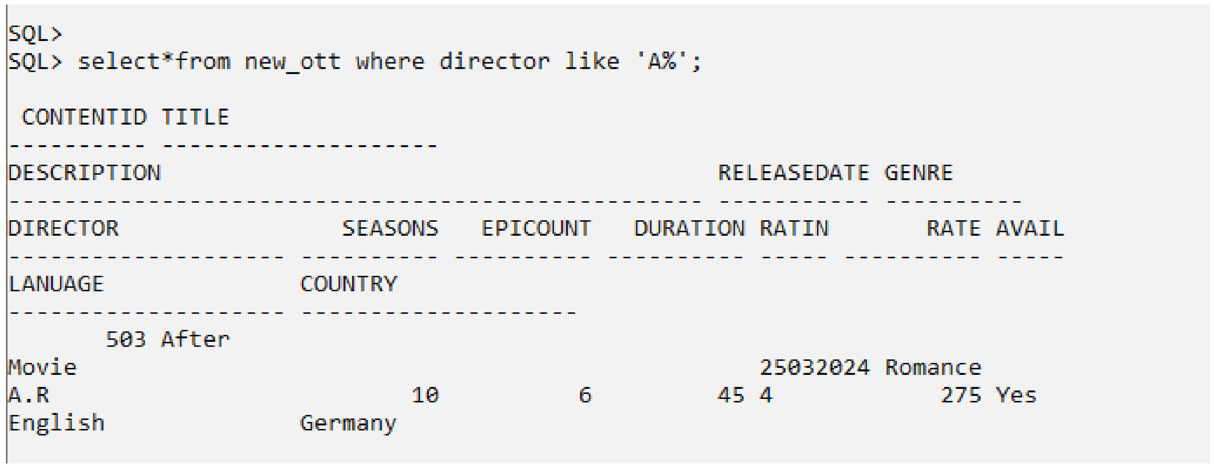
Code:

select* from Suppliers

Where(state='Maharashtra' AND supplier_name='Sahara')

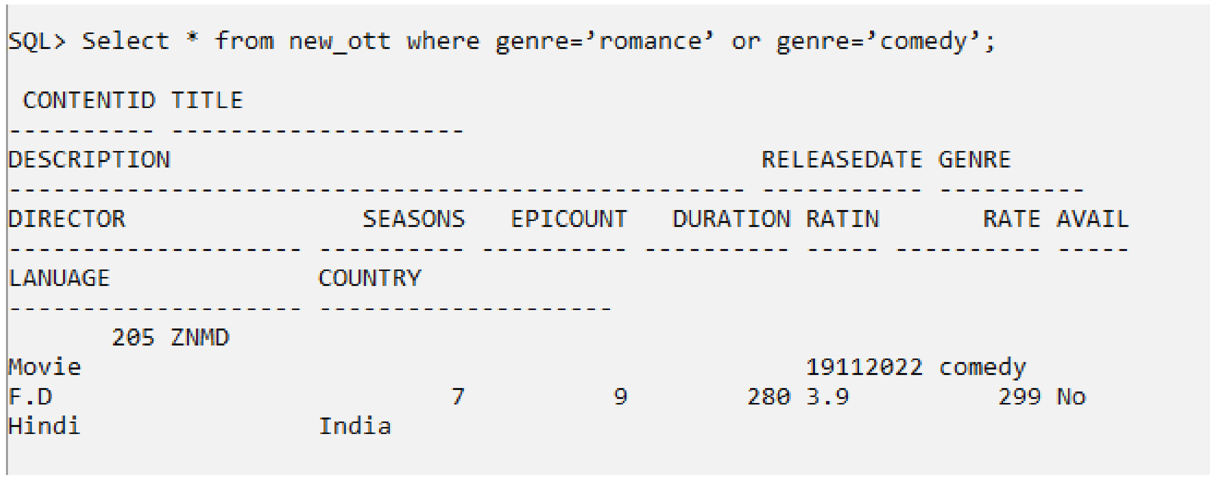
OR(supplier_id<5000);

select*from new_ott where director like 'A%';



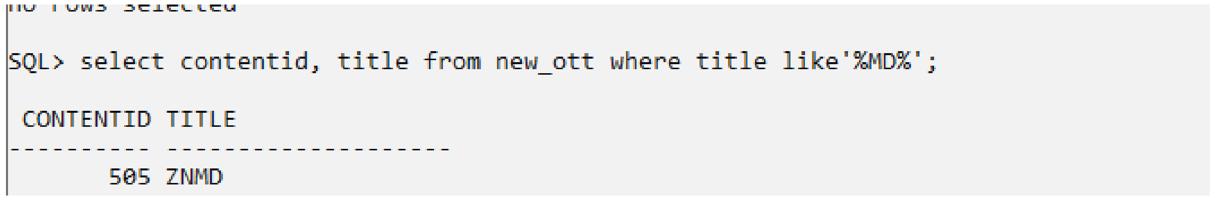
```
SQL>
SQL> select*from new_ott where director like 'A%';

 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       503 After
Movie                                                 25032024 Romance
A.R                          10          6         45 4            275 Yes
English              Germany
```

#The details of series of ‘thriller’ and ‘horror’ genre
Select * from new_ott where genre=’romance’ or genre=’comedy’;
#2 way using in operator
Select * from new_ott where genre in (‘comedy’,’romance’);



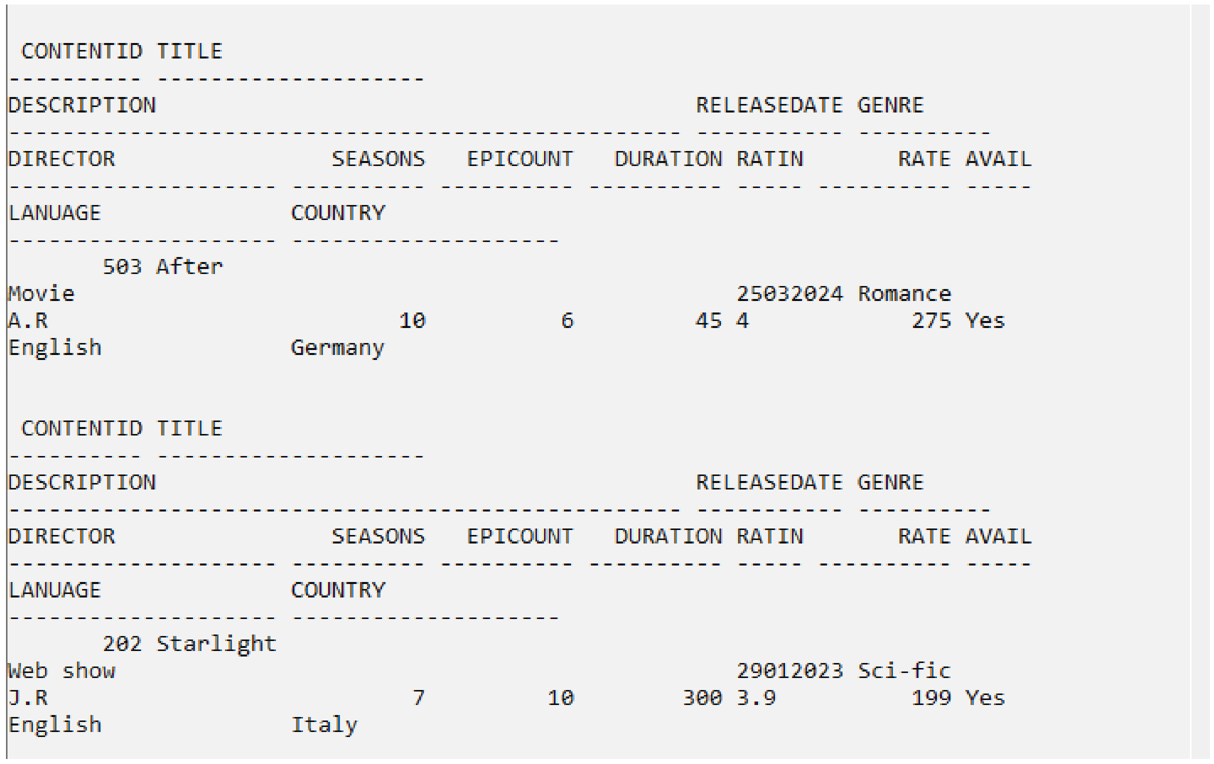
```
SQL> Select * from new_ott where genre=’romance’ or genre=’comedy’;

 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       205 ZNMD
Movie                                                 19112022 comedy
F.D                           7          9        280 3.9          299 No
Hindi                India
```

Exercise for logical operations
#get seires id and titile which contains 'money'
select contentid, title from new_ott where title like'%MD%';



```
SQL> select contentid, title from new_ott where title like'%MD%';

 CONTENTID TITLE
---------- --------------------
       505 ZNMD
```

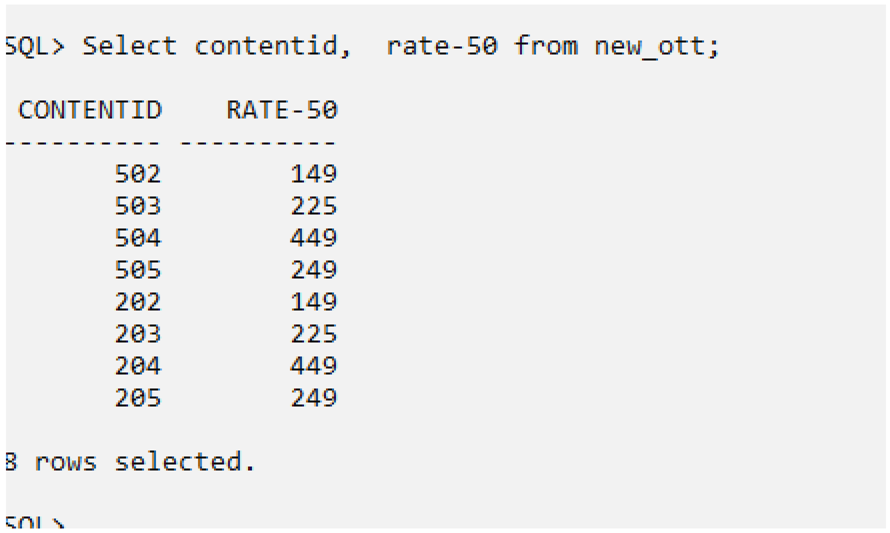
#GET THE DIRECTORS NAME ENDING WITH'H'
select * from new_ott where director like '%R';



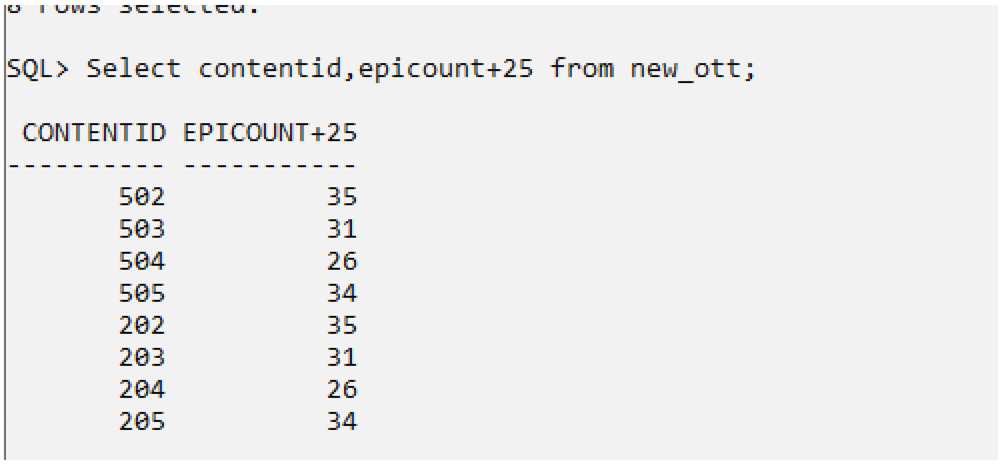
```
 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       503 After
Movie                                                 25032024 Romance
A.R                          10          6         45 4            275 Yes
English              Germany


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       202 Starlight
Web show                                              29012023 Sci-fic
J.R                           7         10        300 3.9          199 Yes
English              Italy
```

d) Arithmetic OPERATORS
Select 10+20 from DUAL;

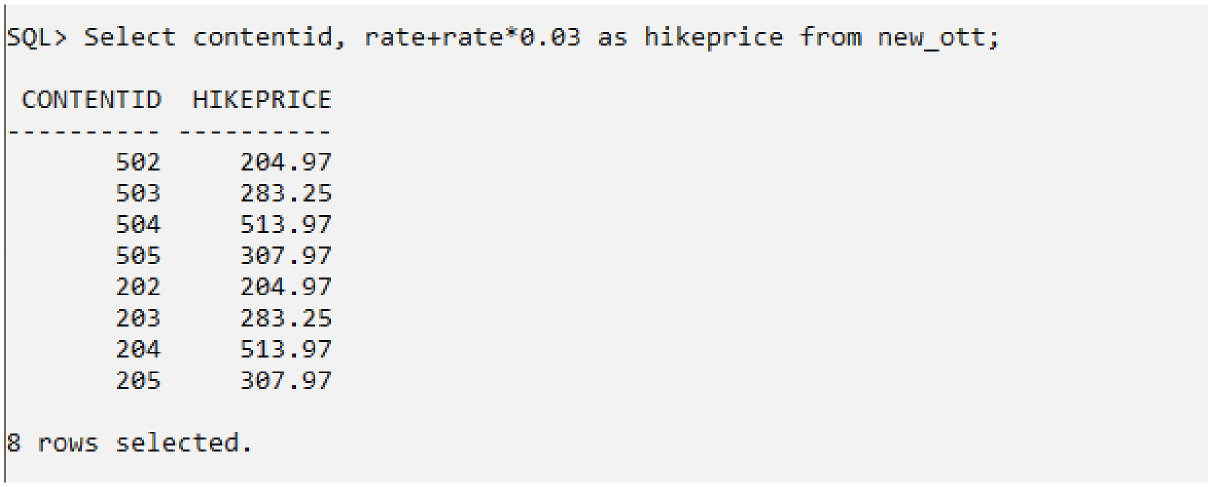
#Display the rate after deducting 30rs
Select contentid, rate-50 from new_ott;


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

 CONTENTID    RATE-50
---------- ----------
       502        149
       503        225
       504        449
       505        249
       202        149
       203        225
       204        449
       205        249

8 rows selected.

SQL>
```

#Display the epicount increased by 2
Select contentid,epicount+25 from new_ott;


```
8 rows selected.

SQL> Select contentid,epicount+25 from new_ott;

 CONTENTID EPICOUNT+25
---------- -----------
       502          35
       503          31
       504          26
       505          34
       202          35
       203          31
       204          26
       205          34
```

#Display the price after increasing by 3%
Select contentid, rate+rate*0.03 as hikeprice from new_ott;


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

 CONTENTID  HIKEPRICE
---------- ----------
       502     204.97
       503     283.25
       504     513.97
       505     307.97
       202     204.97
       203     283.25
       204     513.97
       205     307.97

8 rows selected.
```

e) Comparison operators

Descrpition:

Comparison conditions are used to determine which records to select.

a) Greater than(>)

Syntax:

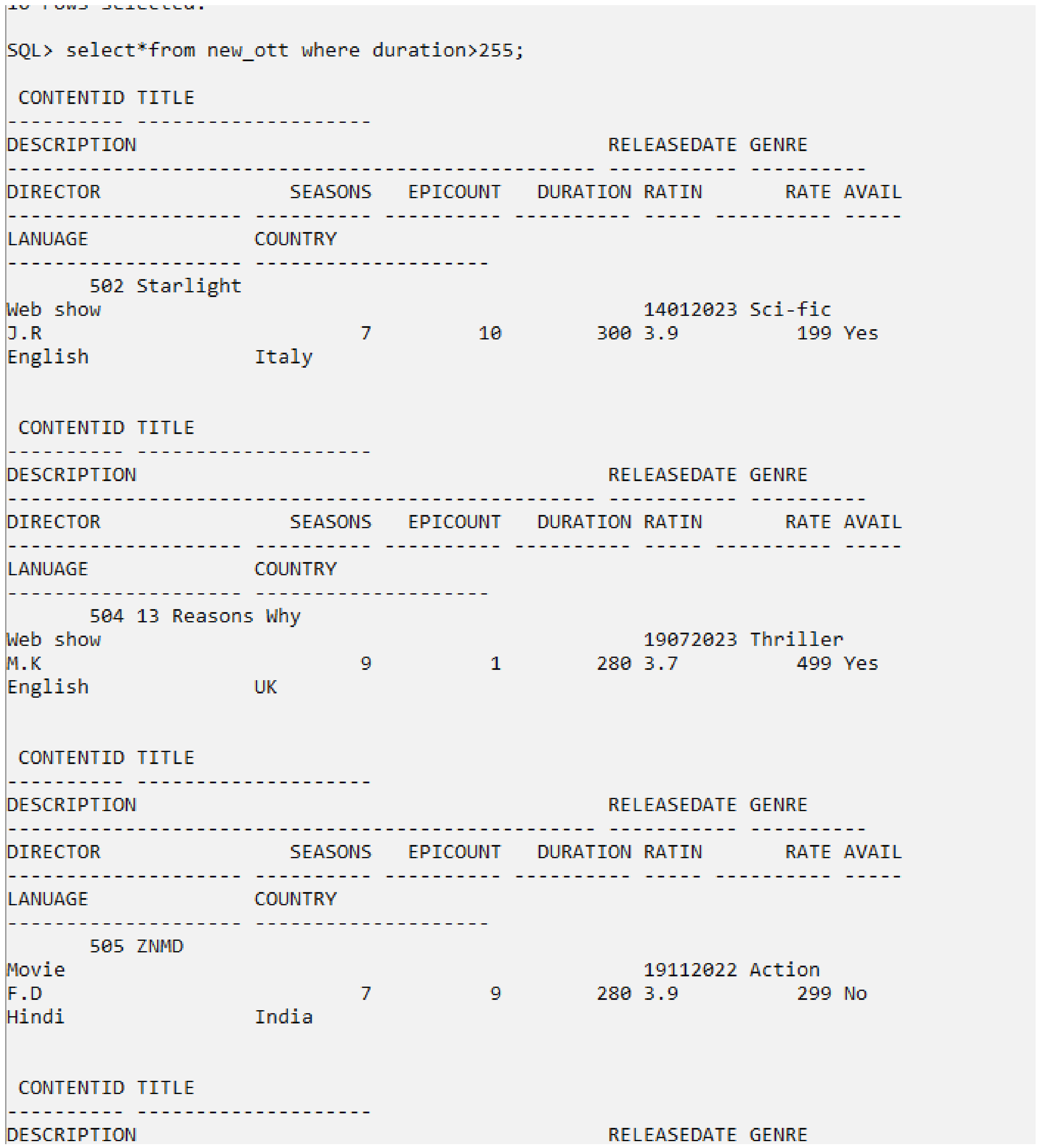
Select * from table_name where col_name>value;

Code:

Duration:

select*from new_ott where duration>255;

Output:



```
10 rows selected.

SQL> select*from new_ott where duration>255;

 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       502 Starlight
Web show                                              14012023 Sci-fic
J.R                           7         10        300 3.9          199 Yes
English              Italy


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       504 13 Reasons Why
Web show                                              19072023 Thriller
M.K                           9          1        280 3.7          499 Yes
English              UK


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       505 ZNMD
Movie                                                 19112022 Action
F.D                           7          9        280 3.9          299 No
Hindi                India


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
```

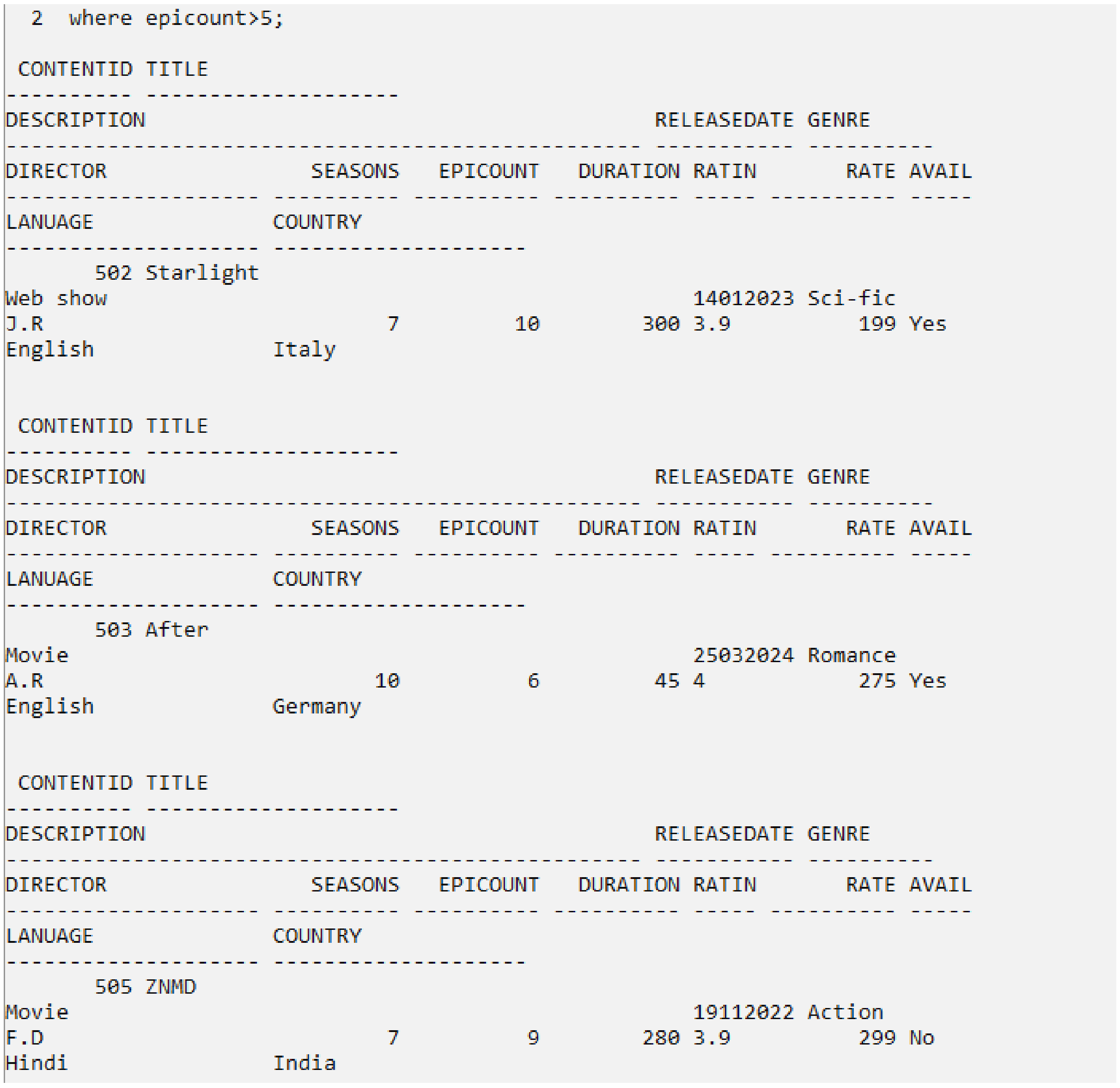
#Exercise for greater than

Code:

select * from new_ott

where epicount>5;

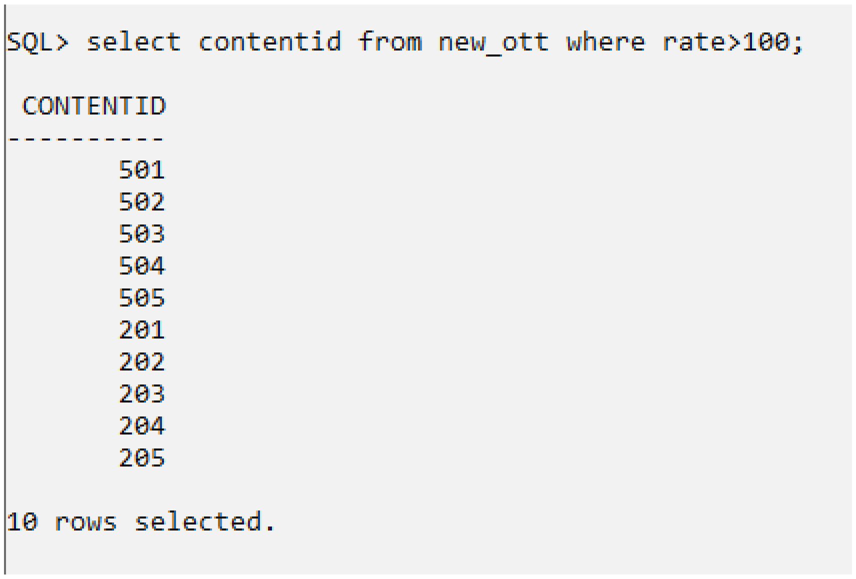
Output:



```
  2  where epicount>5;

 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       502 Starlight
Web show                                              14012023 Sci-fic
J.R                           7         10        300 3.9          199 Yes
English              Italy


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       503 After
Movie                                                 25032024 Romance
A.R                          10          6         45 4            275 Yes
English              Germany


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       505 ZNMD
Movie                                                 19112022 Action
F.D                           7          9        280 3.9          299 No
Hindi                India
```

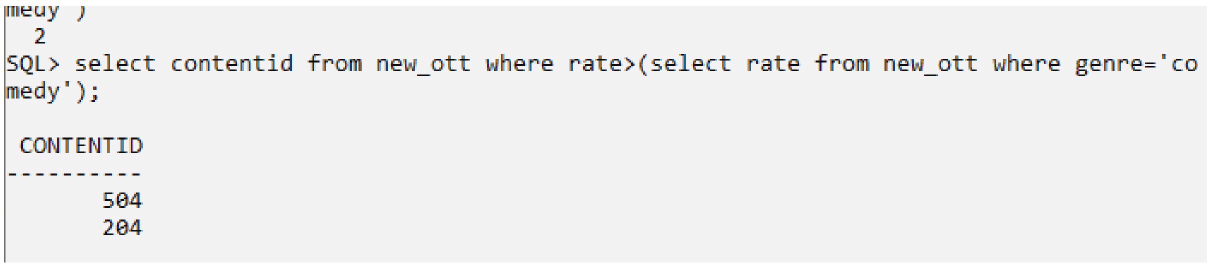
#fetch content id of series with price more than 100

select contentid from new_ott where rate>100;


```
SQL> select contentid from new_ott where rate>100;

 CONTENTID
----------
       501
       502
       503
       504
       505
       201
       202
       203
       204
       205

10 rows selected.
```

Exercise on greater than

select contentid from new_ott where rate>(select rate from new_ott where genre='comedy')


```
medy')
  2
SQL> select contentid from new_ott where rate>(select rate from new_ott where genre='co
medy');

 CONTENTID
----------
       504
       204
```

Equal(=)

Syntax:

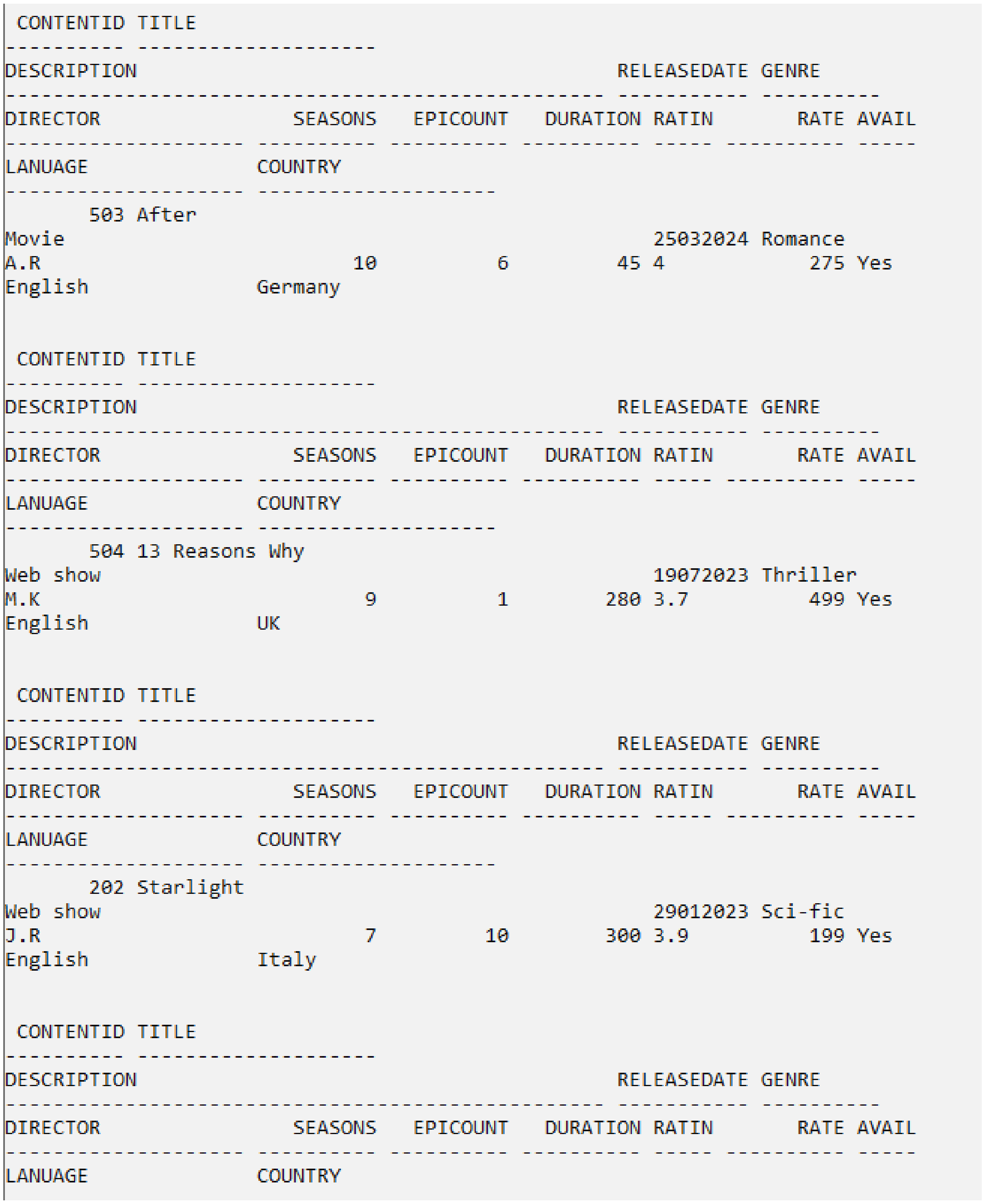
Select 8 from table_name where col_name=value;

Code:

#Fetch the available series details

select * from new_ott where availability='Yes';

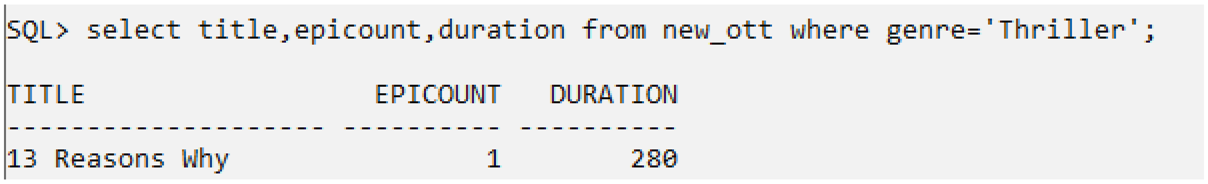
Output:



```
 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       503 After
Movie                                                 25032024 Romance
A.R                          10          6         45 4            275 Yes
English              Germany


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       504 13 Reasons Why
Web show                                              19072023 Thriller
M.K                           9          1        280 3.7          499 Yes
English              UK


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       202 Starlight
Web show                                              29012023 Sci-fic
J.R                           7         10        300 3.9          199 Yes
English              Italy


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
```

Exercise 2 on equal comparison operator

#get the title,episode count and duration of thriller series select title,epicount,duration from new_ott where genre='Thriller';



```
SQL> select title,epicount,duration from new_ott where genre='Thriller';

TITLE                  EPICOUNT   DURATION
-------------------- ---------- ----------
13 Reasons Why                1        280
```

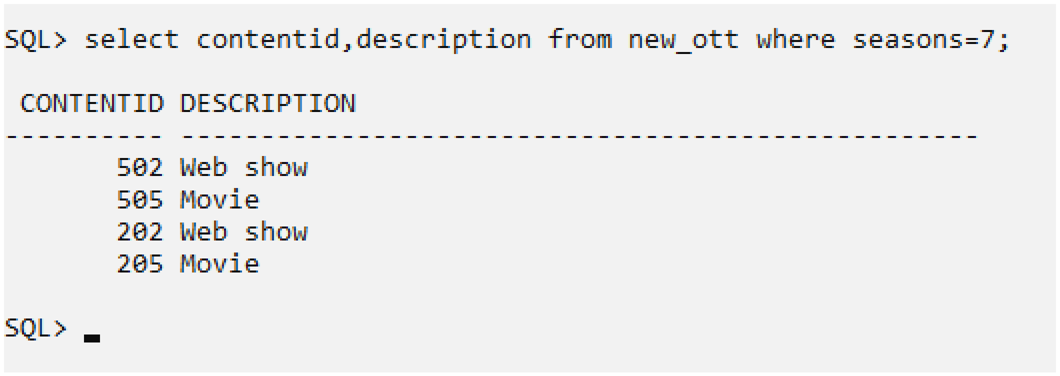
Exercise 3 on equal comparison operator

#get the content id,description of series with 3 seasons;

Code:

select contentid,description from new_ott where seasons=7;

Output:



```
SQL> select contentid,description from new_ott where seasons=7;

 CONTENTID DESCRIPTION
---------- --------------------------------------------------
       502 Web show
       505 Movie
       202 Web show
       205 Movie

SQL>
```

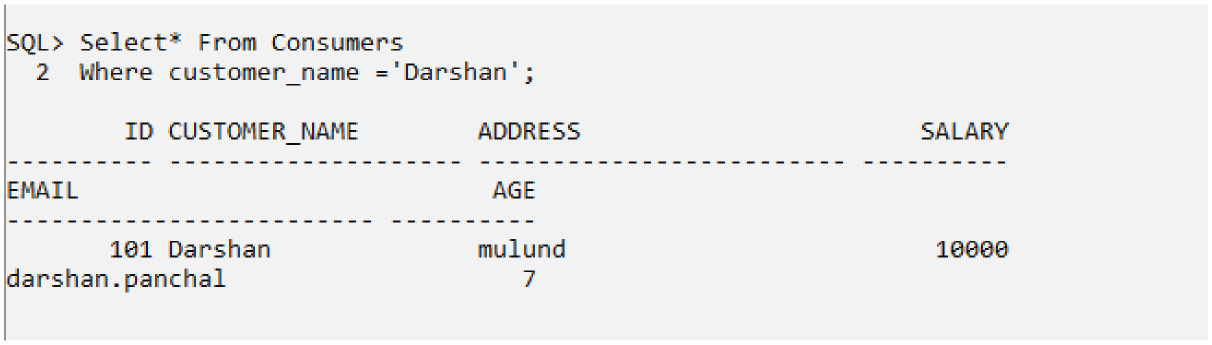
a) **Equal(=)**

   Syntax:

   Select * fron table_name where column=value;

Select* From Consumers

Where last_name ='Darshan';



```
SQL> Select* From Consumers
  2  Where customer_name ='Darshan';

        ID CUSTOMER_NAME        ADDRESS                        SALARY
---------- -------------------- ------------------------- ----------
EMAIL                            AGE
------------------------- ----------
       101 Darshan              mulund                          10000
darshan.panchal                    7
```

b) Not equal(<>)

Syntax:

Select * from table where column<>value;

Code:

Select *

From consumers

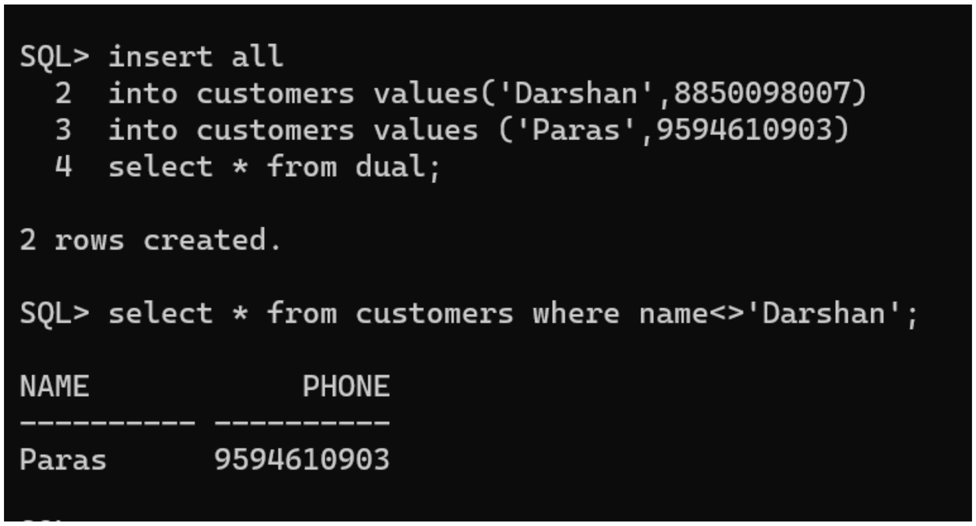
where last_name <> 'marie';

Same for

Select * From Consumers

where last_name != 'Darshan'

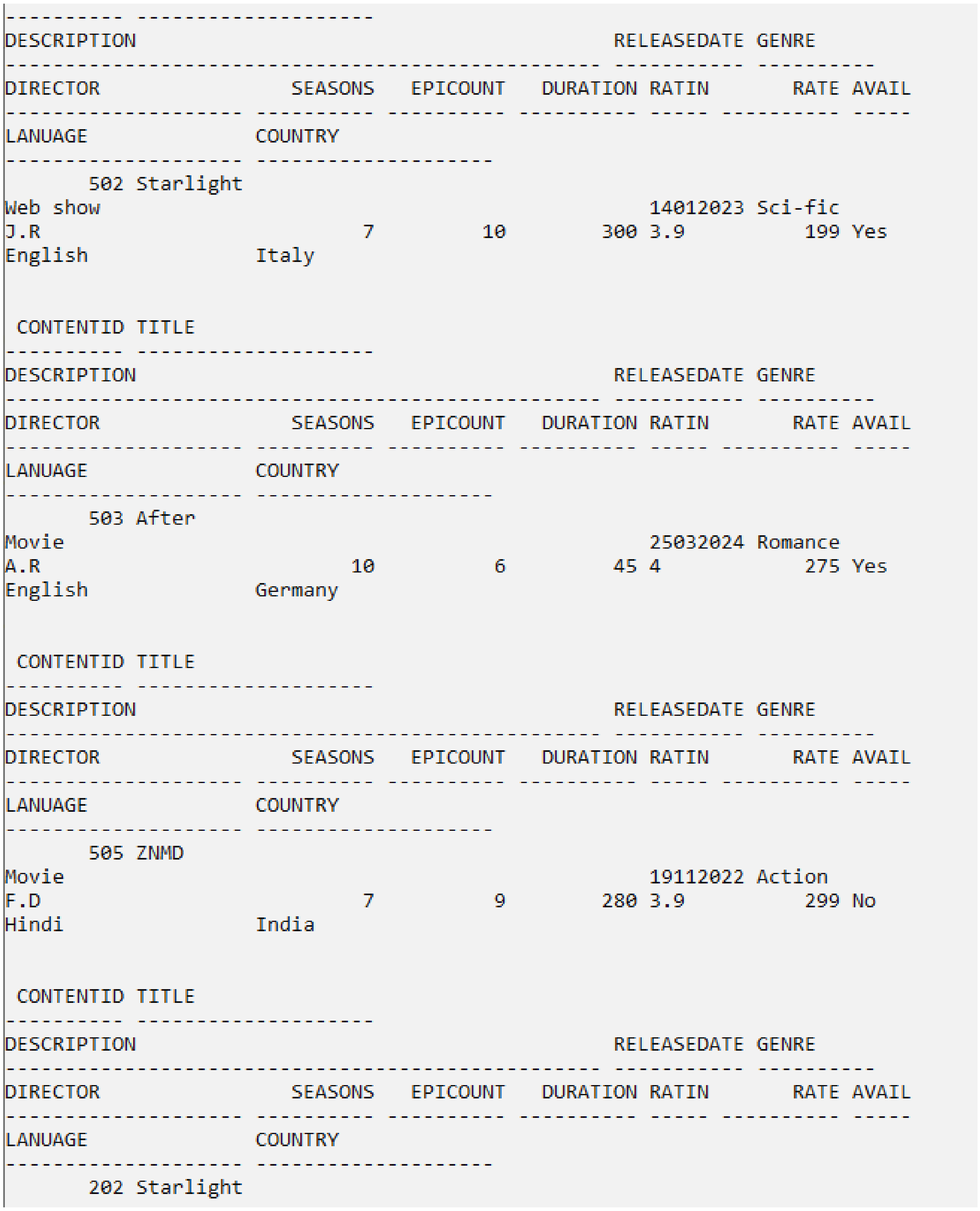
Output:



```
SQL> insert all
  2  into customers values('Darshan',8850098007)
  3  into customers values ('Paras',9594610903)
  4  select * from dual;

2 rows created.

SQL> select * from customers where name<>'Darshan';

NAME            PHONE
---------- ----------
Paras      9594610903
```

Get the details of series not in thriller genre.

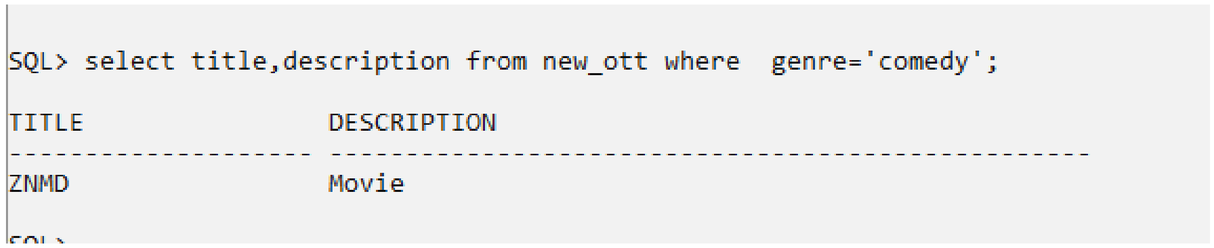
Select * from new_ott where genre<>’Thriller’;



```
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       502 Starlight
Web show                                              14012023 Sci-fic
J.R                           7         10        300 3.9          199 Yes
English              Italy


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       503 After
Movie                                                 25032024 Romance
A.R                          10          6         45 4            275 Yes
English              Germany


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       505 ZNMD
Movie                                                 19112022 Action
F.D                           7          9        280 3.9          299 No
Hindi                India


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       202 Starlight
```

#title and description of series of comedy series;

select title,description from new_ott where genre='comedy';

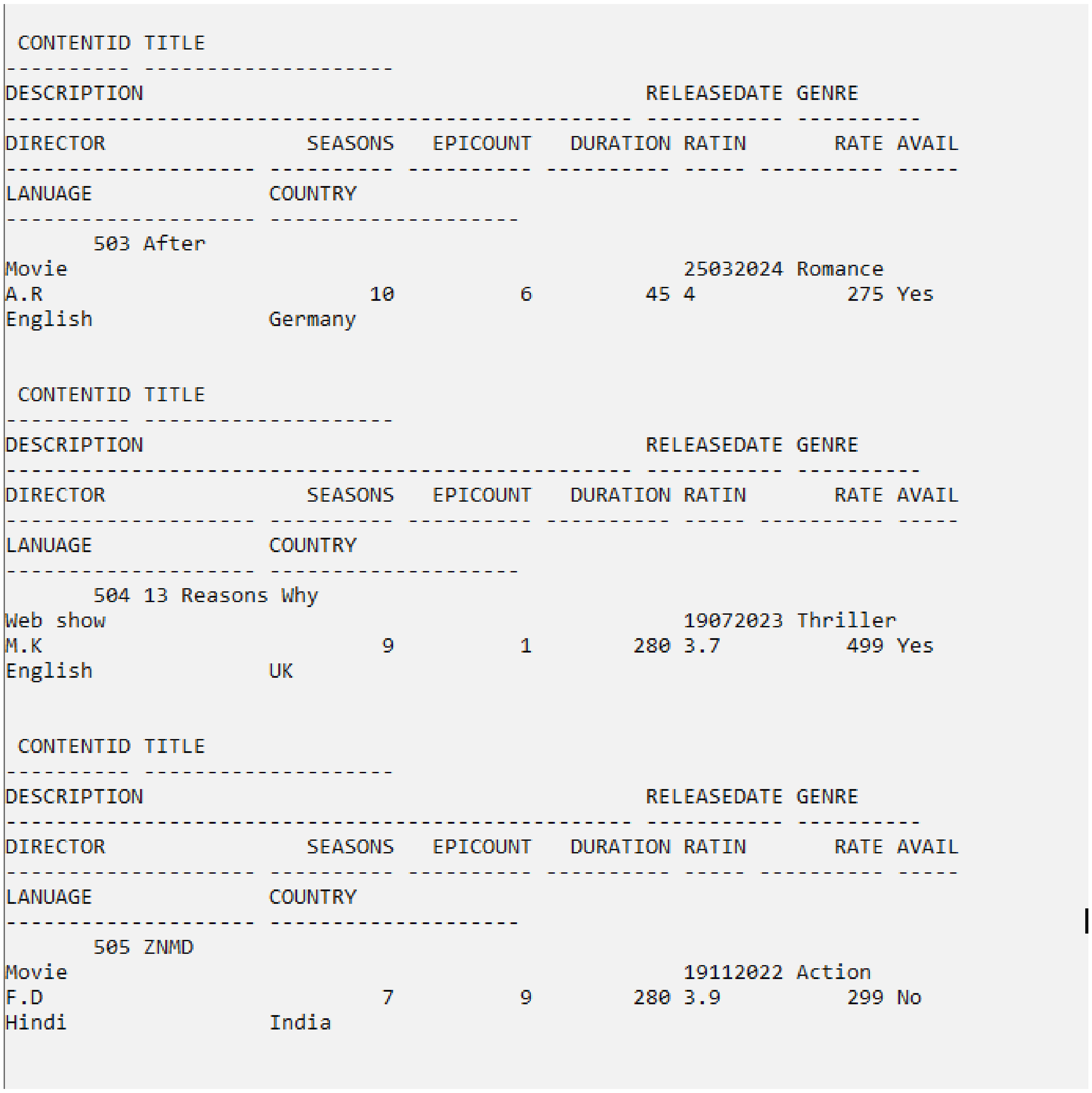


```
SQL> select title,description from new_ott where  genre='comedy';

TITLE                DESCRIPTION
-------------------- --------------------------------------------------
ZNMD                 Movie
```

Exercise for greater than date.

Seires details released before 1-1-2022

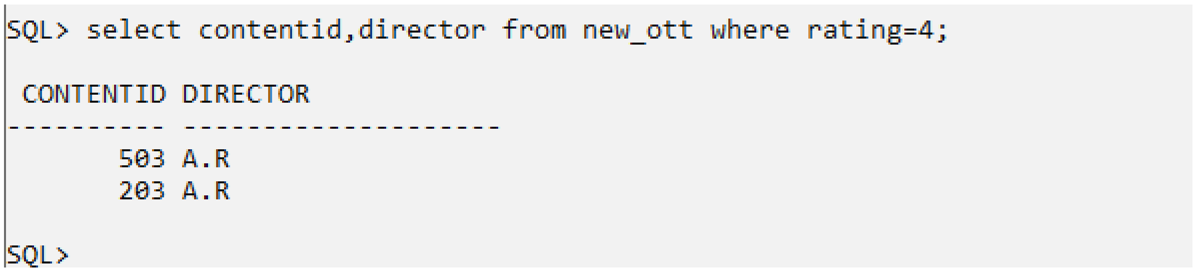
select * from new_ott where releasedate>112022;



```
 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       503 After
Movie                                                 25032024 Romance
A.R                          10          6         45 4            275 Yes
English              Germany


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       504 13 Reasons Why
Web show                                              19072023 Thriller
M.K                           9          1        280 3.7          499 Yes
English              UK


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                        RELEASEDATE GENRE
-------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       505 ZNMD
Movie                                                 19112022 Action
F.D                           7          9        280 3.9          299 No
Hindi                India
```

Exercise for equal to operator

select contentid,director from new_ott where rating=4;



```
SQL> select contentid,director from new_ott where rating=4;

 CONTENTID DIRECTOR
---------- --------------------
       503 A.R
       203 A.R

SQL>
```

c) **Less than or equal (<=)**

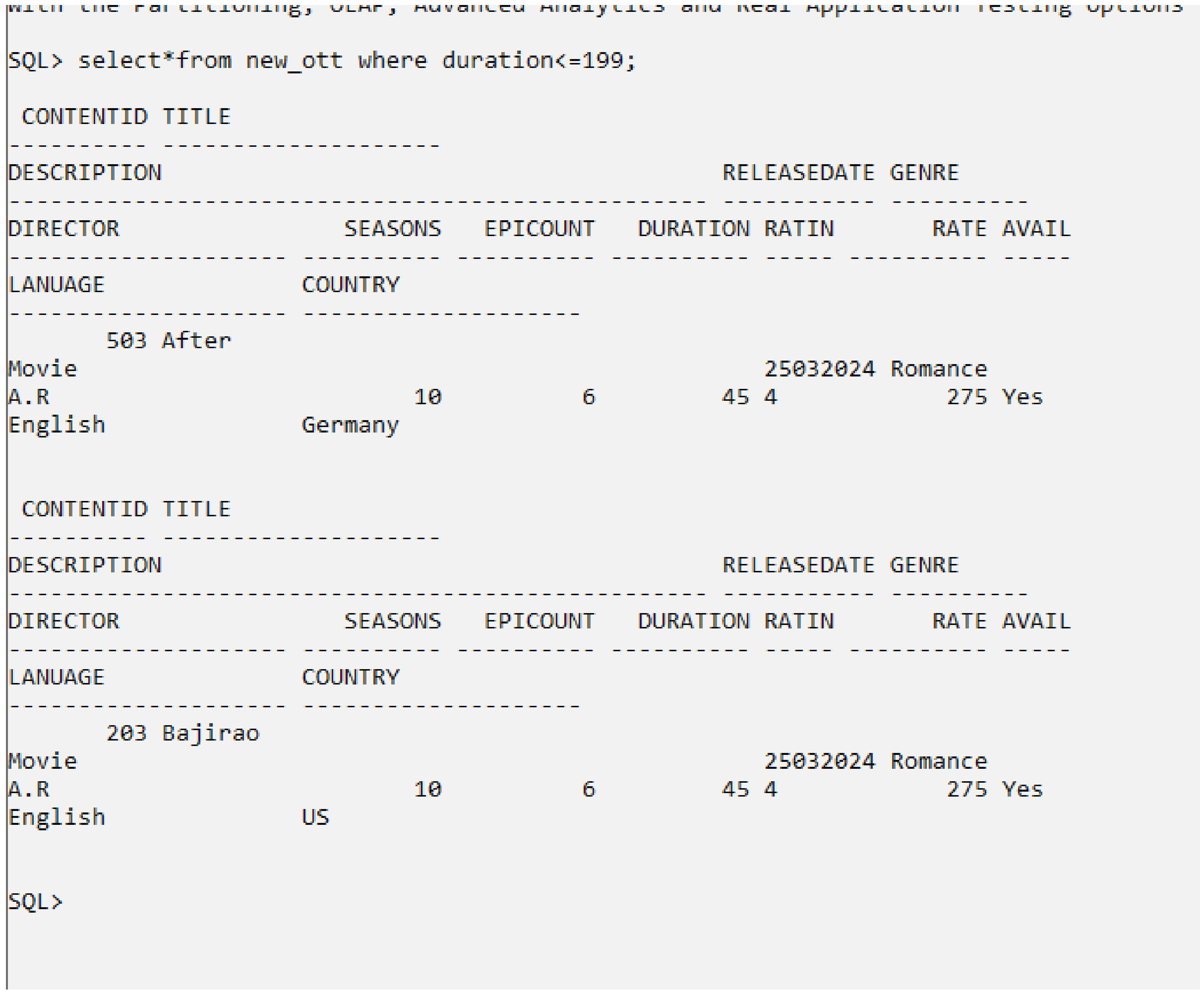
Syntax:

Select * from table_name where col_name<=value;

Code:

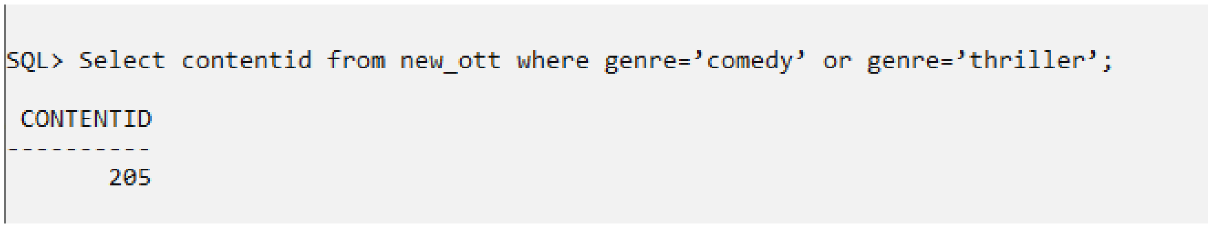
select*from new_ott where duration<=199;

Output:



```
with the Partitioning, OLAP, Advanced Analytics and Real Application Testing options

SQL> select*from new_ott where duration<=199;

 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                         RELEASEDATE GENRE
--------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       503 After
Movie                                                  25032024 Romance
A.R                          10          6         45 4            275 Yes
English              Germany


 CONTENTID TITLE
---------- --------------------
DESCRIPTION                                         RELEASEDATE GENRE
--------------------------------------------------- ----------- ----------
DIRECTOR                SEASONS   EPICOUNT   DURATION RATIN       RATE AVAIL
-------------------- ---------- ---------- ---------- ----- ---------- -----
LANUAGE              COUNTRY
-------------------- --------------------
       203 Bajirao
Movie                                                  25032024 Romance
A.R                          10          6         45 4            275 Yes
English              US


SQL>
```

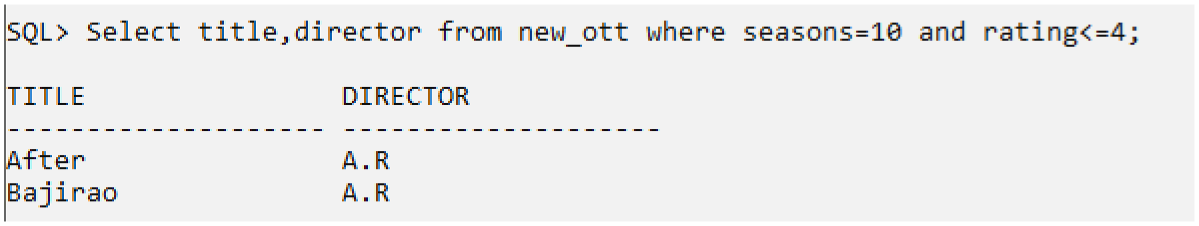
#Get the content ids of comedy or thriller series
Select contentid from new_ott where genre='comedy' or genre='thriller';



```
SQL> Select contentid from new_ott where genre='comedy' or genre='thriller';

 CONTENTID
----------
       205
```

Exercise for less than or equal

#Get the title director of contents with 3 seasons and rating 5/4
Select title,director from new_ott where seasons=10 and rating<=4;

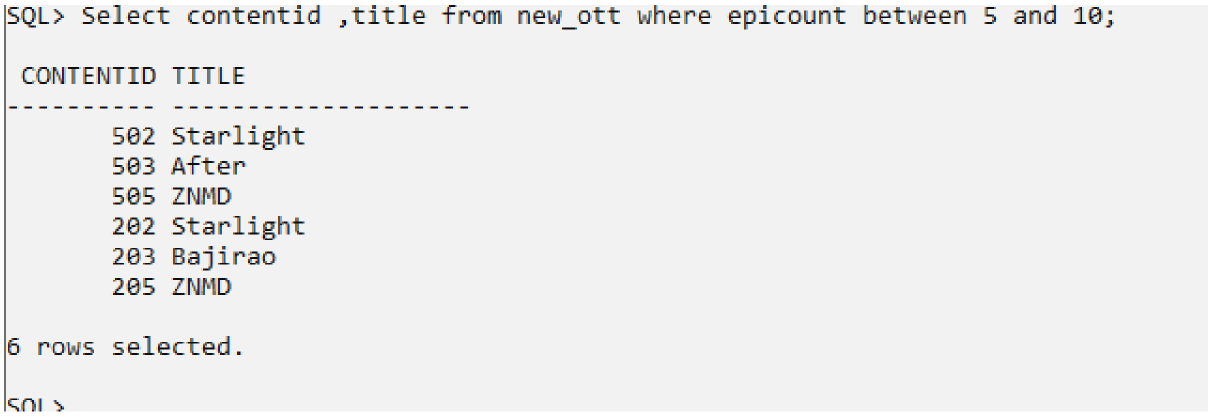


```
SQL> Select title,director from new_ott where seasons=10 and rating<=4;

TITLE                DIRECTOR
-------------------- --------------------
After                A.R
Bajirao              A.R
```

d) Between the Range:
Syntax:

#expression Between value1 and value2;
Code:
Select * from customers
WHERE customer_id BETWEEN 4000 AND 4999;

#Get the content id and title of series with episode count more than 50 and less than 100.
Select contentid ,title from new_ott where epicount between 5 and 10;



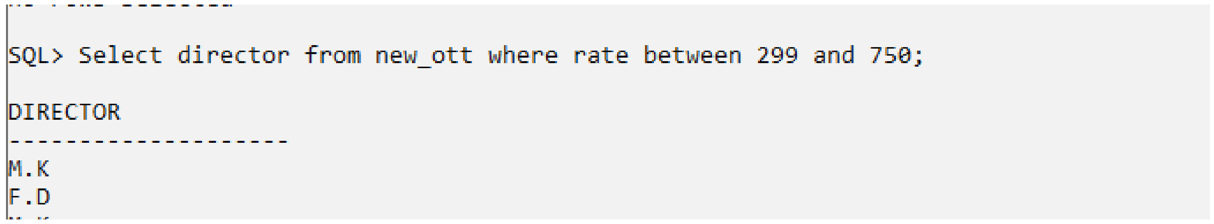
```
SQL> Select contentid ,title from new_ott where epicount between 5 and 10;

 CONTENTID TITLE
---------- --------------------
       502 Starlight
       503 After
       505 ZNMD
       202 Starlight
       203 Bajirao
       205 ZNMD

6 rows selected.

SQL>
```

e) **Omit/skip some range/Not-between**
Syntax:
Select * from table where col not between value1 and value 2;

From consumers WHERE consumer_id NOT BETWEEN 3000 AND 3500;

#GEt the director of series ranges between 500-700
Select director from new_ott where rate between 299 and 750;

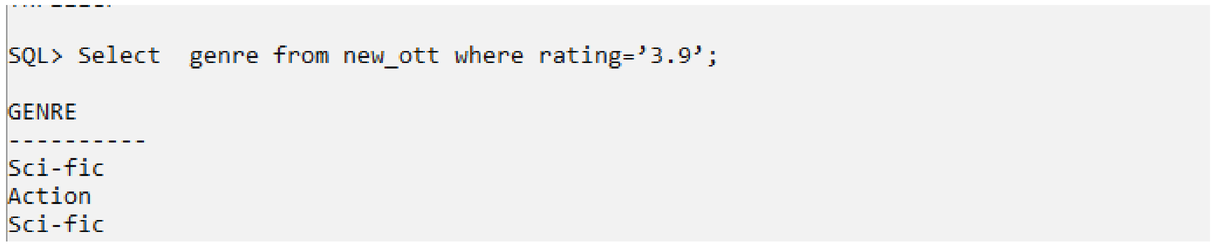
OUTPUT:



```
SQL> Select director from new_ott where rate between 299 and 750;

DIRECTOR
--------------------
M.K
F.D
```

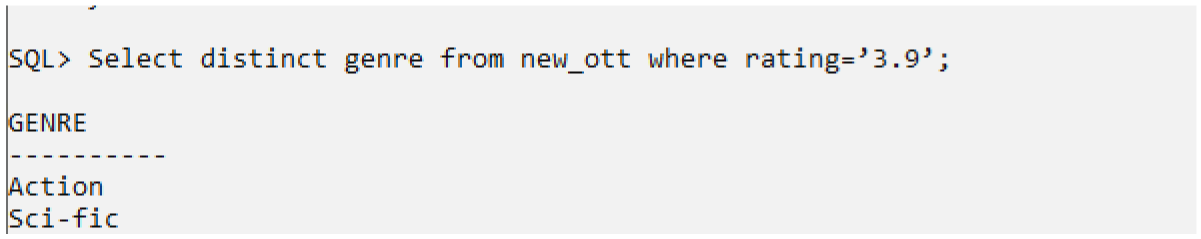
f)DISTINCT Keyword
Description:
In distinct you get unique values even though you ha
Syntax:
SELECT DISTINCT expressions FROM tables [where condition];

Code for normal select statement:
Select genre from new_ott where rating='3.9';



```
SQL> Select  genre from new_ott where rating='3.9';

GENRE
----------
Sci-fic
Action
Sci-fic
```

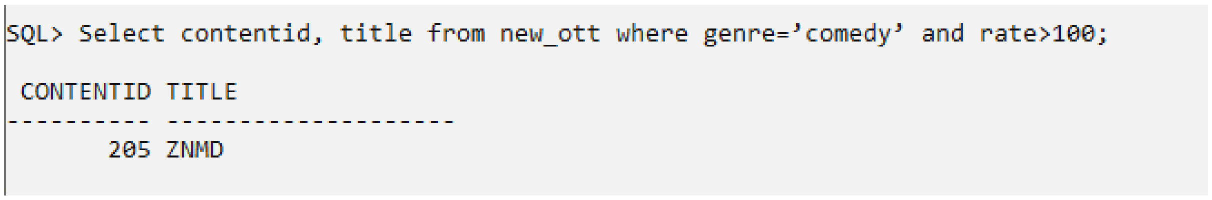
Code for distinct keyword
Select distinct genre from new_ott where rating='3.9';



```
SQL> Select distinct genre from new_ott where rating='3.9';

GENRE
----------
Action
Sci-fic
```

#Comparison Conditions

#Retrieve the id and title of the series in comedy genre with price more than 5000
Select contentid, title from new_ott where genre='comedy' and rate>100;



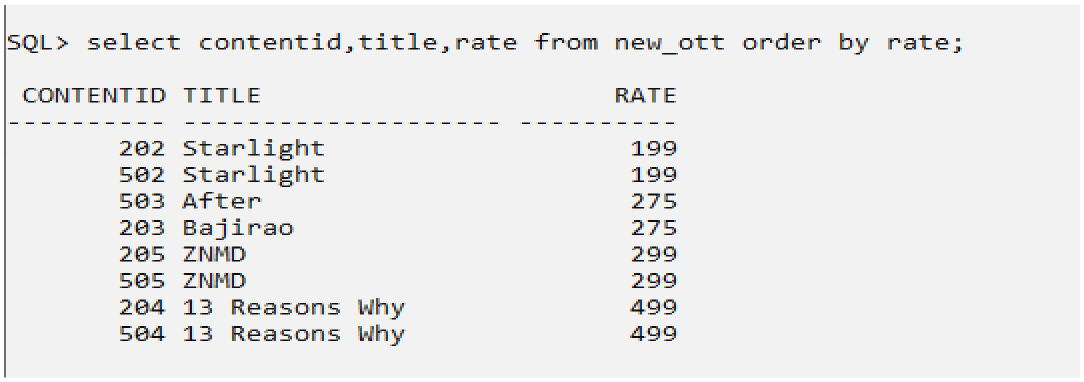
```
SQL> Select contentid, title from new_ott where genre='comedy' and rate>100;

 CONTENTID TITLE
---------- --------------------
       205 ZNMD
```

f) Order by

Two types of orderby is used to sort the table data in entire ascending and descending order.
Ascending order or **descending order** is in ascending (asc) only but for descending we need to write (desc).
Syntax:
select column-list
from rable_name
[where condition]
[order by column1, column2.. column N] [asc|desc];

a. Ascending Order:

Syntax:
Select * from table_name order by col_name;
Question:
fetch contentid and title of series in ascending order of price.
Code:
select contentid,title,rate from new_ott order by rate;

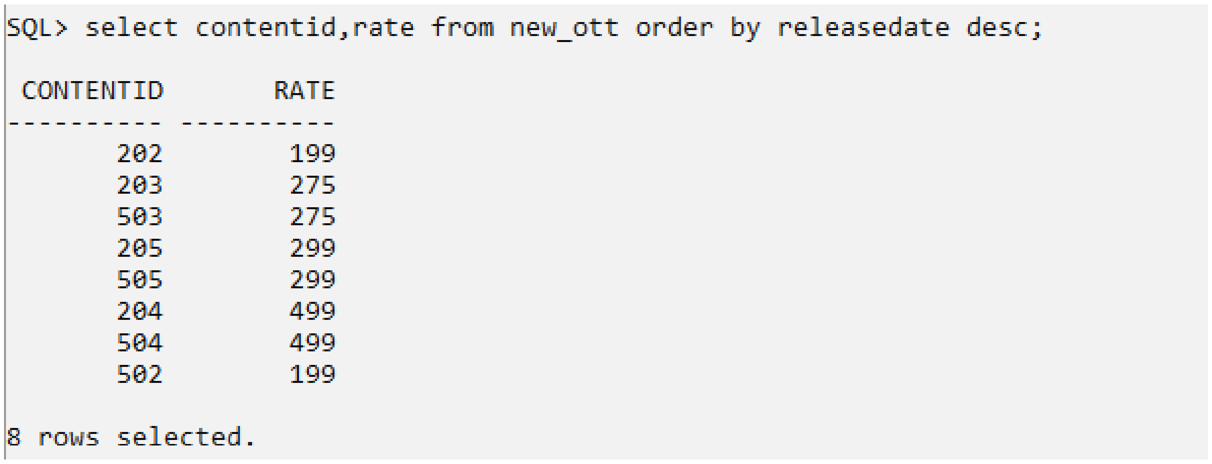


```
SQL> select contentid,title,rate from new_ott order by rate;

 CONTENTID TITLE                      RATE
---------- -------------------- ----------
       202 Starlight                   199
       502 Starlight                   199
       503 After                       275
       203 Bajirao                     275
       205 ZNMD                        299
       505 ZNMD                        299
       204 13 Reasons Why              499
       504 13 Reasons Why              499
```

b. Descending:

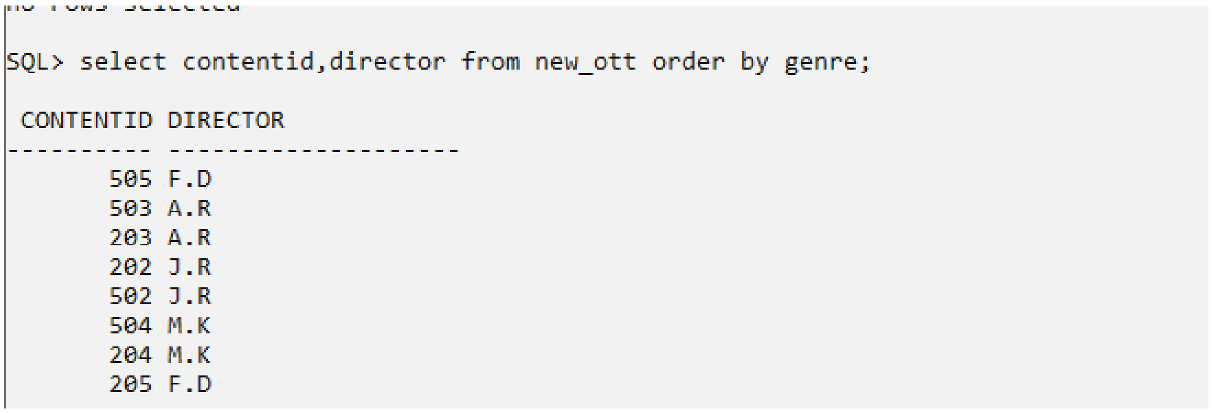
Question:

Fetch the contentid and price in descending order by release date.
Code:
select contentid,rate from new_ott order by releasedate desc;



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

 CONTENTID       RATE
---------- ----------
       202        199
       203        275
       503        275
       205        299
       505        299
       204        499
       504        499
       502        199

8 rows selected.
```

Exercise for ascending order by
fetch the content id director details in ascending order of genre
select contentid,director from new_ott order by genre;

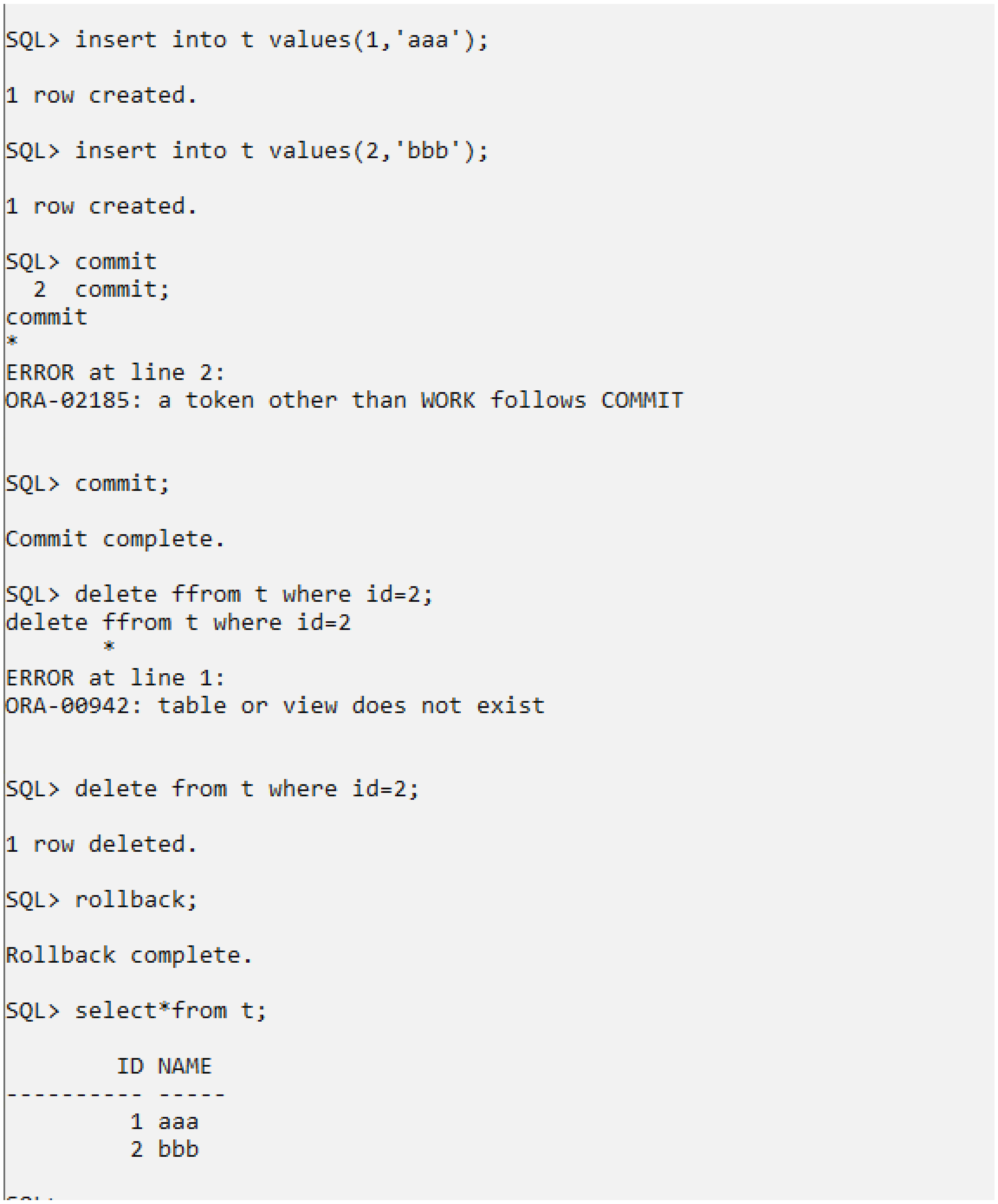


```
SQL> select contentid,director from new_ott order by genre;

 CONTENTID DIRECTOR
---------- --------------------
       505 F.D
       503 A.R
       203 A.R
       202 J.R
       502 J.R
       504 M.K
       204 M.K
       205 F.D
```

Practical -4 : Transaction Control

1. **Commit operation**
   It saves all the table data.
2. **Rollback operation**

It gives the values back from the table if it is deleted.



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

1 row created.

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

1 row created.

SQL> commit
  2  commit;
commit
*
ERROR at line 2:
ORA-02185: a token other than WORK follows COMMIT


SQL> commit;

Commit complete.

SQL> delete ffrom t where id=2;
delete ffrom t where id=2
       *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL> delete from t where id=2;

1 row deleted.

SQL> rollback;

Rollback complete.

SQL> select*from t;

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

```

Practical 5 : Functions

Functions in sql

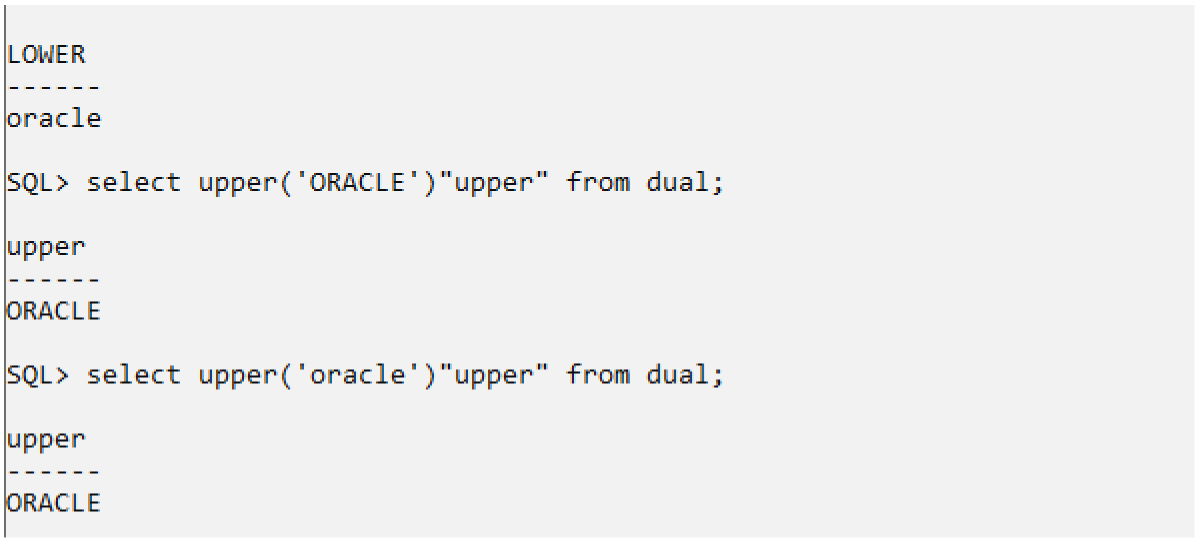
2. Character Functions

#Lower function

select lower('ORACLE')"LOWER" from dual;

#Upper function

select upper('oracle')"upper" from dual;



```
LOWER
------
oracle

SQL> select upper('ORACLE')"upper" from dual;

upper
------
ORACLE

SQL> select upper('oracle')"upper" from dual;

upper
------
ORACLE
```

*Manipulate character strings

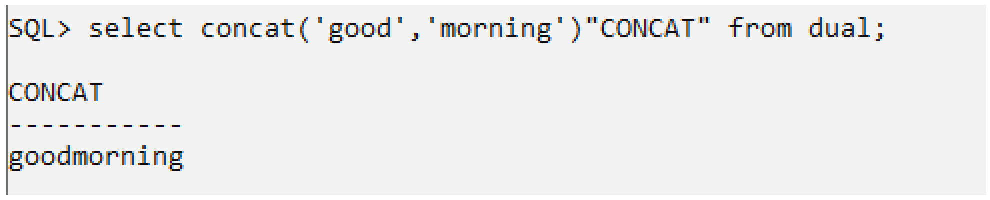
2. Conversion Function

Functions	result
Rpad(name,10,”*”)	han******

Functions	result

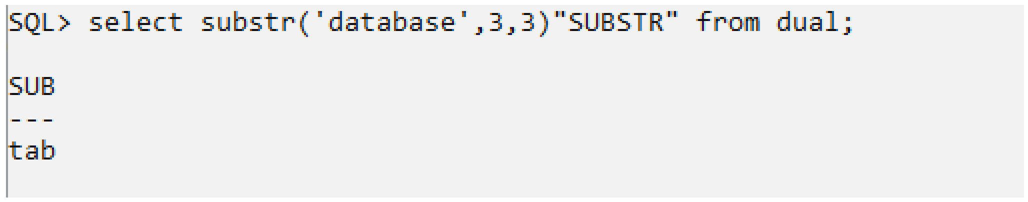
CONCAT(‘Good’,’string’)	Goodstring
SUBSTR(‘string’,1,3)	str
LENGTH(“STRING”)	6
Lpad(sal,10,”*”)	******5000
Rpad(name,10,”*”)	han******
ASCII of (‘a’)	97

#**Concatenate**
select concat('good','morning')"CONCAT" from dual;



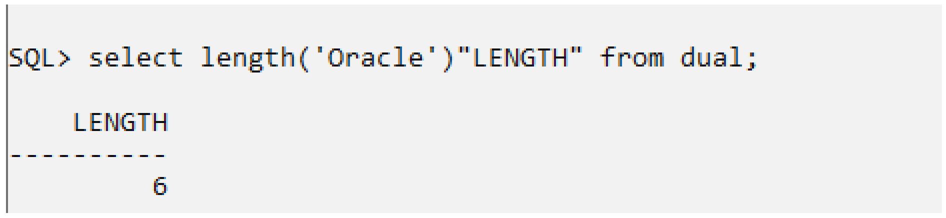
```
SQL> select concat('good','morning')"CONCAT" from dual;

CONCAT
-----------
goodmorning
```

#Substring
select substr('database',3,3)"SUBSTR" from dual;



```
SQL> select substr('database',3,3)"SUBSTR" from dual;

SUB
---
tab
```

#Length
select length('Oracle')"LENGTH" from dual;



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

    LENGTH
----------
         6
```

#intrstring

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

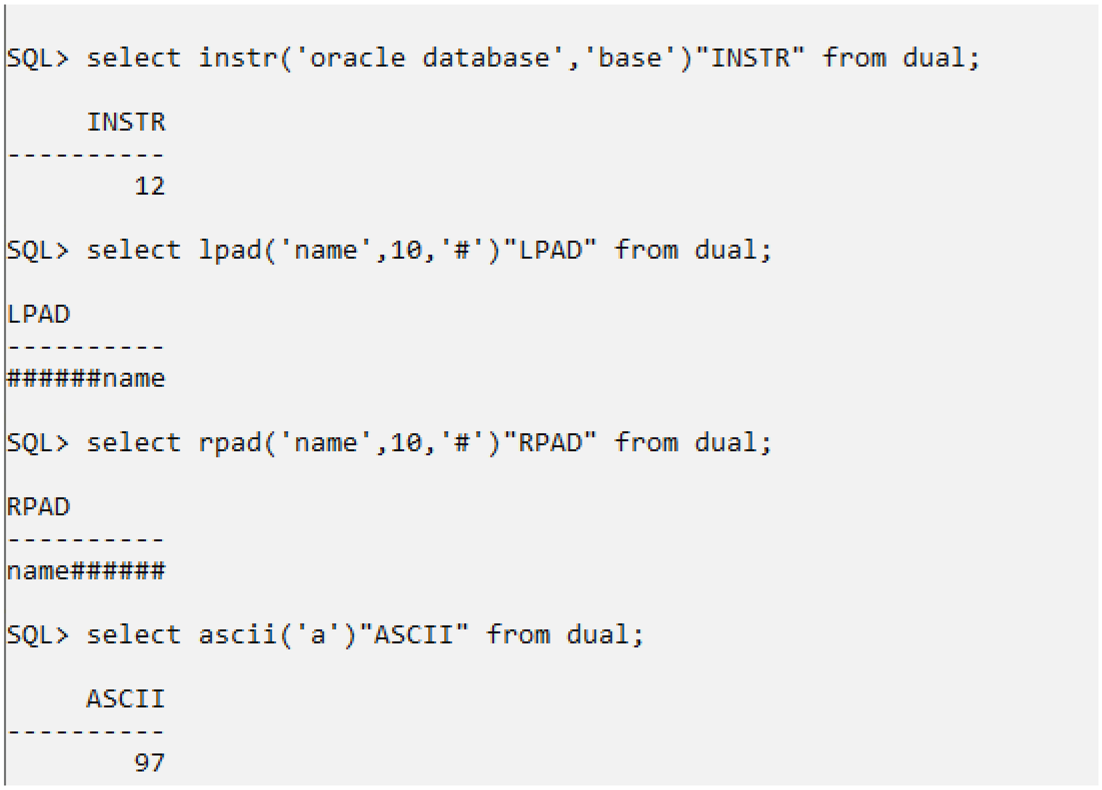
#LPAD
Left padding
select lpad('name',10,'#')"LPAD" from dual;

#RPAD
Right padding
Select rpad('name',10,'#')"rpad" from dual;

select rpad('name',10,'#')"RPAD" from dual;

3. **Conversion Function:**
   a. **ASCII**

Syntax:
select ascii('a')"ASCII" from dual;
Output:



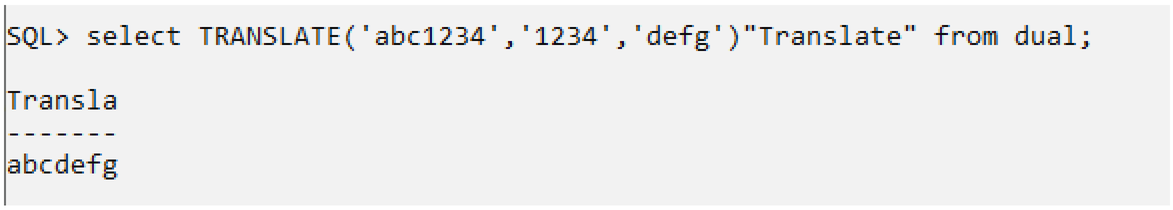
```
SQL> select instr('oracle database','base')"INSTR" from dual;

     INSTR
----------
        12

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

LPAD
----------
######name

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

RPAD
----------
name######

SQL> select ascii('a')"ASCII" from dual;

     ASCII
----------
        97
```

3.Character Functions

Functions	result

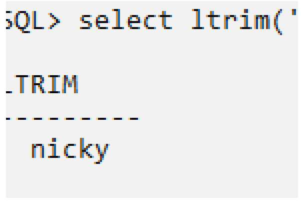
TRANSLATE(‘abc1eb23’,’134’,’abc’)	abcAeBC
Ltrim(‘nicky’,’n’)	icky
rtrim(‘nicky’,’n’)	nicky
TRIM (‘nicky ‘)	nicky

select TRANSLATE('abc1234','1234','defg')"Translate" from dual;



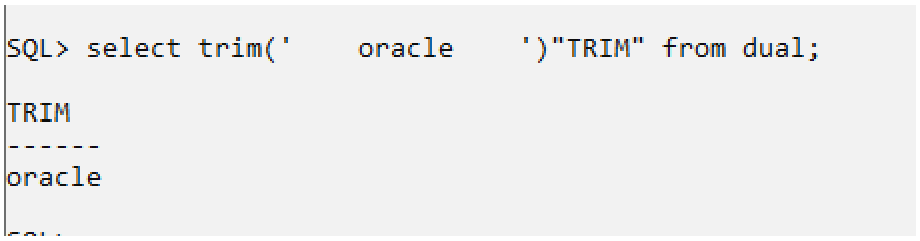
```
SQL> select TRANSLATE('abc1234','1234','defg')"Translate" from dual;

Transla
-------
abcdefg
```

select ltrim(' nicky ','n')"LTRIM" from dual;



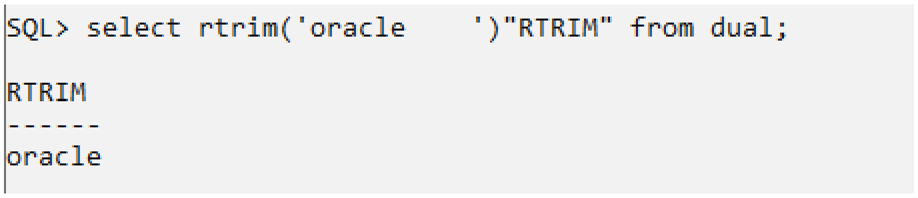
```
QL> select ltrim('

TRIM
---------
  nicky
```

select trim('   oracle   ')"TRIM" from dual;



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

TRIM
------
oracle
```

select trim('oracle   ')"TRIM" from dual;

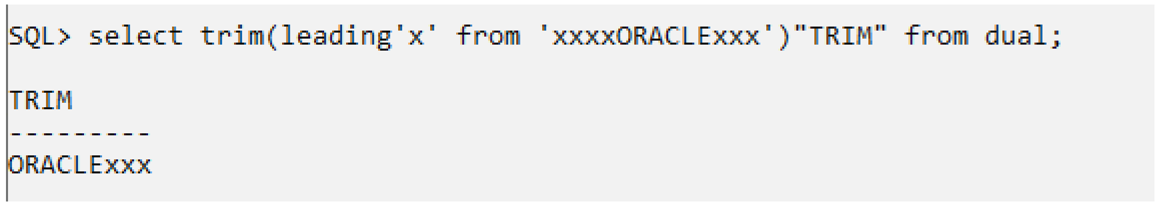


```
SQL> select rtrim('oracle    ')"RTRIM" from dual;

RTRIM
------
oracle
```

select rtrim(‘oracle’,’o’)”RTRIM” from dual;

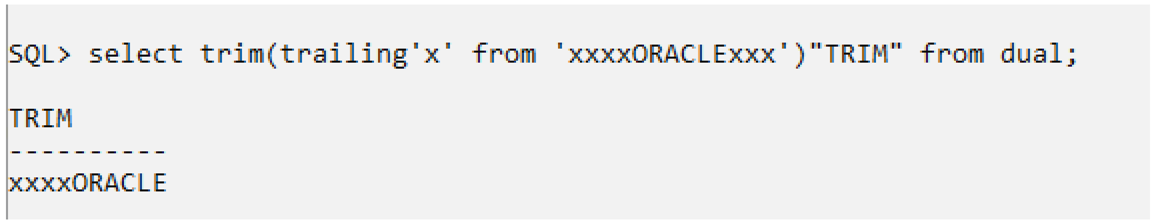
select rtrim(‘oracle’,’cle’)”RTRIM” from dual;

#TRIMMING , TRAILING and BOTH
select trim(leading'x' from 'xxxxORACLExxx')"TRIM" from dual;



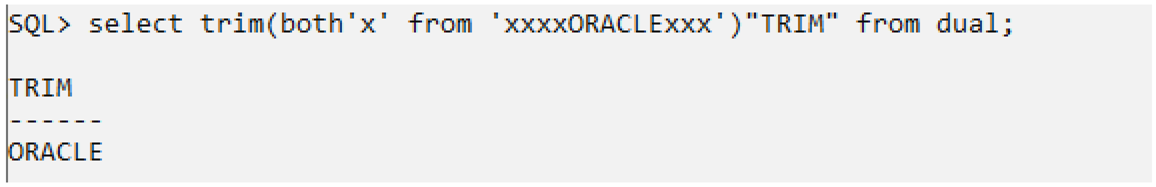
```
SQL> select trim(leading'x' from 'xxxxORACLExxx')"TRIM" from dual;

TRIM
---------
ORACLExxx
```

select trim(trailing'x' from 'xxxxORACLExxx')"TRIM" from dual;



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

TRIM
----------
xxxxORACLE
```

select trim(both'x' from 'xxxxORACLExxx')"TRIM" from dual;



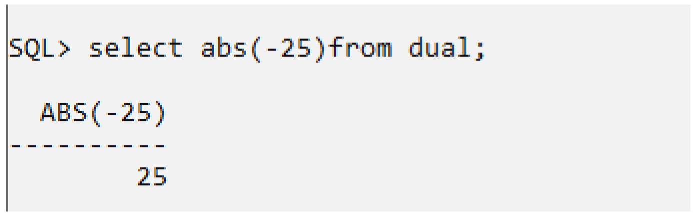
```
SQL> select trim(both'x' from 'xxxxORACLExxx')"TRIM" from dual;

TRIM
------
ORACLE
```

4. List of numeric functions

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

ABS(n)	Absolute value of n
POWER(m,n)	mn
ROUND(n,m)	N round to m places
SQRT(n)	Square root of n
EXT(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()	Smallest integer >=n

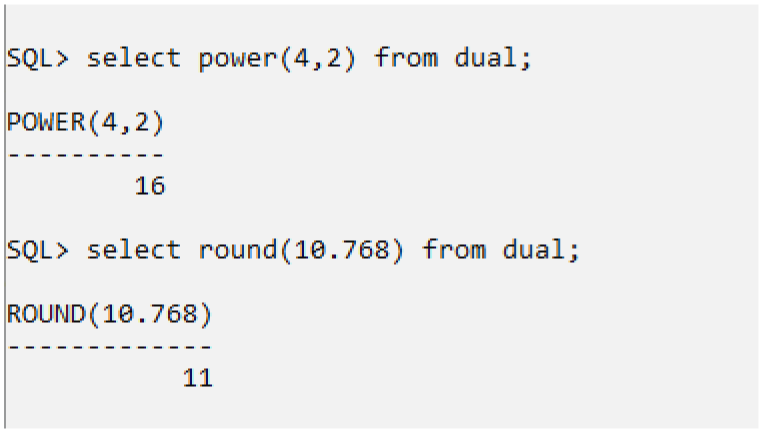
select abs(-25)from dual;



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

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

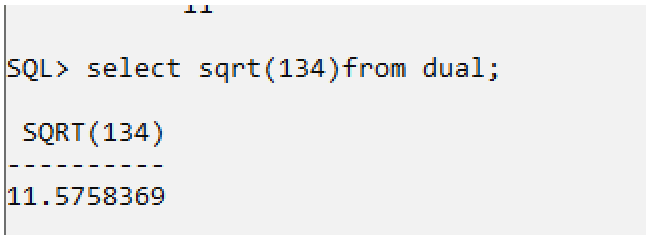
#power
select power(4,2) from dual;

#round
select round(10.768) from dual;



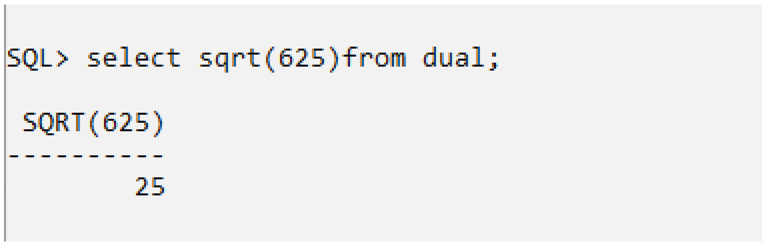
```
SQL> select power(4,2) from dual;

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

SQL> select round(10.768) from dual;

ROUND(10.768)
-------------
           11
```

#square root
select sqrt(134)from dual;

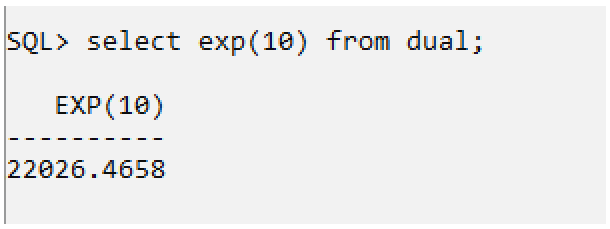


```
           11

SQL> select sqrt(134)from dual;

 SQRT(134)
----------
11.5758369
```

#square root of 625
select sqrt(625)from dual;



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

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



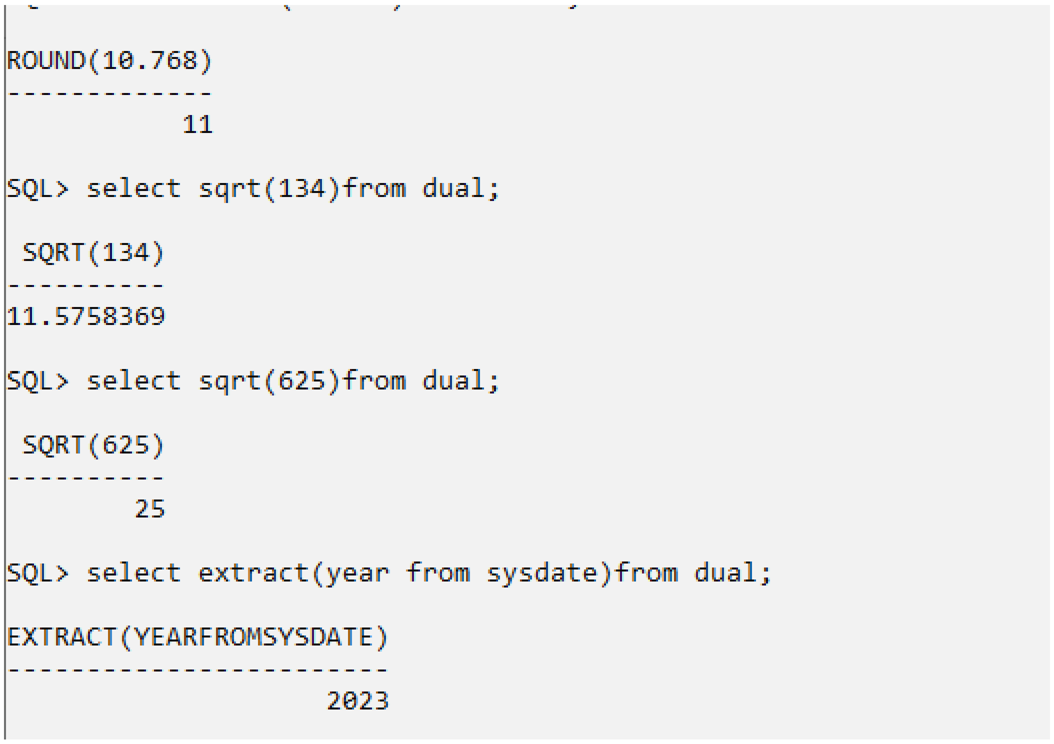
```
SQL> select exp(10) from dual;

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

#extract year
select extract(year from sysdate)from dual;
select extract(year from NOW())from dual;

select extract(day from date '2016-1-14')from dual;

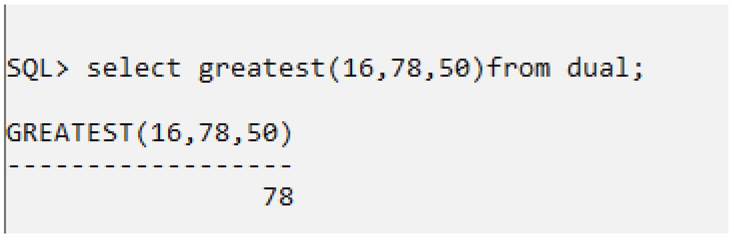
select extract(month from '2020-10-29')from dual;
SELECT EXTRACT(MONTH FROM

select sysdate(0);



```
ROUND(10.768)
-------------
           11

SQL> select sqrt(134)from dual;

 SQRT(134)
----------
11.5758369

SQL> select sqrt(625)from dual;

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

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

EXTRACT(YEARFROMSYSDATE)
------------------------
                    2023
```

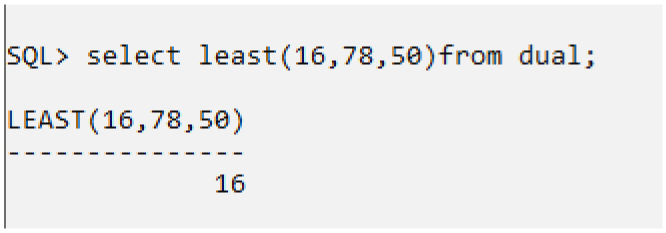
#greatest
select greatest(16,78,50)from dual;



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

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

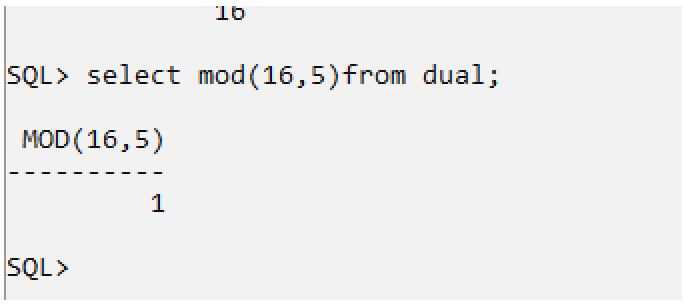
#least number

select least(16,78,50)from dual;



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

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

#mod of numbers
select mod(16,5)from dual;

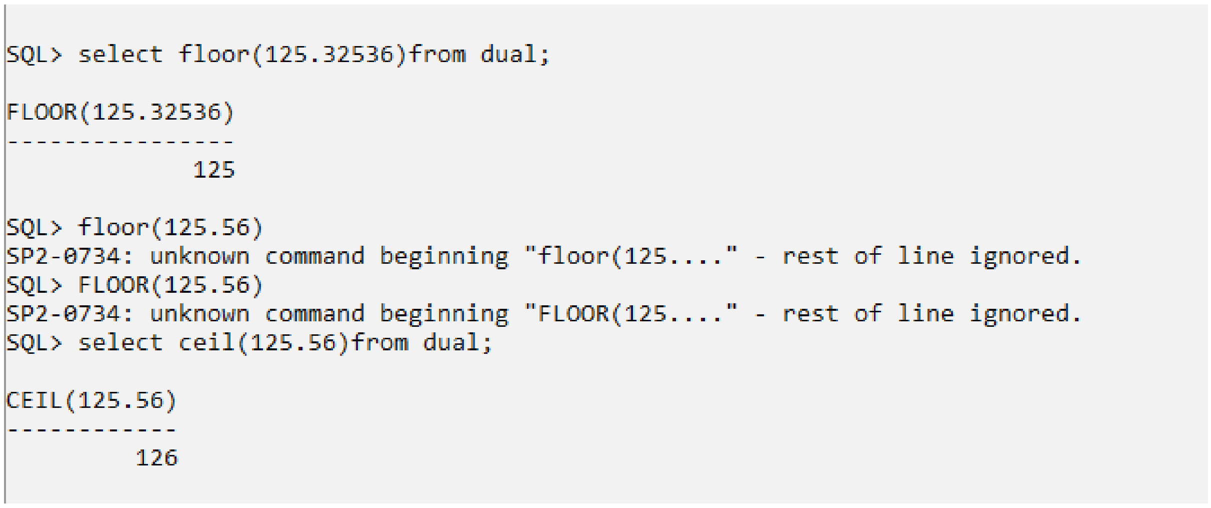


```
             16

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

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

SQL>
```

#select floor
select floor(125.32536)from dual;

#
FLOOR(125.56)

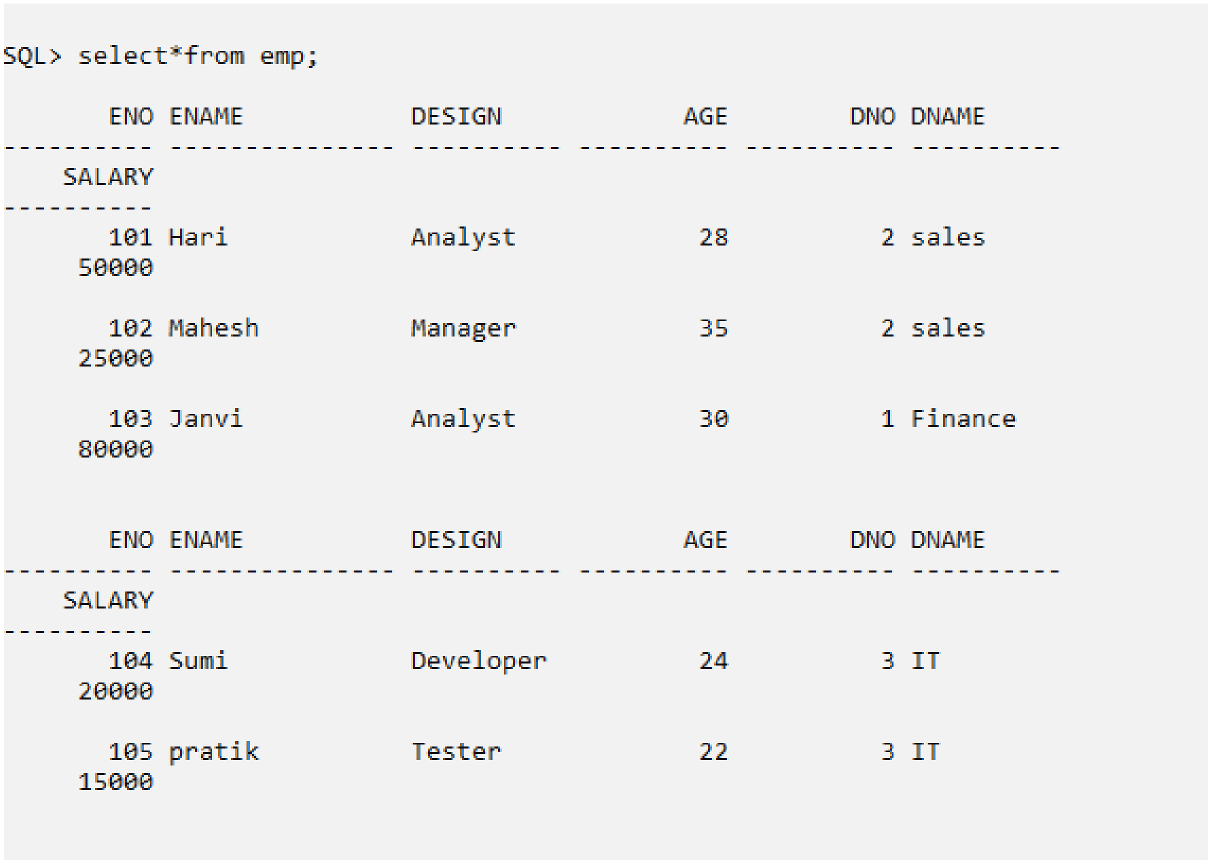
#ceil
select ceil(125.56)from dual;



```
SQL> select floor(125.32536)from dual;

FLOOR(125.32536)
----------------
             125

SQL> floor(125.56)
SP2-0734: unknown command beginning "floor(125...." - rest of line ignored.
SQL> FLOOR(125.56)
SP2-0734: unknown command beginning "FLOOR(125...." - rest of line ignored.
SQL> select ceil(125.56)from dual;

CEIL(125.56)
------------
         126
```

CREATE TABLE emp(eno INT, ename CHAR(15), design char(10), age int, dno int, dname char(10), salary number(7,2))

```
insert all
into emp values(101,'Hari','Analyst',28,2,'sales',50000.00)
into emp values(102,'Mahesh','Manager',35,2,'sales',25000.00)
into emp values(103,'Janvi','Analyst',30,1,'Finance',80000.00)
into emp values(104,'Sumi','Developer',24,3,'IT',20000.00)
into emp values(105,'pratik','Tester',22,3,'IT',15000.00)
select * from dual;
```



```
SQL> select*from emp;

       ENO ENAME           DESIGN            AGE        DNO DNAME
---------- --------------- ---------- ---------- ---------- ----------
    SALARY
----------
       101 Hari            Analyst            28          2 sales
     50000

       102 Mahesh          Manager            35          2 sales
     25000

       103 Janvi           Analyst            30          1 Finance
     80000


       ENO ENAME           DESIGN            AGE        DNO DNAME
---------- --------------- ---------- ---------- ---------- ----------
    SALARY
----------
       104 Sumi            Developer          24          3 IT
     20000

       105 pratik          Tester             22          3 IT
     15000
```

Practical 6 : Subquery

1. **Sub query / nested query**
   Syntax:
   Select * from all tables tabs
   Where tabs.table_name IN(SELECT cols.table_name
   From tab_columns
   WHERE cols.column_name='');

Description:
A subquery is a SELECT statement embedded within another SQL statement using WHERE.

select rate from new_ott where genre='comedy';

2. Type of Subquery

a. **Single Row Subquery**

select contentid from new_ott where rate>(select rate from new_ott where genre='comedy')

# fetch the title and director of the series with duartion more than the duration of the series with eipcount more than 5

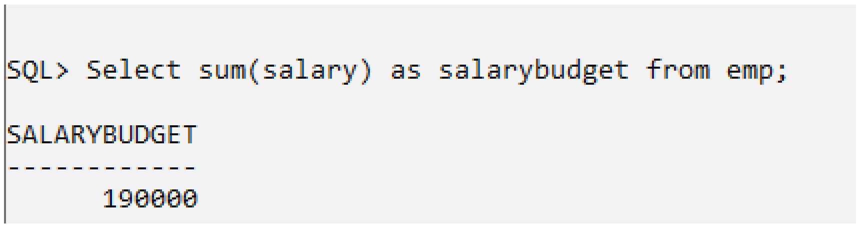
b. **Multiple Row Subquery**

select title,director from new_ott where duration>(select max(duration) from new_ott where title='abc' and epicount>6);

3. Group Function

a. **SUM():**

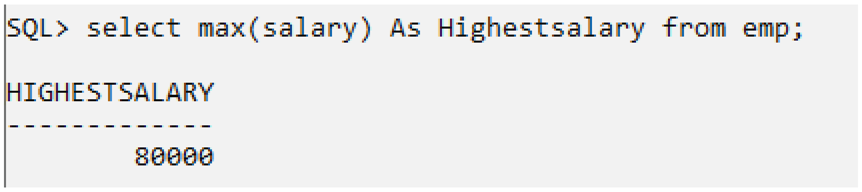
Description:
Returns the total sum
Select sum(salary) as salarybudget from emp;



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

SALARYBUDGET
------------
      190000
```

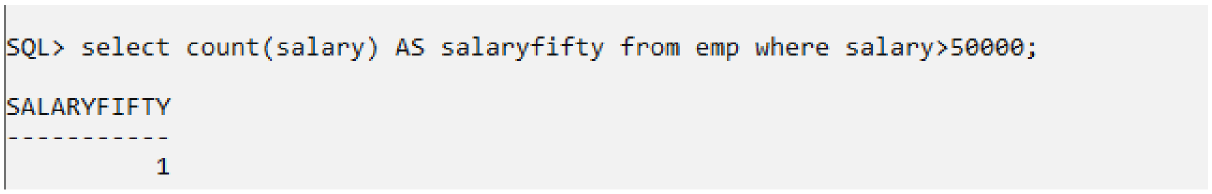
b. **Maximum()** salary
   Description:
   Returns the highest value

select max(salary) As Highestsalary from emp;



```
SQL> select max(salary) As Highestsalary from emp;

HIGHESTSALARY
-------------
        80000
```

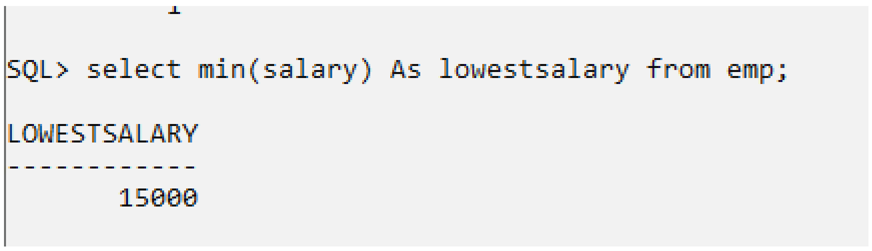
#to count the number of employess getting salary>50000
select count(salary) AS salaryfifty from emp where salary>50000;



```
SQL> select count(salary) AS salaryfifty from emp where salary>50000;

SALARYFIFTY
-----------
          1
```

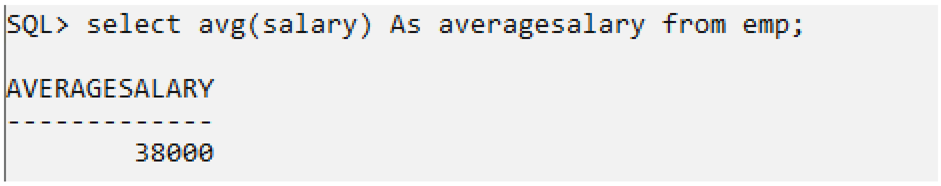
c) Minimum salary of employee
min()
Description:
Returns the lowest value
Code:
select min(salary) As lowestsalary from emp;

Output:



```
          1

SQL> select min(salary) As lowestsalary from emp;

LOWESTSALARY
------------
       15000
```

D. Average() salary of al employees
Description:
Returns the average value
Code:
select avg(salary) As averagesalary from emp;



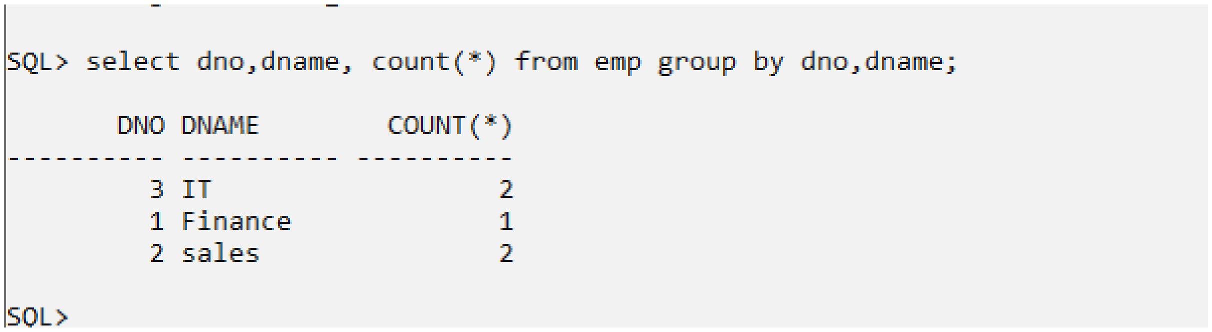
```
SQL> select avg(salary) As averagesalary from emp;

AVERAGESALARY
-------------
        38000
```

4. Having Clause
Syntax:
Select expression1,expression2,...expression n,
FROM tables WHERE conditions GROUP BY expression1, expression2,... expression n HAVING having condition;
Description:

HAVING Clause is used with GROUP BY Clause to restrict the groups of returned rows where condition is TRUE.

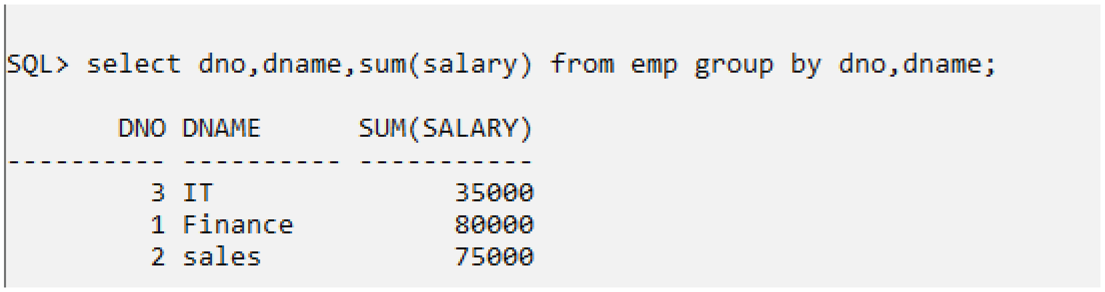
a) **group by**

#how many employees are working in departement
select dno,dname, count(*) from emp group by dno,dname;



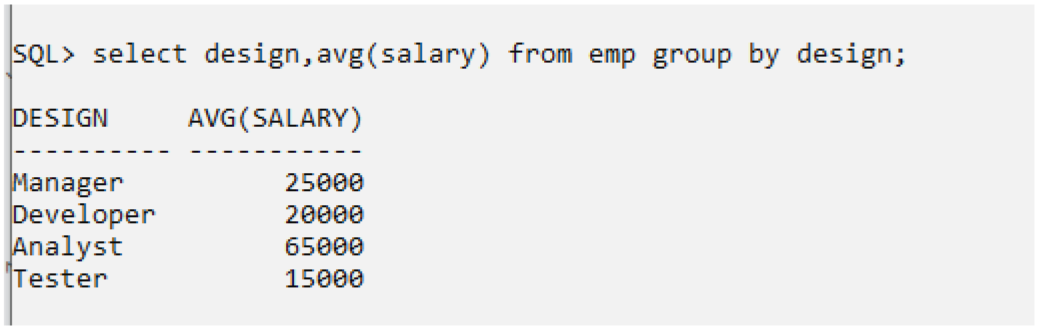
```
SQL> select dno,dname, count(*) from emp group by dno,dname;

       DNO DNAME        COUNT(*)
---------- ---------- ----------
         3 IT                  2
         1 Finance             1
         2 sales               2

SQL>
```

Having Clause
Group by how many employees has total salary for each department
select dno,dname,sum(salary) from emp group by dno,dname;



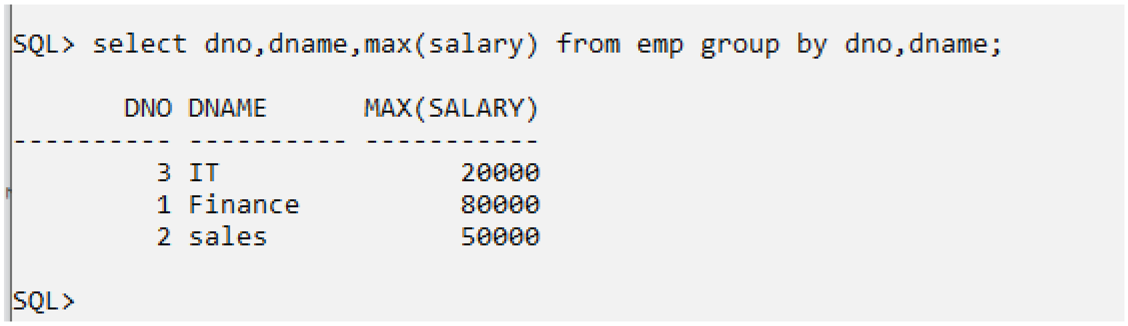
```
SQL> select dno,dname,sum(salary) from emp group by dno,dname;

       DNO DNAME      SUM(SALARY)
---------- ---------- -----------
         3 IT               35000
         1 Finance          80000
         2 sales            75000
```

Exercise removing avg using groupby
average salary for each designation
select design,avg(salary) from emp group by design;



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

DESIGN     AVG(SALARY)
---------- -----------
Manager          25000
Developer        20000
Analyst          65000
Tester           15000
```

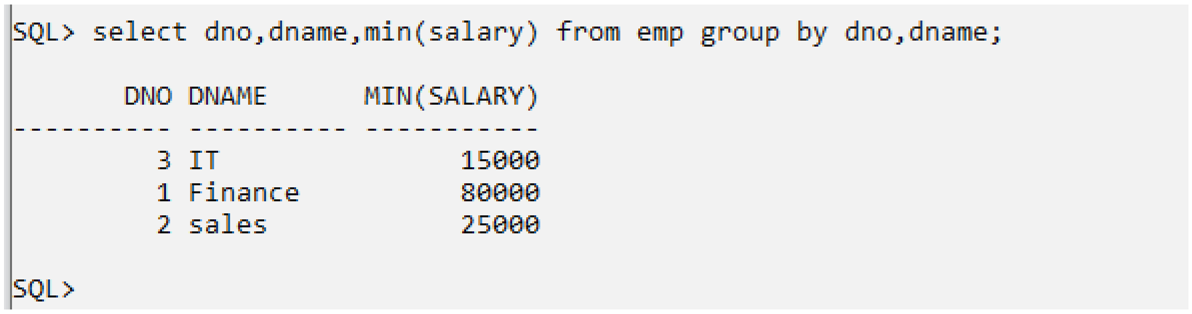
Using group by to get maximum salary of employees
select dno,dname,max(salary) from emp group by dno,dname;
#**maximum** salary for each departement



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

       DNO DNAME      MAX(SALARY)
---------- ---------- -----------
         3 IT               20000
         1 Finance          80000
         2 sales            50000

SQL>
```

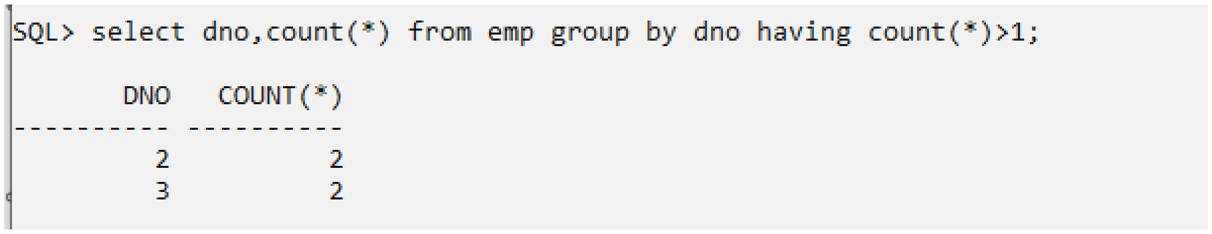
Using group by for getting minimum salary
minimum salary for each departement

select dno,dname,min(salary) from emp group by dno,dname;

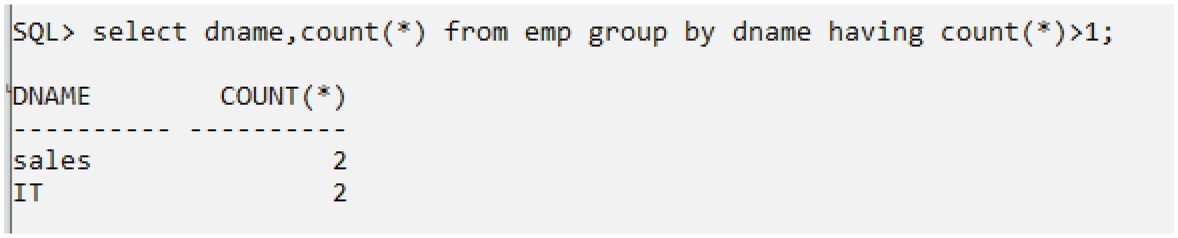


```
SQL> select dno,dname,min(salary) from emp group by dno,dname;

       DNO DNAME      MIN(SALARY)
---------- ---------- -----------
         3 IT               15000
         1 Finance          80000
         2 sales            25000

SQL>
```

#display the departements with the numbers of employees more than one.
select dno,count(*) from emp group by dno having count(*)>1;

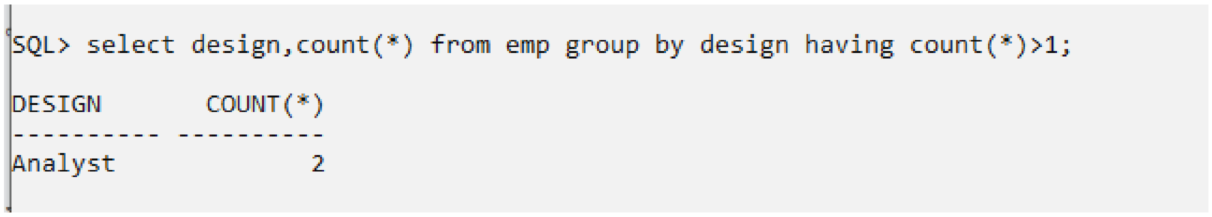


```
SQL> select dno,count(*) from emp group by dno having count(*)>1;

       DNO   COUNT(*)
---------- ----------
         2          2
         3          2
```



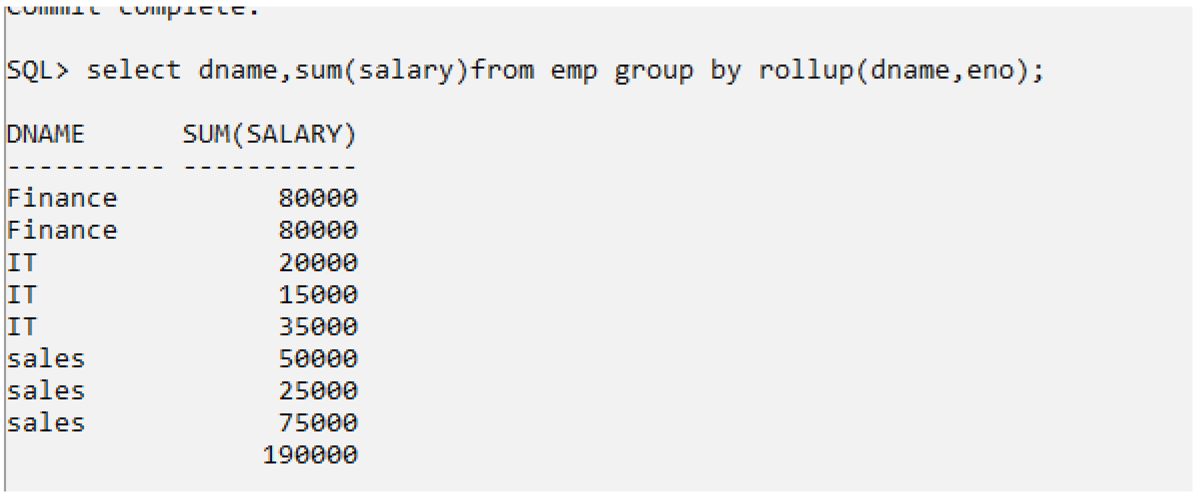
```
SQL> select dname,count(*) from emp group by dname having count(*)>1;

DNAME        COUNT(*)
---------- ----------
sales               2
IT                  2
```

#Display designation with the no of employees more than 1
select design,count(*) from emp group by design having count(*)>1;

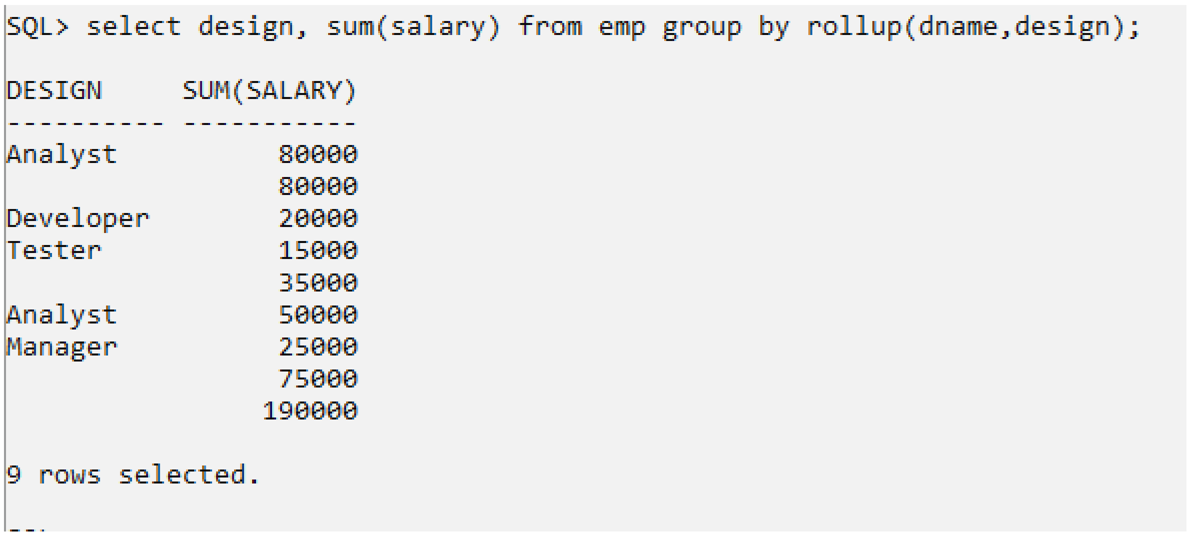


```
SQL> select design,count(*) from emp group by design having count(*)>1;

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

Rollup
select * from employees;
#Roll up
select dname,sum(salary)from emp group by rollup(dname,eno);



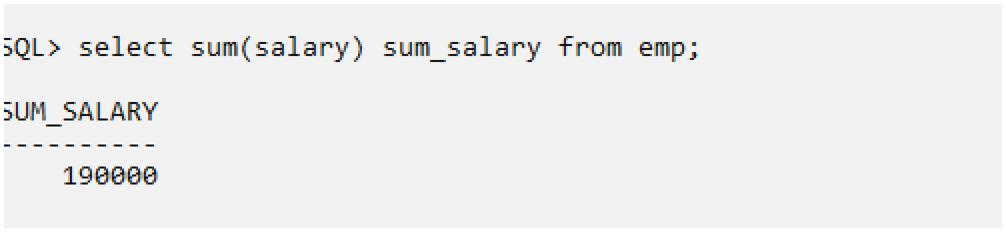
```
Commit complete.

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

DNAME      SUM(SALARY)
---------- -----------
Finance          80000
Finance          80000
IT               20000
IT               15000
IT               35000
sales            50000
sales            25000
sales            75000
                190000
```



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

DESIGN     SUM(SALARY)
---------- -----------
Analyst          80000
                 80000
Developer        20000
Tester           15000
                 35000
Analyst          50000
Manager          25000
                 75000
                190000

9 rows selected.
```

***Aggregate functions**
Description:
It performs a calculation on a set of values and returns a single value.

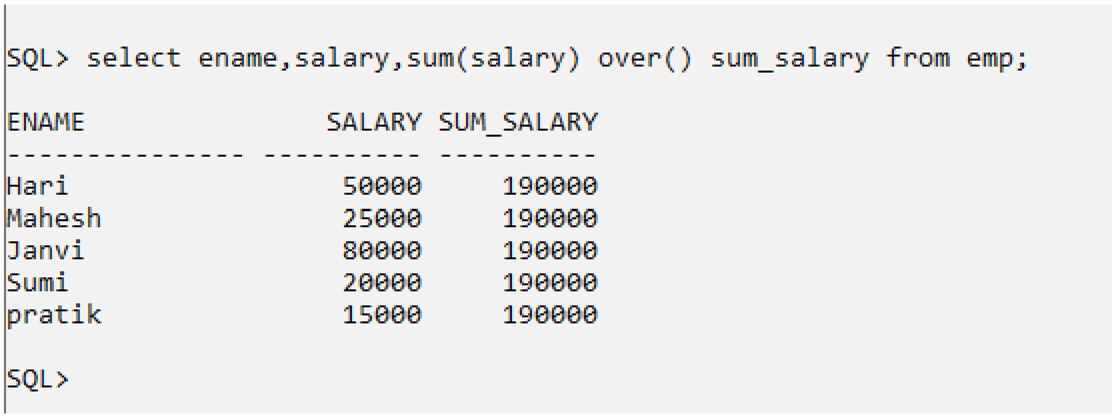
#sum of salary of all employees aggregate functions
Code:
select sum(salary) sum_salary from emp;
Output:



```
SQL> select sum(salary) sum_salary from emp;

SUM_SALARY
----------
    190000
```

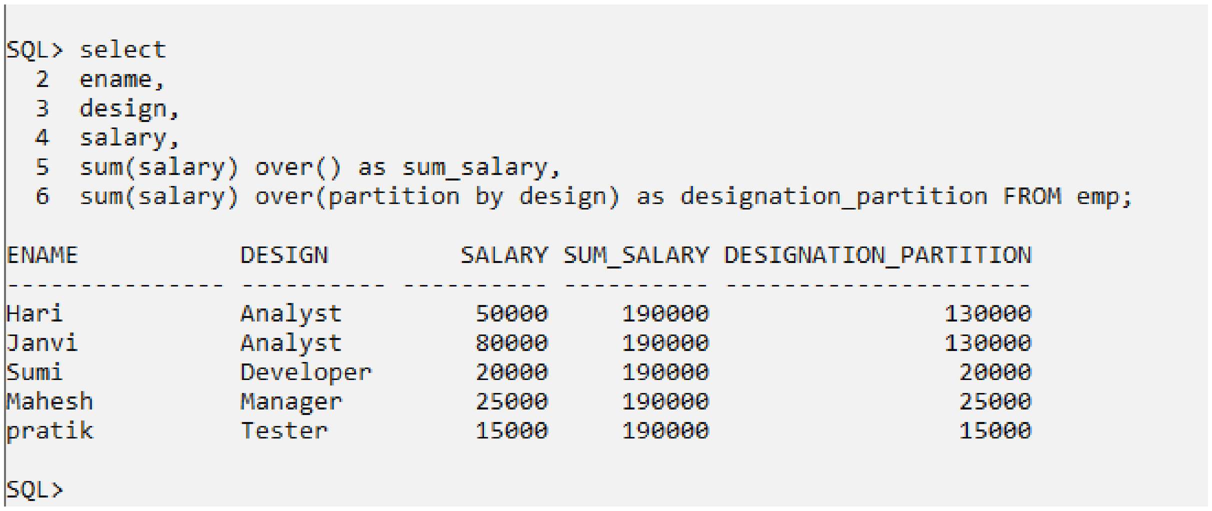
***Window Function**
Description:
It allws all data in the records right before and after the current record.

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
--------------- ---------- ----------
Hari                 50000     190000
Mahesh               25000     190000
Janvi                80000     190000
Sumi                 20000     190000
pratik               15000     190000

SQL>
```

#Window functions using partition by design
select
ename,
design,
salary,
sum(salary) over() as sum_salary,
sum(salary) over(partition by design) as designation_partition FROM emp;



```
SQL> select
  2  ename,
  3  design,
  4  salary,
  5  sum(salary) over() as sum_salary,
  6  sum(salary) over(partition by design) as designation_partition FROM emp;

ENAME           DESIGN         SALARY SUM_SALARY DESIGNATION_PARTITION
--------------- ---------- ---------- ---------- ---------------------
Hari            Analyst         50000     190000                130000
Janvi           Analyst         80000     190000                130000
Sumi            Developer       20000     190000                 20000
Mahesh          Manager         25000     190000                 25000
pratik          Tester          15000     190000                 15000

SQL>
```

*Practical no . 7 Constraints

***table level**
CONSTRAINT student_facid_fk

*column level

*Table level example
```
CREATE TABLE Orders(
        OrderID int NOT NULL,
        OrderNumber int NOT NULL,
        personid int,
        PRIMARY KEY (OrderID),
        Foreign key(PersonID) References Persons
                (personId));
```

*Column Level
```
Create table Orders(
        OrderID int NOT NULL PRIMARY KEY,
        OrderNumber int NOT NULL,
        PersonID
int FOREIGN KEY REFERENCES Persons(PersonID));
```

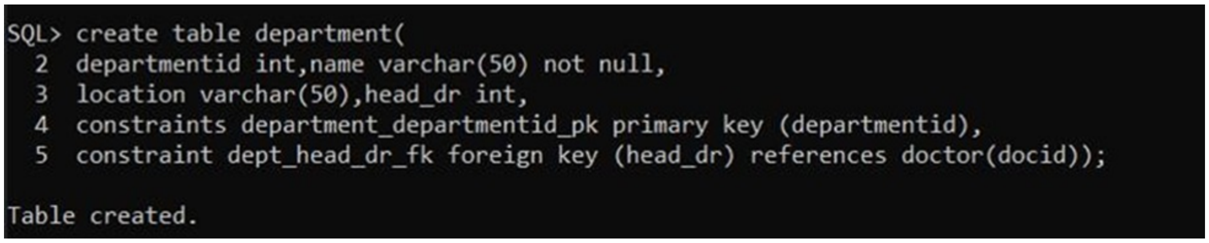
a) **Not Null**
   Syntax:
   Create table table_name(column_name not null);

Description:
The NOT NULL constraint ensures that the column has a value and the value is not null value.

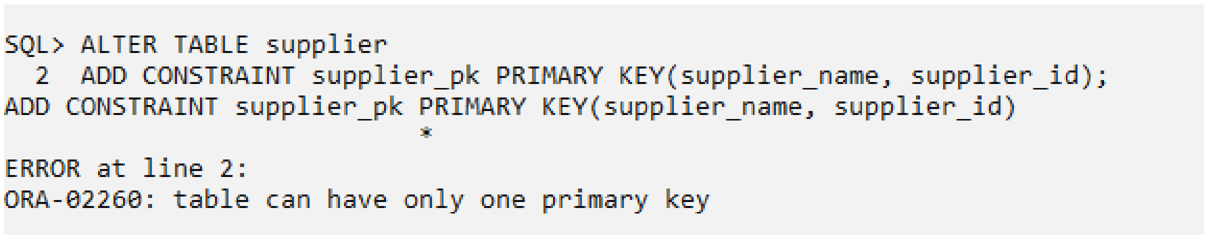
Command:

```
create table department(departmentid int, name varchar(50) not null,location
varchar(50), head_dr int, constraints department_departmentid_pk primary key
(departmentid), constraint dept_head_dr_fk foreign key (head_dr) references
doctor(docid));
```

Output:



```
SQL> create table department(
  2  departmentid int,name varchar(50) not null,
  3  location varchar(50),head_dr int,
  4  constraints department_departmentid_pk primary key (departmentid),
  5  constraint dept_head_dr_fk foreign key (head_dr) references doctor(docid));

Table created.
```



```
SQL> ALTER TABLE supplier
  2  ADD CONSTRAINT supplier_pk PRIMARY KEY(supplier_name, supplier_id);
ADD CONSTRAINT supplier_pk PRIMARY KEY(supplier_name, supplier_id)
                                                     *
ERROR at line 2:
ORA-02260: table can have only one primary key
```

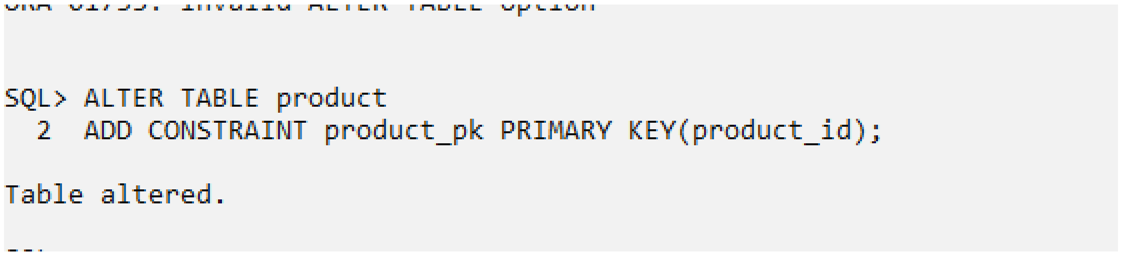
B. MAKING PRIMARY KEY

Syntax:

Create table product(

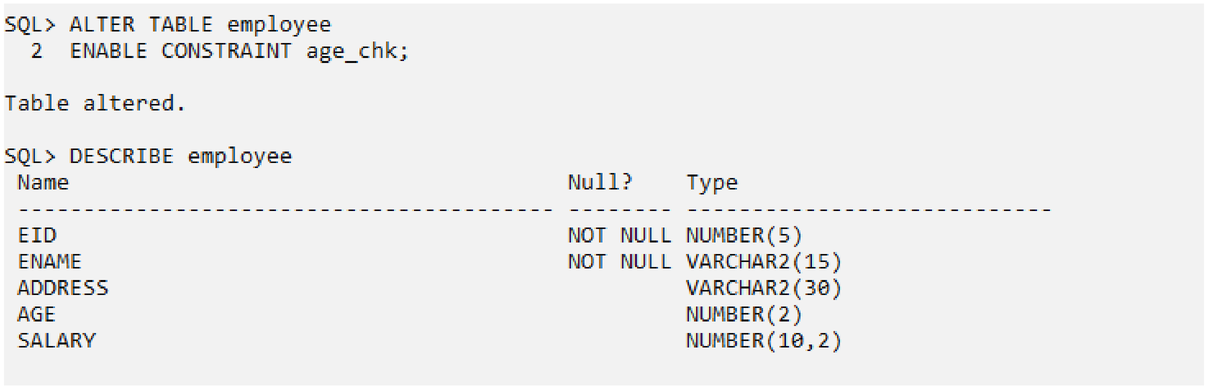
Product_id number primary key,

);



```
SQL> ALTER TABLE product
  2  ADD CONSTRAINT product_pk PRIMARY KEY(product_id);

Table altered.
```

ENABLE CONSTRAINT



```
SQL> ALTER TABLE employee
  2  ENABLE CONSTRAINT age_chk;

Table altered.

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

C. UNIQUE CONSTRAINT
Description:

Define both table level and column level
Ensures taht all values in a column are diffrent

CONSTRAINT dept_deptname_uk UNIQUE(DeptName),

At the column level, the constraint is defined by:
DeptName VARCHAR2(12) CONSTRAINT dept_deptname_uk UNIQUE,

D. CHECK CONSTRAINT
CONSTRAINT dept_deptname_uk UNIQUE(DeptName),

deptname VARCHAR2(12) CONSTRAINT dept_deptname_uk CHECK,

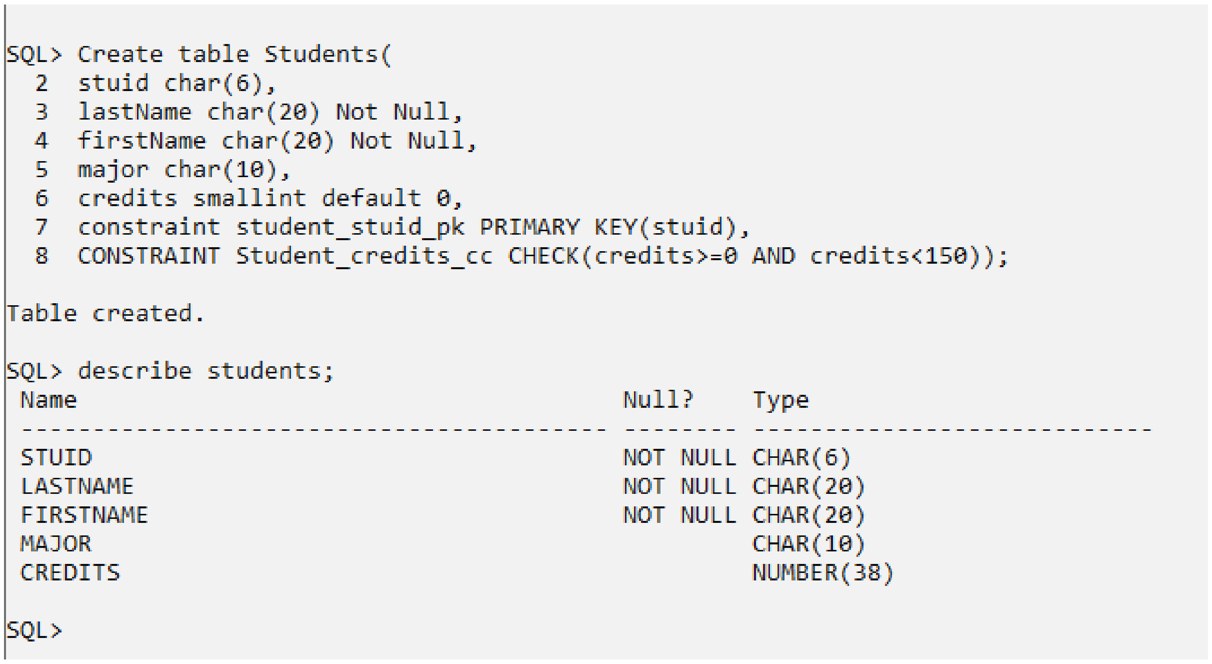
*Check at column level
deptid NUMBER(2) CONSTRAINT dept_deptid_cc
CHECK(DEPTID>=10) and (DeptId<=99))

CONSTRAINT dept_deptid_cc
CHECK((DeptId>=10) and (Deptid<=99)),

#Create table
Create table Students(
stuid char(6),
lastName char(20) Not Null,
firstName char(20) Not Null,
major char(10),
credits smallint default 0,

constraint student_stuid_pk PRIMARY KEY(stuid),
CONSTRAINT Student_credits_cc CHECK(credits>=0 AND credits<150));

Output:

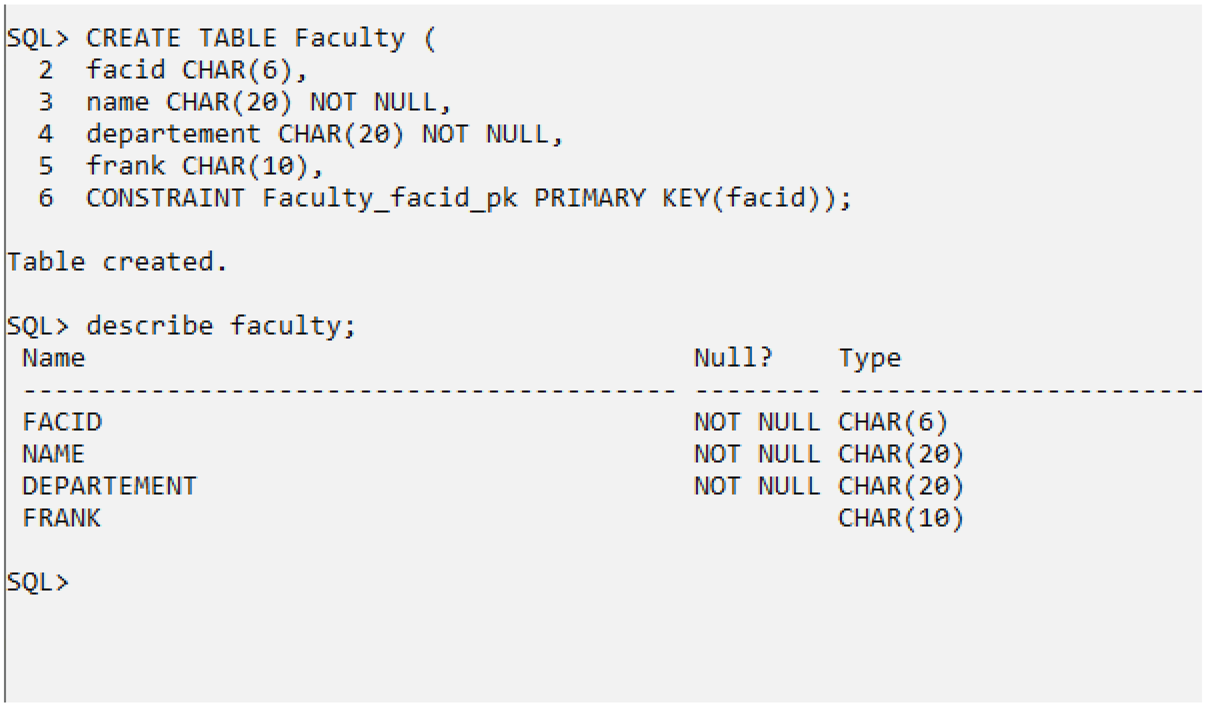


```
SQL> Create table Students(
  2  stuid char(6),
  3  lastName char(20) Not Null,
  4  firstName char(20) Not Null,
  5  major char(10),
  6  credits smallint default 0,
  7  constraint student_stuid_pk PRIMARY KEY(stuid),
  8  CONSTRAINT Student_credits_cc CHECK(credits>=0 AND credits<150));

Table created.

SQL> describe students;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUID                                     NOT NULL CHAR(6)
 LASTNAME                                  NOT NULL CHAR(20)
 FIRSTNAME                                 NOT NULL CHAR(20)
 MAJOR                                              CHAR(10)
 CREDITS                                            NUMBER(38)

SQL>
```

*Create a table of faculty.

CREATE TABLE Faculty (
facid CHAR(6),
name CHAR(20) NOT NULL,
departement CHAR(20) NOT NULL,
frank CHAR(10),
CONSTRAINT Faculty_facid_pk PRIMARY KEY(facid));

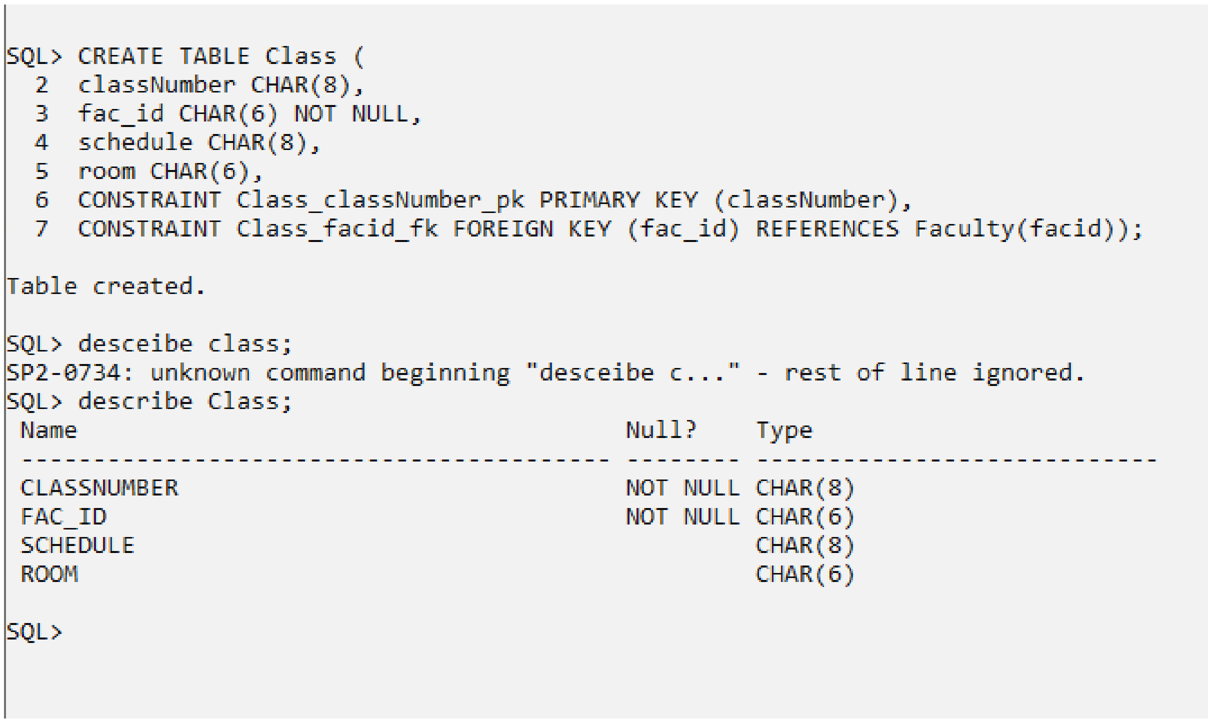
Output:



```
SQL> CREATE TABLE Faculty (
  2  facid CHAR(6),
  3  name CHAR(20) NOT NULL,
  4  departement CHAR(20) NOT NULL,
  5  frank CHAR(10),
  6  CONSTRAINT Faculty_facid_pk PRIMARY KEY(facid));

Table created.

SQL> describe faculty;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 FACID                                     NOT NULL CHAR(6)
 NAME                                      NOT NULL CHAR(20)
 DEPARTEMENT                               NOT NULL CHAR(20)
 FRANK                                              CHAR(10)

SQL>
```

*CREATE TABLE Class

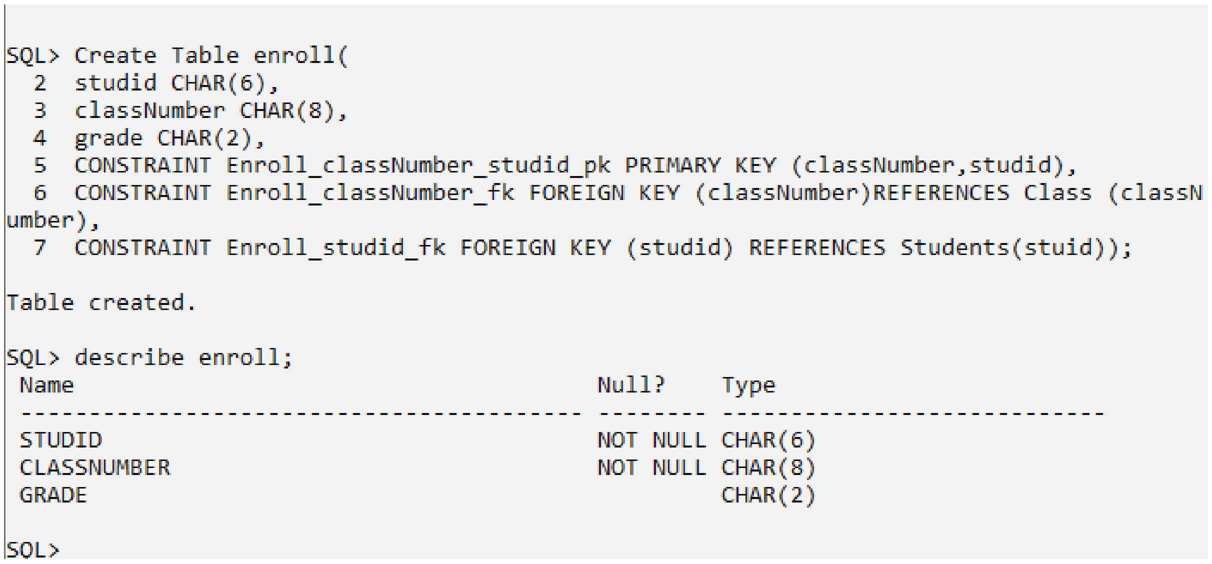
```
CREATE TABLE Class (
classNumber CHAR(8),
fac_id CHAR(6) NOT NULL,
schedule CHAR(8),
room CHAR(6),
CONSTRAINT Class_classNumber_pk PRIMARY KEY (classNumber),
CONSTRAINT Class_facid_fk FOREIGN KEY (fac_id) REFERENCES Faculty(facid));
```



```
SQL> CREATE TABLE Class (
  2  classNumber CHAR(8),
  3  fac_id CHAR(6) NOT NULL,
  4  schedule CHAR(8),
  5  room CHAR(6),
  6  CONSTRAINT Class_classNumber_pk PRIMARY KEY (classNumber),
  7  CONSTRAINT Class_facid_fk FOREIGN KEY (fac_id) REFERENCES Faculty(facid));

Table created.

SQL> desceibe class;
SP2-0734: unknown command beginning "desceibe c..." - rest of line ignored.
SQL> describe Class;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CLASSNUMBER                               NOT NULL CHAR(8)
 FAC_ID                                    NOT NULL CHAR(6)
 SCHEDULE                                           CHAR(8)
 ROOM                                               CHAR(6)

SQL>
```

e.Enroll

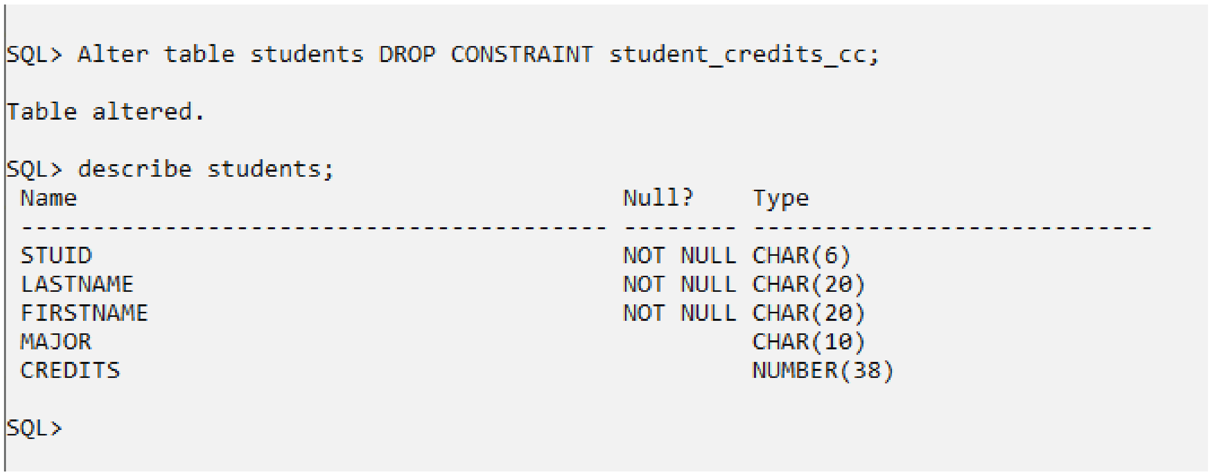
Create Table enroll(
studid CHAR(6),
classNumber CHAR(8),
grade CHAR(2),
CONSTRAINT Enroll_classNumber_studid_pk PRIMARY KEY (classNumber,studid),
CONSTRAINT Enroll_classNumber_fk FOREIGN KEY (classNumber)REFERENCES Class (classNumber),
CONSTRAINT Enroll_studid_fk FOREIGN KEY (studid) REFERENCES Students(stuid));



```
SQL> Create Table enroll(
  2  studid CHAR(6),
  3  classNumber CHAR(8),
  4  grade CHAR(2),
  5  CONSTRAINT Enroll_classNumber_studid_pk PRIMARY KEY (classNumber,studid),
  6  CONSTRAINT Enroll_classNumber_fk FOREIGN KEY (classNumber)REFERENCES Class (classN
umber),
  7  CONSTRAINT Enroll_studid_fk FOREIGN KEY (studid) REFERENCES Students(stuid));

Table created.

SQL> describe enroll;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDID                                    NOT NULL CHAR(6)
 CLASSNUMBER                               NOT NULL CHAR(8)
 GRADE                                              CHAR(2)

SQL>
```

*Alter table command

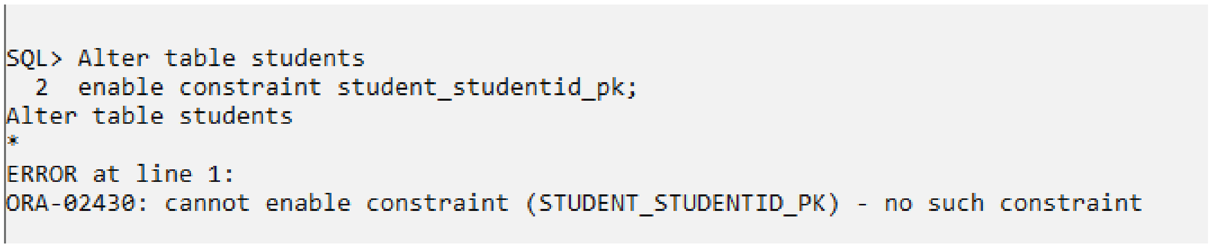
f. Dropping constraint
Description: To drop constraint which has assigned a key
Code:
Alter table students DROP CONSTRAINT student_credits_cc;
Output:



```
SQL> Alter table students DROP CONSTRAINT student_credits_cc;

Table altered.

SQL> describe students;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUID                                     NOT NULL CHAR(6)
 LASTNAME                                  NOT NULL CHAR(20)
 FIRSTNAME                                 NOT NULL CHAR(20)
 MAJOR                                              CHAR(10)
 CREDITS                                            NUMBER(38)

SQL>
```

g.Enable Constraint
Description:
To enable the constraint and make primary key constraint.

Alter table students
enable constraint student_studentid_pk;



```
SQL> Alter table students
  2  enable constraint student_studentid_pk;
Alter table students
*
ERROR at line 1:
ORA-02430: cannot enable constraint (STUDENT_STUDENTID_PK) - no such constraint
```

Practical No-8 SQL JOINS

Join:

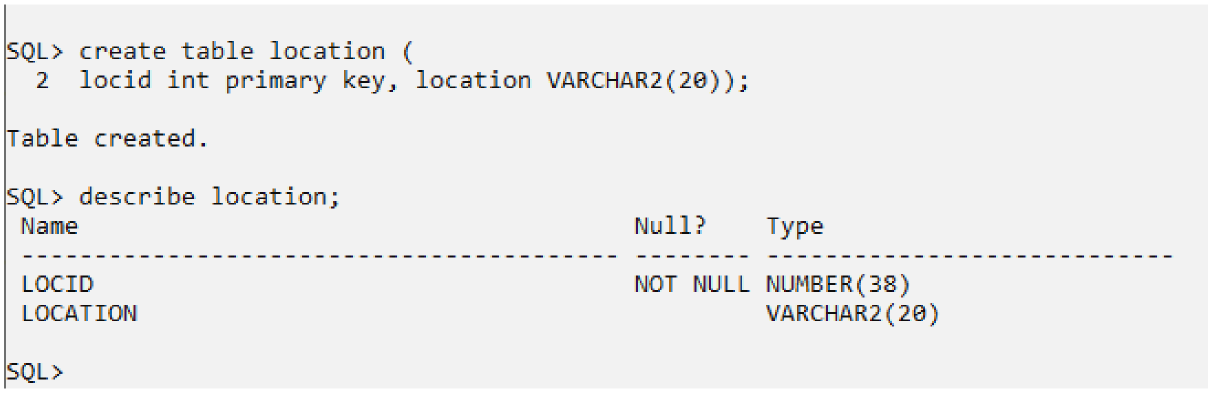
Syntax for joining
SELECT table.column, table2.column
FROM table1, table2
WHERE table1.column1 = table2.column2;

Table structure
Fetch pateintid, patient name, drspecilization
patient id, name-PATIENT
Dspec- DOCTOR

Patient id, name-PATIENT
DSPEC-DOCTOR

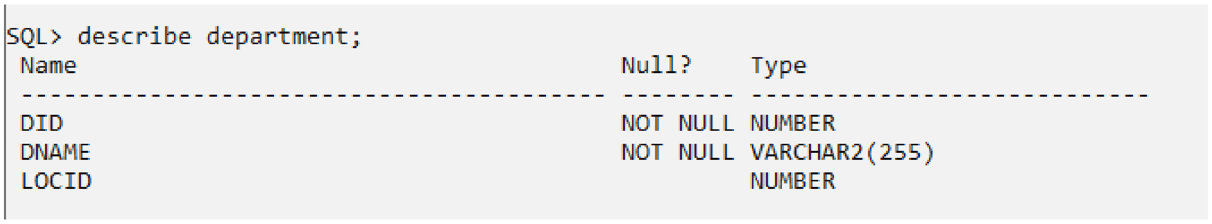
Select patient.pid,patient.name,doctor.dspec
FROM patient,doctor
WHERE Patient.dID=doctor.docID
Alyas name
select p.pid,p.name,d.dspec
FROM patient p,doctor d
WHERE p.DID= d.docID

#Create a table location and department
create table location (
locid int primary key, location VARCHAR2(20));



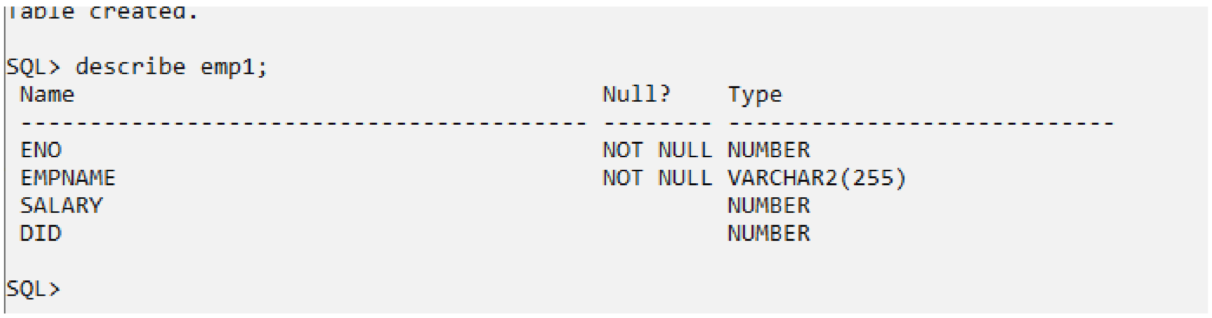
```
SQL> create table location (
  2  locid int primary key, location VARCHAR2(20));

Table created.

SQL> describe location;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 LOCID                                     NOT NULL NUMBER(38)
 LOCATION                                           VARCHAR2(20)

SQL>
```

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



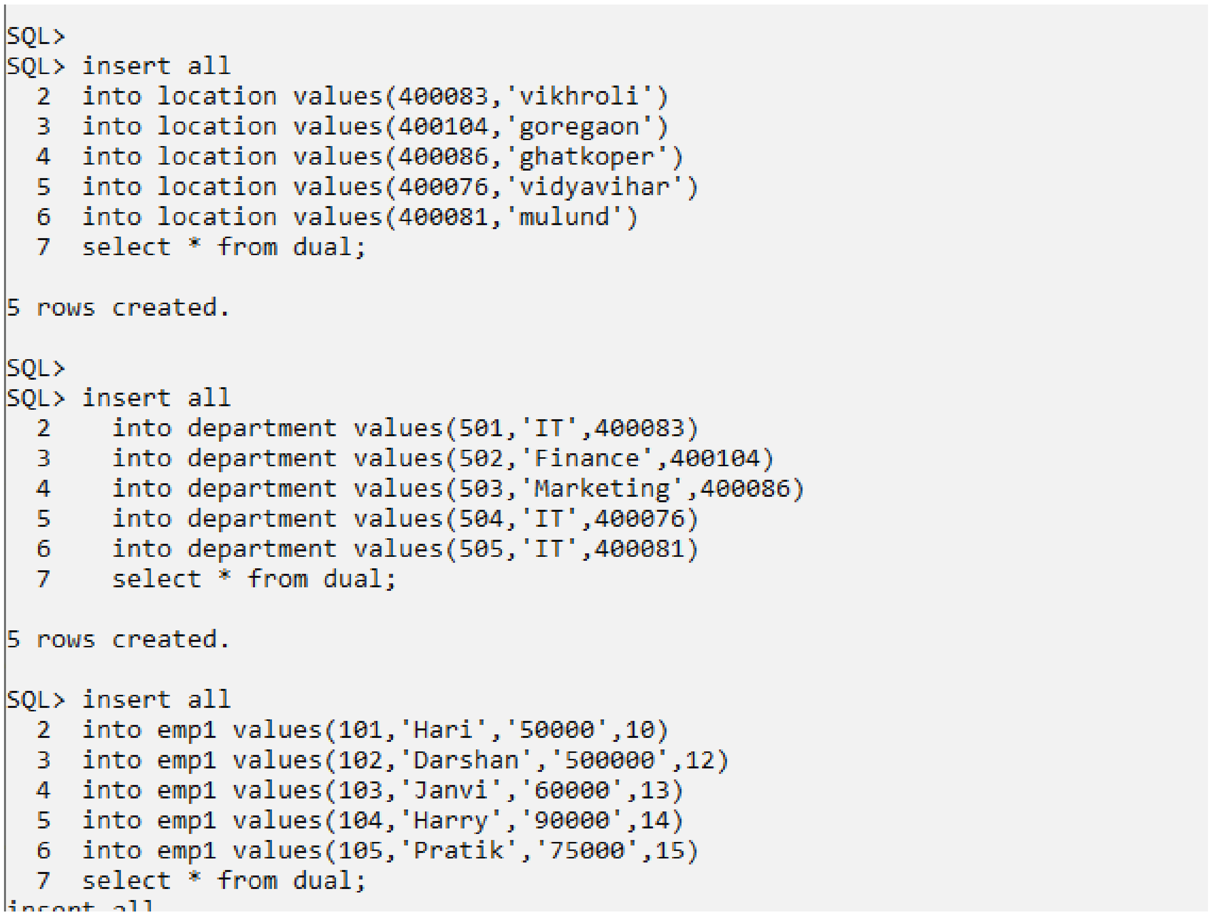
```
SQL> describe department;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DID                                       NOT NULL NUMBER
 DNAME                                     NOT NULL VARCHAR2(255)
 LOCID                                              NUMBER
```

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

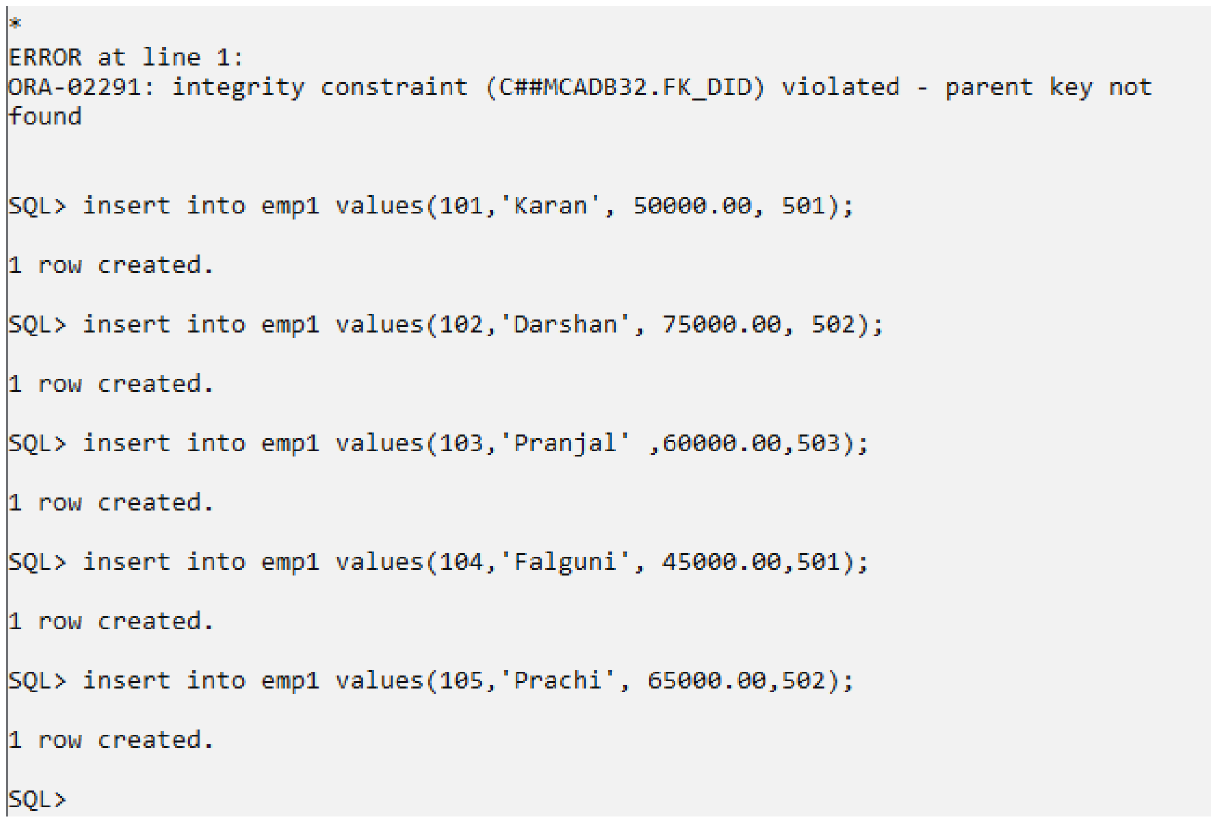


```
Table created.

SQL> describe emp1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ENO                                       NOT NULL NUMBER
 EMPNAME                                   NOT NULL VARCHAR2(255)
 SALARY                                             NUMBER
 DID                                                NUMBER

SQL>
```

*Inserting 5 values in the table



```
SQL>
SQL> 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;

5 rows created.

SQL>
SQL> 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;

5 rows created.

SQL> insert all
into emp1 values(101,'Hari','50000',10)
into emp1 values(102,'Darshan','500000',12)
into emp1 values(103,'Janvi','60000',13)
into emp1 values(104,'Harry','90000',14)
into emp1 values(105,'Pratik','75000',15)
select * from dual;
insert all
```



```
*
ERROR at line 1:
ORA-02291: integrity constraint (C##MCADB32.FK_DID) violated - parent key not
found


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.

SQL>
```

\

1. **Equi Joins**

Retrieving records with equijoins
employee number employee name from department number , department name.
Syntax:

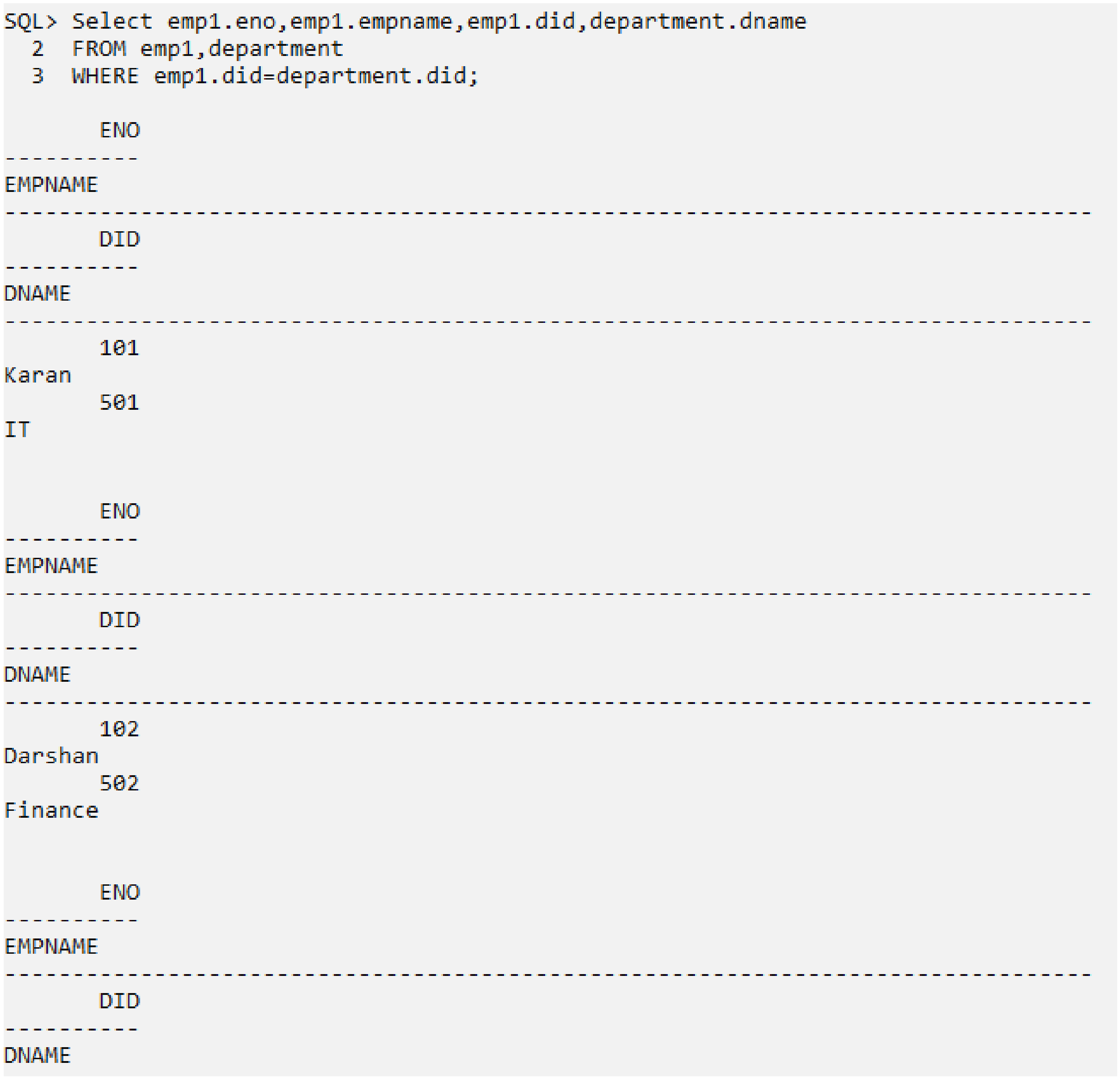
select Select table1.column, table2.column from table1, table 2 where table1.column1 =table2.column2;

Description:

An equi join is a type of join that combines tables based on matching values in specified columns.

Code:

```
Select emp1.eno,emp1.empname,emp1.did,department.dname
FROM emp1,department
WHERE emp1.did=department.did;
```

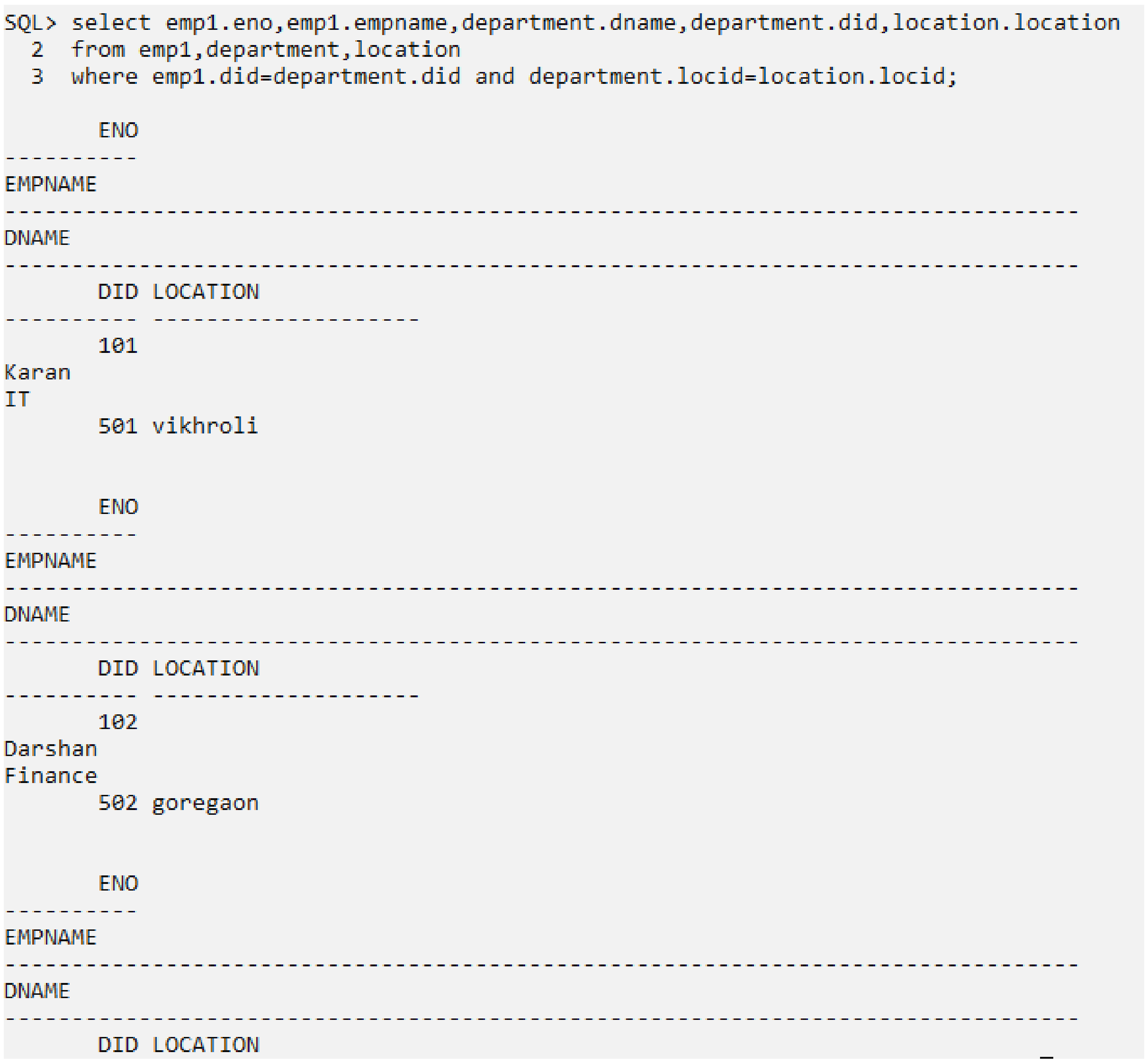

```
SQL> Select emp1.eno,emp1.empname,emp1.did,department.dname
  2  FROM emp1,department
  3  WHERE emp1.did=department.did;

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


       ENO
----------
EMPNAME
--------------------------------------------------------------------------------
       DID
----------
DNAME
--------------------------------------------------------------------------------
       102
Darshan
       502
Finance


       ENO
----------
EMPNAME
--------------------------------------------------------------------------------
       DID
----------
DNAME
```

***Using aliases name**
Description: Alias name is used to give a table or column an temporary name.
*Use aliases name for table
select e.eno,e.empname,e.did,d.dname
from emp1 e, department d
where e.did=d.did;

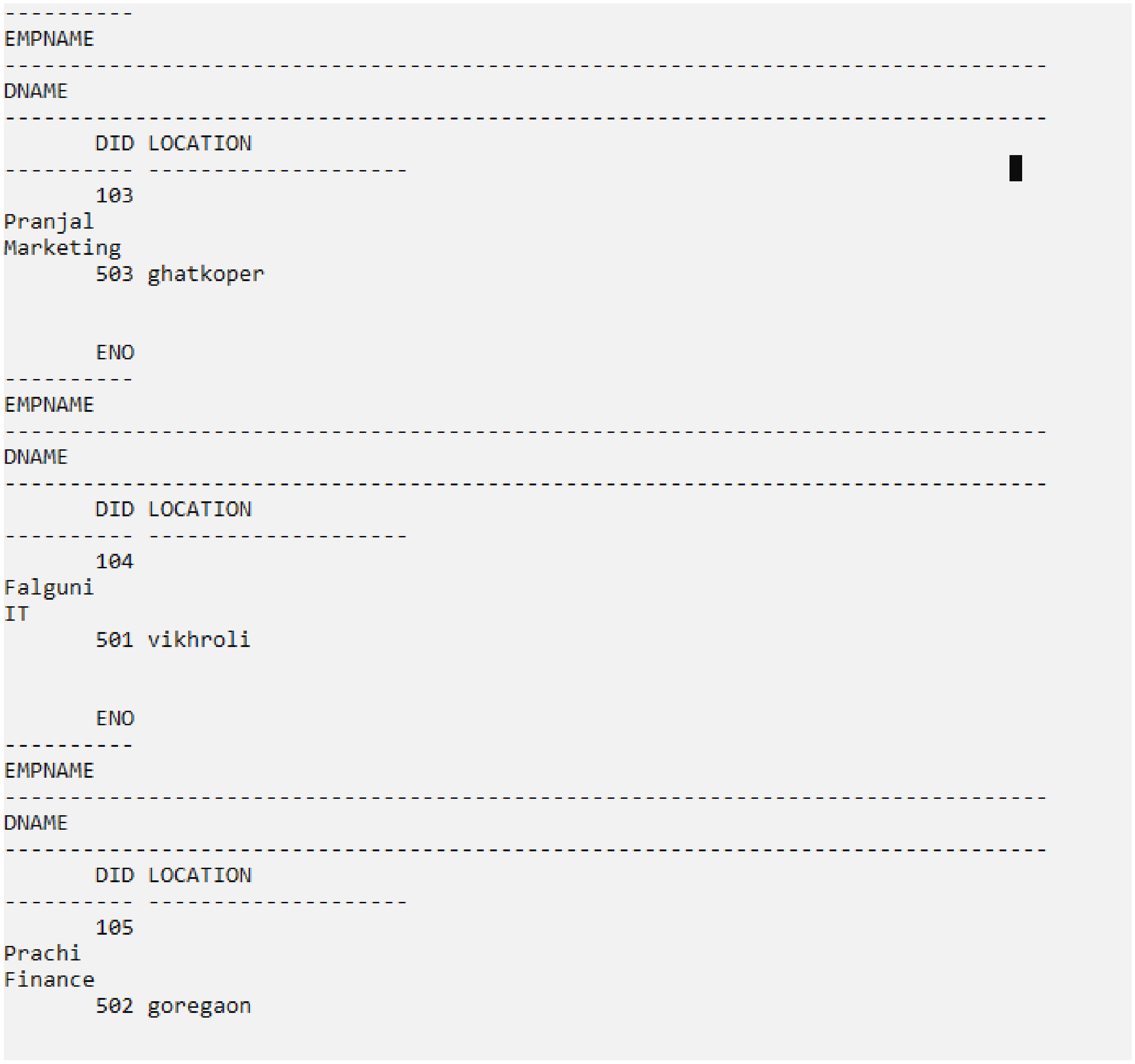
```
EMPNAME
----------------------------------------
       DID
----------
DNAME
----------------------------------------
       101
Karan
       501
IT


       ENO
----------
EMPNAME
----------------------------------------
       DID
----------
DNAME
----------------------------------------
       102
Darshan
       502
Finance


       ENO
----------
EMPNAME
----------------------------------------
       DID
----------
DNAME
----------------------------------------
       103
Pranjal
       503
Marketing


       ENO
----------
EMPNAME
----------------------------------------
       DID
----------
DNAME
----------------------------------------
       104
Falguni
       501
```

***Joining more than two tables**
Description:
Joining more than 2 tables help to fetch the values of both the tables in one entity.

employee no, department name, department no, location id,location name
select emp1.eno,emp1.empname,department.dname,department.did,location.location
from emp1,department,location
where emp1.did=department.did and department.locid=location.locid;



```
SQL> select emp1.eno,emp1.empname,department.dname,department.did,location.location
  2  from emp1,department,location
  3  where emp1.did=department.did and department.locid=location.locid;

       ENO
----------
EMPNAME
--------------------------------------------------------------------------------
DNAME
--------------------------------------------------------------------------------
       DID LOCATION
---------- --------------------
       101
Karan
IT
       501 vikhroli


       ENO
----------
EMPNAME
--------------------------------------------------------------------------------
DNAME
--------------------------------------------------------------------------------
       DID LOCATION
---------- --------------------
       102
Darshan
Finance
       502 goregaon


       ENO
----------
EMPNAME
--------------------------------------------------------------------------------
DNAME
--------------------------------------------------------------------------------
       DID LOCATION
```



```
----------
EMPNAME
--------------------------------------------------------------------------------
DNAME
--------------------------------------------------------------------------------
       DID LOCATION
---------- --------------------
       103
Pranjal
Marketing
       503 ghatkoper


       ENO
----------
EMPNAME
--------------------------------------------------------------------------------
DNAME
--------------------------------------------------------------------------------
       DID LOCATION
---------- --------------------
       104
Falguni
IT
       501 vikhroli


       ENO
----------
EMPNAME
--------------------------------------------------------------------------------
DNAME
--------------------------------------------------------------------------------
       DID LOCATION
---------- --------------------
       105
Prachi
Finance
       502 goregaon
```

b) Non Equijoins

Description:

Nn-equi joins are joins whose join condition use conditional operators other than equals.

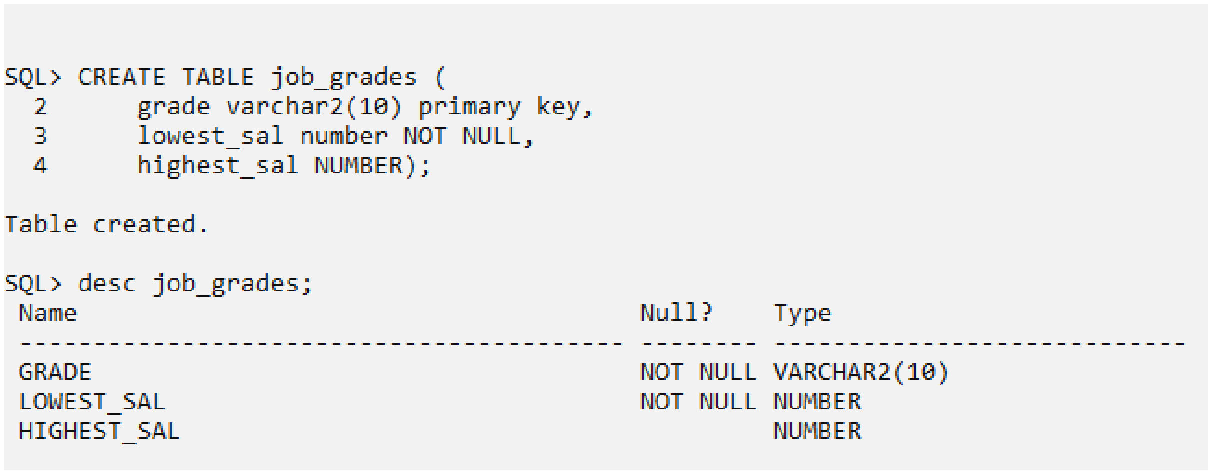
Syntax:

select table1.column, table2.column from table1, table 2 where table1.column1 condition table2.column2

Code:

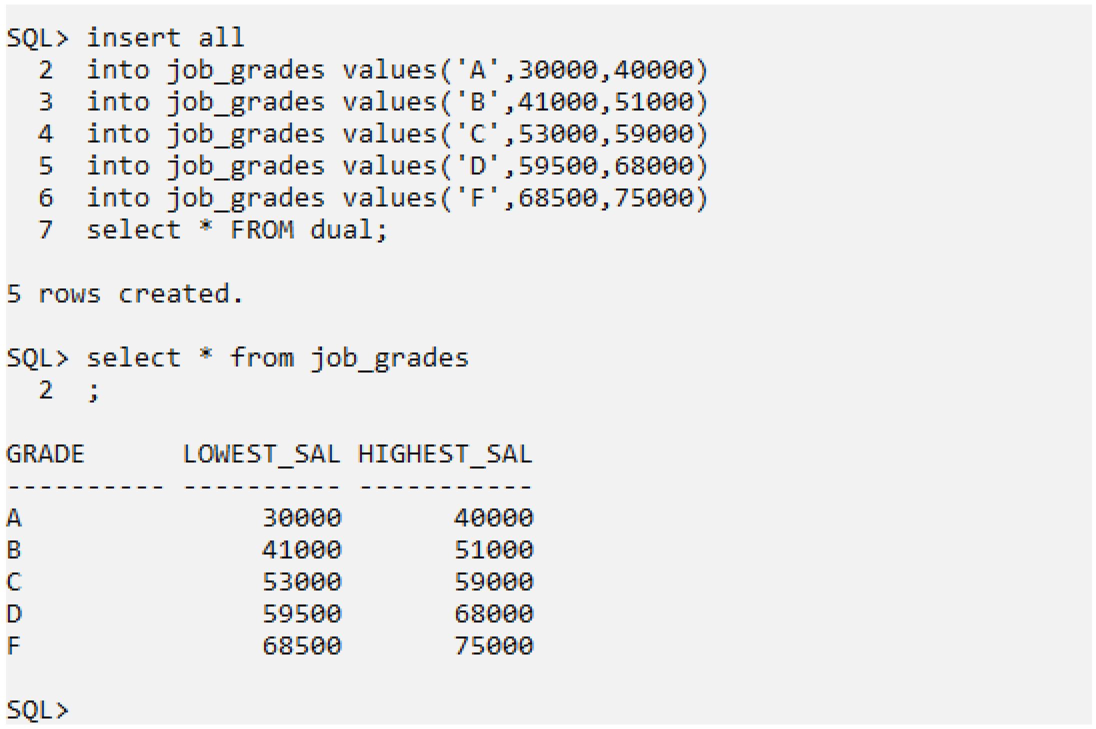
salary in the employees table must be between lowest salary and highest salary
create table job grades(

```
CREATE TABLE job_grades (
    grade varchar2(10) primary key,
    lowest_sal number NOT NULL,
    highest_sal NUMBER);
```



```
SQL> CREATE TABLE job_grades (
  2      grade varchar2(10) primary key,
  3      lowest_sal number NOT NULL,
  4      highest_sal NUMBER);

Table created.

SQL> desc job_grades;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 GRADE                                     NOT NULL VARCHAR2(10)
 LOWEST_SAL                                NOT NULL NUMBER
 HIGHEST_SAL                                        NUMBER
```

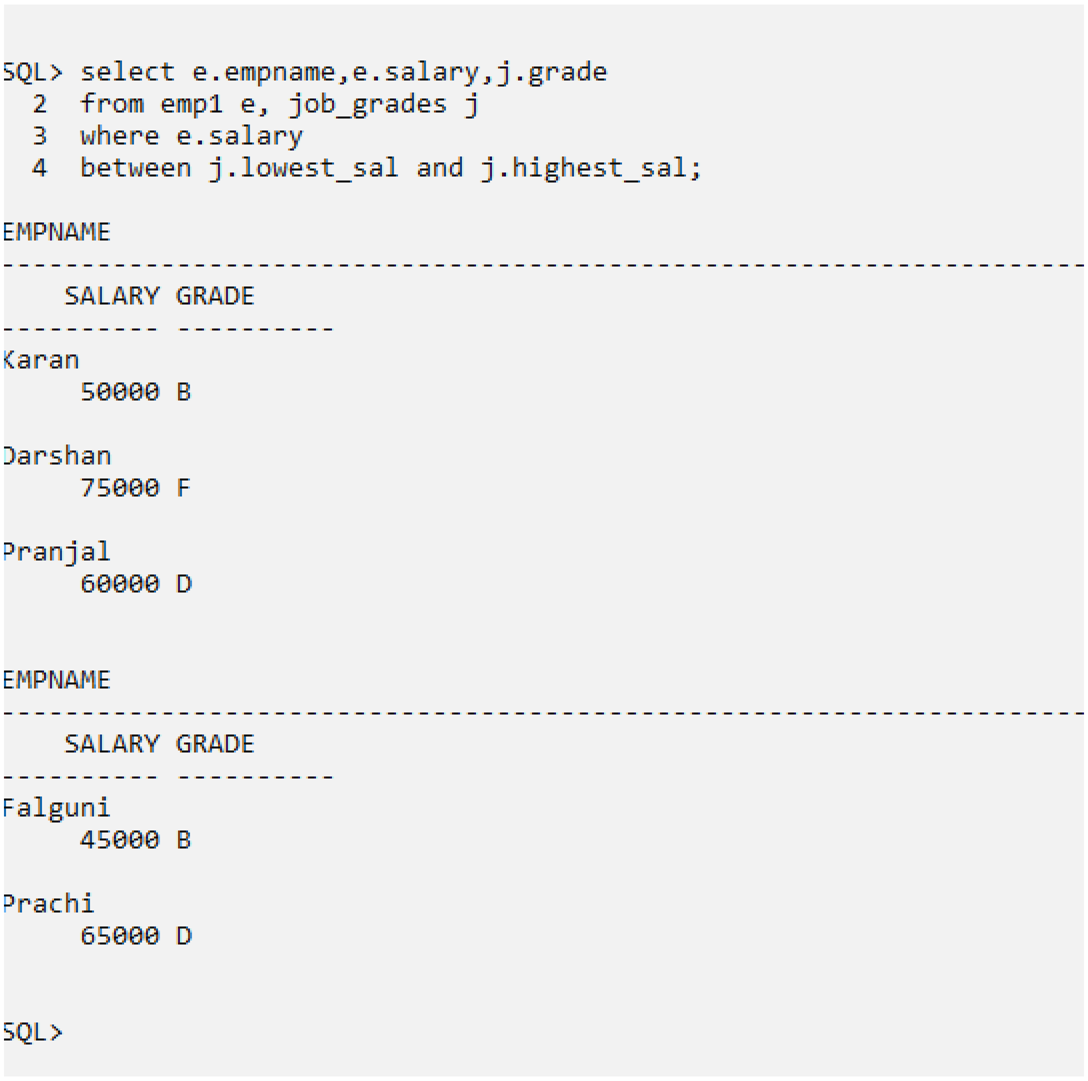
```
insert all
into job_grades values('A',30000,40000)
into job_grades values('B',41000,51000)
into job_grades values('C',53000,59000)
into job_grades values('D',59500,68000)
into job_grades values('F',68500,75000)
select * FROM dual;
```



```
SQL> insert all
  2  into job_grades values('A',30000,40000)
  3  into job_grades values('B',41000,51000)
  4  into job_grades values('C',53000,59000)
  5  into job_grades values('D',59500,68000)
  6  into job_grades values('F',68500,75000)
  7  select * FROM dual;

5 rows created.

SQL> select * from job_grades
  2  ;

GRADE      LOWEST_SAL HIGHEST_SAL
---------- ---------- -----------
A               30000       40000
B               41000       51000
C               53000       59000
D               59500       68000
F               68500       75000

SQL>
```

NON-EQUIJOINS
using comparator operator
Putting different types of salary of employees into grades and comparing it.
Code:
select e.empname,e.salary,j.grade
from emp1 e, job_grades j
where e.salary
between j.lowest_sal and j.highest_sal;

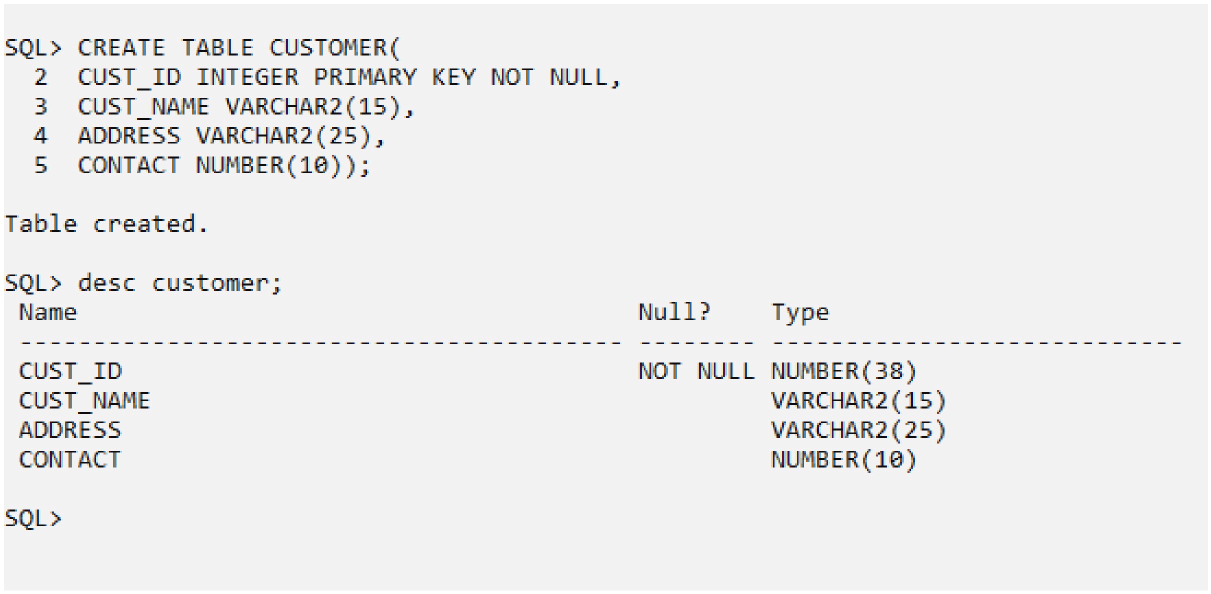
Output:



```
SQL> select e.empname,e.salary,j.grade
  2  from emp1 e, job_grades j
  3  where e.salary
  4  between j.lowest_sal and j.highest_sal;

EMPNAME
--------------------------------------------------------------------
    SALARY GRADE
---------- ----------
Karan
     50000 B

Darshan
     75000 F

Pranjal
     60000 D


EMPNAME
--------------------------------------------------------------------
    SALARY GRADE
---------- ----------
Falguni
     45000 B

Prachi
     65000 D


SQL>
```

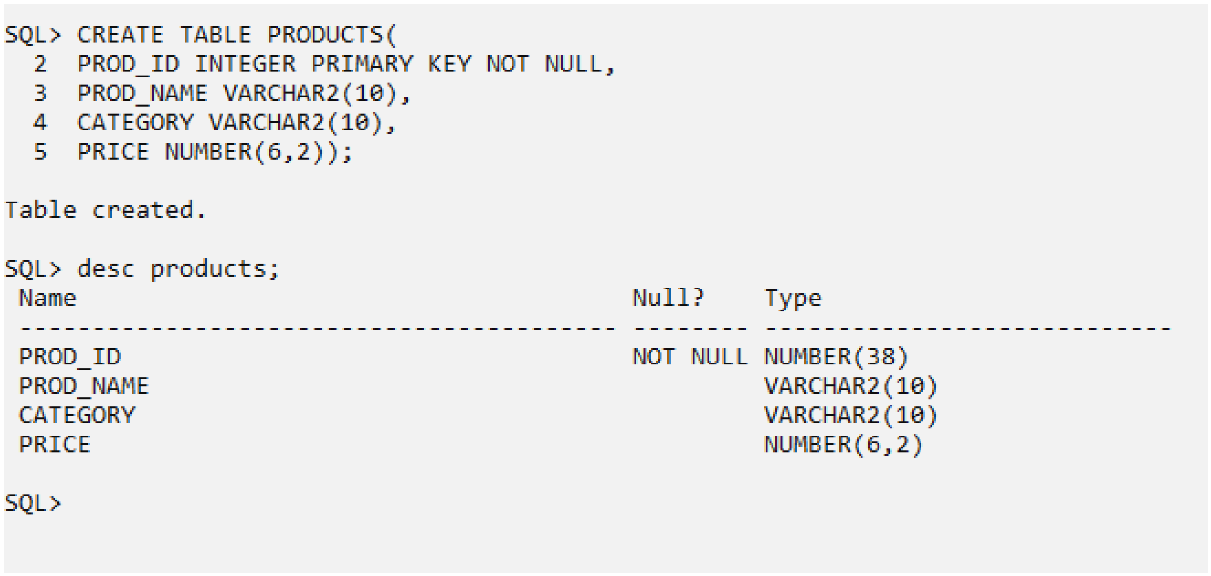
CUSTOMER TABLE

CREATE TABLE CUSTOMER(
CUST_ID INTEGER PRIMARY KEY NOT NULL,
CUST_NAME VARCHAR2(15),
ADDRESS VARCHAR2(25),
CONTACT NUMBER(10));



```
SQL> CREATE TABLE CUSTOMER(
  2  CUST_ID INTEGER PRIMARY KEY NOT NULL,
  3  CUST_NAME VARCHAR2(15),
  4  ADDRESS VARCHAR2(25),
  5  CONTACT NUMBER(10));

Table created.

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

SQL>
```

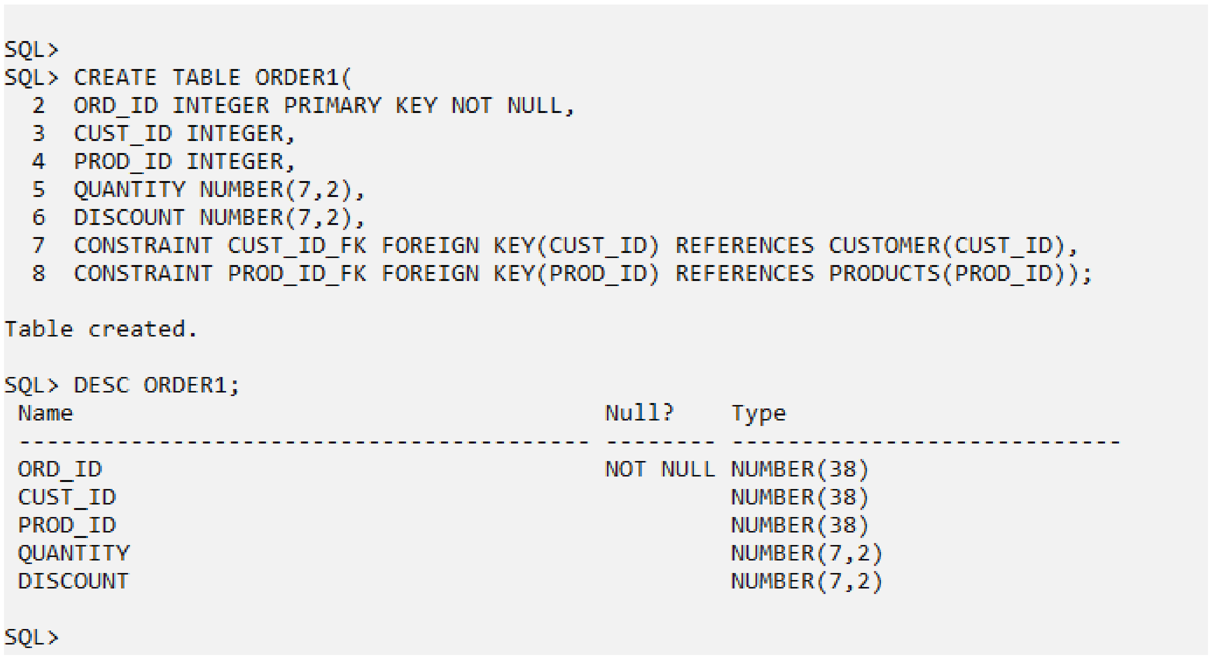
```
CREATE TABLE PRODUCTS(
PROD_ID INTEGER PRIMARY KEY NOT NULL,
PROD_NAME VARCHAR2(10),
CATEGORY VARCHAR2(10),
PRICE NUMBER(6,2));
```



```
SQL> CREATE TABLE PRODUCTS(
  2  PROD_ID INTEGER PRIMARY KEY NOT NULL,
  3  PROD_NAME VARCHAR2(10),
  4  CATEGORY VARCHAR2(10),
  5  PRICE NUMBER(6,2));

Table created.

SQL> desc products;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PROD_ID                                   NOT NULL NUMBER(38)
 PROD_NAME                                          VARCHAR2(10)
 CATEGORY                                           VARCHAR2(10)
 PRICE                                              NUMBER(6,2)

SQL>
```

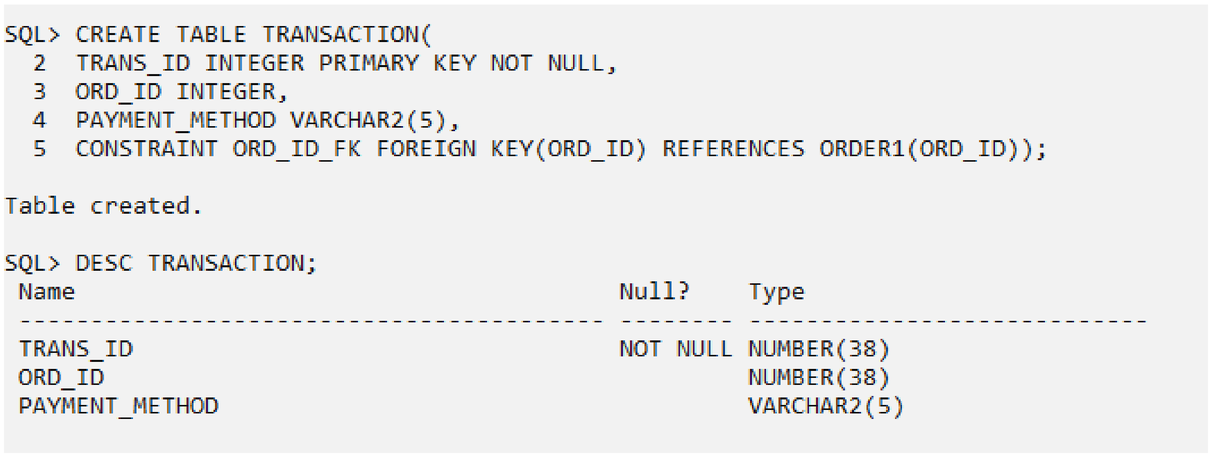
```
CREATE TABLE ORDER1(
ORD_ID INTEGER PRIMARY KEY NOT NULL,
CUST_ID INTEGER,
PROD_ID INTEGER,
QUANTITY NUMBER(7,2),
```

DISCOUNT NUMBER(7,2),
CONSTRAINT CUST_ID_FK FOREIGN KEY(CUST_ID) REFERENCES CUSTOMER(CUST_ID),
CONSTRAINT PROD_ID_FK FOREIGN KEY(PROD_ID) REFERENCES PRODUCTS(PROD_ID));



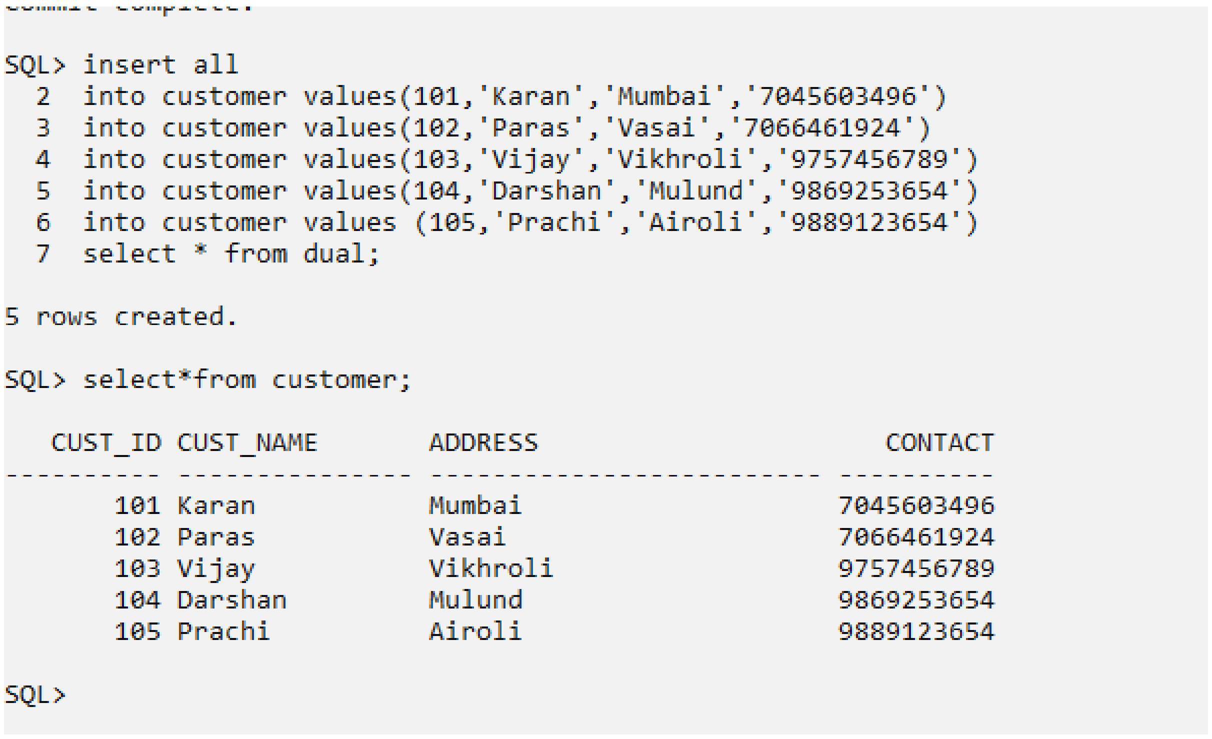
```
SQL>
SQL> CREATE TABLE ORDER1(
  2  ORD_ID INTEGER PRIMARY KEY NOT NULL,
  3  CUST_ID INTEGER,
  4  PROD_ID INTEGER,
  5  QUANTITY NUMBER(7,2),
  6  DISCOUNT NUMBER(7,2),
  7  CONSTRAINT CUST_ID_FK FOREIGN KEY(CUST_ID) REFERENCES CUSTOMER(CUST_ID),
  8  CONSTRAINT PROD_ID_FK FOREIGN KEY(PROD_ID) REFERENCES PRODUCTS(PROD_ID));

Table created.

SQL> DESC ORDER1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORD_ID                                    NOT NULL NUMBER(38)
 CUST_ID                                            NUMBER(38)
 PROD_ID                                            NUMBER(38)
 QUANTITY                                           NUMBER(7,2)
 DISCOUNT                                           NUMBER(7,2)

SQL>
```

CREATE TABLE TRANSACTION
(
TRANS_ID INTEGER PRIMARY KEY NOT NULL,
ORD_ID INTEGER,
PAYMENT_METHOD VARCHAR2(5),
CONSTRAINT ORD_ID_FK FOREIGN KEY(ORD_ID) REFERENCES ORDER1(ORD_ID));


```
SQL> CREATE TABLE TRANSACTION(
  2  TRANS_ID INTEGER PRIMARY KEY NOT NULL,
  3  ORD_ID INTEGER,
  4  PAYMENT_METHOD VARCHAR2(5),
  5  CONSTRAINT ORD_ID_FK FOREIGN KEY(ORD_ID) REFERENCES ORDER1(ORD_ID));

Table created.

SQL> DESC TRANSACTION;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 TRANS_ID                                  NOT NULL NUMBER(38)
 ORD_ID                                             NUMBER(38)
 PAYMENT_METHOD                                     VARCHAR2(5)
```

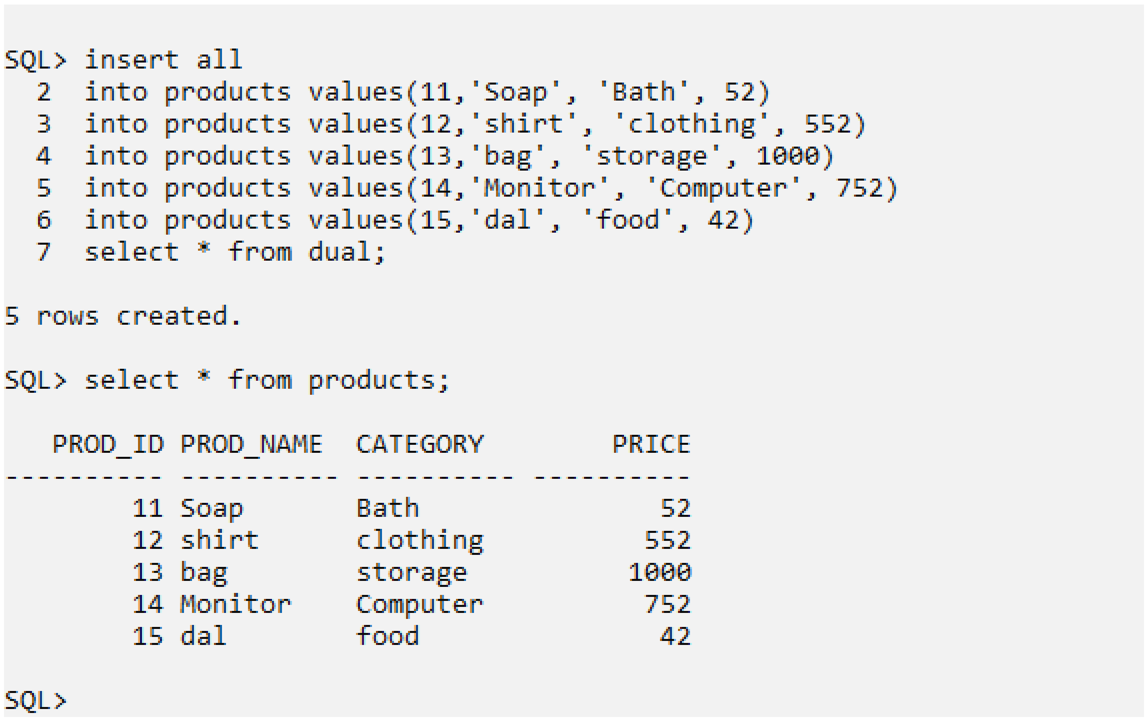
*Inserting values in customer code

```
insert all
into customer values(101,'Karan','Mumbai','7045603496')
into customer values(102,'Paras','Vasai','7066461924')
into customer values(103,'Vijay','Vikhroli','9757456789')
into customer values(104,'Darshan','Mulund','9869253654')
into customer values (105,'Prachi','Airoli','9889123654')
select * from dual;
```



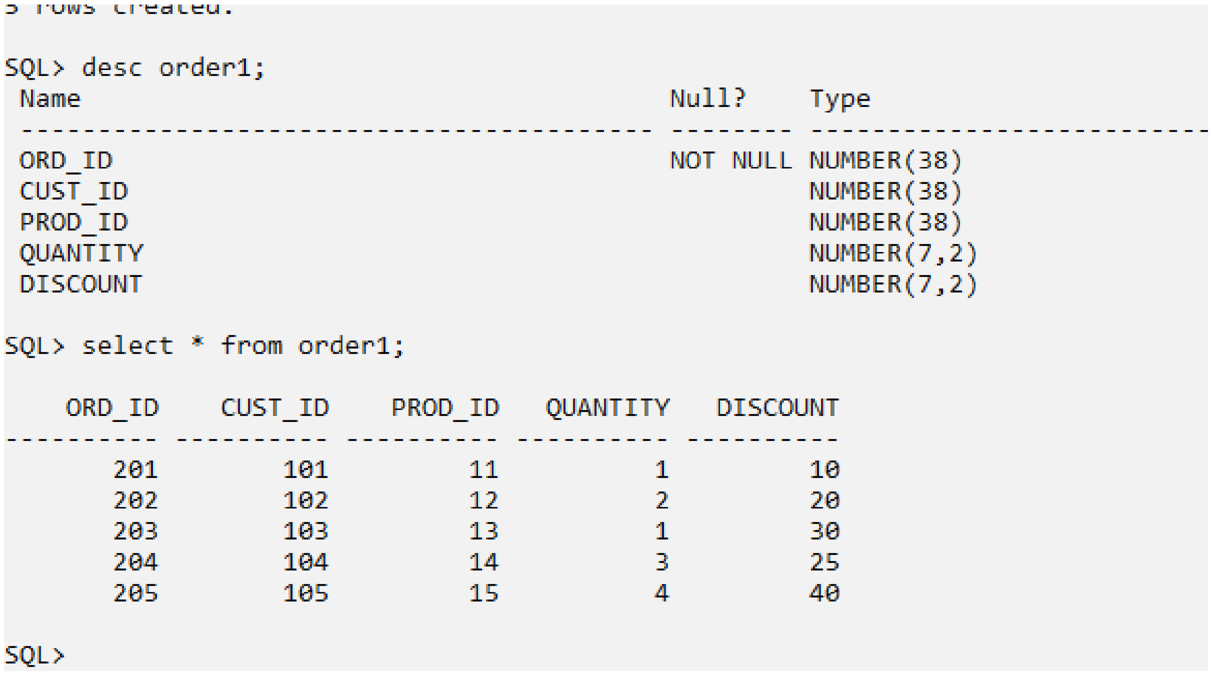
```
SQL> insert all
  2  into customer values(101,'Karan','Mumbai','7045603496')
  3  into customer values(102,'Paras','Vasai','7066461924')
  4  into customer values(103,'Vijay','Vikhroli','9757456789')
  5  into customer values(104,'Darshan','Mulund','9869253654')
  6  into customer values (105,'Prachi','Airoli','9889123654')
  7  select * from dual;

5 rows created.

SQL> select*from customer;

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

SQL>
```

```
insert all
into products values(11,'Soap', 'Bath', 52)
into products values(12,'shirt', 'clothing', 552)
into products values(13,'bag', 'storage', 1000)
into products values(14,'Monitor', 'Computer', 752)
into products values(15,'dal', 'food', 42)
select * from dual;
```



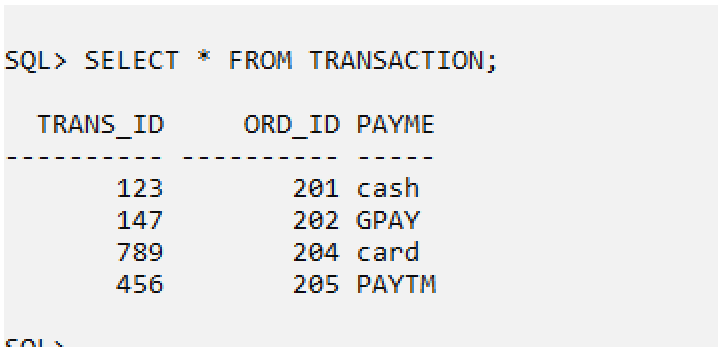
```
SQL> insert all
  2  into products values(11,'Soap', 'Bath', 52)
  3  into products values(12,'shirt', 'clothing', 552)
  4  into products values(13,'bag', 'storage', 1000)
  5  into products values(14,'Monitor', 'Computer', 752)
  6  into products values(15,'dal', 'food', 42)
  7  select * from dual;

5 rows created.

SQL> select * from products;

   PROD_ID PROD_NAME  CATEGORY        PRICE
---------- ---------- ---------- ----------
        11 Soap       Bath               52
        12 shirt      clothing          552
        13 bag        storage          1000
        14 Monitor    Computer          752
        15 dal        food               42

SQL>
```

insert all
into order1 values(201,101,11,1,10)
into order1 values(202,102,12,2,20)
into order1 values(203,103,13,1,30)
into order1 values(204,104,14,3,25)
into order1 values(205,105,15,4,40)
select * from dual;



```
5 rows created.

SQL> desc order1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORD_ID                                    NOT NULL NUMBER(38)
 CUST_ID                                            NUMBER(38)
 PROD_ID                                            NUMBER(38)
 QUANTITY                                           NUMBER(7,2)
 DISCOUNT                                           NUMBER(7,2)

SQL> select * from order1;

    ORD_ID    CUST_ID    PROD_ID   QUANTITY   DISCOUNT
---------- ---------- ---------- ---------- ----------
       201        101         11          1         10
       202        102         12          2         20
       203        103         13          1         30
       204        104         14          3         25
       205        105         15          4         40

SQL>
```

insert all
into transaction values(123,201,'cash')
into transaction values(147,202,'GPAY')
into transaction values(567,203,'PHOPE')
into transaction values(789,204,'card')
into transaction values(456,205,'PAYTM')
select * from dual;



```
SQL> SELECT * FROM TRANSACTION;

  TRANS_ID     ORD_ID PAYME
---------- ---------- -----
       123        201 cash
       147        202 GPAY
       789        204 card
       456        205 PAYTM

SQL>
```

c) Inner Join
Description:
The inner join keywords selects records that have matching values in both tables.
Syntax:
select column_name(s)
FROM table1
join table2
ON table1.coulmn_name=table2.column_name;
or
select column_name(s)
from table1
inner join table2
on table1.column_name=table2.column_name;

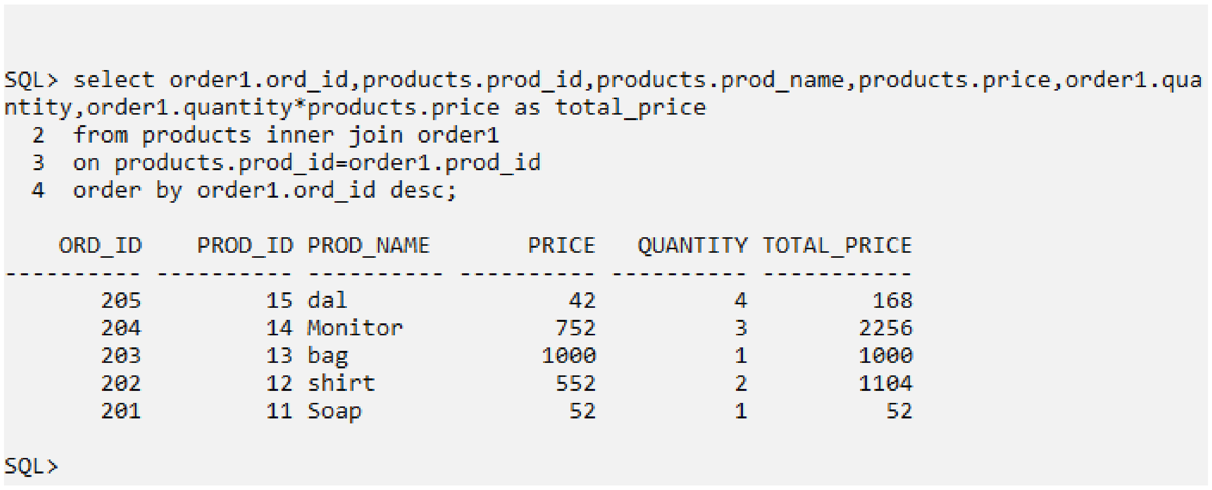
Question and query

Code:

select
order1.ord_id,products.prod_id,products.prod_name,products.price,order1.quantity,order1.quantity*products.price as total_price
from products inner join order1
on products.prod_id=order1.prod_id

order by order1.ord_id desc;

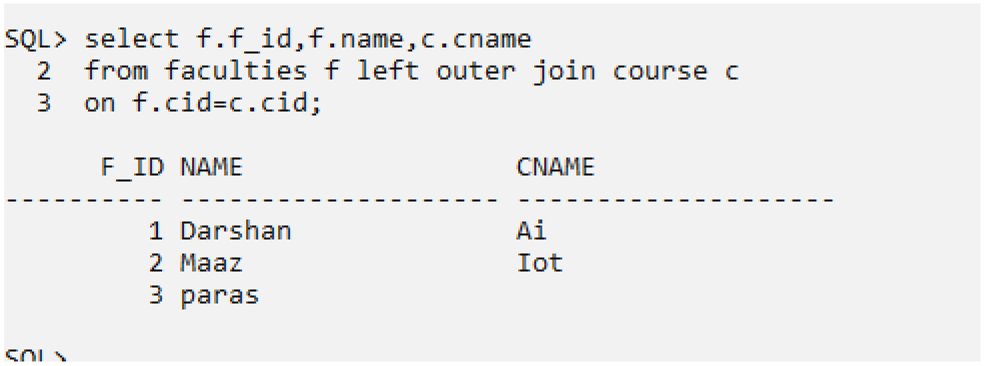
Output:



```
SQL> select order1.ord_id,products.prod_id,products.prod_name,products.price,order1.qua
ntity,order1.quantity*products.price as total_price
  2  from products inner join order1
  3  on products.prod_id=order1.prod_id
  4  order by order1.ord_id desc;

    ORD_ID    PROD_ID PROD_NAME       PRICE   QUANTITY TOTAL_PRICE
---------- ---------- ---------- ---------- ---------- -----------
       205         15 dal                42          4         168
       204         14 Monitor           752          3        2256
       203         13 bag              1000          1        1000
       202         12 shirt             552          2        1104
       201         11 Soap               52          1          52

SQL>
```

d) Left outer joins
Description:
Returns all records from the left table, and the matched records from the right table
Syntax:
select column_name
from table1
left join table2
on table1.column_name=table2.column_name
or
select column_name(s)
from table1
left outer join table2
on table1.column_name=table2.column_name;

code:
select f.f_id,f.name,c.cname
from faculty f left outer join course c
on f.cid=c.cid;
Output:

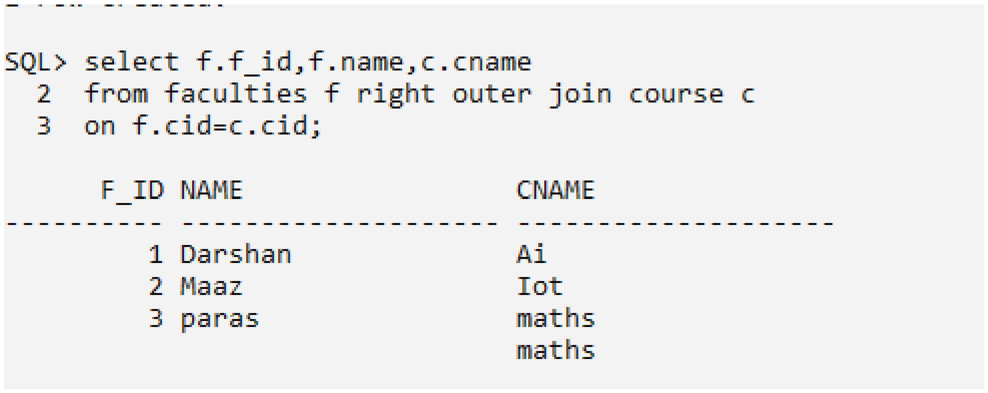


```
SQL> select f.f_id,f.name,c.cname
  2  from faculties f left outer join course c
  3  on f.cid=c.cid;

      F_ID NAME                 CNAME
---------- -------------------- --------------------
         1 Darshan              Ai
         2 Maaz                 Iot
         3 paras

SQL>
```

e) Right Outer Join:
Description:
Returns all records from the right table, and the matched records from the left table

Syntax:

select column_name
from table1
right join table2
on table1.column_name=table2.column_name
or
select column_name(s)
from table1
right outer join table2
on table1.column_name=table2.column_name;
Code:



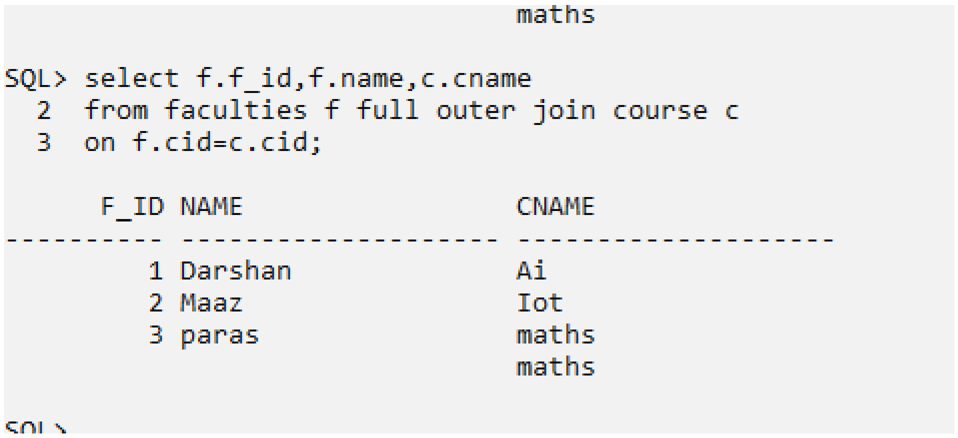
```
SQL> select f.f_id,f.name,c.cname
  2  from faculties f right outer join course c
  3  on f.cid=c.cid;

      F_ID NAME                 CNAME
---------- -------------------- --------------------
         1 Darshan              Ai
         2 Maaz                 Iot
         3 paras                maths
                                maths
```

g) Full Outer join
Description:
Returns all records when there is a match in either left or right table

Code:
select f.f_id,f.name,c.cname

from faculties f full outer join course c
on f.cid=c.cid;

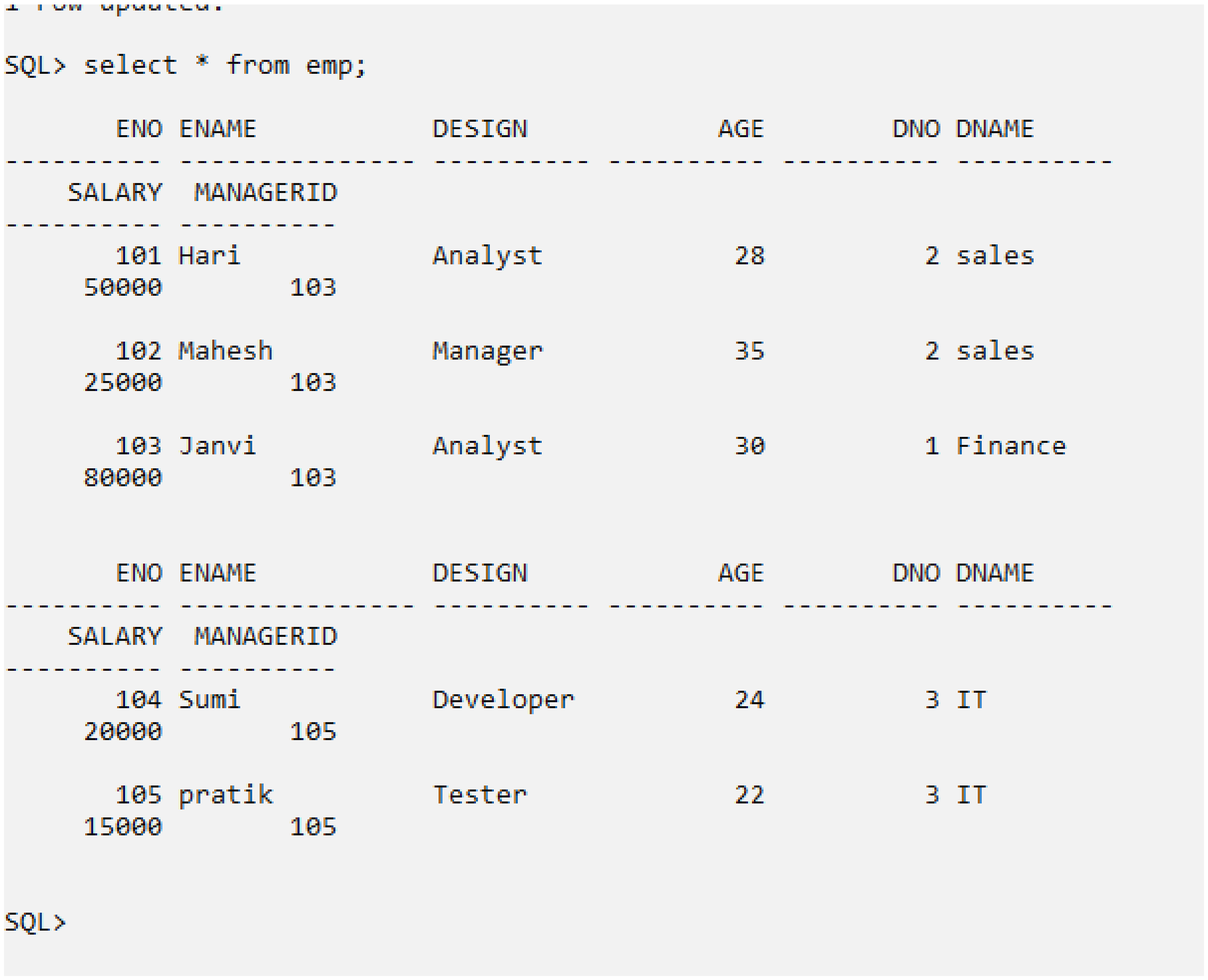
Output:



```
                                maths

SQL> select f.f_id,f.name,c.cname
  2  from faculties f full outer join course c
  3  on f.cid=c.cid;

      F_ID NAME                 CNAME
---------- -------------------- --------------------
         1 Darshan              Ai
         2 Maaz                 Iot
         3 paras                maths
                                maths

SQL>
```

alter table emp
ADD (managerid int);

alter table emp
update emp set managerid=103 where eno=101;
update emp set managerid=103 where eno=102;
update emp set managerid=103 where eno=103;
update emp set managerid=105 where eno=104;
update emp set managerid=105 where eno=105;

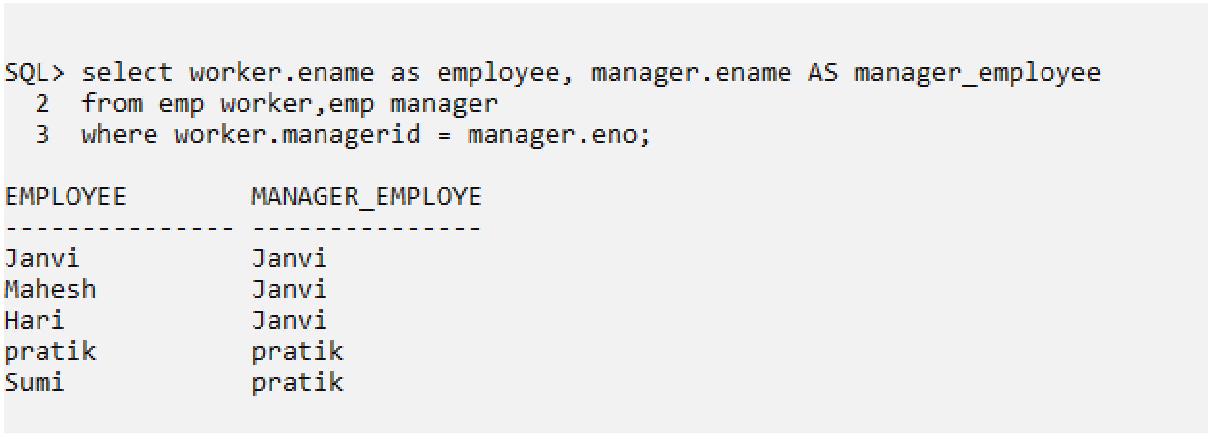


```
1 row updated.

SQL> select * from emp;

       ENO ENAME           DESIGN            AGE        DNO DNAME
---------- --------------- ---------- ---------- ---------- ----------
    SALARY  MANAGERID
---------- ----------
       101 Hari            Analyst            28          2 sales
     50000        103

       102 Mahesh          Manager            35          2 sales
     25000        103

       103 Janvi           Analyst            30          1 Finance
     80000        103


       ENO ENAME           DESIGN            AGE        DNO DNAME
---------- --------------- ---------- ---------- ---------- ----------
    SALARY  MANAGERID
---------- ----------
       104 Sumi            Developer          24          3 IT
     20000        105

       105 pratik          Tester             22          3 IT
     15000        105


SQL>
```

Code for joining the table

Code:

```
select worker.ename as employee, manager.ename AS manager_employee
from emp worker,emp manager
where worker.managerid = manager.eno;
```



```
SQL> select worker.ename as employee, manager.ename AS manager_employee
  2  from emp worker,emp manager
  3  where worker.managerid = manager.eno;

EMPLOYEE        MANAGER_EMPLOYE
--------------- ---------------
Janvi           Janvi
Mahesh          Janvi
Hari            Janvi
pratik          pratik
Sumi            pratik
```

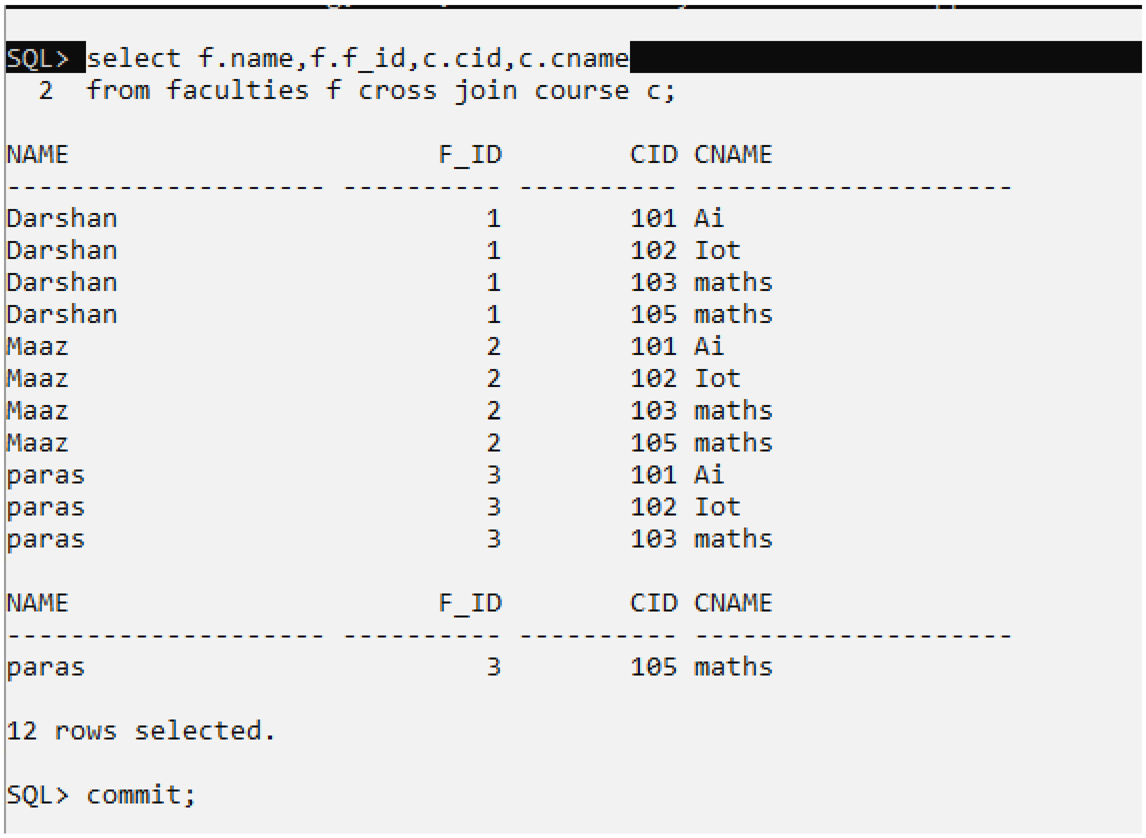
h) Creating cross Joins it is cartesian product

Description: It produces a result set which is the number of rows in the first table multiplied by the number of rows in the second table

Syntax:

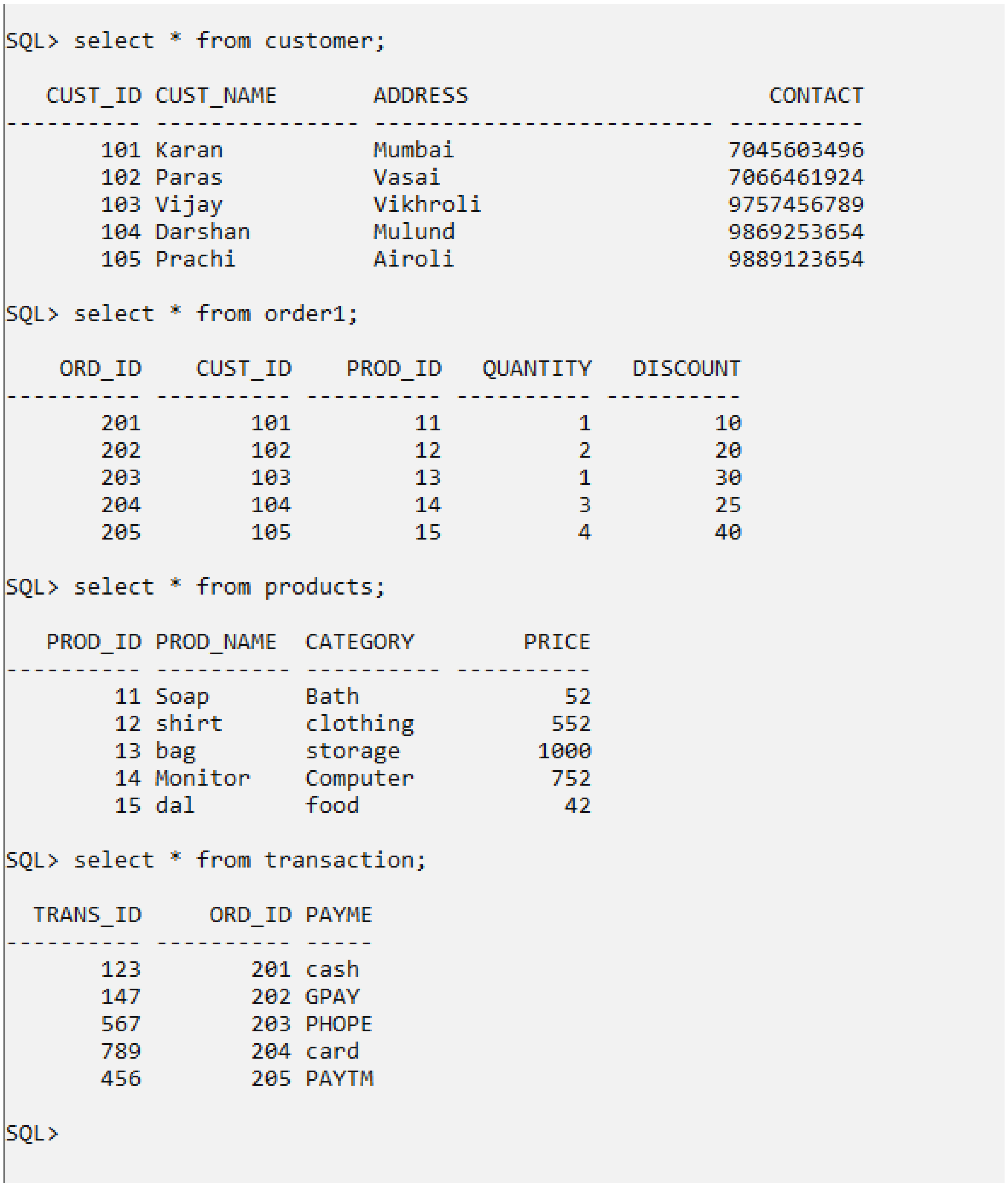
select f.name,f.f_id,c.cid,c.cname
from faculties f cross join course c;

Output:



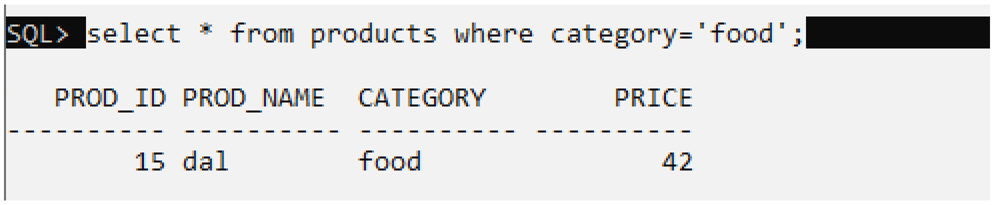
```
SQL> select f.name,f.f_id,c.cid,c.cname
  2  from faculties f cross join course c;

NAME                       F_ID        CID CNAME
-------------------- ---------- ---------- --------------------
Darshan                       1        101 Ai
Darshan                       1        102 Iot
Darshan                       1        103 maths
Darshan                       1        105 maths
Maaz                          2        101 Ai
Maaz                          2        102 Iot
Maaz                          2        103 maths
Maaz                          2        105 maths
paras                         3        101 Ai
paras                         3        102 Iot
paras                         3        103 maths

NAME                       F_ID        CID CNAME
-------------------- ---------- ---------- --------------------
paras                         3        105 maths

12 rows selected.

SQL> commit;
```

* fetch all 4 four tables



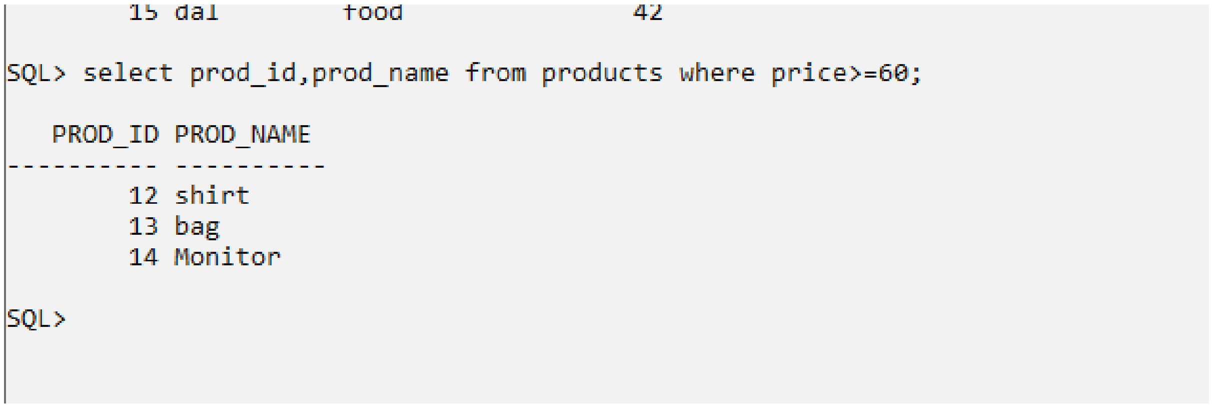
```
SQL> select * from customer;

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

SQL> select * from order1;

    ORD_ID    CUST_ID    PROD_ID   QUANTITY   DISCOUNT
---------- ---------- ---------- ---------- ----------
       201        101         11          1         10
       202        102         12          2         20
       203        103         13          1         30
       204        104         14          3         25
       205        105         15          4         40

SQL> select * from products;

   PROD_ID PROD_NAME  CATEGORY        PRICE
---------- ---------- ---------- ----------
        11 Soap       Bath               52
        12 shirt      clothing          552
        13 bag        storage          1000
        14 Monitor    Computer          752
        15 dal        food               42

SQL> select * from transaction;

  TRANS_ID     ORD_ID PAYME
---------- ---------- -----
       123        201 cash
       147        202 GPAY
       567        203 PHOPE
       789        204 card
       456        205 PAYTM

SQL>
```

- **Exercise:**
- Get product details in home category
  Code:



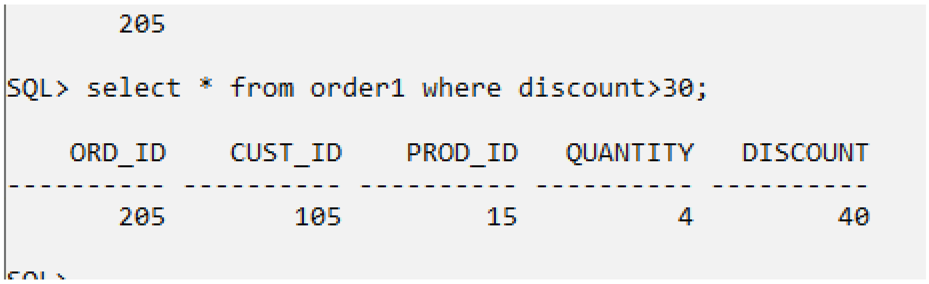
```
SQL> select * from products where category='food';

   PROD_ID PROD_NAME  CATEGORY        PRICE
---------- ---------- ---------- ----------
        15 dal        food               42
```

*get id and name of proudcts with price more than 60.
select prod_id,prod_name from products where price>=60;



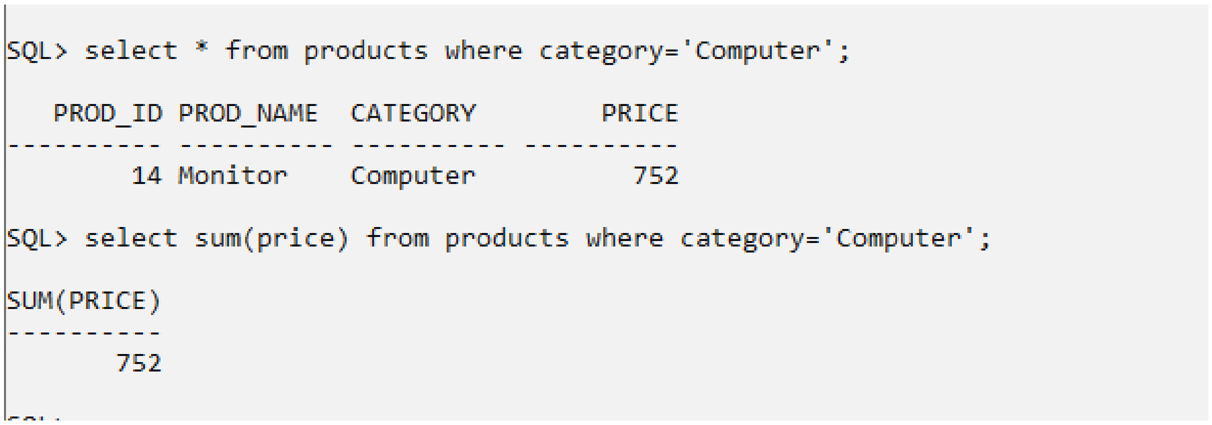
```
        15 dal        food               42

SQL> select prod_id,prod_name from products where price>=60;

   PROD_ID PROD_NAME
---------- ----------
        12 shirt
        13 bag
        14 Monitor

SQL>
```

*get order details with discount more than 30%
select * from order1 where discount>30;



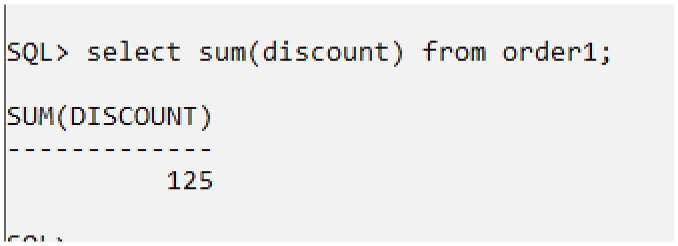
```
       205

SQL> select * from order1 where discount>30;

    ORD_ID    CUST_ID    PROD_ID   QUANTITY   DISCOUNT
---------- ---------- ---------- ---------- ----------
       205        105         15          4         40

SQL>
```

Get the total price of products in computer category
select sum(price) from products where category='Computer';



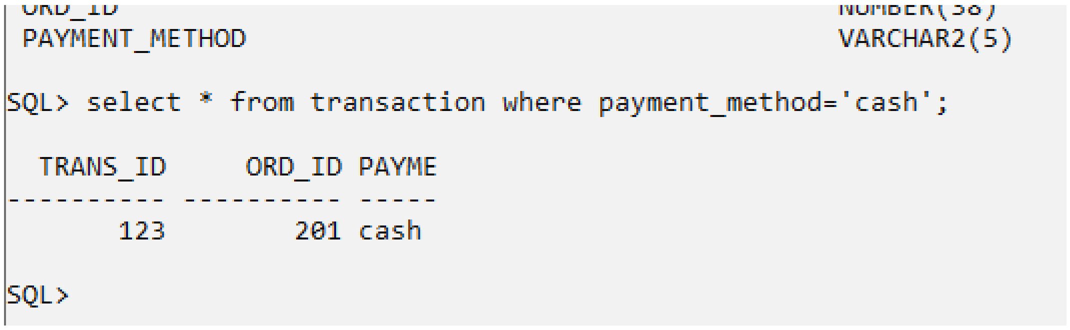
```
SQL> select * from products where category='Computer';

   PROD_ID PROD_NAME  CATEGORY        PRICE
---------- ---------- ---------- ----------
        14 Monitor    Computer          752

SQL> select sum(price) from products where category='Computer';

SUM(PRICE)
----------
       752
```

*Get the total discount given in all orders
select sum(discount) from order1;

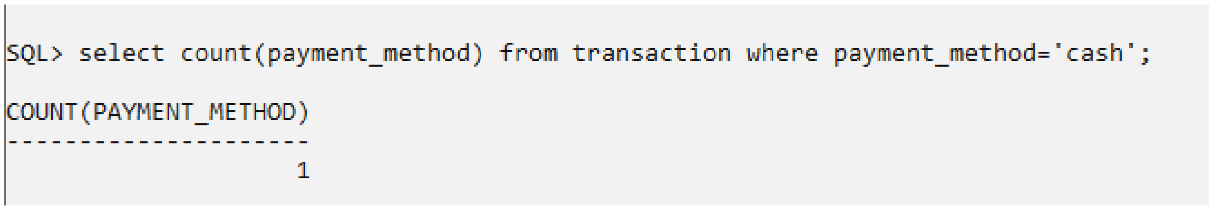


```
SQL> select sum(discount) from order1;

SUM(DISCOUNT)
-------------
          125
```

*how many transaction used in cash
select * from transaction where payment_method='cash';

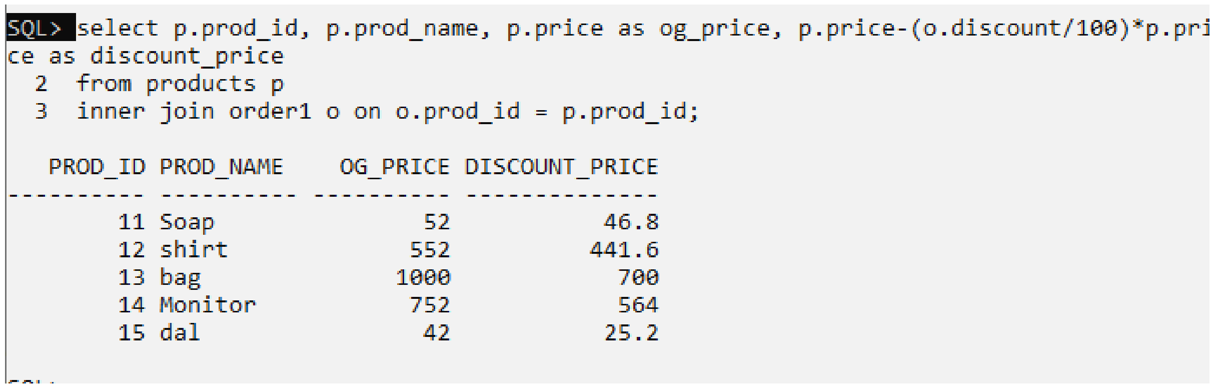


```
 PAYMENT_METHOD                                        VARCHAR2(5)

SQL> select * from transaction where payment_method='cash';

  TRANS_ID     ORD_ID PAYME
---------- ---------- -----
       123        201 cash

SQL>
```



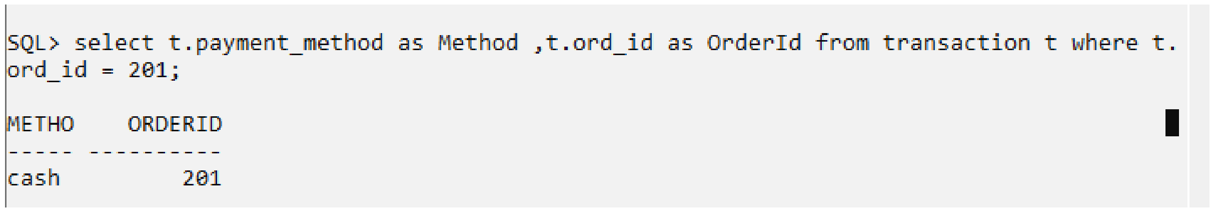
```
SQL> select count(payment_method) from transaction where payment_method='cash';

COUNT(PAYMENT_METHOD)
---------------------
                    1
```

get the product wise total discounted price
select p.prod_id, p.prod_name, p.price as og_price, p.price-(o.discount/100)*p.price as discount_price
from products p
inner join order1 o on o.prod_id = p.prod_id;



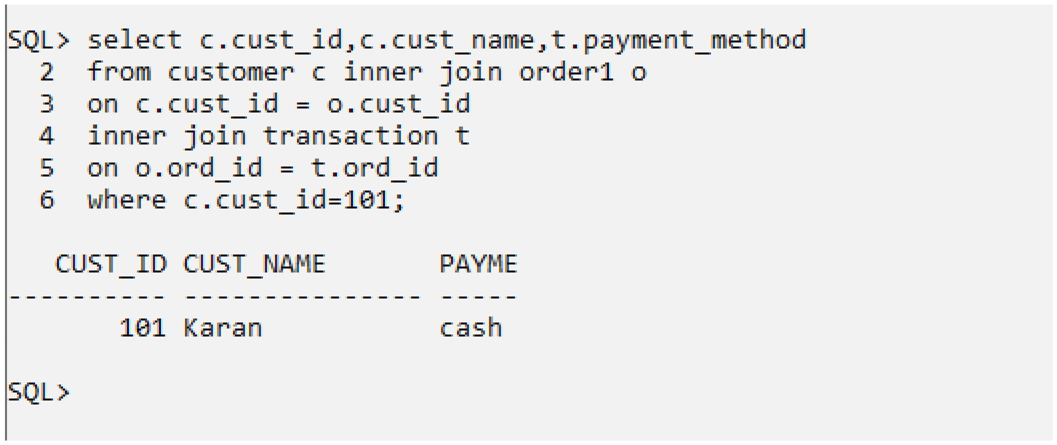
```
SQL> select p.prod_id, p.prod_name, p.price as og_price, p.price-(o.discount/100)*p.pri
ce as discount_price
  2  from products p
  3  inner join order1 o on o.prod_id = p.prod_id;

   PROD_ID PROD_NAME    OG_PRICE DISCOUNT_PRICE
---------- ---------- ---------- --------------
        11 Soap               52           46.8
        12 shirt             552          441.6
        13 bag              1000            700
        14 Monitor           752            564
        15 dal                42           25.2
```

second last
*Get the transaction method for an order 111.
select t.payment_method as Method ,t.ord_id as OrderId from transaction t where t.ord_id = 201;



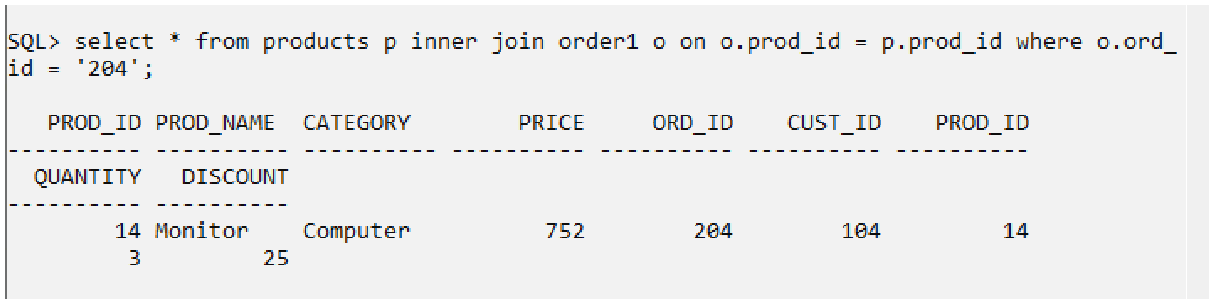
```
SQL> select t.payment_method as Method ,t.ord_id as OrderId from transaction t where t.
ord_id = 201;

METHO     ORDERID
----- ----------
cash          201
```

Last one
*Get the transaction method used by a customer 201
last question
select c.cust_id,c.cust_name,t.payment_method
from customer c inner join order1 o
on c.cust_id = o.cust_id
inner join transaction t
on o.ord_id = t.ord_id
where c.cust_id=101;



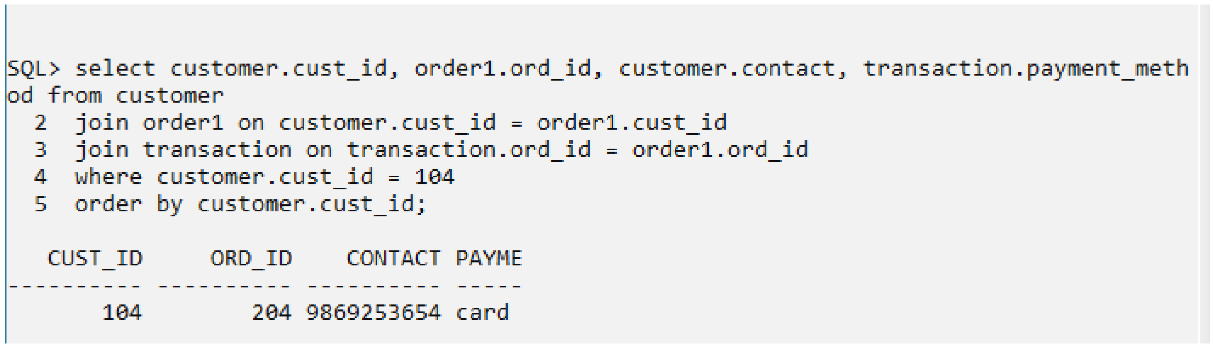
```
SQL> select c.cust_id,c.cust_name,t.payment_method
     from customer c inner join order1 o
     on c.cust_id = o.cust_id
     inner join transaction t
     on o.ord_id = t.ord_id
     where c.cust_id=101;

   CUST_ID CUST_NAME       PAYME
---------- --------------- -----
       101 Karan           cash

SQL>
```

15.Get the product details of an order id 111.
select * from products p inner join order1 o on o.prod_id = p.prod_id where o.ord_id = '204';



```
SQL> select * from products p inner join order1 o on o.prod_id = p.prod_id where o.ord_
id = '204';

   PROD_ID PROD_NAME  CATEGORY        PRICE     ORD_ID    CUST_ID    PROD_ID
---------- ---------- ---------- ---------- ---------- ---------- ----------
  QUANTITY   DISCOUNT
---------- ----------
        14 Monitor    Computer          752        204        104         14
         3         25
```

16. *16. Get the customer contact and payment method for an order with id 210.
Code:
select customer.cust_id, order1.ord_id, customer.contact, transaction.payment_method from customer
join order1 on customer.cust_id = order1.cust_id
join transaction on transaction.ord_id = order1.ord_id
where customer.cust_id = 104
order by customer.cust_id;
Output:

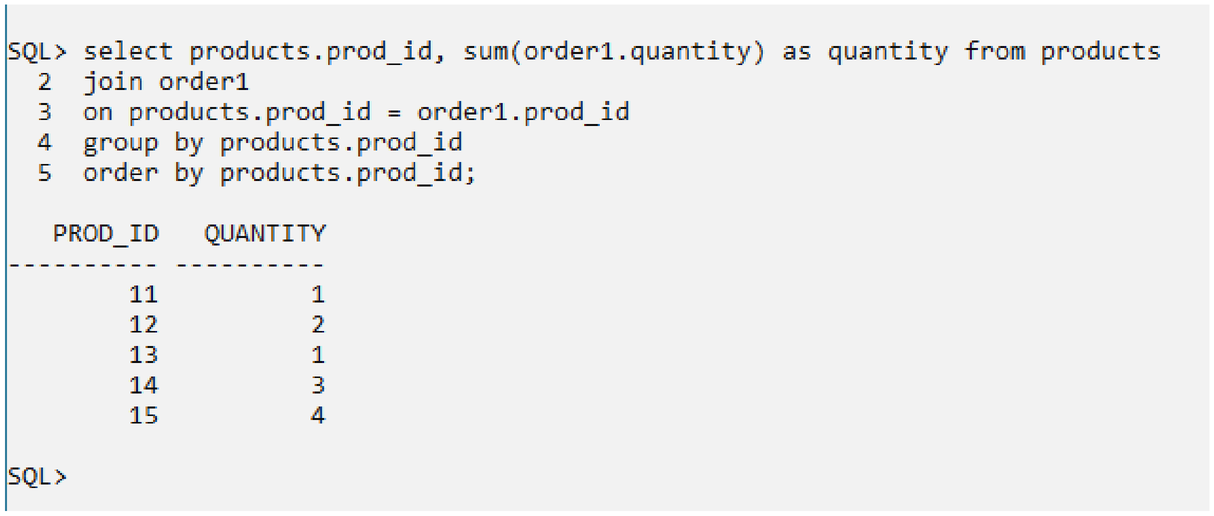


```
SQL> select customer.cust_id, order1.ord_id, customer.contact, transaction.payment_meth
od from customer
     join order1 on customer.cust_id = order1.cust_id
     join transaction on transaction.ord_id = order1.ord_id
     where customer.cust_id = 104
     order by customer.cust_id;

   CUST_ID     ORD_ID    CONTACT PAYME
---------- ---------- ---------- -----
       104        204 9869253654 card
```

*17. Get the product wise total quantity in all orders:
select products.prod_id, sum(order1.quantity) as quantity from products

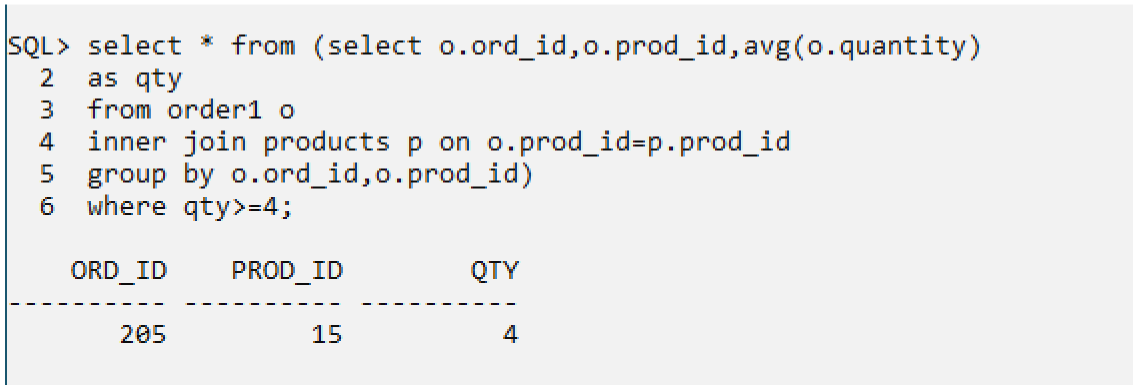
join order1
on products.prod_id = order1.prod_id
group by products.prod_id
order by products.prod_id;

Output:



```
SQL> select products.prod_id, sum(order1.quantity) as quantity from products
  2  join order1
  3  on products.prod_id = order1.prod_id
  4  group by products.prod_id
  5  order by products.prod_id;

   PROD_ID   QUANTITY
---------- ----------
        11          1
        12          2
        13          1
        14          3
        15          4

SQL>
```

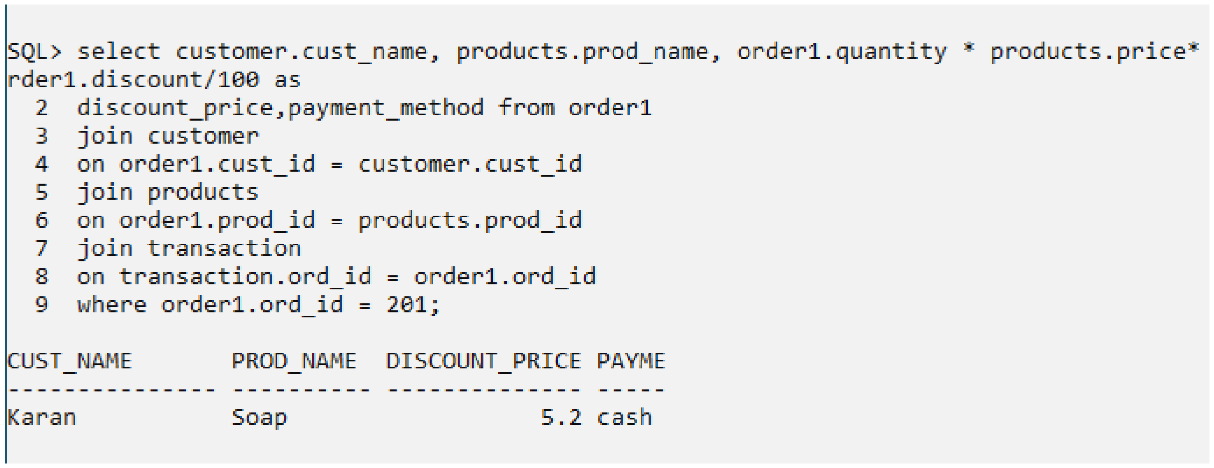
*18. Get the order id with the average quantity of products more than 20:
Code:
select * from (select o.ord_id,o.prod_id,avg(o.quantity)
as qty
from order1 o
inner join products p on o.prod_id=p.prod_id
group by o.ord_id,o.prod_id)
where qty>=4;



```
SQL> select * from (select o.ord_id,o.prod_id,avg(o.quantity)
  2  as qty
  3  from order1 o
  4  inner join products p on o.prod_id=p.prod_id
  5  group by o.ord_id,o.prod_id)
  6  where qty>=4;

    ORD_ID    PROD_ID        QTY
---------- ---------- ----------
       205         15          4
```

*19.Get the customer name, product name, discounted price, payment method for an id 1:
select customer.cust_name, products.prod_name, order1.quantity * products.price*
order1.discount/100 as
discount_price,payment_method from order1

join customer
on order1.cust_id = customer.cust_id
join products
on order1.prod_id = products.prod_id
join transaction
on transaction.ord_id = order1.ord_id
where order1.ord_id = 201;
Output:



```
SQL> select customer.cust_name, products.prod_name, order1.quantity * products.price*
rder1.discount/100 as
  2  discount_price,payment_method from order1
  3  join customer
  4  on order1.cust_id = customer.cust_id
  5  join products
  6  on order1.prod_id = products.prod_id
  7  join transaction
  8  on transaction.ord_id = order1.ord_id
  9  where order1.ord_id = 201;

CUST_NAME       PROD_NAME  DISCOUNT_PRICE PAYME
--------------- ---------- -------------- -----
Karan           Soap                  5.2 cash
```

Practical 9 : Views, synonyms, index and synonym

1. VIEW

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

Syntax for view

CREATE[OR REPLACE][FORCE|NOFORCE]

VIEW<view name>

[(column alias name...)]

AS <query> [WITH[CHECK OPTION]

[READ ONLY][CONSTRAINT]];

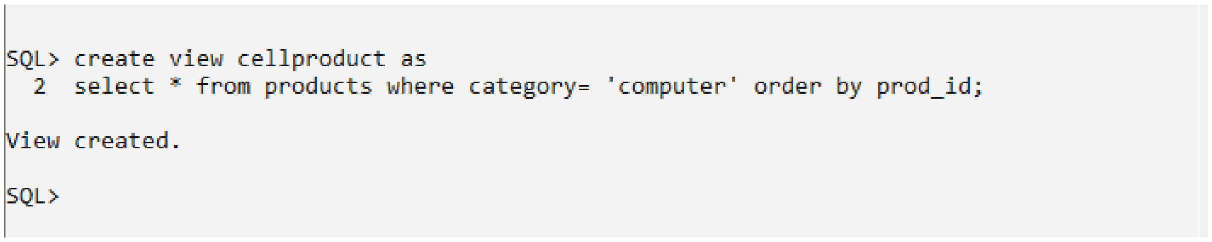
Creating view

Code:

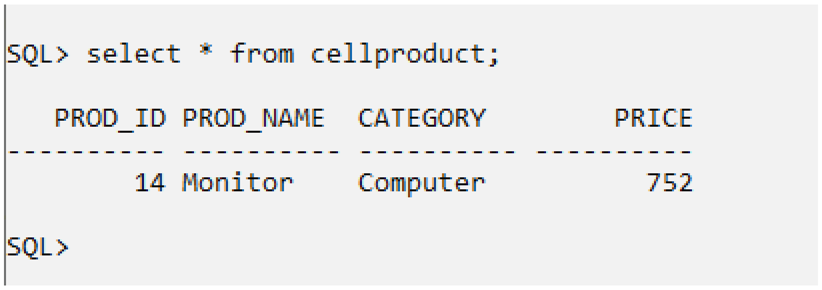
create view cellproduct as

select * from products where category= 'computer' order by prod_id;

OUTPUT:



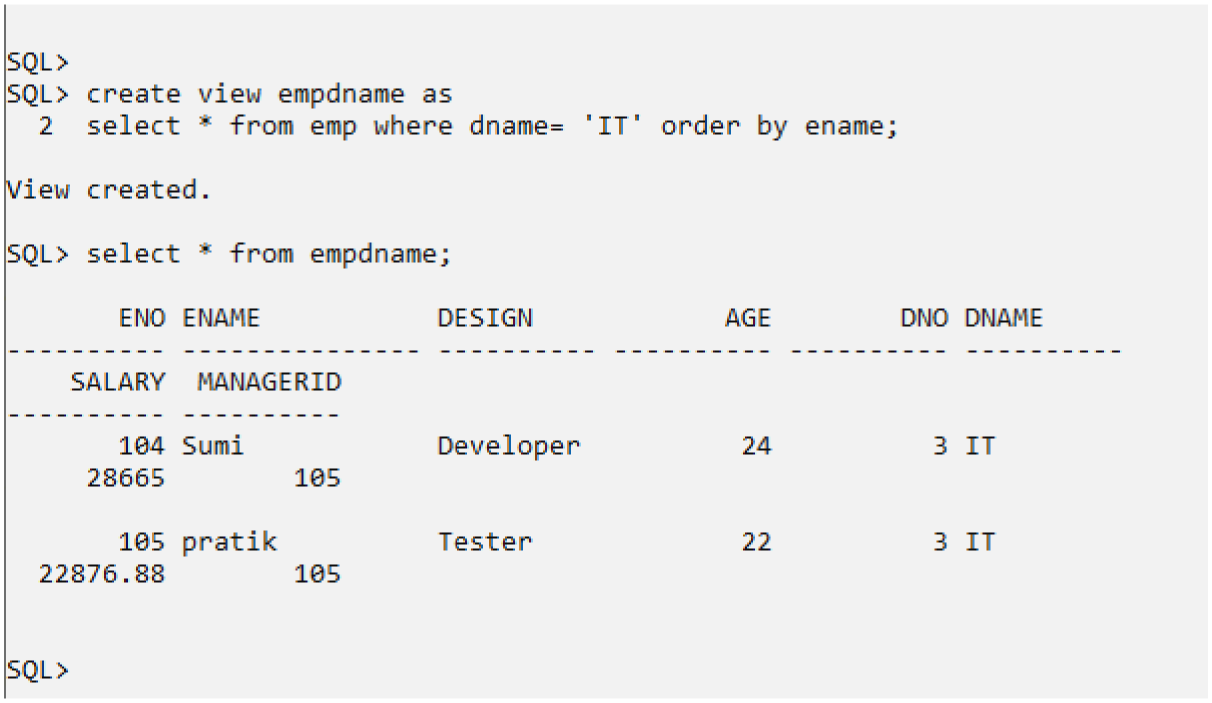
```
SQL> create view cellproduct as
  2  select * from products where category= 'computer' order by prod_id;

View created.

SQL>
```



```
SQL> select * from cellproduct;

   PROD_ID PROD_NAME  CATEGORY        PRICE
---------- ---------- ---------- ----------
        14 Monitor    Computer          752

SQL>
```

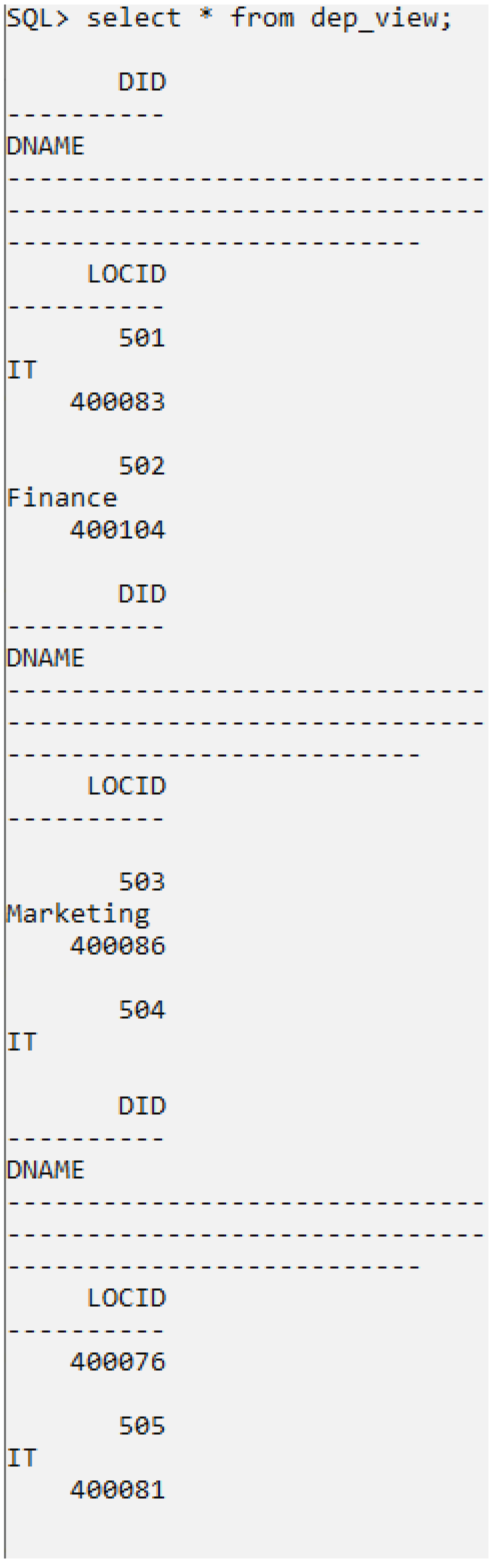
create view empdname as

select * from emp where dname= 'IT' order by ename;


```
SQL>
SQL> create view empdname as
  2  select * from emp where dname= 'IT' order by ename;

View created.

SQL> select * from empdname;

       ENO ENAME           DESIGN            AGE        DNO DNAME
---------- --------------- ---------- ---------- ---------- ----------
    SALARY  MANAGERID
---------- ----------
       104 Sumi            Developer          24          3 IT
     28665        105

       105 pratik          Tester             22          3 IT
  22876.88        105


SQL>
```

*Create a view which has exact details of a table.
create view dep_view AS
SELECT * FROM department;

Output:

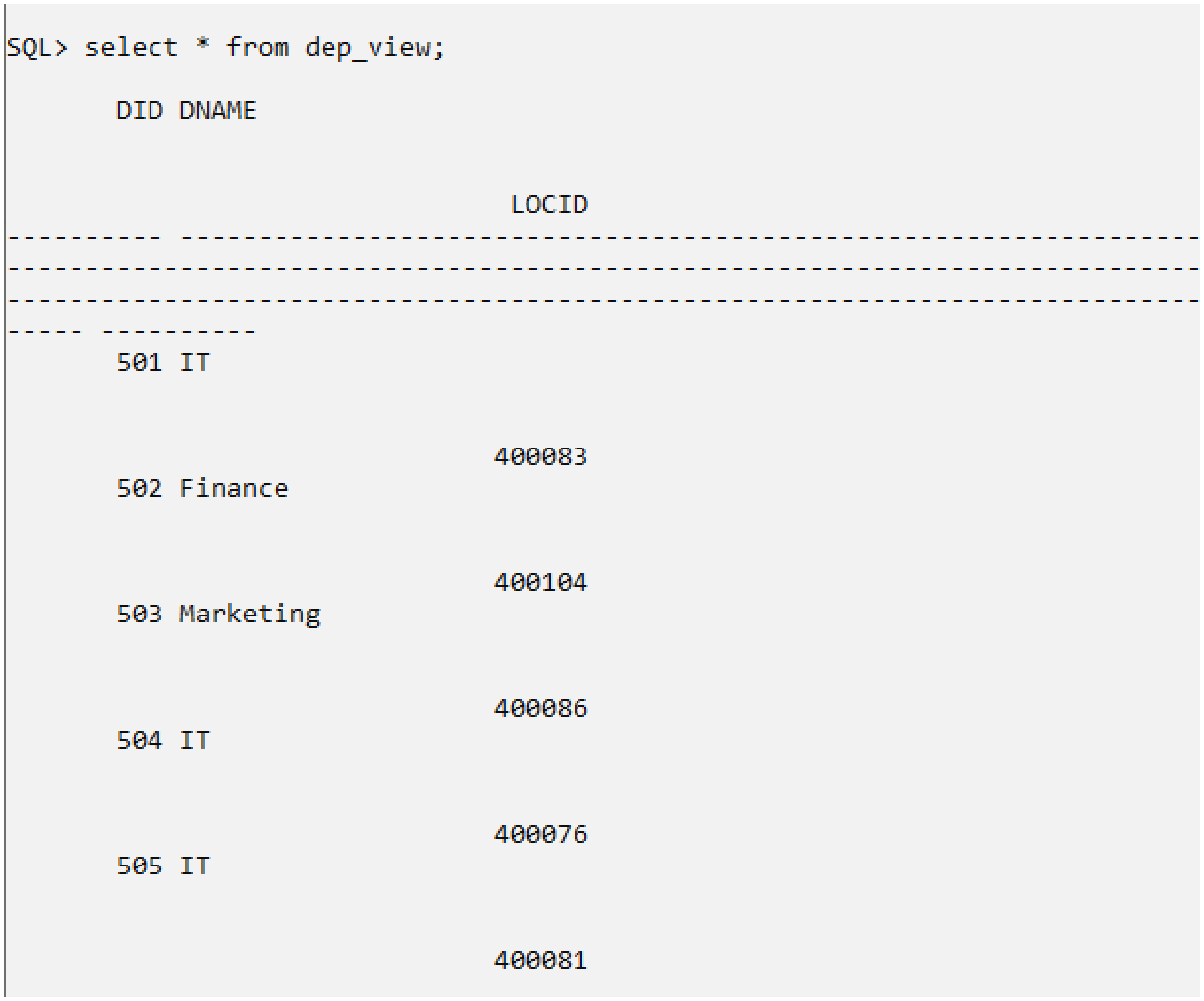


```
SQL> select * from dep_view;

       DID
----------
DNAME
------------------------------
------------------------------
--------------------------
     LOCID
----------
       501
IT
    400083

       502
Finance
    400104

       DID
----------
DNAME
------------------------------
------------------------------
--------------------------
     LOCID
----------

       503
Marketing
    400086

       504
IT

       DID
----------
DNAME
------------------------------
------------------------------
--------------------------
     LOCID
----------
    400076

       505
IT
    400081
```

Code:

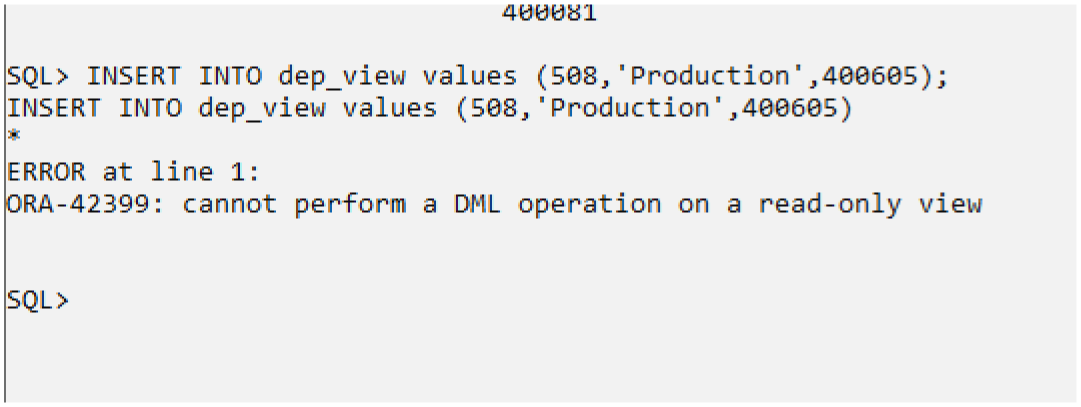
#using new read only constraint

create or replace view dep_view AS
SELECT * FROM department
WITH READ ONLY CONSTRAINT
vw_dept_view_read_only;



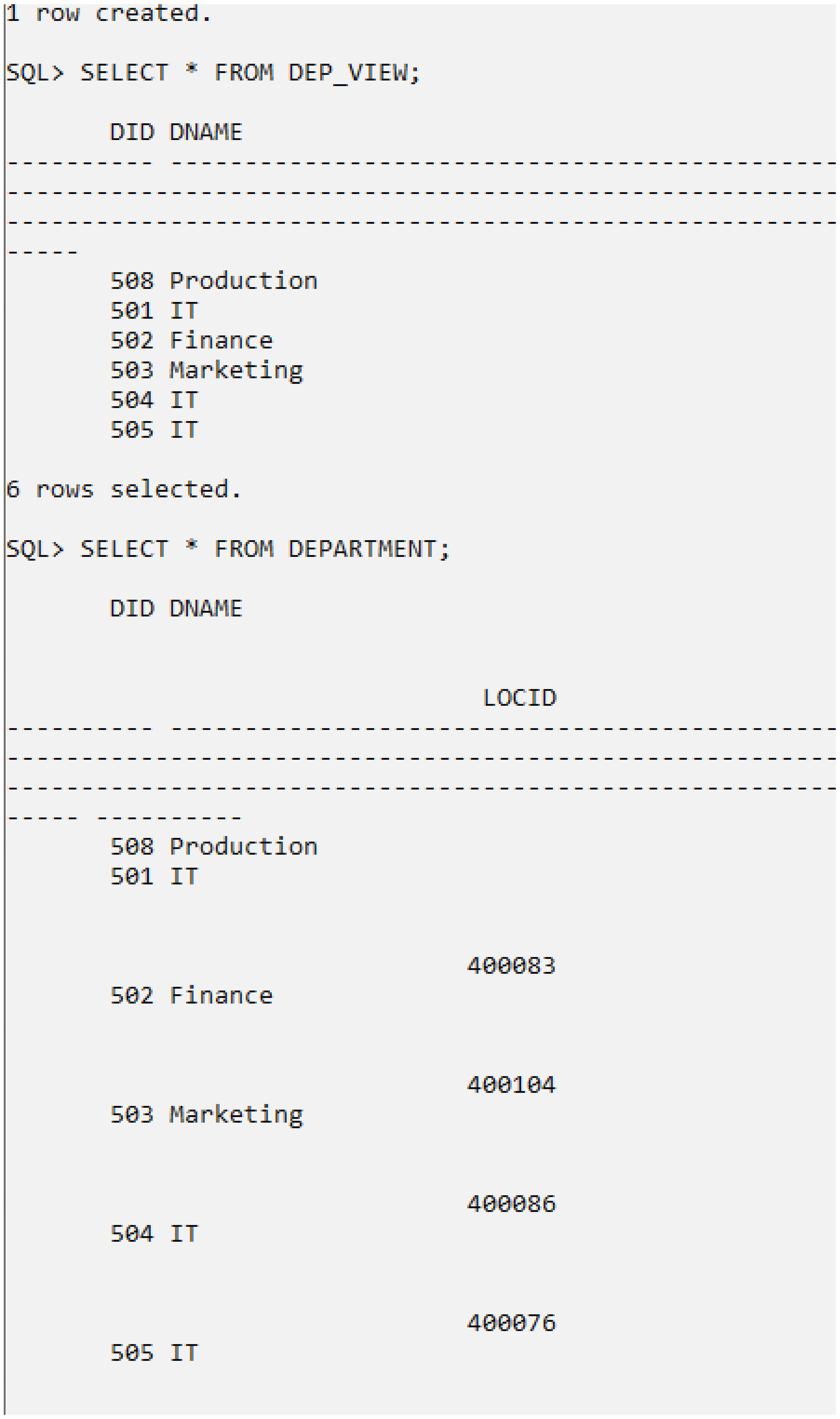
```
SQL> select * from dep_view;

       DID DNAME


                                LOCID
---------- ------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
----- ----------
       501 IT


                               400083
       502 Finance


                               400104
       503 Marketing


                               400086
       504 IT


                               400076
       505 IT


                               400081
```

***Inserting values in dep_view**
INSERT INTO dep_view values (508,'Production',400605);



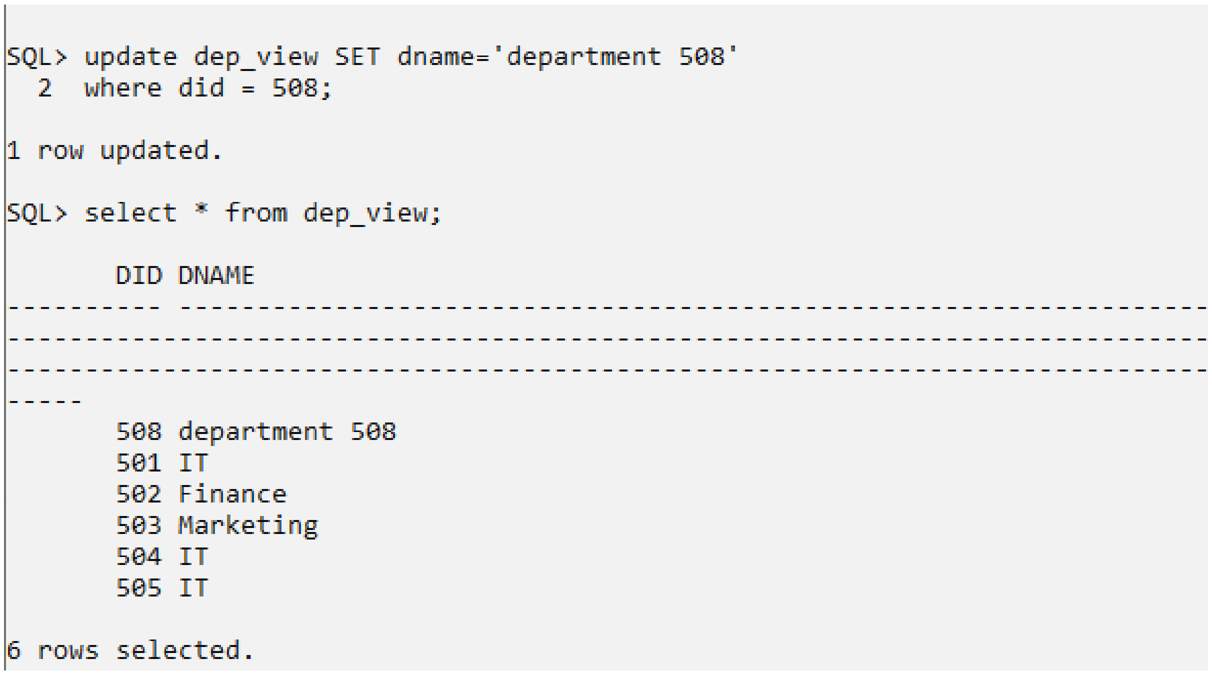
```
                              400081

SQL> INSERT INTO dep_view values (508,'Production',400605);
INSERT INTO dep_view values (508,'Production',400605)
*
ERROR at line 1:
ORA-42399: cannot perform a DML operation on a read-only view


SQL>
```

CREATE OR REPLACE VIEW DEP_VIEW AS
select did,dname
FROM department;
INSERT INTO dep_view values (508,'Production');



```
1 row created.

SQL> SELECT * FROM DEP_VIEW;

       DID DNAME
---------- ---------------------------------------------
--------------------------------------------------------
--------------------------------------------------------
-----
       508 Production
       501 IT
       502 Finance
       503 Marketing
       504 IT
       505 IT

6 rows selected.

SQL> SELECT * FROM DEPARTMENT;

       DID DNAME


                                LOCID
---------- ---------------------------------------------
--------------------------------------------------------
--------------------------------------------------------
----- ----------
       508 Production
       501 IT


                               400083
       502 Finance


                               400104
       503 Marketing


                               400086
       504 IT


                               400076
       505 IT
```

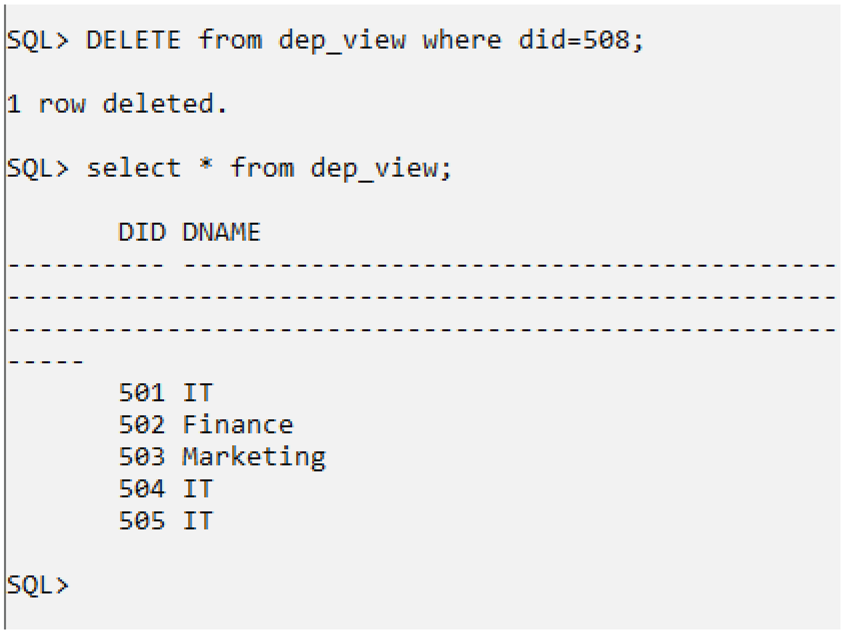
***UPDATE**
update dep_view SET dname='department 508'
where did = 508;



```
SQL> update dep_view SET dname='department 508'
  2  where did = 508;

1 row updated.

SQL> select * from dep_view;

       DID DNAME
---------- ------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
-----
       508 department 508
       501 IT
       502 Finance
       503 Marketing
       504 IT
       505 IT

6 rows selected.
```

***DELETE**

DELETE from dep_view where did=508;

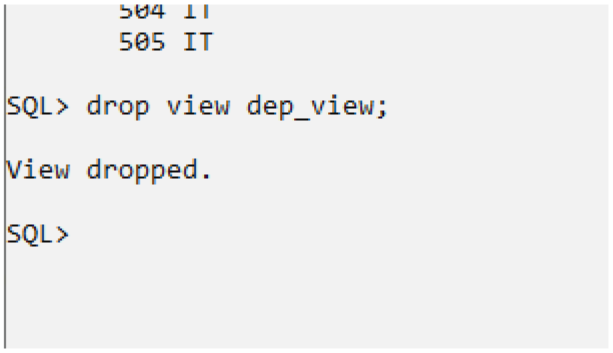


```
SQL> DELETE from dep_view where did=508;

1 row deleted.

SQL> select * from dep_view;

       DID DNAME
---------- -----------------------------------------
-----------------------------------------------------
-----------------------------------------------------
-----
       501 IT
       502 Finance
       503 Marketing
       504 IT
       505 IT

SQL>
```

***DROP**

*Dropping view

drop view dep_view;

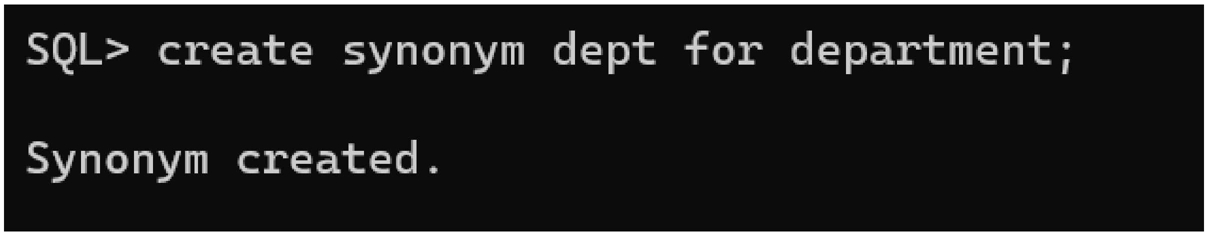

```
       504  IT
       505  IT

SQL> drop view dep_view;

View dropped.

SQL>
```

2. SYNONYMS

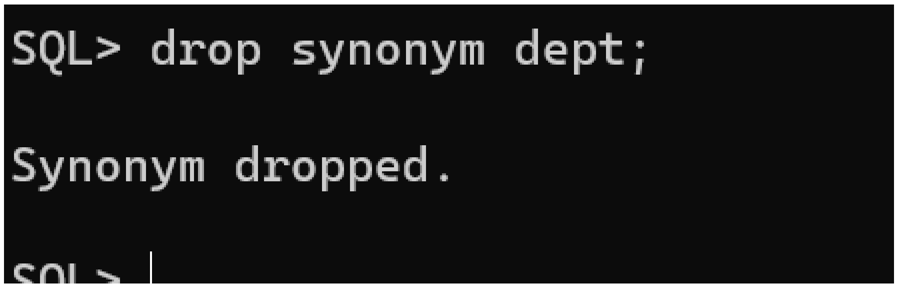
A synonym is an alias ,that is a form of shorthand used to simplify the task of referencing a database object./other name for the table

Creating synonyms
Syntax
CREATE [PUBLIC] SYNONYM synonym_name
for object_name;

Code:
CREATE SYNONYM DEPT FOR DEPARTMENT;


```
SQL> create synonym dept for department;

Synonym created.
```

Code:
DROP SYNONYM DEPT FOR DEPARTMENT;

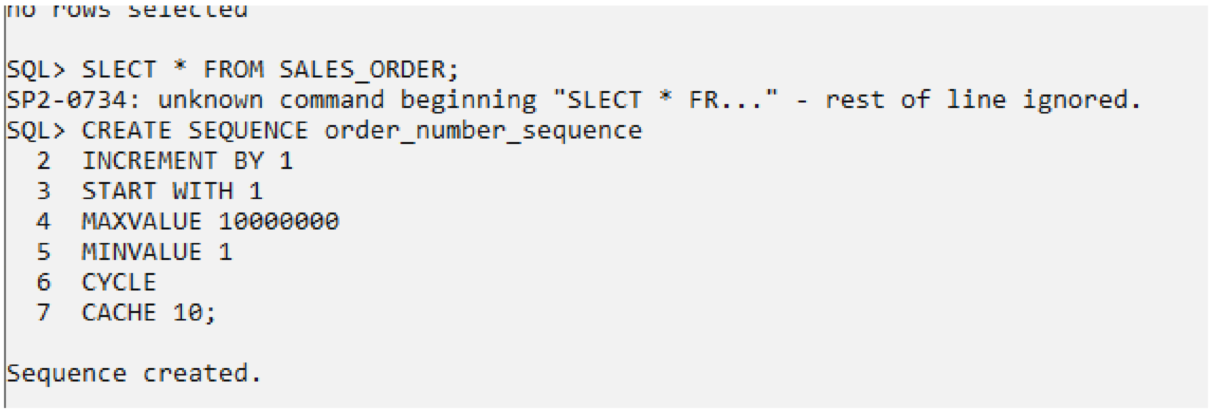

```
SQL> drop synonym dept;

Synonym dropped.

SQL>
```

3. 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];
```

Code:

```
CREATE SEQUENCE order_number_sequence
INCREMENT BY 1
START WITH 1
MAXVALUE 10000000
MINVALUE 1
CYCLE
CACHE 10;
```



```
no rows selected

SQL> SLECT * FROM SALES_ORDER;
SP2-0734: unknown command beginning "SLECT * FR..." - rest of line ignored.
SQL> CREATE SEQUENCE order_number_sequence
  2  INCREMENT BY 1
  3  START WITH 1
  4  MAXVALUE 10000000
  5  MINVALUE 1
  6  CYCLE
  7  CACHE 10;

Sequence created.
```

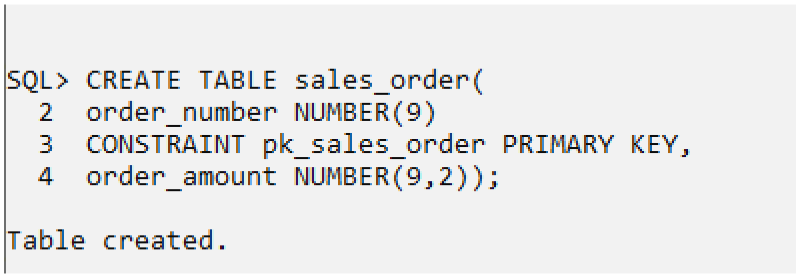
a) **Ascending Sequence Value**s

Code:

```
CREATE TABLE sales_order(
order_number NUMBER(9)
```

CONSTRAINT pk_sales_order PRIMARY KEY,
order_amount NUMBER(9,2));

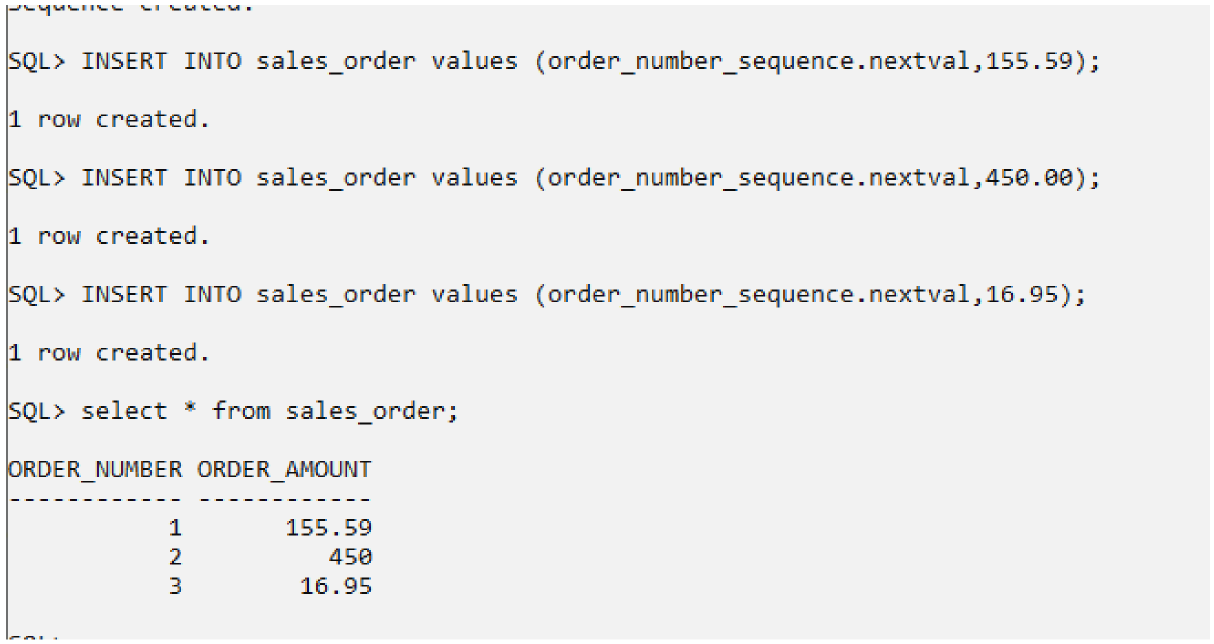
Output:



```
SQL> CREATE TABLE sales_order(
  2  order_number NUMBER(9)
  3  CONSTRAINT pk_sales_order PRIMARY KEY,
  4  order_amount NUMBER(9,2));

Table created.
```

Inserting values

INSERT INTO sales_order values (order_number_sequence.nextval,155.59);
INSERT INTO sales_order values (order_number_sequence.nextval,450.00);
INSERT INTO sales_order values (order_number_sequence.nextval,16.95);

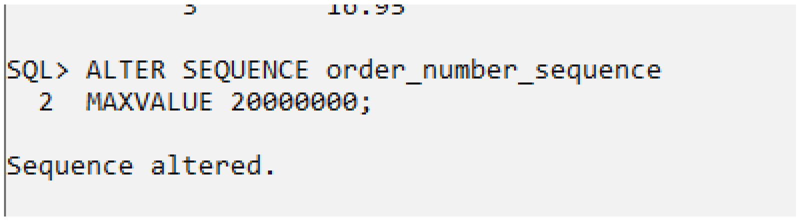


```
SQL> INSERT INTO sales_order values (order_number_sequence.nextval,155.59);

1 row created.

SQL> INSERT INTO sales_order values (order_number_sequence.nextval,450.00);

1 row created.

SQL> INSERT INTO sales_order values (order_number_sequence.nextval,16.95);

1 row created.

SQL> select * from sales_order;

ORDER_NUMBER ORDER_AMOUNT
------------ ------------
           1       155.59
           2          450
           3        16.95
```

b) Altering a sequence

Code:

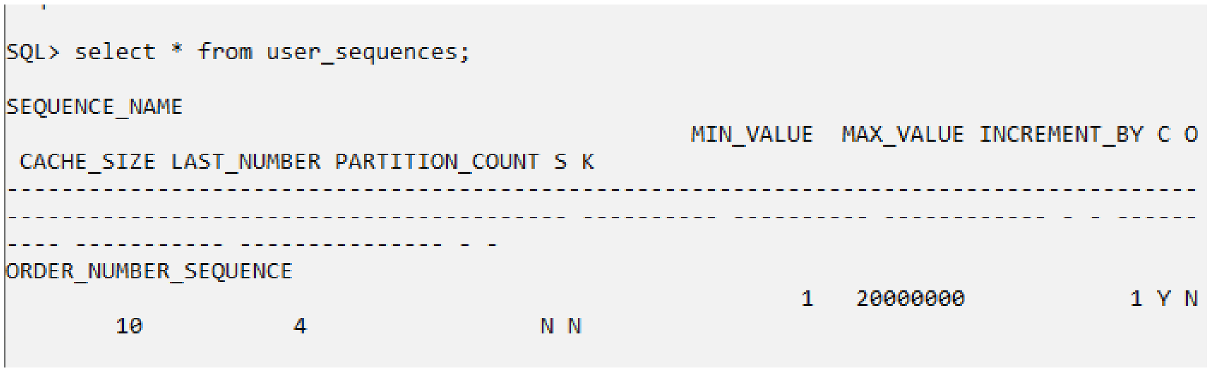
ALTER SEQUENCE order_number_sequence
MAXVALUE 20000000;



```
             5          10.95

SQL> ALTER SEQUENCE order_number_sequence
  2  MAXVALUE 20000000;

Sequence altered.
```

c) view sequence

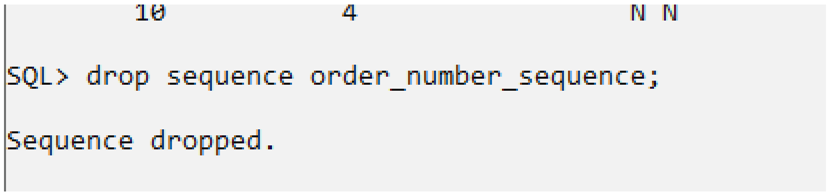
select * from user_sequences;



```
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   20000000            1 Y N
        10           4                  N N
```

d) Drop sequences:

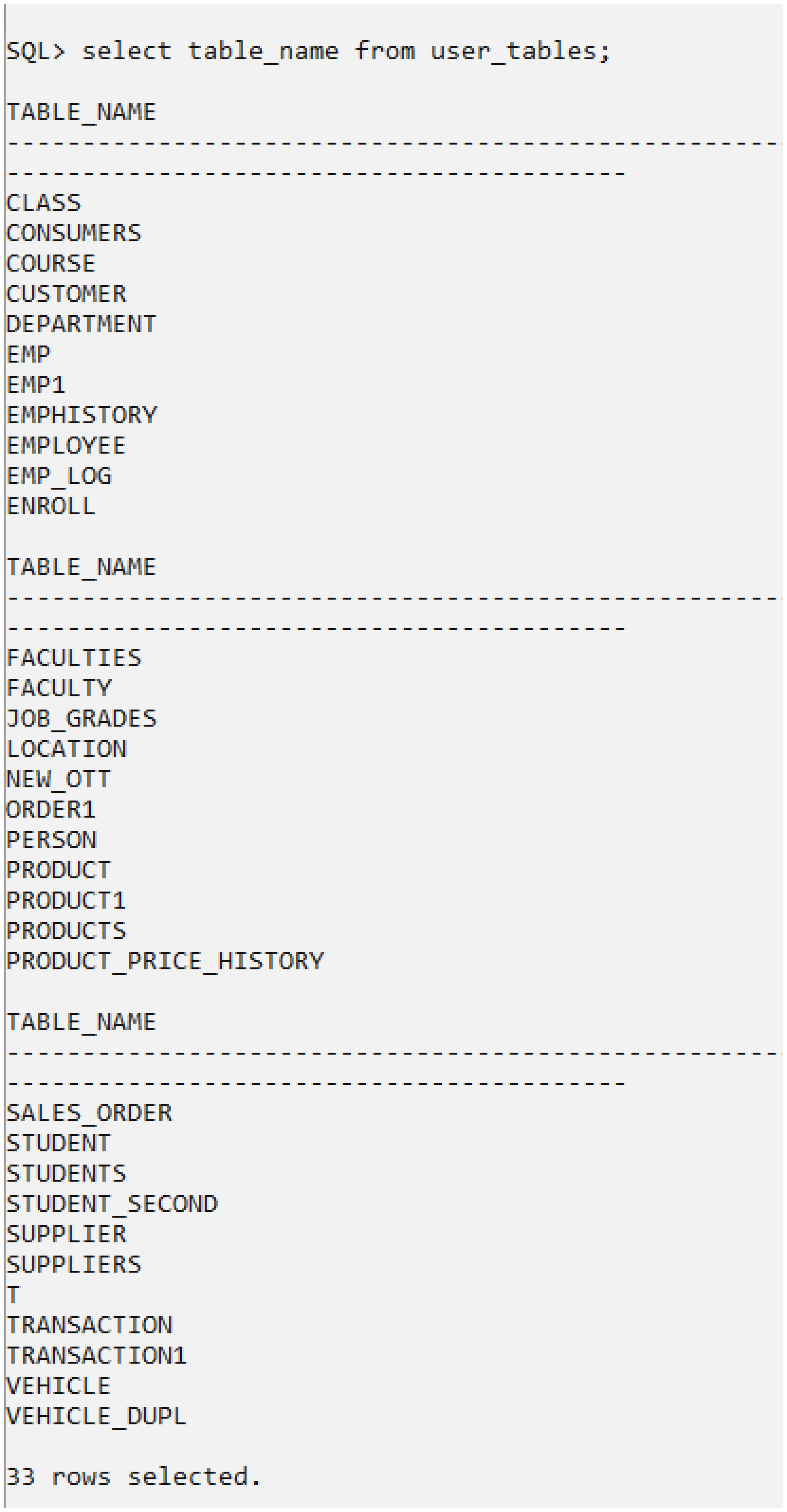
drop sequence order_number-sequence



```
        10           4                  N N

SQL> drop sequence order_number_sequence;

Sequence dropped.
```

To get all tables

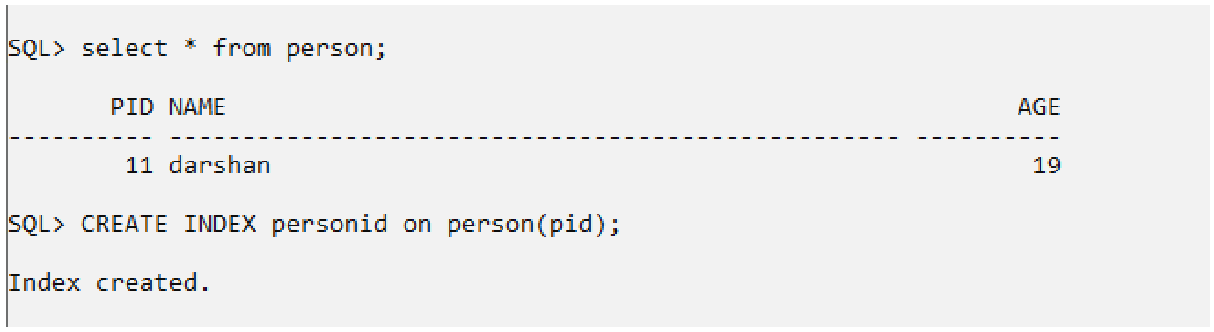
select table_name from user_tables



```
SQL> select table_name from user_tables;

TABLE_NAME
----------------------------------------------------
-----------------------------------------
CLASS
CONSUMERS
COURSE
CUSTOMER
DEPARTMENT
EMP
EMP1
EMPHISTORY
EMPLOYEE
EMP_LOG
ENROLL

TABLE_NAME
----------------------------------------------------
-----------------------------------------
FACULTIES
FACULTY
JOB_GRADES
LOCATION
NEW_OTT
ORDER1
PERSON
PRODUCT
PRODUCT1
PRODUCTS
PRODUCT_PRICE_HISTORY

TABLE_NAME
----------------------------------------------------
-----------------------------------------
SALES_ORDER
STUDENT
STUDENTS
STUDENT_SECOND
SUPPLIER
SUPPLIERS
T
TRANSACTION
TRANSACTION1
VEHICLE
VEHICLE_DUPL

33 rows selected.
```

4. INDEX

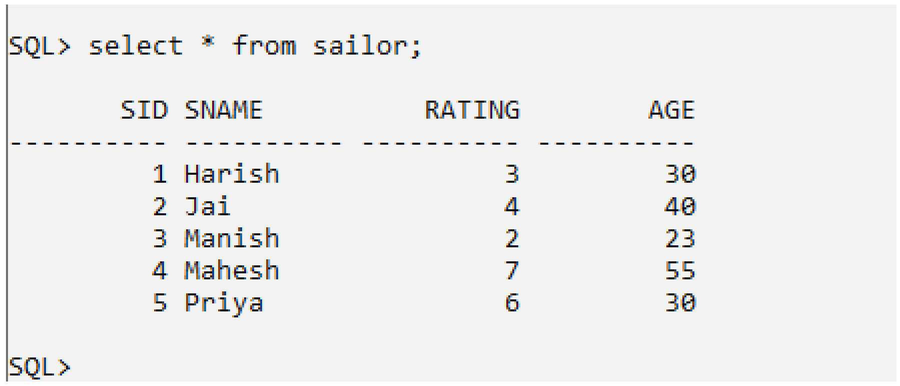
Indexes are important to speed up query processing time
INDEX
CREATE INDEX personid on person(pid);



```
SQL> select * from person;

       PID NAME                                                     AGE
---------- -------------------------------------------------- ----------
        11 darshan                                                   19

SQL> CREATE INDEX personid on person(pid);

Index created.
```

Set operations:
1. UNION
2. INTERSECT
3. MINUS
Creating table .
create table sailor(
sid number, sname varchar(10), rating number, age number(2));

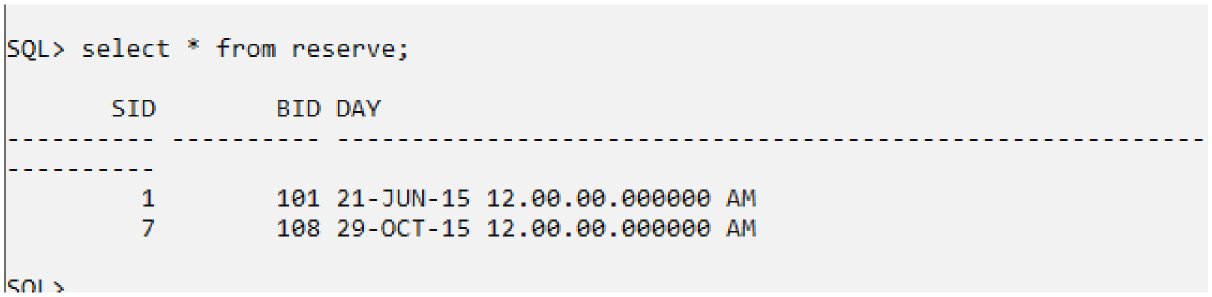
insert into sailor(&sid,'&sname',&rating,&age)
Output:



```
SQL> select * from sailor;

       SID SNAME          RATING        AGE
---------- ---------- ---------- ----------
         1 Harish              3         30
         2 Jai                 4         40
         3 Manish              2         23
         4 Mahesh              7         55
         5 Priya               6         30

SQL>
```

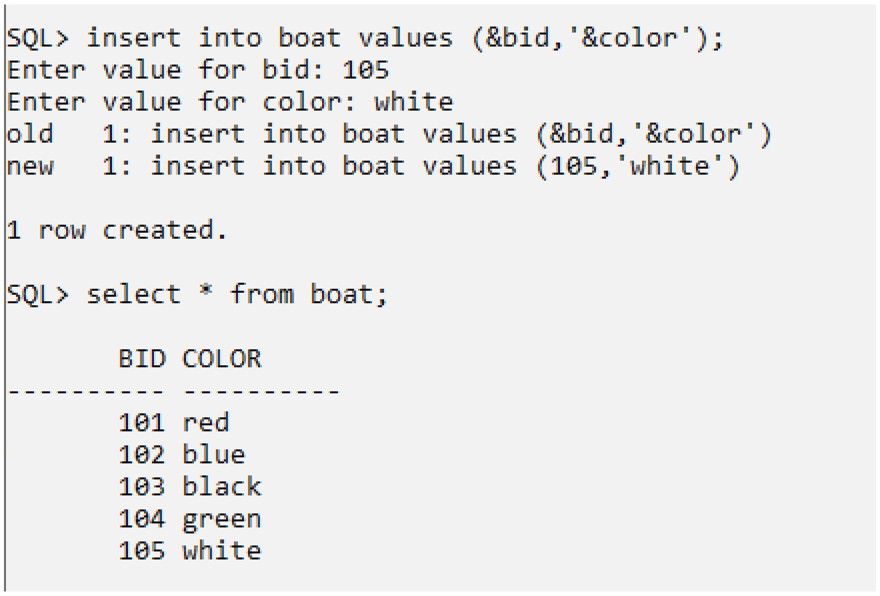
CREATE TABLE RESERVE(
sid number, bid number, day timestamp);

insert into reserve values (&sid,&bid,&day);
Output:



```
SQL> select * from reserve;

       SID        BID DAY
---------- ---------- -------------------------------------------------------------
----------
         1        101 21-JUN-15 12.00.00.000000 AM
         7        108 29-OCT-15 12.00.00.000000 AM

SQL>
```

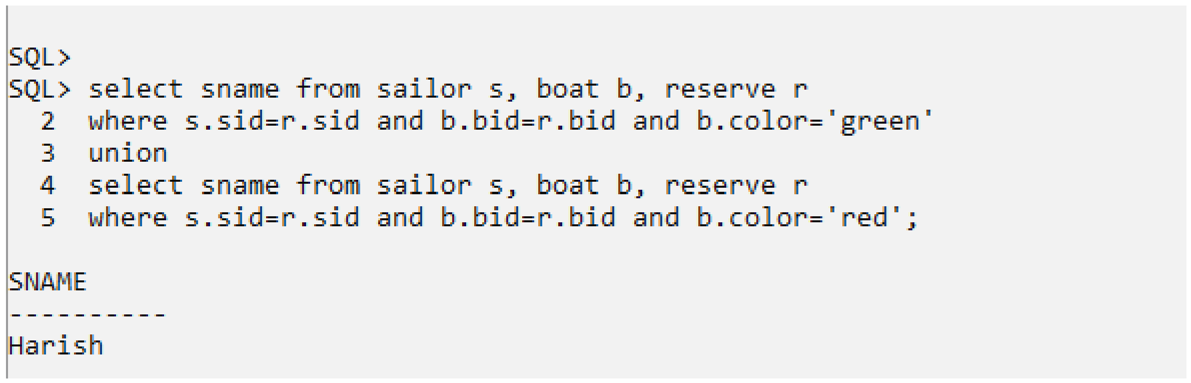
CREATE TABLE boat(bid number, color varchar(10));
insert into boat values (&bid,'&color');
Output:



```
SQL> insert into boat values (&bid,'&color');
Enter value for bid: 105
Enter value for color: white
old   1: insert into boat values (&bid,'&color')
new   1: insert into boat values (105,'white')

1 row created.

SQL> select * from boat;

       BID COLOR
---------- ----------
       101 red
       102 blue
       103 black
       104 green
       105 white
```

***UNION**
Combining two tables
Display name of sailors in boat with colors green and red
Code:
select sname from sailor s, boat b, reserve r
where s.sid=r.sid and b.bid=r.bid and b.color='green'
union
select sname from sailor s, boat b, reserve r
where s.sid=r.sid and b.bid=r.bid and b.color='red';

Output:



```
SQL>
SQL> select sname from sailor s, boat b, reserve r
  2  where s.sid=r.sid and b.bid=r.bid and b.color='green'
  3  union
  4  select sname from sailor s, boat b, reserve r
  5  where s.sid=r.sid and b.bid=r.bid and b.color='red';

SNAME
----------
Harish
```

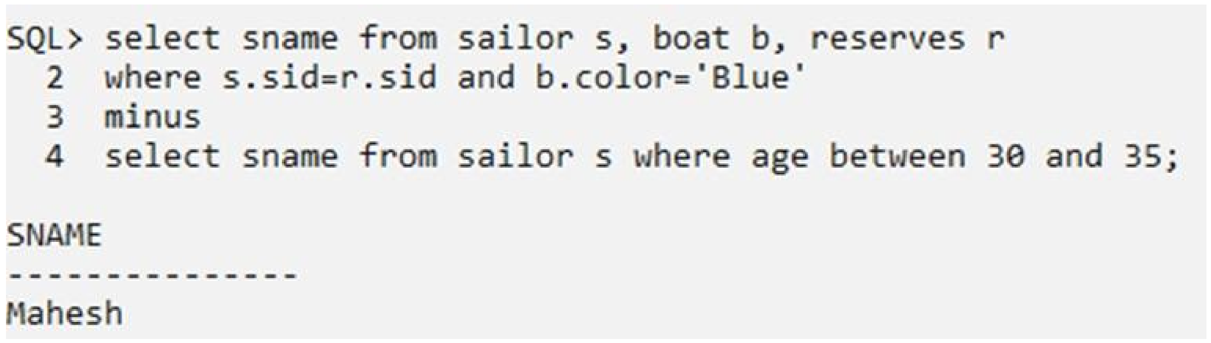
Intersect

(Used to combine two select statements .Returns the records which are common)

Display the sailor name in boat with color BLUE and age between 30 and 35.

select sname from sailor s, boat b, reserve r
where s.sid=r.sid and b.color='blue'
intersect
select sname from sailor where age between 30 and 35;

***MINUS**
Display sailors name in boat with color blue and age not between 30 and 35.
Code:
Select snake from sailor s, boat b , reserve r
Where s.sid=r.sid and b.color=’blue’
Minus
Select sanme from sailor s where age between 30 and 35;
Output:



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

SNAME
---------------
Mahesh
```

PRACTICAL-10 Pl/SQL PROGRAMMING

1. Variables & Identifiers

Syntax

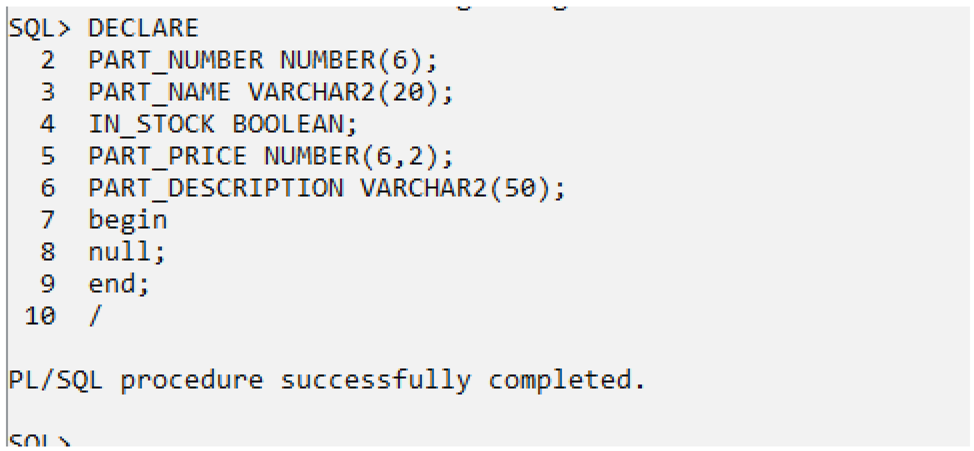
variable_name [CONSTANT] datatype [NOT NULL] [:= |
DEFAULT initial_value]

Description:

PL/SQL variables must be declared in the declaration section or in a package as a global variable. When you declare a variable, PL/SQL allocates memory for the variable's value and the storage location is identified by the variable name.

Code:

```
Declare
part_number NUMBER(6); –SQL data type
part_name VARCHAR2(20); –SQL dats atype
in_stock BOOLEAN; –PL/SQL- only data type
part_price NUMBER(6,2); –SQL data type
part_(description VARCHAR2(50);--SQL data type
BEGIN
        NULL;
END;
```



```
SQL> DECLARE
PART_NUMBER NUMBER(6);
PART_NAME VARCHAR2(20);
IN_STOCK BOOLEAN;
PART_PRICE NUMBER(6,2);
PART_DESCRIPTION VARCHAR2(50);
begin
null;
end;
/

PL/SQL procedure successfully completed.

SQL>
```

*** variable as constant**

declare

credit constant real :=5000.00;

days_year constant integer :=366;

val constant boolean :=false;

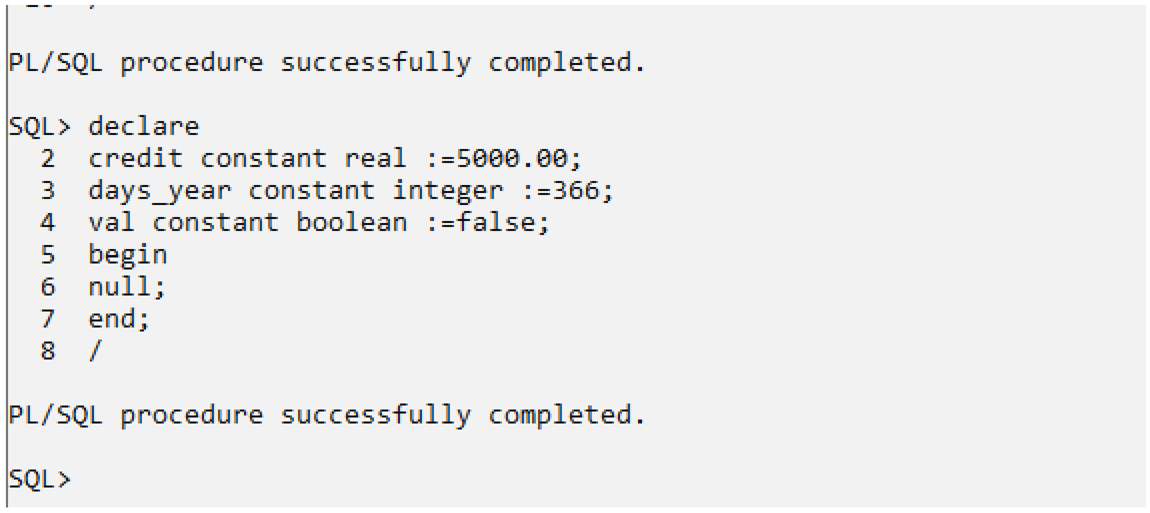
begin

null;

end;

/

Output:



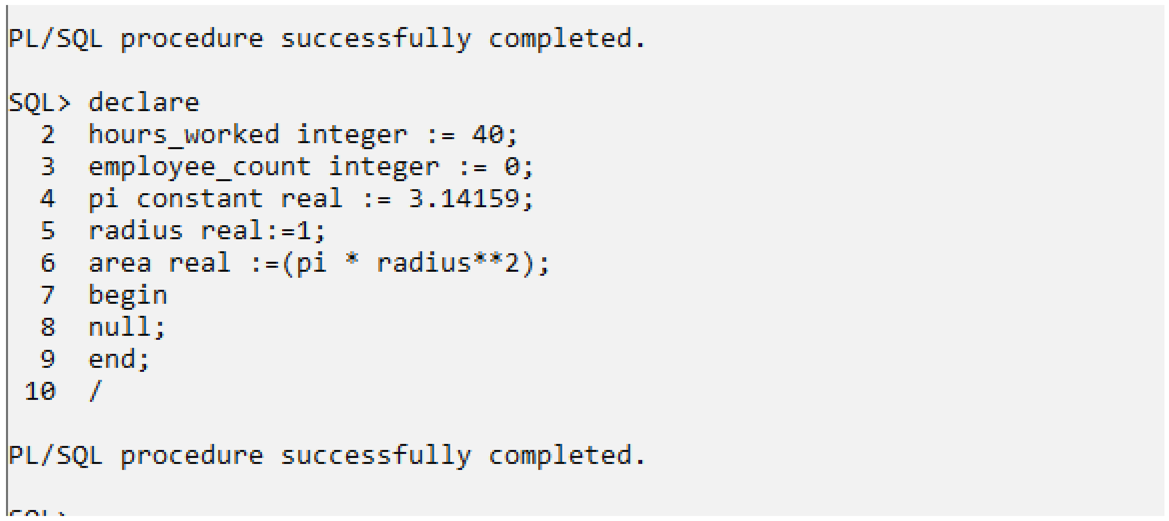
```
PL/SQL procedure successfully completed.

SQL> declare
credit constant real :=5000.00;
days_year constant integer :=366;
val constant boolean :=false;
begin
null;
end;
/

PL/SQL procedure successfully completed.

SQL>
```

***Assigning Values**

declare

hours_worked integer := 40;
employee_count integer := 0;
pi constant real := 3.14159;
radius real:=1;
area real :=(pi * radius**2);
begin
null;
end;
/

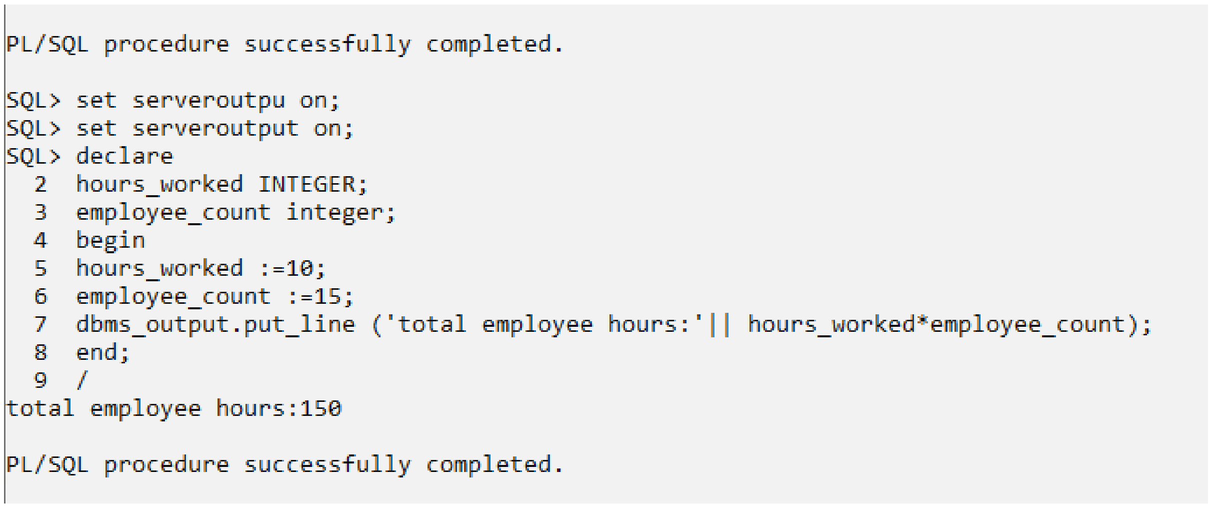
Output:



```
PL/SQL procedure successfully completed.

SQL> declare
  2  hours_worked integer := 40;
  3  employee_count integer := 0;
  4  pi constant real := 3.14159;
  5  radius real:=1;
  6  area real :=(pi * radius**2);
  7  begin
  8  null;
  9  end;
 10  /

PL/SQL procedure successfully completed.
```

***Assigning values to variables Displaying Output**

DBMS_OUTPUT.PUT_LINE
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;
/



```
PL/SQL procedure successfully completed.

SQL> set serveroutpu on;
SQL> set serveroutput on;
SQL> declare
  2  hours_worked INTEGER;
  3  employee_count integer;
  4  begin
  5  hours_worked :=10;
  6  employee_count :=15;
  7  dbms_output.put_line ('total employee hours:'|| hours_worked*employee_count);
  8  end;
  9  /
total employee hours:150

PL/SQL procedure successfully completed.
```

2. Comment

Syntax:

a) —comment you want to include

b) /* comment you want to

include*/ Description

Comments let you include arbitrary text within your code to explain what the code does. You can also disable obsolete or unfinished pieces of code by turning them into comments.

3. Pl/sql block Structure

1. PL/SQL block structure

Syntax:

```
DECLARE
   <declarations
section>
BEGIN
  <executable
command(s)>
EXCEPTION
   <exception
handling> END;
```

Description:

A block is a unit of code that provides execution and scoping boundaries for variable declarations and exception handling. PL/SQL allows you to create anonymous blocks (blocks of code that have no name) and named blocks, which may be packages, procedures, functions, triggers, or object types

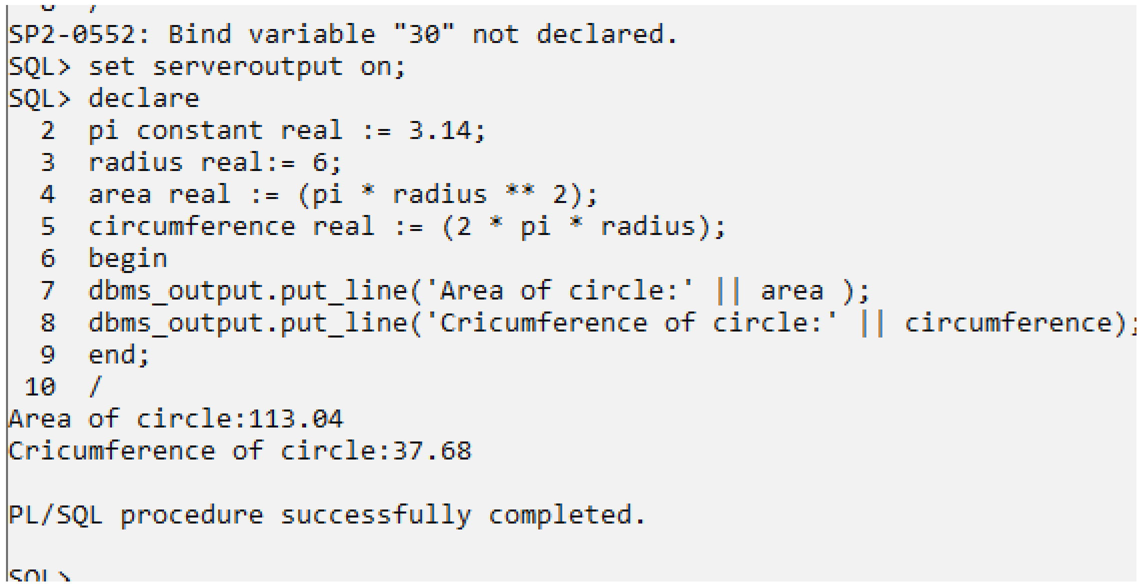
find the area and circumference of a circle
average of three marks
square of a number

1.find the area and circumference of a circle
Code:

```
set serveroutput on;
declare
pi constant real := 3.14;
```

```
radius real:= 6;
area real := (pi * radius ** 2);
circumference real := (2 * pi * radius);
begin
dbms_output.put_line('Area of circle:' || area );
dbms_output.put_line('Cricumference of circle:' || circumference);
end;
/
```

Output:

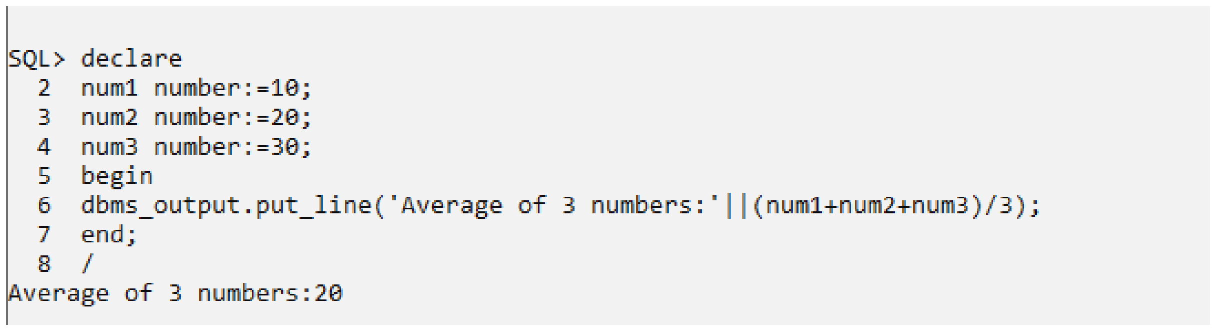


```
SP2-0552: Bind variable "30" not declared.
SQL> set serveroutput on;
SQL> declare
pi constant real := 3.14;
radius real:= 6;
area real := (pi * radius ** 2);
circumference real := (2 * pi * radius);
begin
dbms_output.put_line('Area of circle:' || area );
dbms_output.put_line('Cricumference of circle:' || circumference);
end;
/
Area of circle:113.04
Cricumference of circle:37.68

PL/SQL procedure successfully completed.

SQL>
```

b)Average of marks

```
declare
num1 number:=10;
num2 number:=20;
num3 number:=30;
begin
dbms_output.put_line('Average of 3 numbers:'||(num1+num2+num3)/3);
end;
/
```

Output:

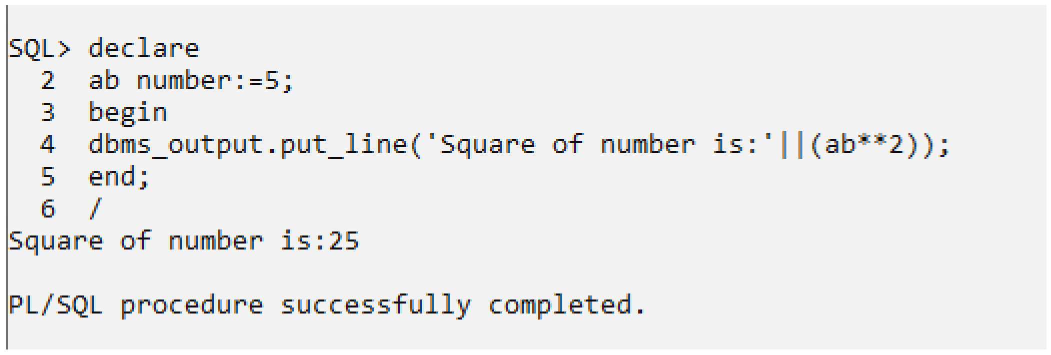


```
SQL> declare
num1 number:=10;
num2 number:=20;
num3 number:=30;
begin
dbms_output.put_line('Average of 3 numbers:'||(num1+num2+num3)/3);
end;
/
Average of 3 numbers:20
```

c)square of number

```
declare
ab number:=5;
begin
dbms_output.put_line('Square of number is:'||(ab**2));
end;
/
```

Output:



```
SQL> declare
ab number:=5;
begin
dbms_output.put_line('Square of number is:'||(ab**2));
end;
/
Square of number is:25

PL/SQL procedure successfully completed.
```

Practical 11. PL/SQL Control Statements

***Conditional Statements**
If
If Else

***sequential control statements:**
goto which goes to a specified statement
null which does nothing.

Syntax for if condition:

```
if condition
then
        statements
end if;
```

Description:

The condition is a Boolean expression that always evaluates to TRUE, FALSE, or NULL.If the condition evaluates to TRUE, the statements after the THEN execute. Otherwise, the IF statement does nothing

syntax for if-else elsif statement:

```
if condition
then
statements
else
else_statements
end if;
```

Syntax for if condition:

```
if condition
then
        statements
```

```
end if;
```

example:

```
if new_balance<minimum_balance THEN
overdrawn := True;
else
overdrawn := false;
end if;
```

syntax for if-else elsif statement:

```
if condition
then
        statements
else
        else_statements
end if;
```

example;

```
if sales>(quota + 200)
then
bonus :=(sales - quota)/4;
else
bonus := 50;
end if;
```

Syntax of if then elsif statement

```
if condition_1
then
        statements_1
elsif condition_2
        then
           statements_2
        (ELSIF condition_3
        THEN
```

```
            statements_3]...
         [ELSE else_statements]
END IF;
```

example:

```
if sales>50000 then
bonus :=1500;
elsif sales>35000 then
bonus := 500;
else
bonus := 100;
end if;
```

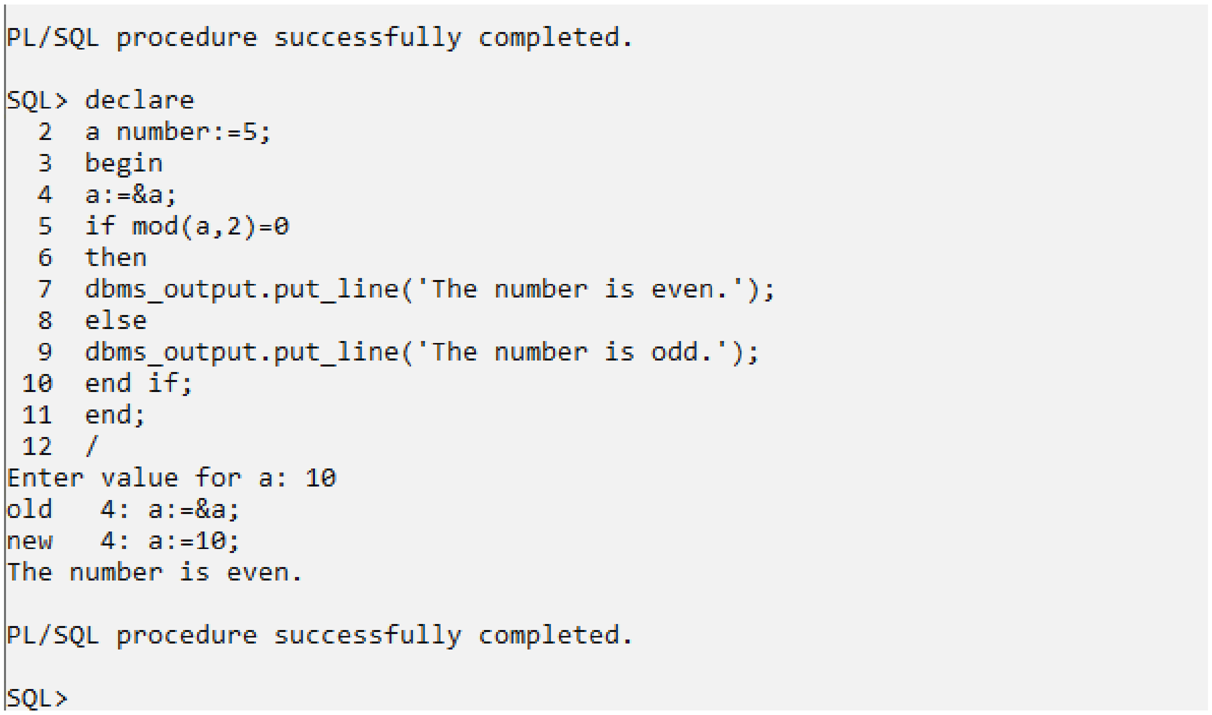
CODE for if then else:

sum 1 Pl/sql block to check whether a number is even or odd.

Code:

```
declare
a number:=5;
begin
a:=&a;
if mod(a,2)=0
then
dbms_output.put_line('The number is even.');
else
dbms_output.put_line('The number is odd.');
end if;
end;
/
```

OUTPUT:



```
PL/SQL procedure successfully completed.

SQL> declare
  2  a number:=5;
  3  begin
  4  a:=&a;
  5  if mod(a,2)=0
  6  then
  7  dbms_output.put_line('The number is even.');
  8  else
  9  dbms_output.put_line('The number is odd.');
 10  end if;
 11  end;
 12  /
Enter value for a: 10
old   4: a:=&a;
new   4: a:=10;
The number is even.

PL/SQL procedure successfully completed.

SQL>
```

2.. Compound if statements:

Syntax:

```
IF condition_1 THEN
  statements_1
ELSIF condition_2
  THEN statements_2
 [ ELSIF condition_3
    THEN
    statements_3

]

...
```

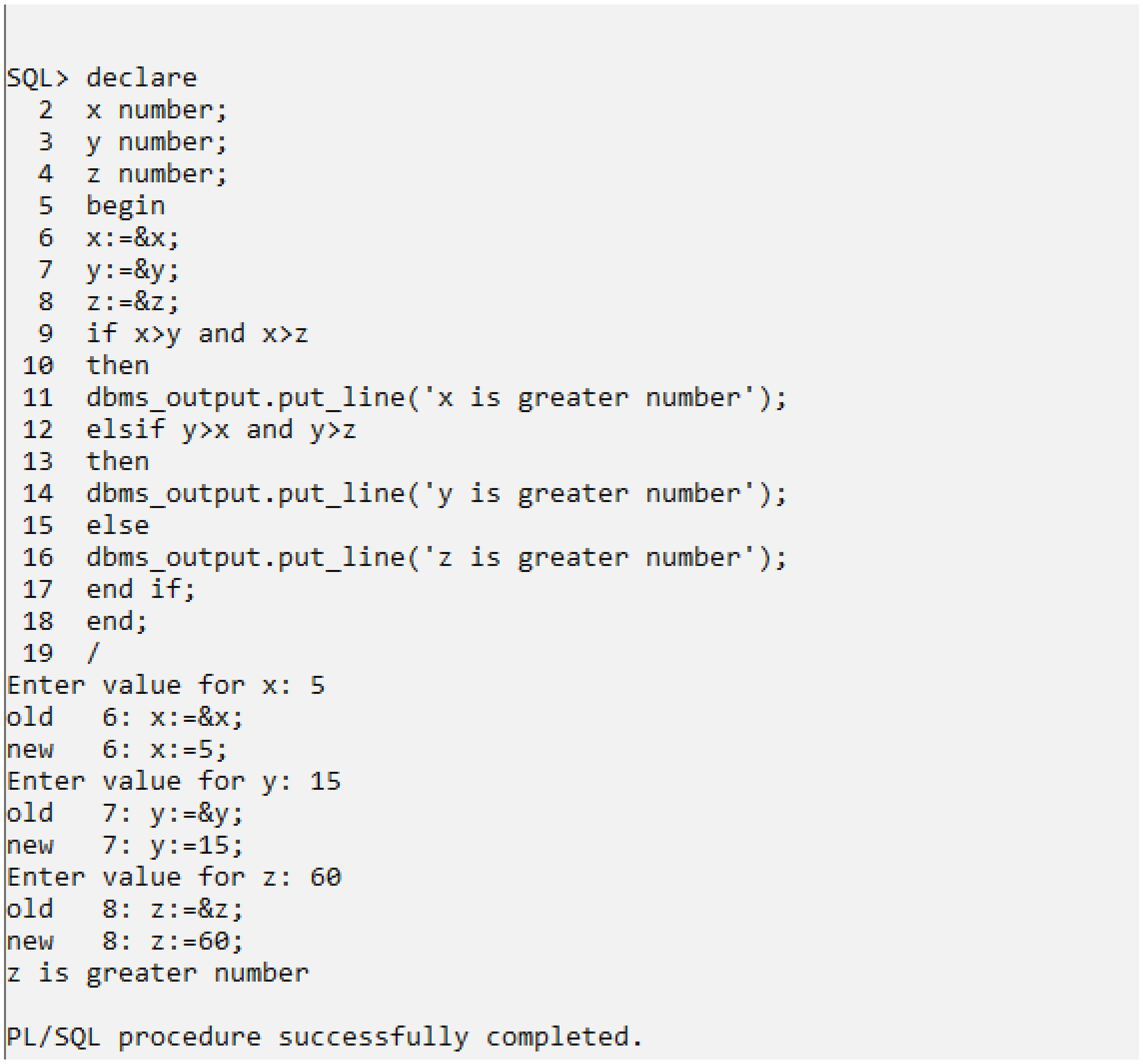
```
[ ELSE
   else_statements
]
END IF;
```

2. Write a code for pl/sql block to find the largest of three numbers;

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  /
Enter value for x: 5
old   6: x:=&x;
new   6: x:=5;
Enter value for y: 15
old   7: y:=&y;
new   7: y:=15;
Enter value for z: 60
old   8: z:=&z;
new   8: z:=60;
z is greater number

PL/SQL procedure successfully completed.
```

3. IF THEN ELSE statement:

Syntax:

IF condition THEN statements;

ELSE

else_st

atements;

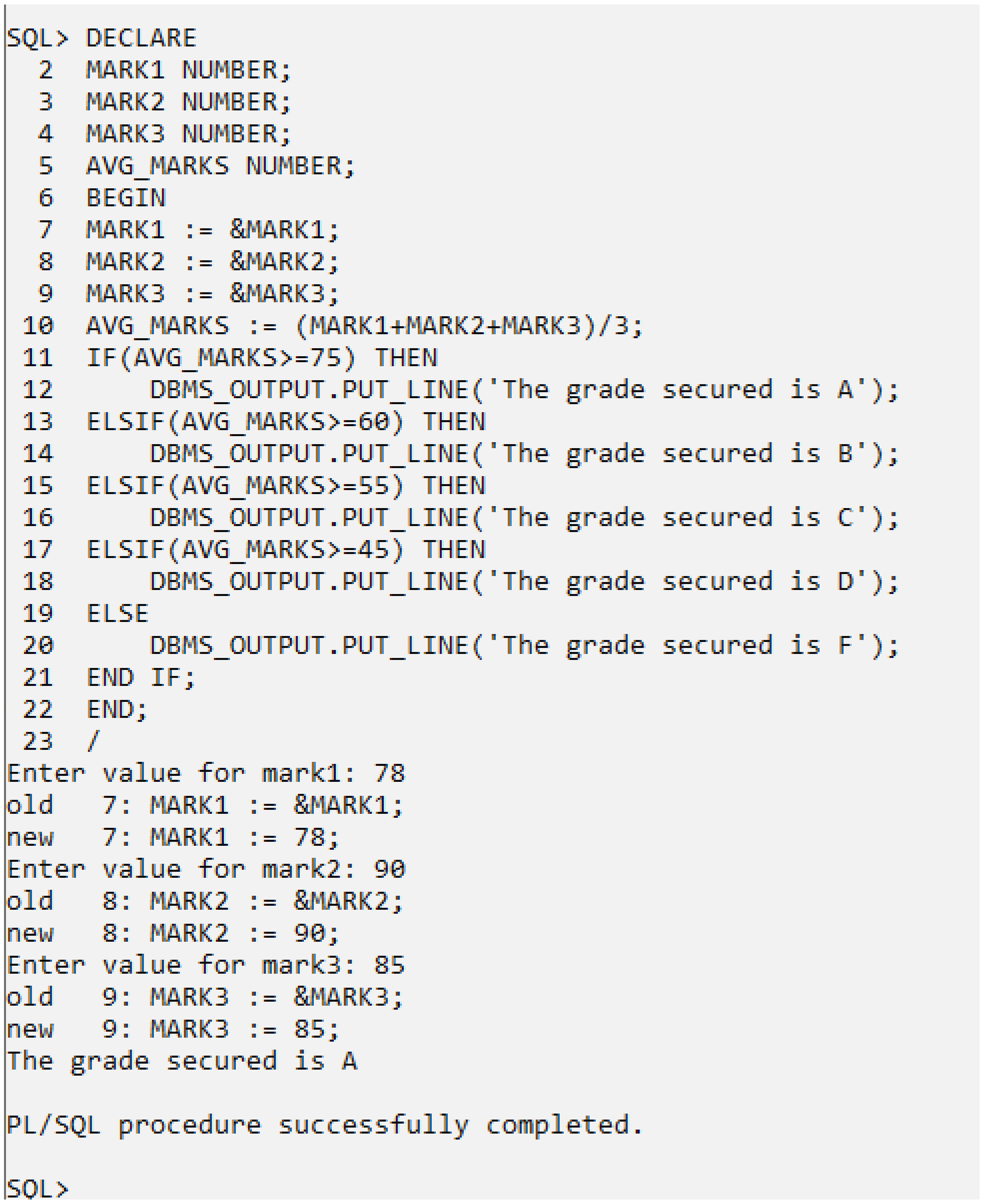
END IF;

3.Write a pl/sql block to accept marks of 3 subjects and find the average marks

and do the grading.
average>=75 grade=A
average>=60 grade=B
average>=55 grade=C
average>=45 grade=D
else grade = F

```
DECLARE
MARK1 NUMBER;
MARK2 NUMBER;
MARK3 NUMBER;
AVG_MARKS NUMBER;
BEGIN
MARK1 := &MARK1;
MARK2 := &MARK2;
MARK3 := &MARK3;
AVG_MARKS := (MARK1+MARK2+MARK3)/3;
IF(AVG_MARKS>=75) THEN
   DBMS_OUTPUT.PUT_LINE('The grade secured is A');
ELSIF(AVG_MARKS>=60) THEN
   DBMS_OUTPUT.PUT_LINE('The grade secured is B');
ELSIF(AVG_MARKS>=55) THEN
   DBMS_OUTPUT.PUT_LINE('The grade secured is C');
ELSIF(AVG_MARKS>=45) THEN
   DBMS_OUTPUT.PUT_LINE('The grade secured is D');
ELSE
   DBMS_OUTPUT.PUT_LINE('The grade secured is F');
END IF;
END;
/
```

Output:



```
SQL> DECLARE
  2  MARK1 NUMBER;
  3  MARK2 NUMBER;
  4  MARK3 NUMBER;
  5  AVG_MARKS NUMBER;
  6  BEGIN
  7  MARK1 := &MARK1;
  8  MARK2 := &MARK2;
  9  MARK3 := &MARK3;
 10  AVG_MARKS := (MARK1+MARK2+MARK3)/3;
 11  IF(AVG_MARKS>=75) THEN
 12      DBMS_OUTPUT.PUT_LINE('The grade secured is A');
 13  ELSIF(AVG_MARKS>=60) THEN
 14      DBMS_OUTPUT.PUT_LINE('The grade secured is B');
 15  ELSIF(AVG_MARKS>=55) THEN
 16      DBMS_OUTPUT.PUT_LINE('The grade secured is C');
 17  ELSIF(AVG_MARKS>=45) THEN
 18      DBMS_OUTPUT.PUT_LINE('The grade secured is D');
 19  ELSE
 20      DBMS_OUTPUT.PUT_LINE('The grade secured is F');
 21  END IF;
 22  END;
 23  /
Enter value for mark1: 78
old   7: MARK1 := &MARK1;
new   7: MARK1 := 78;
Enter value for mark2: 90
old   8: MARK2 := &MARK2;
new   8: MARK2 := 90;
Enter value for mark3: 85
old   9: MARK3 := &MARK3;
new   9: MARK3 := 85;
The grade secured is A

PL/SQL procedure successfully completed.

SQL>
```

Practical 12. Loops

***the loop statements**
loop
for loop
while loop

***1. Basic Loop**
Loop
Syntax:

```
program satements
if condition then
   exit;
end if;
[additional program statements]
end loop;
```

Description:

Basic loop structure encloses sequence of statements in between the LOOP and END LOOP statements. With each iteration, the sequence of statements is executed and then control resumes at the top of the loop.

Basic Loop
(do While loop type)

```
loop
   program statements
   exit when condition;
 end loop;
```

While Loop

```
WHILE condition LOOP
PROGRAM STATEMENTS
End loop;
```

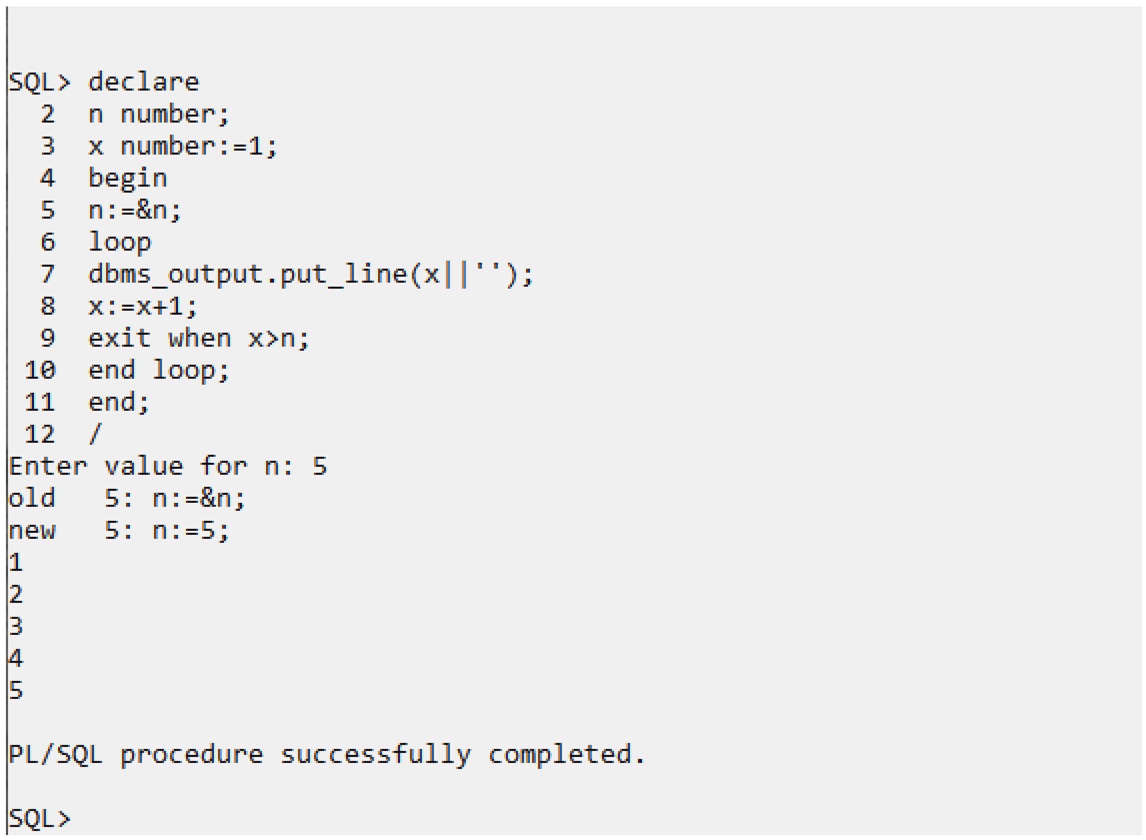
The Numeric FOR LOOP
syntax

```
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;
```

plsql block to display numbers from 1 to 10 using simple loop, while loop and for loop
using Simple loop

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

Output:



```
SQL> declare
  2  n number;
  3  x number:=1;
  4  begin
  5  n:=&n;
  6  loop
  7  dbms_output.put_line(x||'');
  8  x:=x+1;
  9  exit when x>n;
 10  end loop;
 11  end;
 12  /
Enter value for n: 5
old   5: n:=&n;
new   5: n:=5;
1
2
3
4
5

PL/SQL procedure successfully completed.

SQL>
```

2. Using while loop
Syntax:

WHILE condition LOOP

Program

statements

END LOOP;

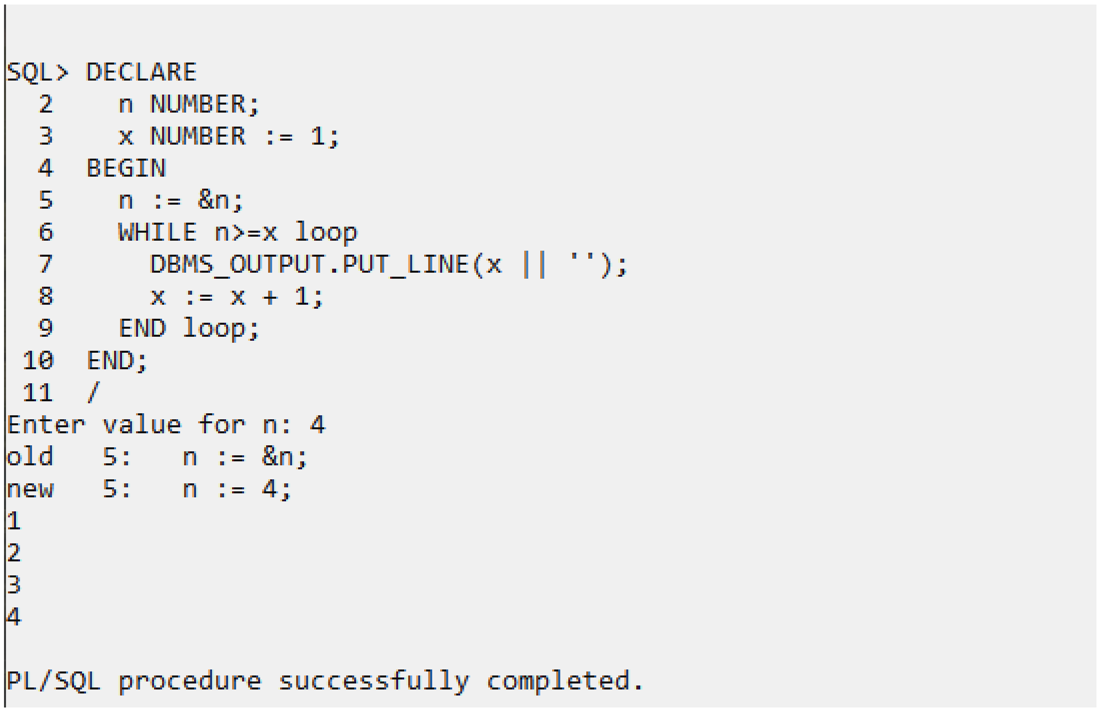
<u>Description:</u>

Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.

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> 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: 4
old   5:   n := &n;
new   5:   n := 4;
1
2
3
4

PL/SQL procedure successfully completed.
```

3. **Using for loop**

<u>Syntax:</u>

FOR counter_variable IN

start_value….end value LOOP

Program statement

END LOOP

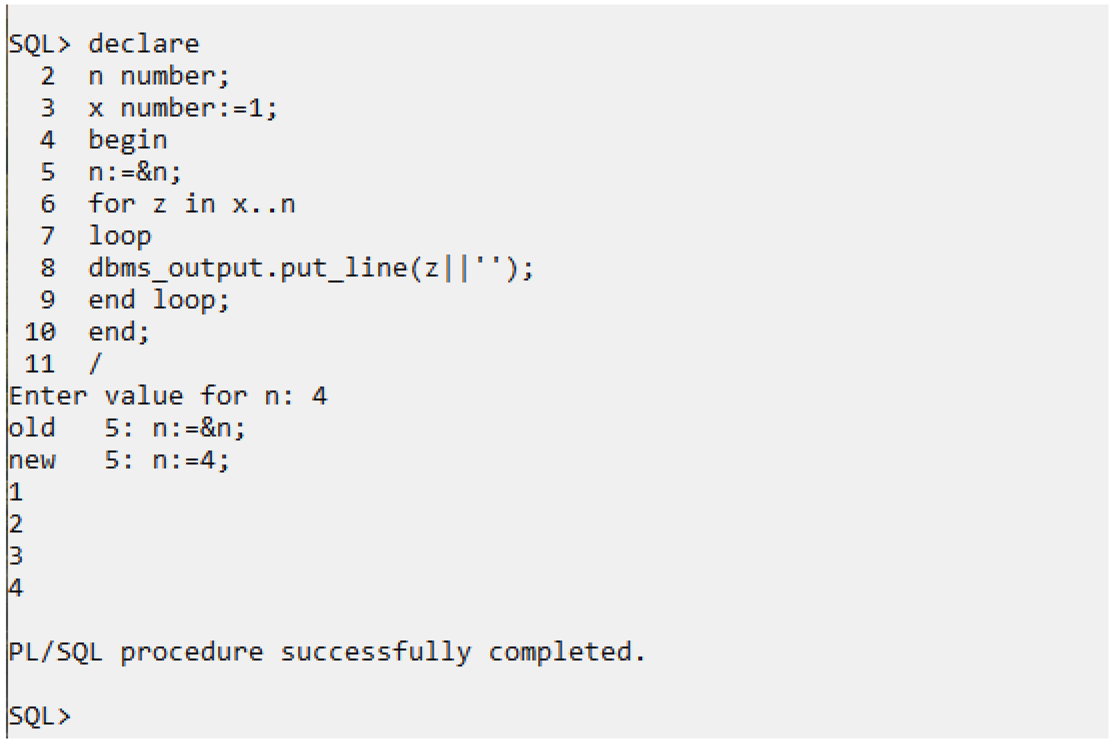
<u>Description:</u>

Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable.

Code:

```
declare
n number;
x number:=1;
begin
n:=&n;
for z in x..n
loop
dbms_output.put_line(z||'');
end loop;
end;
/
```

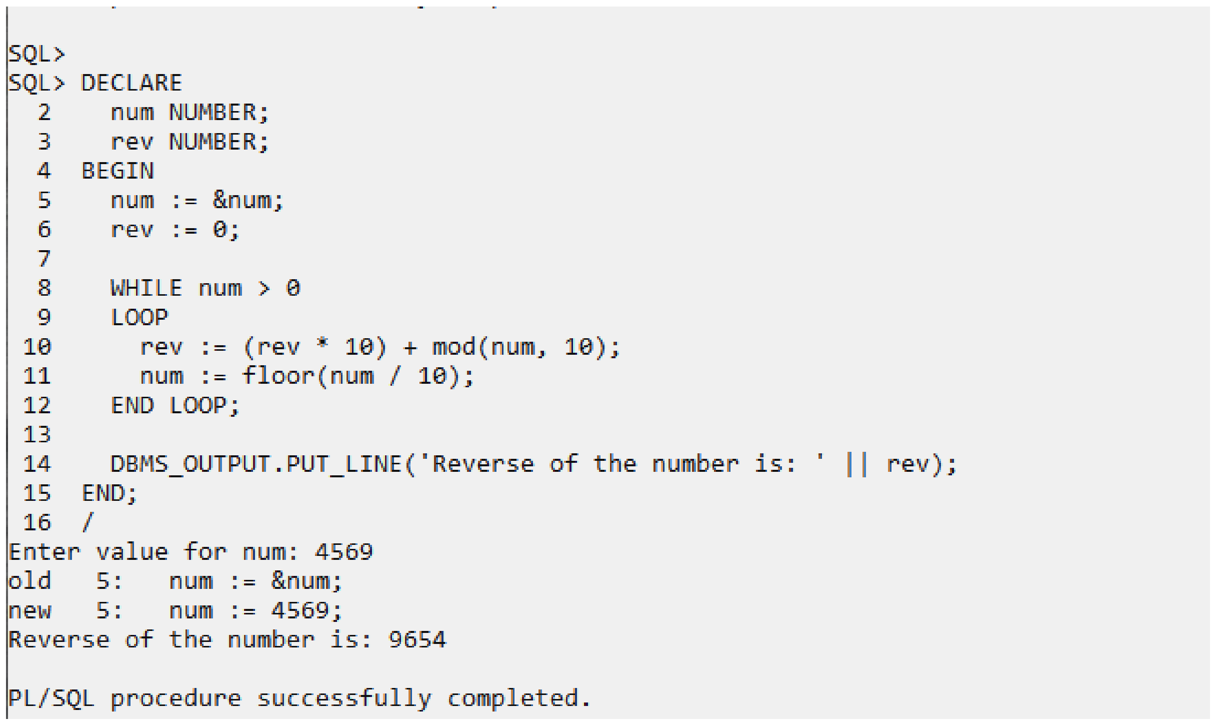
Output:



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

PL/SQL procedure successfully completed.

SQL>
```

Write a plsql block to reverse the number

```
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;
/
```

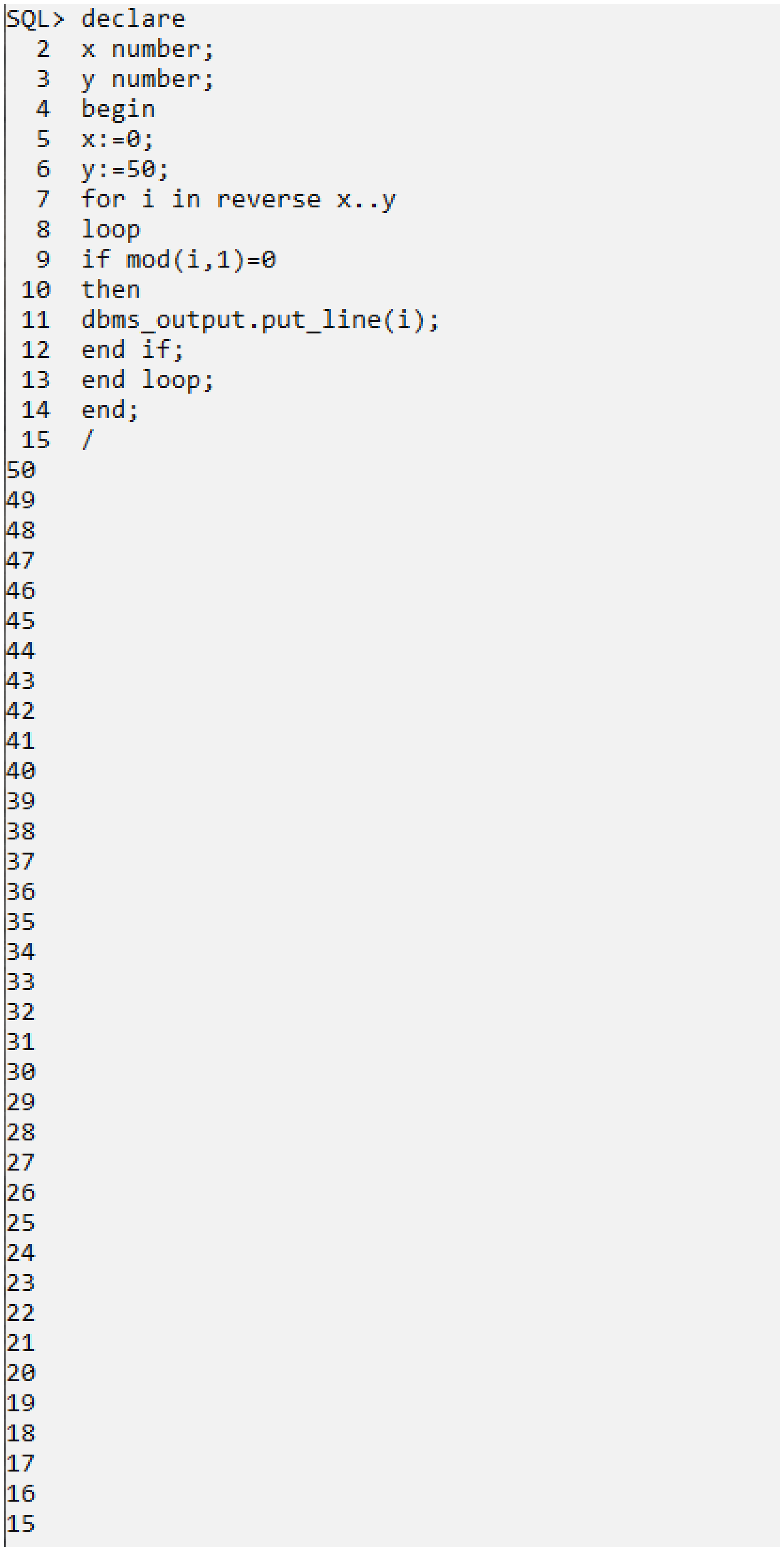


```
SQL>
SQL> DECLARE
  2    num NUMBER;
  3    rev NUMBER;
  4  BEGIN
  5    num := &num;
  6    rev := 0;
  7
  8    WHILE num > 0
  9    LOOP
 10      rev := (rev * 10) + mod(num, 10);
 11      num := floor(num / 10);
 12    END LOOP;
 13
 14    DBMS_OUTPUT.PUT_LINE('Reverse of the number is: ' || rev);
 15  END;
 16  /
Enter value for num: 4569
old   5:   num := &num;
new   5:   num := 4569;
Reverse of the number is: 9654

PL/SQL procedure successfully completed.
```

to find the even even numbers between 1 and 50 in reverse order

```
declare
x number;
y number;
begin
x:=0;
y:=50;
for i in reverse x..y
loop
if mod(i,1)=0
then
dbms_output.put_line(i);
end if;
end loop;
end;
/
```

Output:



```
SQL> declare
x number;
y number;
begin
x:=0;
y:=50;
for i in reverse x..y
loop
if mod(i,1)=0
then
dbms_output.put_line(i);
end if;
end loop;
end;
/
50
49
48
47
46
45
44
43
42
41
40
39
38
37
36
35
34
33
32
31
30
29
28
27
26
25
24
23
22
21
20
19
18
17
16
15
```

Practical No-13 DML Operations Using pl/sql

insert
update
delete
merge

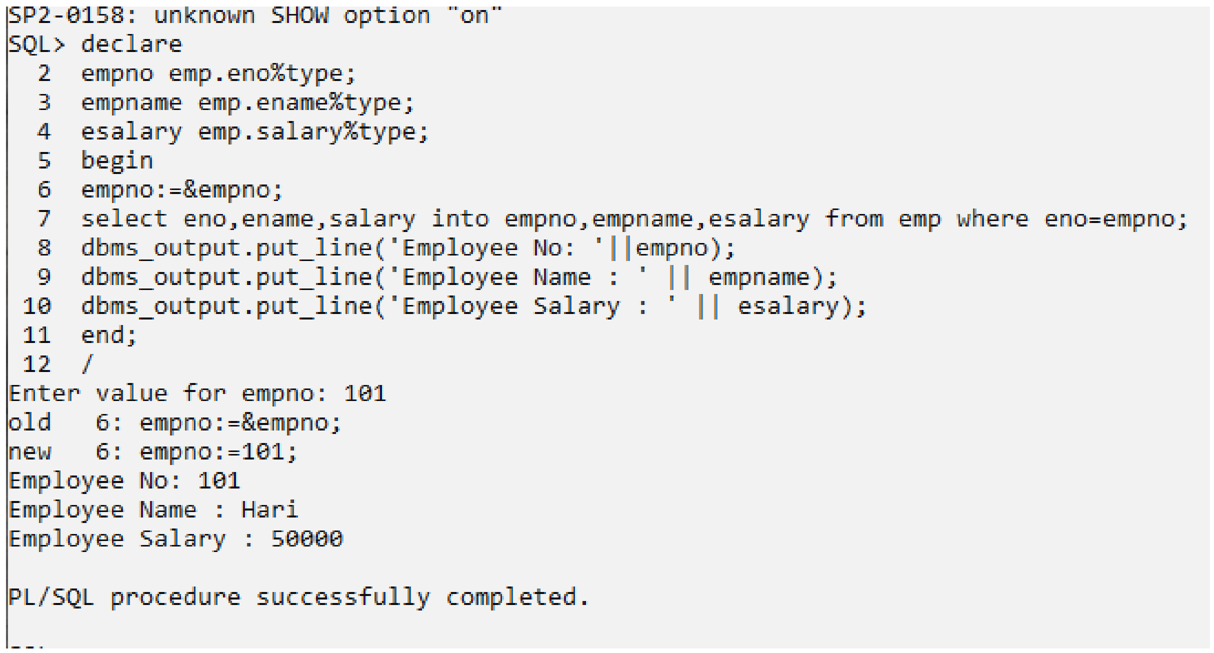
1. **Insert**

Syntax:

```
 INSERT INTO `table_name`(column_1,column_2,...)
VALUES(value_1,value_2,...);
```

```
declare
empno emp.eno%type;
empname emp.ename%type;
esalary emp.salary%type;
begin
empno:=&empno;
select eno,ename,esalary into empno,empname,salary from emp where
eno=empno;
dbms_output.put_line('Employee No: '||empno);
dbms_output.put_line('Employee Name : ' || empname);
dbms_output.put_line('Employee Salary : ' || esalary);
end;
/
```

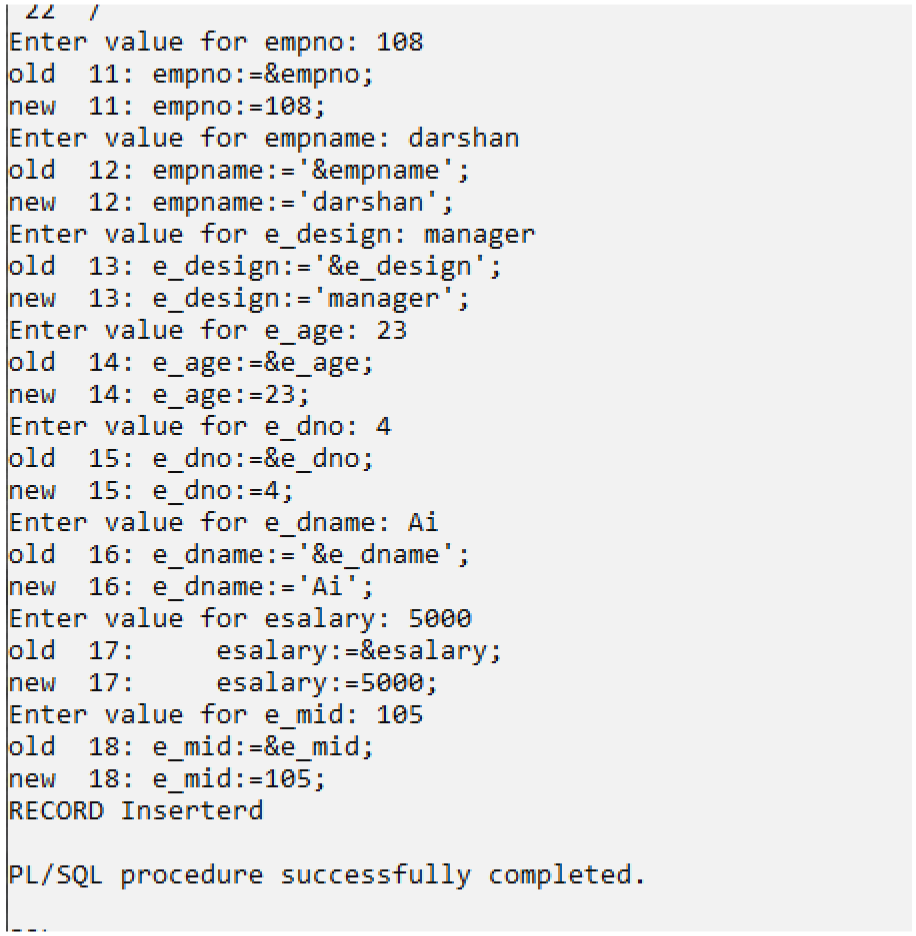
Output:



```
SP2-0158: unknown SHOW option "on"
SQL> declare
  2  empno emp.eno%type;
  3  empname emp.ename%type;
  4  esalary emp.salary%type;
  5  begin
  6  empno:=&empno;
  7  select eno,ename,salary into empno,empname,esalary from emp where eno=empno;
  8  dbms_output.put_line('Employee No: '||empno);
  9  dbms_output.put_line('Employee Name : ' || empname);
 10  dbms_output.put_line('Employee Salary : ' || esalary);
 11  end;
 12  /
Enter value for empno: 101
old   6: empno:=&empno;
new   6: empno:=101;
Employee No: 101
Employee Name : Hari
Employee Salary : 50000

PL/SQL procedure successfully completed.
```

```
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.managerid%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);
        dbms_output.put_line('RECORD Inserterd');
```

end;
/
Output:



```
 22  /
Enter value for empno: 108
old  11: empno:=&empno;
new  11: empno:=108;
Enter value for empname: darshan
old  12: empname:='&empname';
new  12: empname:='darshan';
Enter value for e_design: manager
old  13: e_design:='&e_design';
new  13: e_design:='manager';
Enter value for e_age: 23
old  14: e_age:=&e_age;
new  14: e_age:=23;
Enter value for e_dno: 4
old  15: e_dno:=&e_dno;
new  15: e_dno:=4;
Enter value for e_dname: Ai
old  16: e_dname:='&e_dname';
new  16: e_dname:='Ai';
Enter value for esalary: 5000
old  17:     esalary:=&esalary;
new  17:     esalary:=5000;
Enter value for e_mid: 105
old  18: e_mid:=&e_mid;
new  18: e_mid:=105;
RECORD Inserterd

PL/SQL procedure successfully completed.
```

Exercise:

2. Update

<u>Syntax:</u>

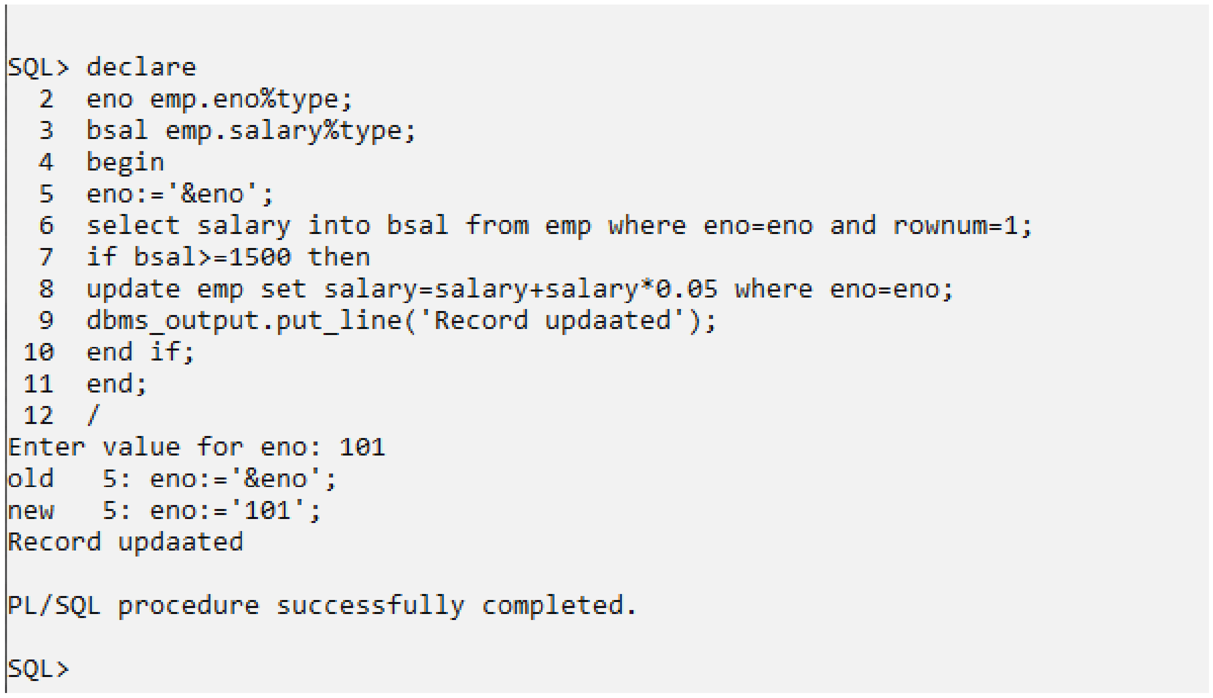
UPDATE `table_name` SET `column_name` = `new_value&#39;
[WHERE condition];

to increase the salary of an employee by 5% if salary>30000

declare

```
        eno emp.eno%type;
        bsal emp.salary%type;
begin
eno:='&eno';
select salary into bsal from emp where eno=eno and rownum=1;
if bsal>=1500 then
        update emp set salary=salary+salary*0.05 where eno=eno;
        dbms_output.put_line('Record updaated');
end if;
end;
/
```

output:



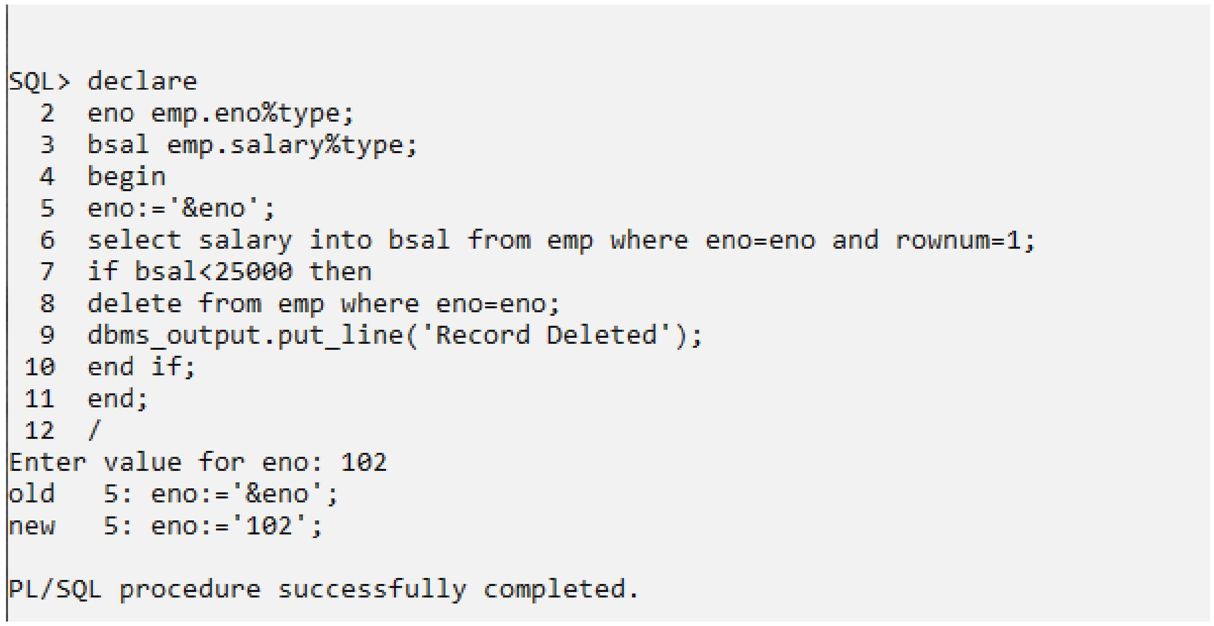
```
SQL> declare
  2  eno emp.eno%type;
  3  bsal emp.salary%type;
  4  begin
  5  eno:='&eno';
  6  select salary into bsal from emp where eno=eno and rownum=1;
  7  if bsal>=1500 then
  8  update emp set salary=salary+salary*0.05 where eno=eno;
  9  dbms_output.put_line('Record updaated');
 10  end if;
 11  end;
 12  /
Enter value for eno: 101
old   5: eno:='&eno';
new   5: eno:='101';
Record updaated

PL/SQL procedure successfully completed.

SQL>
```

Code:

```
declare
        eno emp.eno%type;
        bsal emp.salary%type;
begin
eno:='&eno';
select salary into bsal from emp where eno=eno and rownum=1;
```

```
if bsal<25000 then
        delete from emp where eno=eno;
        dbms_output.put_line('Record Deleted');
end if;
end;
/
```

Output:



```
SQL> declare
  2  eno emp.eno%type;
  3  bsal emp.salary%type;
  4  begin
  5  eno:='&eno';
  6  select salary into bsal from emp where eno=eno and rownum=1;
  7  if bsal<25000 then
  8  delete from emp where eno=eno;
  9  dbms_output.put_line('Record Deleted');
 10  end if;
 11  end;
 12  /
Enter value for eno: 102
old   5: eno:='&eno';
new   5: eno:='102';

PL/SQL procedure successfully completed.
```

***Write pl sql block to retrive customer details when customer id is given at run time**

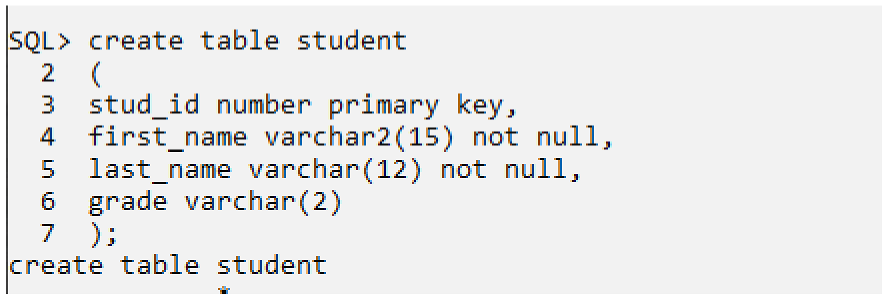
Code:

```
declare
        a_customer_id customer.cust_id%type;
        b_customer_name customer.cust_name%type;
        c_customer_contact customer.contact%type;
begin
        a_customer_id := &customer_id;
select cust_id,cust_name,contact into
a_customer_id,b_customer_name,c_customer_contact from customer where
cust_id=a_customer_id;

dbms_output.put_line('Customer Name:'|| b_customer_name);
```

dbms_output.put_line('Customer contact:'|| c_customer_contact);
end;
/

*Merge the tables using pl/sql
Code:
create table student
(
stud_id number primary key,
first_name varchar2(15) not null,
last_name varchar(12) not null,
grade varchar(2)
);

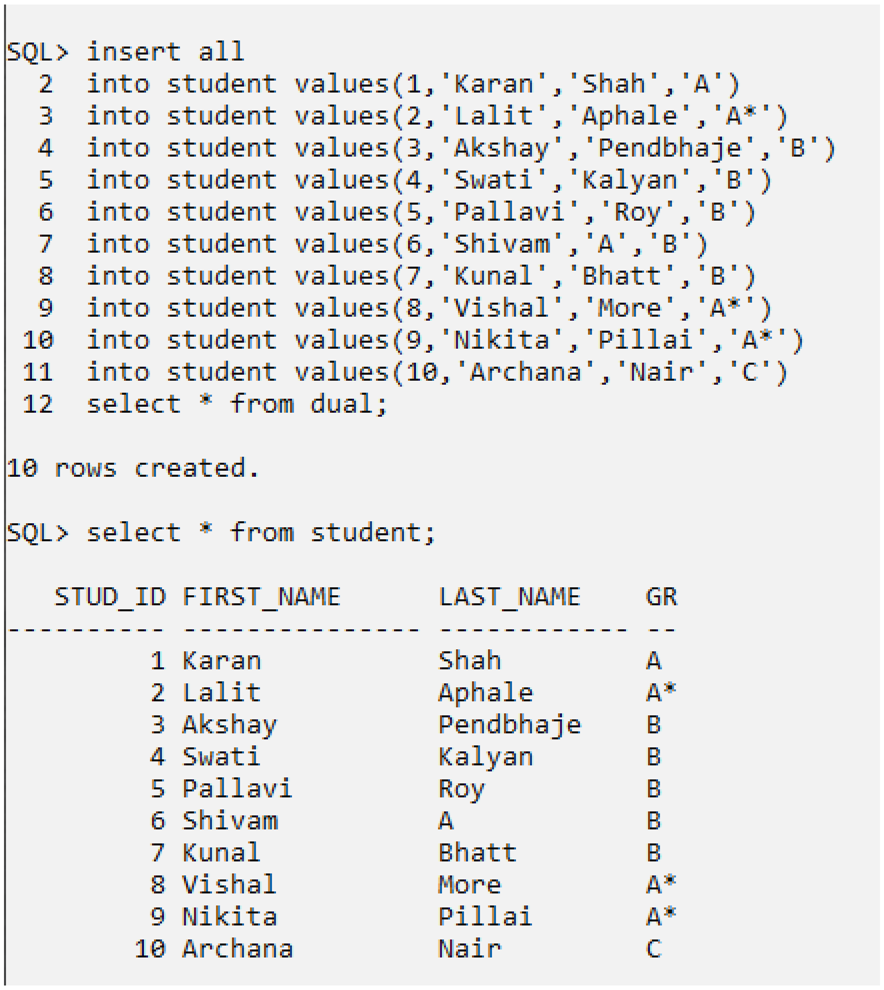
Output:



```
SQL> 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  );
create table student
```

***Inserting values in students**
Code:
insert all
into student values(1,'Karan','Shah','A');
into student values(2,'lalit','Aphale','A');
into student values(3,'Akshay','Pendbhaje','B');
into student values(4,'Swati','Kalyan','B');
into student values(5,'Pallavi','Roy','B');
into student values(6,'Shivam','A','B');
into student values(7,'Kunal','Bhatt','A');
into student values(8,'Vishal','More','A');
into student values(9,'Nikita','Pillai','A');

into student values(10,'Archana','Nair','C');
select * from dual;
Output:



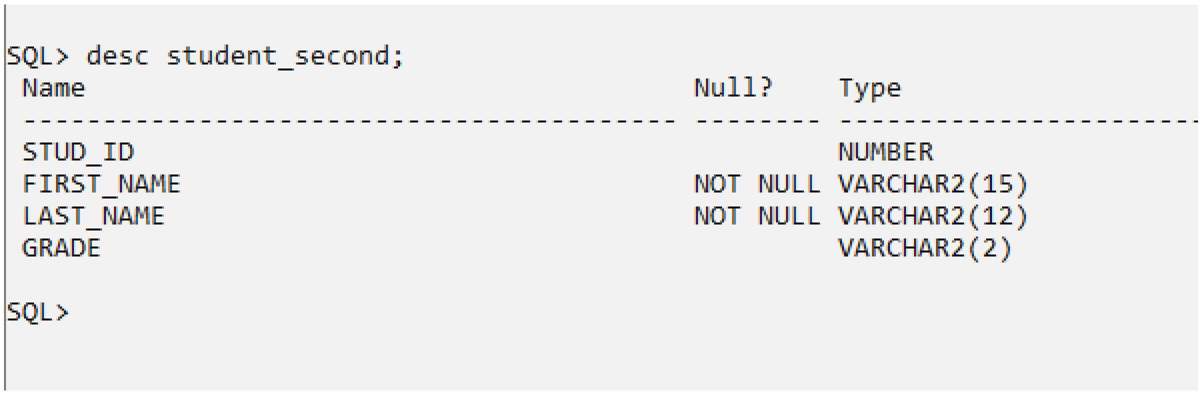
```
SQL> insert all
  2  into student values(1,'Karan','Shah','A')
  3  into student values(2,'Lalit','Aphale','A*')
  4  into student values(3,'Akshay','Pendbhaje','B')
  5  into student values(4,'Swati','Kalyan','B')
  6  into student values(5,'Pallavi','Roy','B')
  7  into student values(6,'Shivam','A','B')
  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.

SQL> select * from student;

   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
```

Creating table student_second

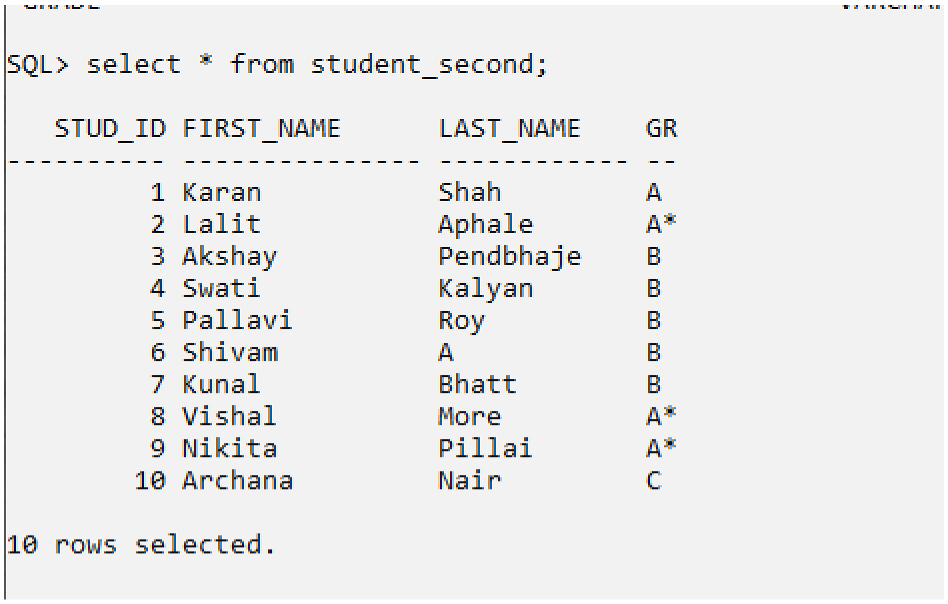
code

create table student_second as select * from student;

Output:



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

SQL>
```



```
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.
```

3. Delete

Syntax:

DELETE FROM table_name

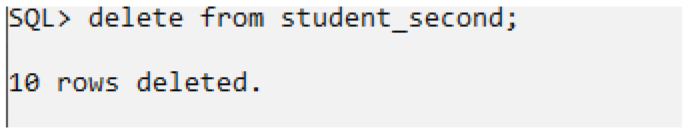
WHERE some_column =

some_value;

Description:
The DELETE statement is used to delete records from a table.

Code:

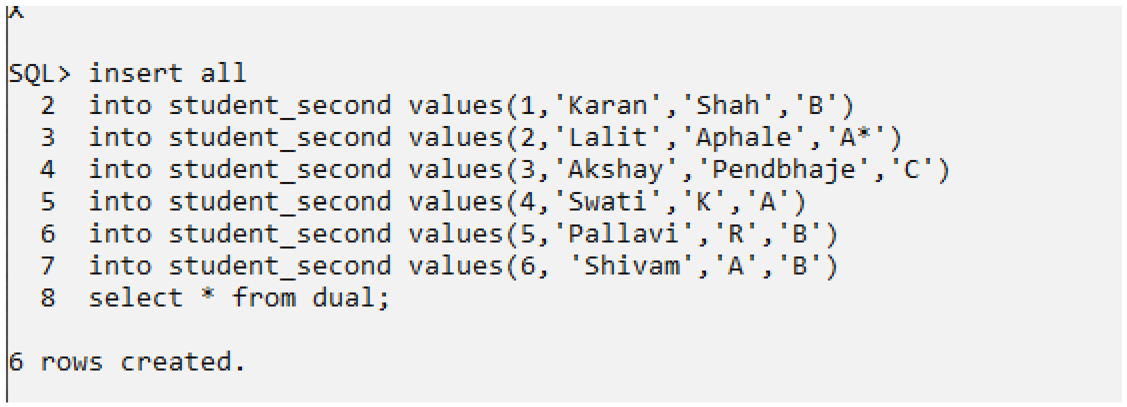
delete from student_second;
Output:



```
SQL> delete from student_second;

10 rows deleted.
```

Inserting new values:
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:



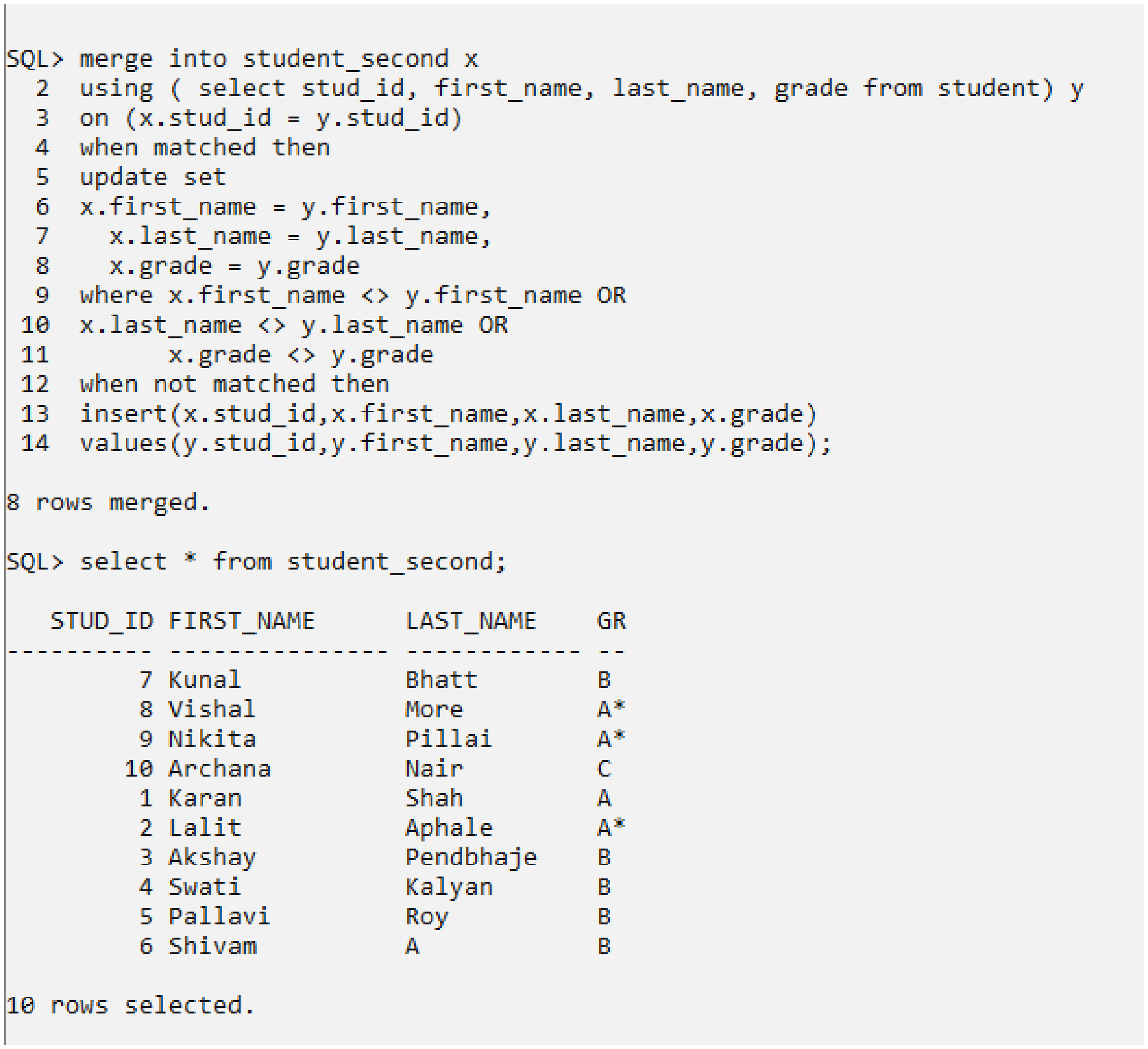
```
SQL> insert all
  2  into student_second values(1,'Karan','Shah','B')
  3  into student_second values(2,'Lalit','Aphale','A*')
  4  into student_second values(3,'Akshay','Pendbhaje','C')
  5  into student_second values(4,'Swati','K','A')
  6  into student_second values(5,'Pallavi','R','B')
  7  into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

6 rows created.
```

***Merge table with Pl/Sql**
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
---------- --------------- ------------ --
         7 Kunal           Bhatt        B
         8 Vishal          More         A*
         9 Nikita          Pillai       A*
        10 Archana         Nair         C
         1 Karan           Shah         A
         2 Lalit           Aphale       A*
         3 Akshay          Pendbhaje    B
         4 Swati           Kalyan       B
         5 Pallavi         Roy          B
         6 Shivam          A            B

10 rows selected.
```

Inserting two new tables

Code:

*Create the following relations

create table vehicle

(

model_no number,

name varchar2(15),

year number,

noofwheels number

);

```
create table vehicle_dupl
(
model_no number,
name varchar2(15),
year number,
noofwheels number,
);
```

Inserting values in both the tables

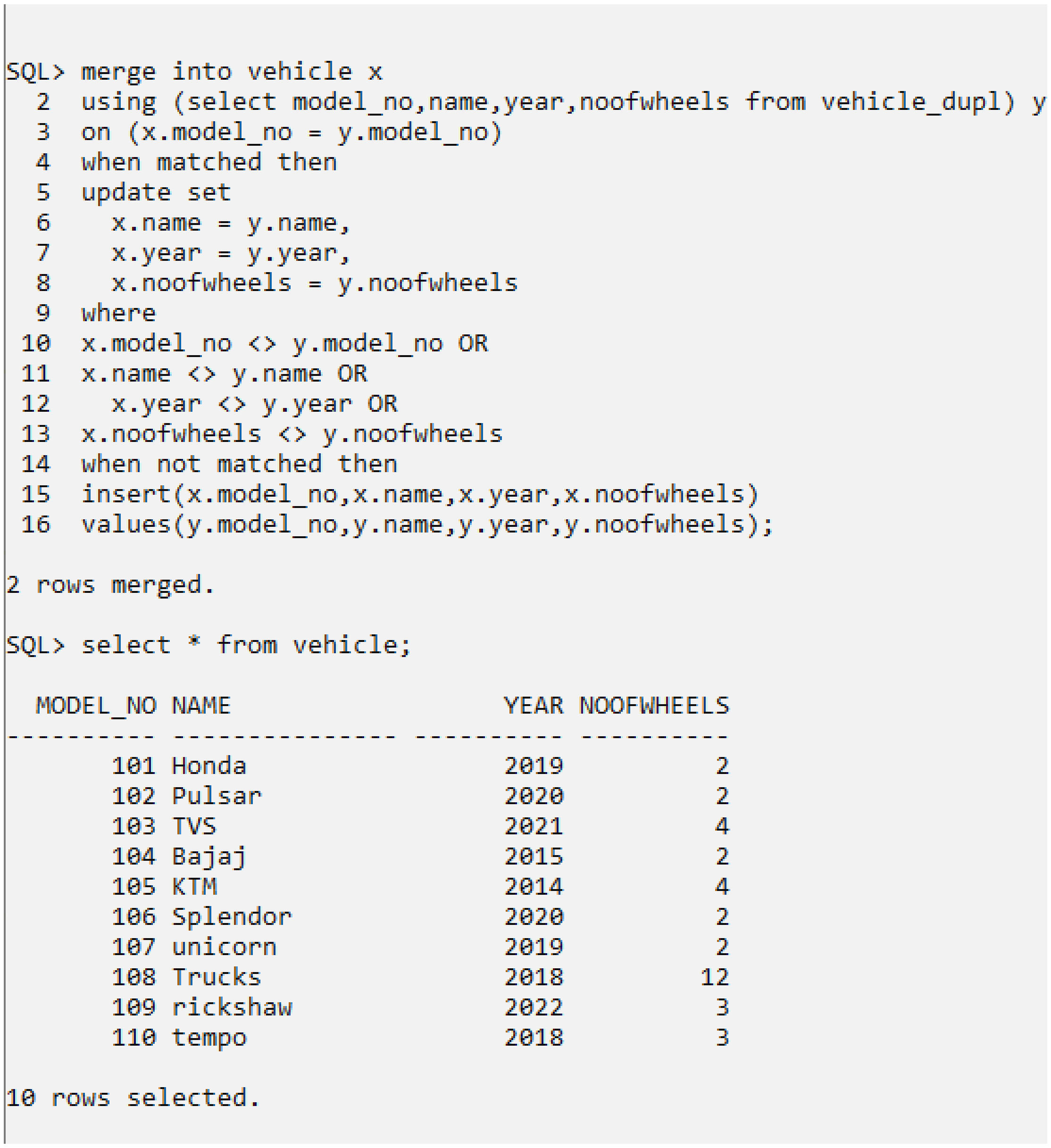
```
insert all
  into vehicle values(101,'Honda',2019,2)
  into vehicle values(102,'Pulsar',2020,2)
  into vehicle values(103,'TVS',2021,4)
  into vehicle values(104,'Bajaj',2015,2)
  into vehicle values(105,'KTM',2014,4)
  into vehicle values(106,'Splendor',2020,2)
  into vehicle values(107,'unicorn',2019,2)
  into vehicle values(108,'Trucks',2018,12)
  select * from dual;

insert all
  into vehicle_dupl values(101,'Honda',2019,2)
  into vehicle_dupl values(102,'Pulsar',2020,2)
  into vehicle_dupl values(103,'TVS',2021,4)
  into vehicle_dupl values(104,'Bajaj',2015,2)
  into vehicle_dupl values(105,'KTM',2014,4)
  into vehicle_dupl values(106,'Splendor',2020,2)
  into vehicle_dupl values(107,'unicorn',2019,2)
  into vehicle_dupl values(108,'Trucks',2018,12)
into vehicle_dupl values(109,'rickshaw',2022,3)
into vehicle_dupl values(110,'tempo',2018,3)
  select * from dual;
```

*Merging the tables

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);
```

Output:



```
SQL> merge into vehicle x
  2  using (select model_no,name,year,noofwheels from vehicle_dupl) y
  3  on (x.model_no = y.model_no)
  4  when matched then
  5  update set
  6    x.name = y.name,
  7    x.year = y.year,
  8    x.noofwheels = y.noofwheels
  9  where
 10  x.model_no <> y.model_no OR
 11  x.name <> y.name OR
 12    x.year <> y.year OR
 13  x.noofwheels <> y.noofwheels
 14  when not matched then
 15  insert(x.model_no,x.name,x.year,x.noofwheels)
 16  values(y.model_no,y.name,y.year,y.noofwheels);

2 rows merged.

SQL> select * from vehicle;

  MODEL_NO NAME                  YEAR NOOFWHEELS
---------- --------------- ---------- ----------
       101 Honda                 2019          2
       102 Pulsar                2020          2
       103 TVS                   2021          4
       104 Bajaj                 2015          2
       105 KTM                   2014          4
       106 Splendor              2020          2
       107 unicorn               2019          2
       108 Trucks                2018         12
       109 rickshaw              2022          3
       110 tempo                 2018          3

10 rows selected.
```

Practical 14. Exception Handling

1)type of exception
2)An error code
3) a message also comes

Syntax for exception handling is

```
declare
        declaration section
begin
        exception section
exception
        when ex_name1 then-error handling statements
        when others then error handling-statements
end;
```

when an exception is raised oracle searches for an appropriate exception handler int the exception handling.

There are 3 types of Exceptions
1)Named system exception
2)unnamed system exceptions
3)User-defined exceptions

a) **Type of exception**
1) **Named System exceptions**

Example

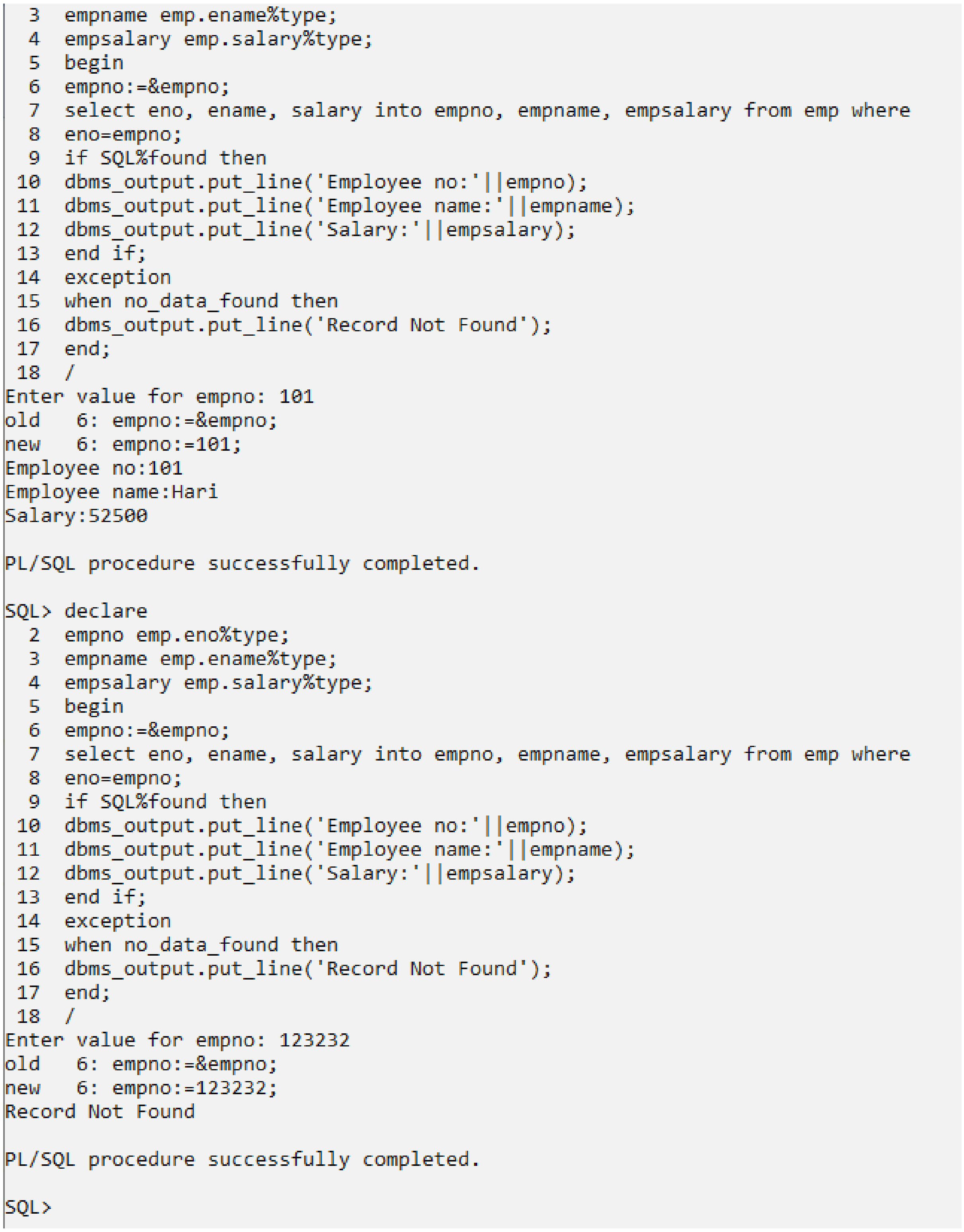
```
Begin
execution section
Exception
        WHEN NO_DATA_FOUND THEN dbms_output.put_line
        (A SELect _INTO did not return any row ‘);
END;
```

Example

Code:

```
declare
empno emp.eno%type;
empname emp.ename%type;
empsalary emp.salary%type;
begin
empno:=&empno;
select eno, ename, salary into empno, empname, empsalary from emp where
eno=empno;
if SQL%found then
dbms_output.put_line('Employee no:'||empno);
dbms_output.put_line('Employee name:'||empname);
dbms_output.put_line('Salary:'||empsalary);
end if;
exception
when no_data_found then
dbms_output.put_line('Record Not Found');
end;
/
```

Output:



```
  3  empname emp.ename%type;
  4  empsalary emp.salary%type;
  5  begin
  6  empno:=&empno;
  7  select eno, ename, salary into empno, empname, empsalary from emp where
  8  eno=empno;
  9  if SQL%found then
 10  dbms_output.put_line('Employee no:'||empno);
 11  dbms_output.put_line('Employee name:'||empname);
 12  dbms_output.put_line('Salary:'||empsalary);
 13  end if;
 14  exception
 15  when no_data_found then
 16  dbms_output.put_line('Record Not Found');
 17  end;
 18  /
Enter value for empno: 101
old   6: empno:=&empno;
new   6: empno:=101;
Employee no:101
Employee name:Hari
Salary:52500

PL/SQL procedure successfully completed.

SQL> declare
  2  empno emp.eno%type;
  3  empname emp.ename%type;
  4  empsalary emp.salary%type;
  5  begin
  6  empno:=&empno;
  7  select eno, ename, salary into empno, empname, empsalary from emp where
  8  eno=empno;
  9  if SQL%found then
 10  dbms_output.put_line('Employee no:'||empno);
 11  dbms_output.put_line('Employee name:'||empname);
 12  dbms_output.put_line('Salary:'||empsalary);
 13  end if;
 14  exception
 15  when no_data_found then
 16  dbms_output.put_line('Record Not Found');
 17  end;
 18  /
Enter value for empno: 123232
old   6: empno:=&empno;
new   6: empno:=123232;
Record Not Found

PL/SQL procedure successfully completed.

SQL>
```

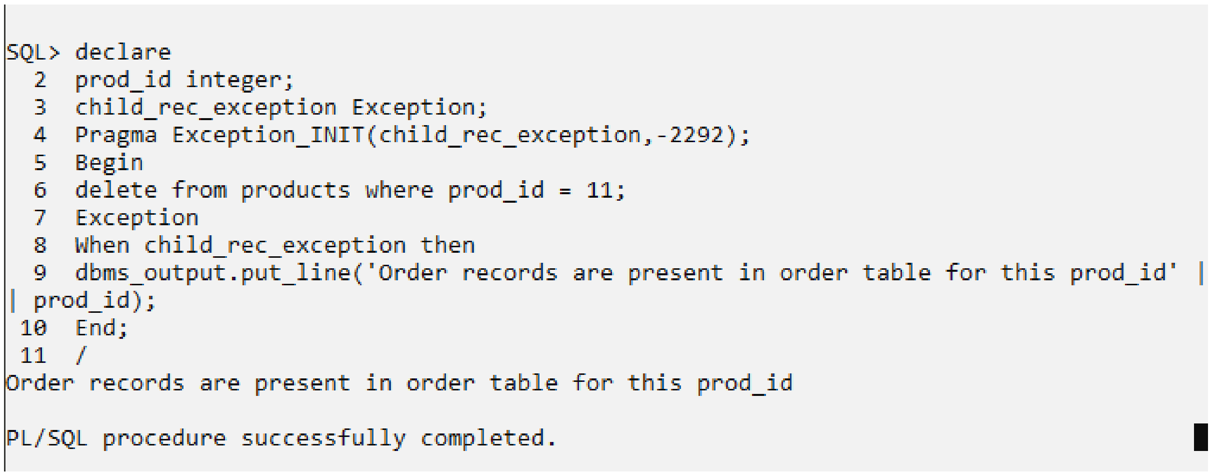
2)Unnamed system exception

Syntax:

```
declare
exception_name EXCEPTION;
Pragma
EXCEPTION_INT(exception_name,err_code);
Begin
Exception section
exception when exception_name then handle the exception
end;
```

pragma exception _INT:
the directive binds a user defined exception to a particular error number.

```
declare
        prod_id integer;
        child_rec_exception Exception;
        Pragma Exception_INIT(child_rec_exception,-2292);
Begin
        delete from products where prod_id = 11;
Exception
        When child_rec_exception then
        dbms_output.put_line('Order records are present in order table for this
prod_id' || prod_id);
End;
/
```



```
SQL> declare
  2  prod_id integer;
  3  child_rec_exception Exception;
  4  Pragma Exception_INIT(child_rec_exception,-2292);
  5  Begin
  6  delete from products where prod_id = 11;
  7  Exception
  8  When child_rec_exception then
  9  dbms_output.put_line('Order records are present in order table for this prod_id' |
| prod_id);
 10  End;
 11  /
Order records are present in order table for this prod_id

PL/SQL procedure successfully completed.
```

3)User-defined exceptions

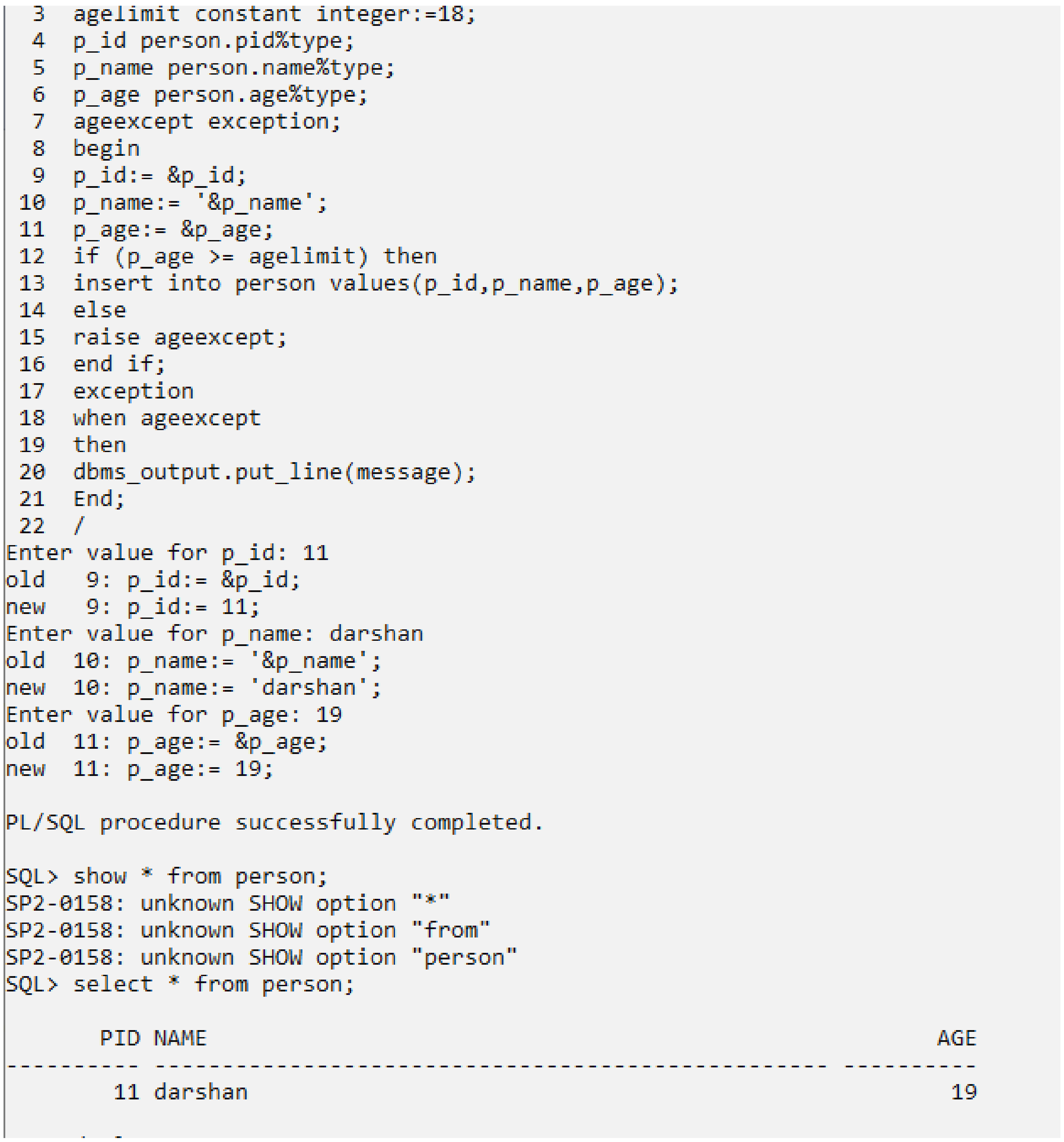
Create table

```
create table person(
pid int,
name varchar2(50),
age int
);
```

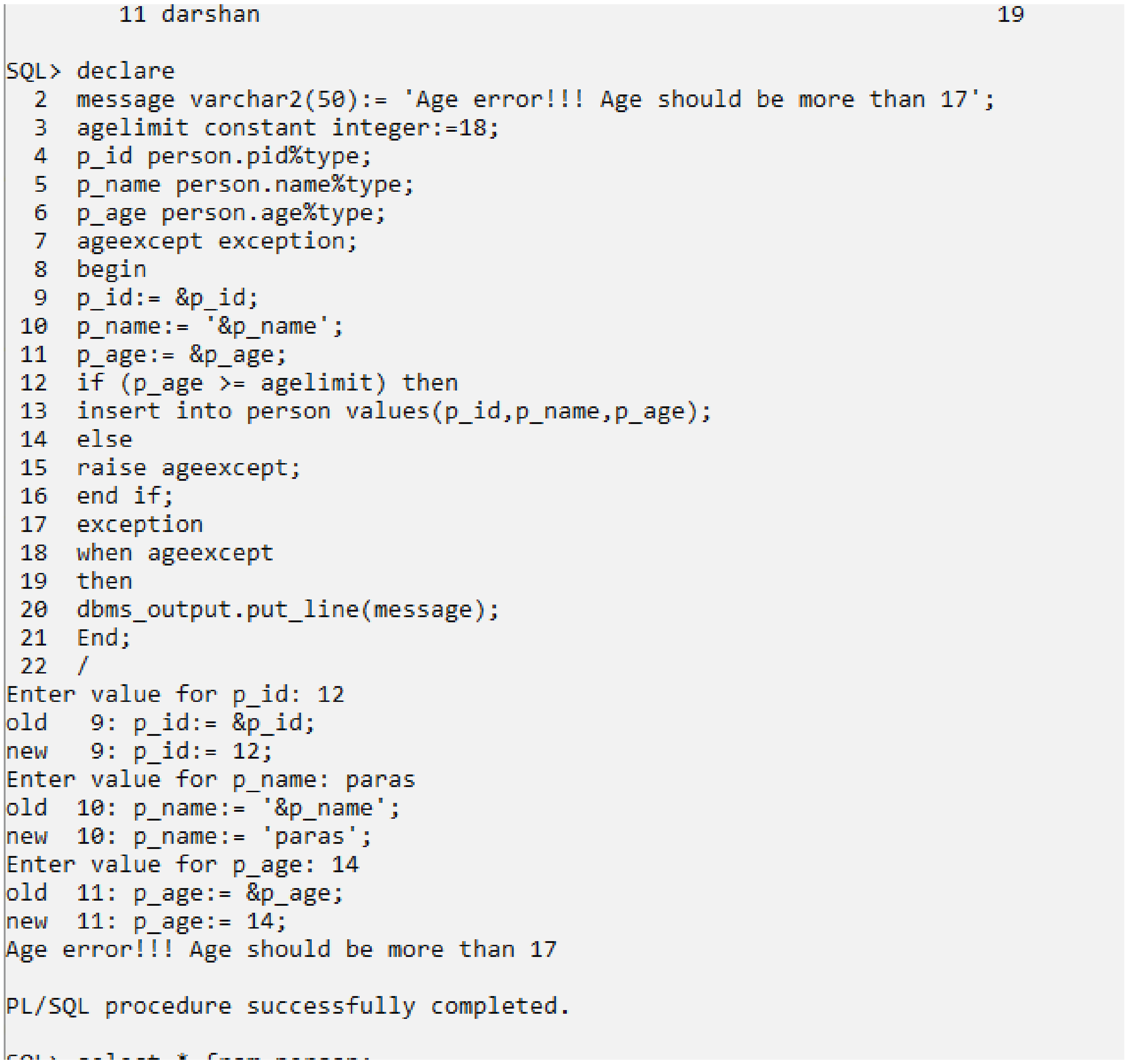
Code:

```
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:



```
  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: 11
old   9: p_id:= &p_id;
new   9: p_id:= 11;
Enter value for p_name: darshan
old  10: p_name:= '&p_name';
new  10: p_name:= 'darshan';
Enter value for p_age: 19
old  11: p_age:= &p_age;
new  11: p_age:= 19;

PL/SQL procedure successfully completed.

SQL> show * from person;
SP2-0158: unknown SHOW option "*"
SP2-0158: unknown SHOW option "from"
SP2-0158: unknown SHOW option "person"
SQL> select * from person;

       PID NAME                                                      AGE
---------- -------------------------------------------------- ----------
        11 darshan                                                    19
```



```
        11 darshan                                                        19

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: 12
old   9: p_id:= &p_id;
new   9: p_id:= 12;
Enter value for p_name: paras
old  10: p_name:= '&p_name';
new  10: p_name:= 'paras';
Enter value for p_age: 14
old  11: p_age:= &p_age;
new  11: p_age:= 14;
Age error!!! Age should be more than 17

PL/SQL procedure successfully completed.

SQL> select * from person;
```

Practical 15. Cursor

*Point on a particular data

1.Implicit Cursor

a. %NOTFOUND

Description:

The logical opposite of %FOUND. It returns TRUE if an INSERT, UPDATE, or DELETE statement affected no rows, or a SELECT INTO statement returned no rows. Otherwise, it returns FALSE.

Code:

```
Declare
        num_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
num_rows:=SQL%ROWCOUNT;
dbms_output.put_line(num_rows||'records updated');
end if;
end;
/
```

Output:

```
PL/SQL: Statement ignored


SQL> Declare
  2  num_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  num_rows:=SQL%ROWCOUNT;
  9  dbms_output.put_line(num_rows||'records updated');
 10  end if;
 11  end;
 12  /
6records updated

PL/SQL procedure successfully completed.
```

```
PL/SQL procedure successfully completed.

SQL> select * from emp;

       ENO ENAME           DESIGN            AGE        DNO DNAME
---------- --------------- ---------- ---------- ---------- ----------
    SALARY  MANAGERID
---------- ----------
       101 Hari            Analyst            28          2 sales
     53500        103

       102 Mahesh          Manager            35          2 sales
     27250        103

       103 Janvi           Analyst            30          1 Finance
     85000        103


       ENO ENAME           DESIGN            AGE        DNO DNAME
---------- --------------- ---------- ---------- ---------- ----------
    SALARY  MANAGERID
---------- ----------
       104 Sumi            Developer          24          3 IT
     22000        105

       105 pratik          Tester             22          3 IT
     16750        105

       108 darshan         manager            23          4 Ai
      6250        105


6 rows selected.
```

b. Rowtype

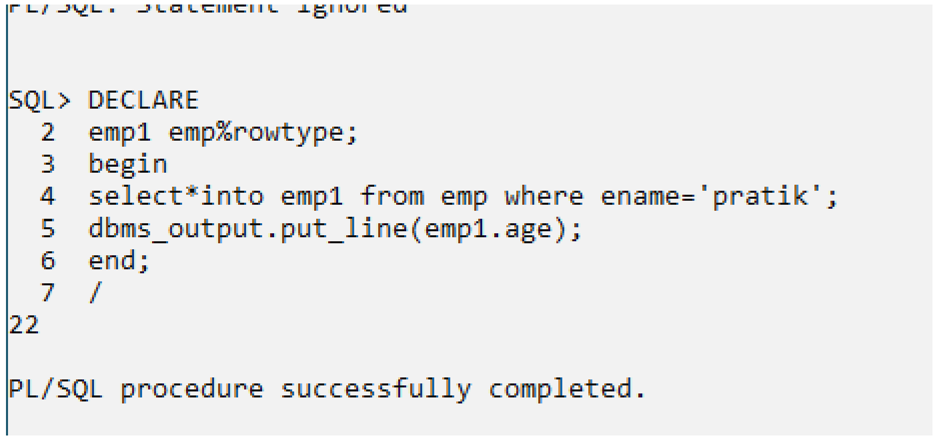
To display the age from emp whose name is pratik

*records

Code:

```
DECLARE
emp1 emp%rowtype;
begin
select*into emp1 from emp where ename='pratik';
dbms_output.put_line(emp1.age);
end;
/
```

Output:



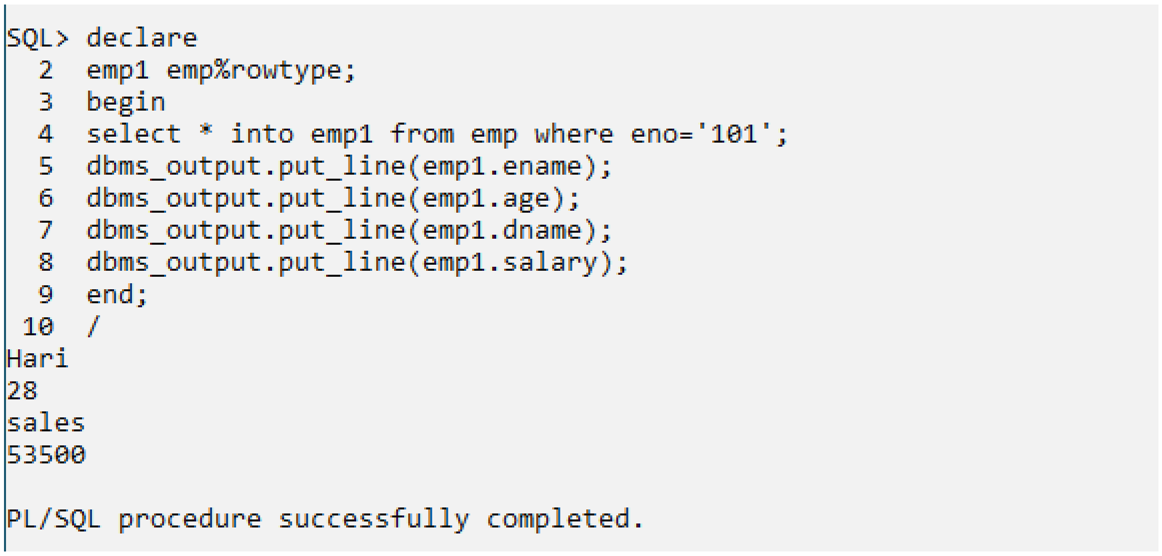
```
SQL> DECLARE
  2  emp1 emp%rowtype;
  3  begin
  4  select*into emp1 from emp where ename='pratik';
  5  dbms_output.put_line(emp1.age);
  6  end;
  7  /
22

PL/SQL procedure successfully completed.
```

2. To display the details of an employee with eid 101 using %rowtype

Code:

```
declare
emp1 emp%rowtype;
begin
select * into emp1 from emp where eno='101';
dbms_output.put_line(emp1.ename);
dbms_output.put_line(emp1.age);
dbms_output.put_line(emp1.dname);
```

dbms_output.put_line(emp1.salary);
end;
/
Output:



```
SQL> declare
  2  emp1 emp%rowtype;
  3  begin
  4  select * into emp1 from emp where eno='101';
  5  dbms_output.put_line(emp1.ename);
  6  dbms_output.put_line(emp1.age);
  7  dbms_output.put_line(emp1.dname);
  8  dbms_output.put_line(emp1.salary);
  9  end;
 10  /
Hari
28
sales
53500

PL/SQL procedure successfully completed.
```

***Explicit Cursor**
Syntax:
CURSOR cursor_name IS select _statement;

four types in the declaration of cursor
declare
DECLARE CURSOR mep_cur IS SELECT * FROM emp_tbl WHERE salary>5000;

2. OPEN:The cursor in the execution section
OPEN cursor_name; OPEN emp_cur;

3.FETCH :the data from cursor into pl/sql variables or records in the execution section. go through all the records.
*FETCH cursor_name INTO record_name;

4.CLOSE:The cursor in the execution

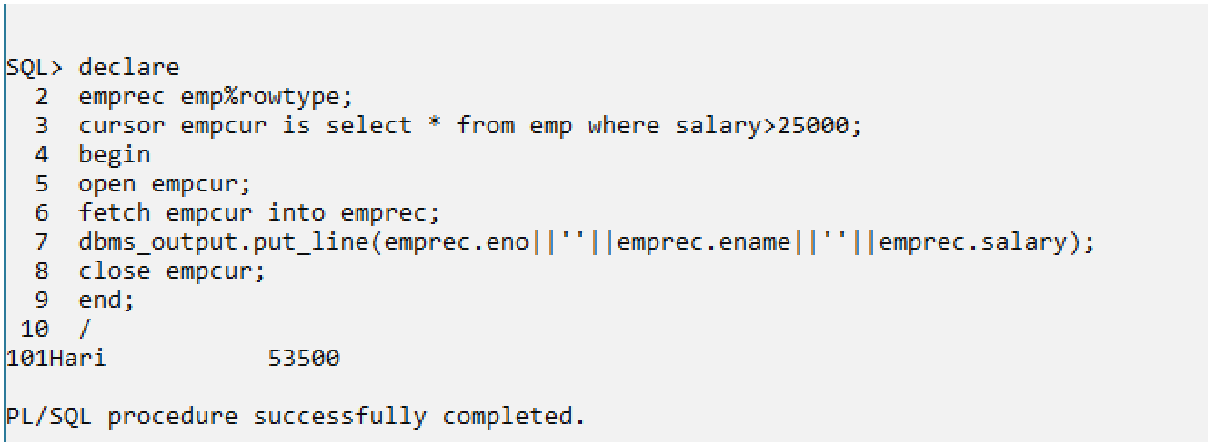
CLOSE THE CURSOR

EXAMPLE:
```
DECLARE
        variables;
        records;
        create a cursor;
BEGIN
OPEN cursor;
        FETCH cursor;
        process the records;
CLOSE cursor;
END;
/
```

Code:

Display empno,fname and salary using cursor
```
declare
        emprec emp%rowtype;
        cursor empcur is select * from emp where salary>25000;
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>25000;
  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  /
101Hari           53500

PL/SQL procedure successfully completed.
```

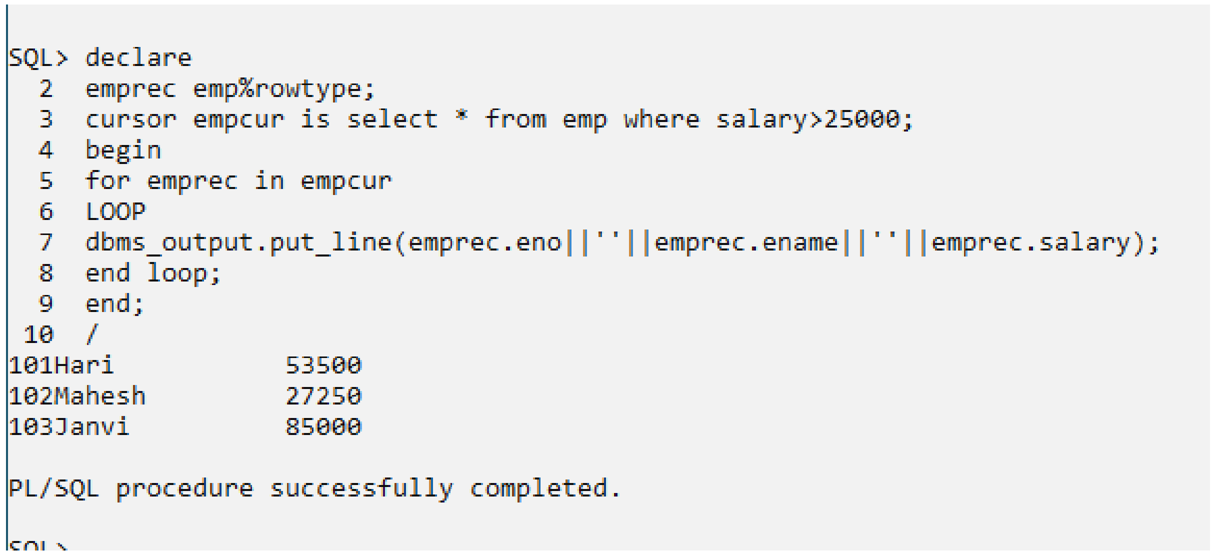
3. Explicit cursor USING for LOOP

Code:

for loop open , fetch and close be used default

```
declare
        emprec emp%rowtype;
        cursor empcur is select * from emp where salary>25000;
begin
        for emprec in empcur
        LOOP
        dbms_output.put_line(emprec.eno||''||emprec.ename||''||emprec.salary);
end loop;
end;
/
```

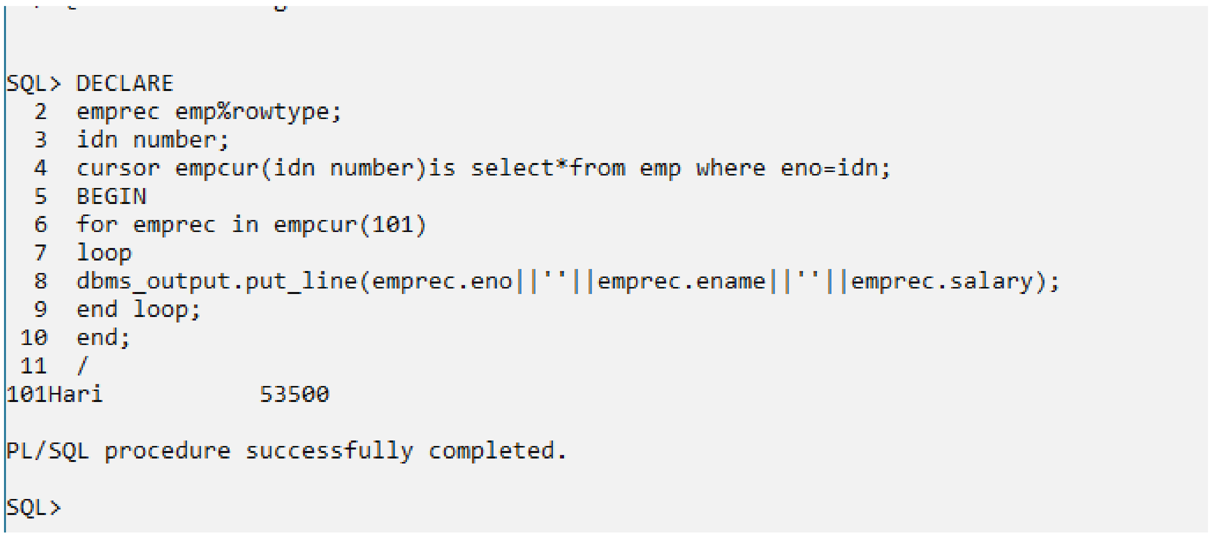
Output:



```
SQL> declare
emprec emp%rowtype;
cursor empcur is select * from emp where salary>25000;
begin
for emprec in empcur
LOOP
dbms_output.put_line(emprec.eno||''||emprec.ename||''||emprec.salary);
end loop;
end;
/
101Hari           53500
102Mahesh         27250
103Janvi          85000

PL/SQL procedure successfully completed.

SQL>
```

4.Cursor with parameters
SYNTAX:
CURSOR cursor_name(paramter_list) IS cursor_query;

example:
DECLARE
emprec emp%rowtype;
idn number;
cursor empcur(idn number)is select*from emp where eno=idn;
BEGIN
for emprec in empcur(101)
loop
dbms_output.put_line(emprec.eno||''||emprec.ename||''||emprec.salary);
end loop;
end;
/
Output:



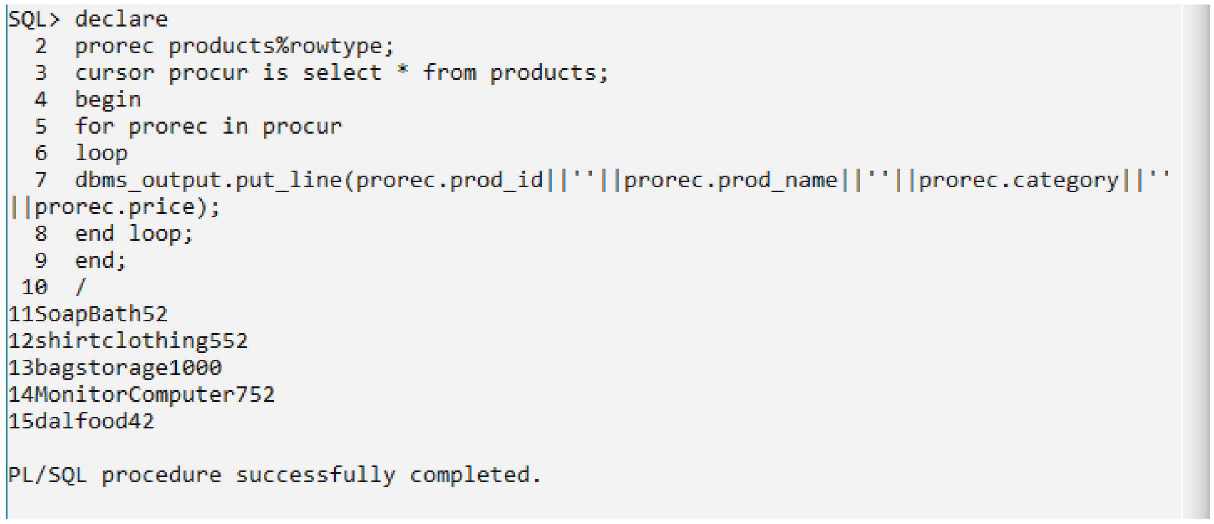
```
SQL> DECLARE
  emprec emp%rowtype;
  idn number;
  cursor empcur(idn number)is select*from emp where eno=idn;
  BEGIN
  for emprec in empcur(101)
  loop
  dbms_output.put_line(emprec.eno||''||emprec.ename||''||emprec.salary);
  end loop;
  end;
  /
101Hari           53500

PL/SQL procedure successfully completed.

SQL>
```

1. Write a Pl/Sql block to print the product details using cursor.(loop)

Code:

```
declare
prorec products%rowtype;
cursor procur is select * from products;
begin
for prorec in procur
loop
dbms_output.put_line(prorec.prod_id||''||prorec.prod_name||''||prorec.category||''||
prorec.price);
end loop;
end;
/
```

Output:

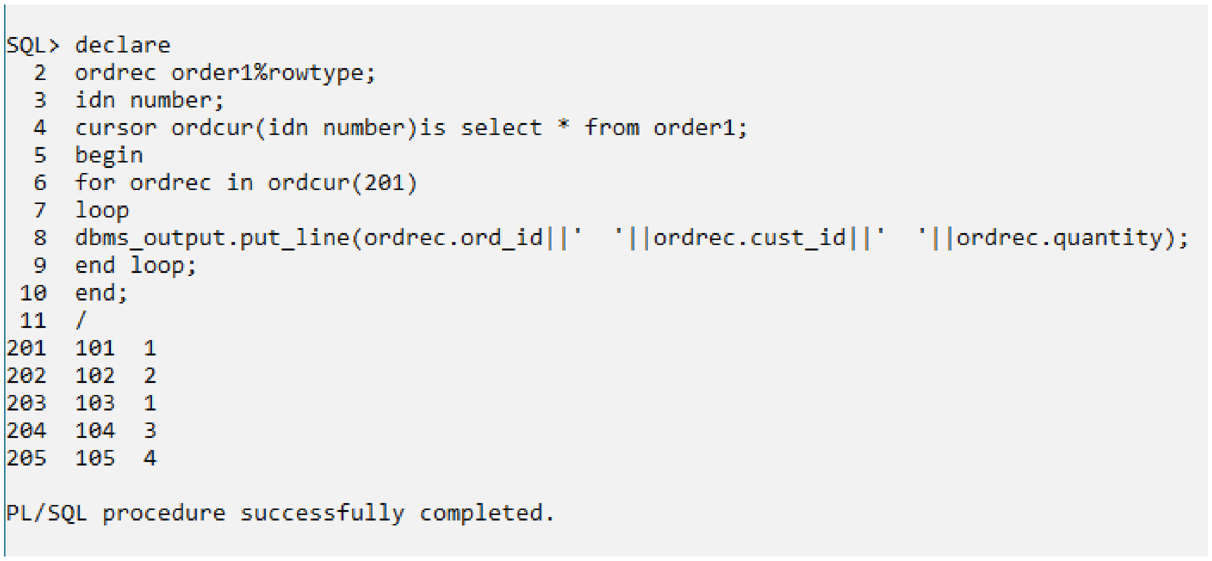

```
SQL> declare
prorec products%rowtype;
cursor procur is select * from products;
begin
for prorec in procur
loop
dbms_output.put_line(prorec.prod_id||''||prorec.prod_name||''||prorec.category||''
||prorec.price);
end loop;
end;
/
11SoapBath52
12shirtclothing552
13bagstorage1000
14MonitorComputer752
15dalfood42

PL/SQL procedure successfully completed.
```

2. Write a PL/SQL block to print the order details for a particular order.(cursor with the parameters)

Code:

```
declare
ordrec order1%rowtype;
idn number;
cursor ordcur(idn number)is select * from order1;
begin
for ordrec in ordcur(201)
loop
dbms_output.put_line(ordrec.ord_id||'  '||ordrec.cust_id||'  '||ordrec.quantity);
end loop;
end;
/
```

Output:



```
SQL> declare
  2  ordrec order1%rowtype;
  3  idn number;
  4  cursor ordcur(idn number)is select * from order1;
  5  begin
  6  for ordrec in ordcur(201)
  7  loop
  8  dbms_output.put_line(ordrec.ord_id||'  '||ordrec.cust_id||'  '||ordrec.quantity);
  9  end loop;
 10  end;
 11  /
201  101  1
202  102  2
203  103  1
204  104  3
205  105  4

PL/SQL procedure successfully completed.
```

Practical 16 : Records

1) Inserting records for books:

composite attribute(one single attribute has many values)
1. It is composite data types, which means it is a combination of different scalar datatypes like char,varchar,number.
2. each scalar data types in the record holds a value.
3. A record can be visualised as a row of data.

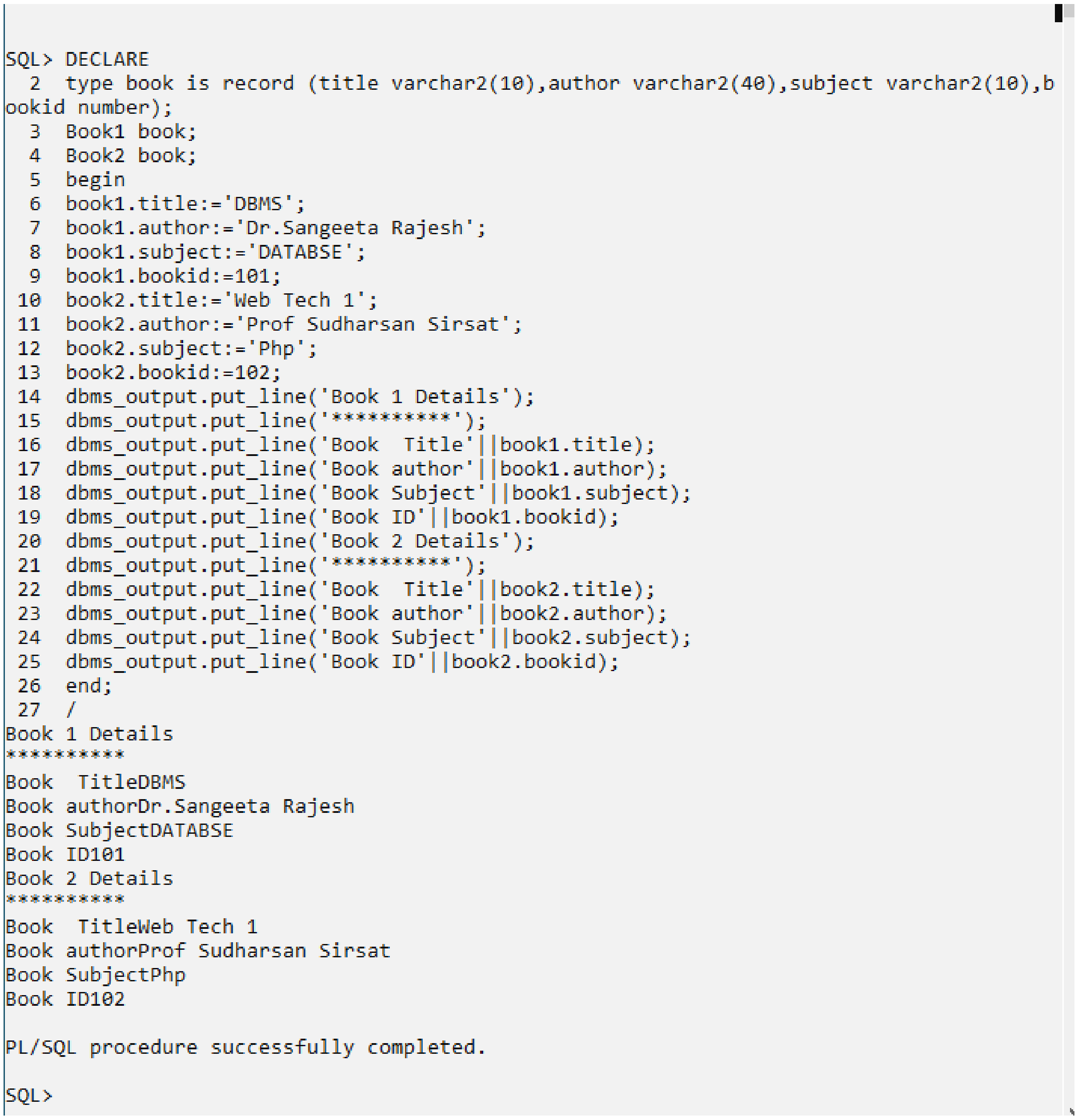
SYNTAX to define a composite data type is:

```
TYPE record_type_name IS RECORD
(first_col_name column_datatype,
second_col-name column_datatype,..);
```

*User defined records

```
DECLARE
type book is record (title varchar2(10),author varchar2(40),subject
varchar2(10),bookid number);
Book1 book;
Book2 book;
begin
book1.title:='DBMS';
book1.author:='Dr.Sangeeta Rajesh';
book1.subject:='DATABSE';
book1.bookid:=101;
book2.title:='Web Tech 1';
book2.author:='Prof Sangeetha ma’am';
book2.subject:='Php';
book2.bookid:=102;
dbms_output.put_line('Book 1 Details');
dbms_output.put_line('**********');
dbms_output.put_line('Book  Title'||book1.title);
dbms_output.put_line('Book author'||book1.author);
dbms_output.put_line('Book Subject'||book1.subject);
dbms_output.put_line('Book ID'||book1.bookid);
dbms_output.put_line('Book 2 Details');
dbms_output.put_line('**********');
dbms_output.put_line('Book  Title'||book2.title);
dbms_output.put_line('Book author'||book2.author);
dbms_output.put_line('Book Subject'||book2.subject);
dbms_output.put_line('Book ID'||book2.bookid);
end;
/
```

Output:

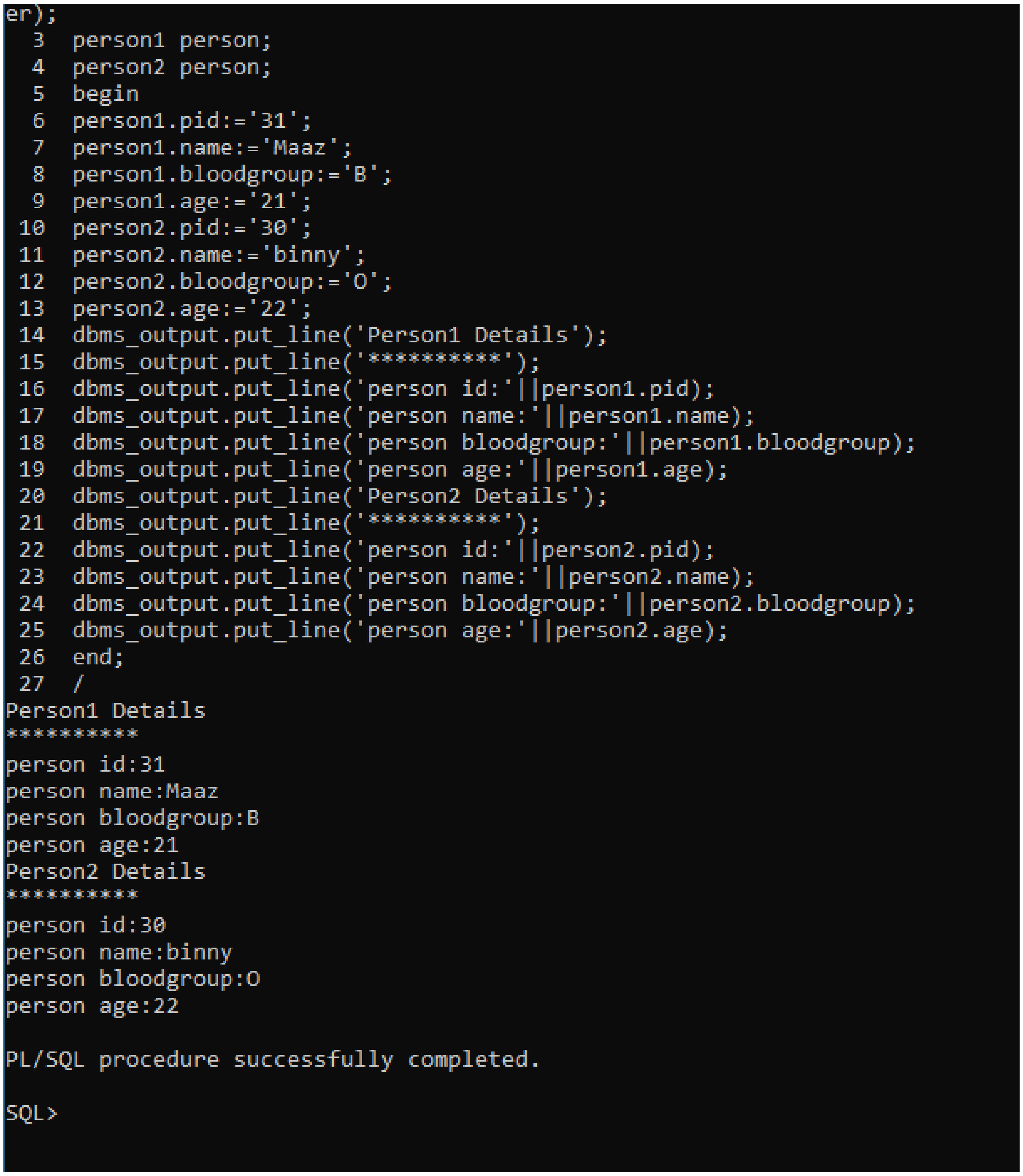


```
SQL> DECLARE
  2  type book is record (title varchar2(10),author varchar2(40),subject varchar2(10),b
ookid number);
  3  Book1 book;
  4  Book2 book;
  5  begin
  6  book1.title:='DBMS';
  7  book1.author:='Dr.Sangeeta Rajesh';
  8  book1.subject:='DATABSE';
  9  book1.bookid:=101;
 10  book2.title:='Web Tech 1';
 11  book2.author:='Prof Sudharsan Sirsat';
 12  book2.subject:='Php';
 13  book2.bookid:=102;
 14  dbms_output.put_line('Book 1 Details');
 15  dbms_output.put_line('**********');
 16  dbms_output.put_line('Book  Title'||book1.title);
 17  dbms_output.put_line('Book author'||book1.author);
 18  dbms_output.put_line('Book Subject'||book1.subject);
 19  dbms_output.put_line('Book ID'||book1.bookid);
 20  dbms_output.put_line('Book 2 Details');
 21  dbms_output.put_line('**********');
 22  dbms_output.put_line('Book  Title'||book2.title);
 23  dbms_output.put_line('Book author'||book2.author);
 24  dbms_output.put_line('Book Subject'||book2.subject);
 25  dbms_output.put_line('Book ID'||book2.bookid);
 26  end;
 27  /
Book 1 Details
**********
Book  TitleDBMS
Book authorDr.Sangeeta Rajesh
Book SubjectDATABSE
Book ID101
Book 2 Details
**********
Book  TitleWeb Tech 1
Book authorProf Sudharsan Sirsat
Book SubjectPhp
Book ID102

PL/SQL procedure successfully completed.

SQL>
```

2) Inserting records for persons blood group

```
declare
type person is record(pid number,name varchar2(20),bloodgroup
varchar2(5),age number);
person1 person;
person2 person;
begin
person1.pid:='31';
person1.name:='Maaz';
person1.bloodgroup:='B';
person1.age:='21';
person2.pid:='30';
person2.name:='binny';
person2.bloodgroup:='O';
person2.age:='22';
dbms_output.put_line('Person1 Details');
dbms_output.put_line('**********');
dbms_output.put_line('person id:'||person1.pid);
dbms_output.put_line('person name:'||person1.name);
dbms_output.put_line('person bloodgroup:'||person1.bloodgroup);
dbms_output.put_line('person age:'||person1.age);
dbms_output.put_line('Person2 Details');
dbms_output.put_line('**********');
dbms_output.put_line('person id:'||person2.pid);
dbms_output.put_line('person name:'||person2.name);
dbms_output.put_line('person bloodgroup:'||person2.bloodgroup);
dbms_output.put_line('person age:'||person2.age);
end;
/
```

Output:



```
er);
person1 person;
person2 person;
begin
person1.pid:='31';
person1.name:='Maaz';
person1.bloodgroup:='B';
person1.age:='21';
person2.pid:='30';
person2.name:='binny';
person2.bloodgroup:='O';
person2.age:='22';
dbms_output.put_line('Person1 Details');
dbms_output.put_line('**********');
dbms_output.put_line('person id:'||person1.pid);
dbms_output.put_line('person name:'||person1.name);
dbms_output.put_line('person bloodgroup:'||person1.bloodgroup);
dbms_output.put_line('person age:'||person1.age);
dbms_output.put_line('Person2 Details');
dbms_output.put_line('**********');
dbms_output.put_line('person id:'||person2.pid);
dbms_output.put_line('person name:'||person2.name);
dbms_output.put_line('person bloodgroup:'||person2.bloodgroup);
dbms_output.put_line('person age:'||person2.age);
end;
/
Person1 Details
**********
person id:31
person name:Maaz
person bloodgroup:B
person age:21
Person2 Details
**********
person id:30
person name:binny
person bloodgroup:O
person age:22

PL/SQL procedure successfully completed.

SQL>
```

Practical 17. TRIGGER

Description:
A trigger is a pl/sql block structure which is fired when a DML statements like INSERT, DELETE, UPDATE is executed/

Syntax for Creating a Trigger

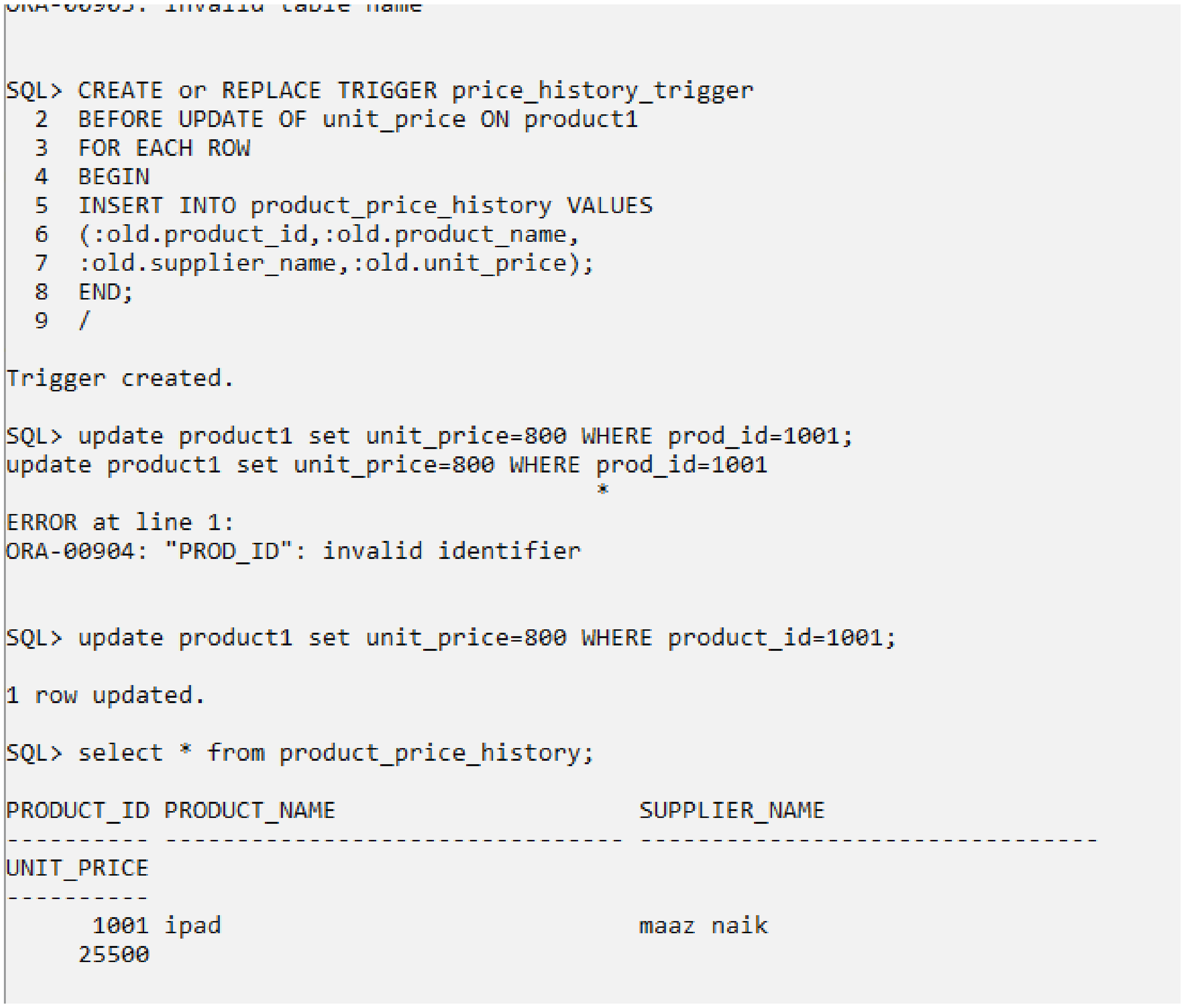
```
CREATE [OR REPLACE] TRIGGER trigger-name
{BEFORE | AFTER | INSTEAD OF}
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name] ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
BEGIN
--- sql statements
END;
/
```

2 TYPES OF TRIGGER
1) Row level trigger
2) statement level trigger

1) **Row level trigger**
   ***Creating Trigger**

```
CREATE or REPLACE TRIGGER price_history_trigger
BEFORE UPDATE OF unit_price ON product1
FOR EACH ROW
BEGIN
INSERT INTO product_price_history VALUES
(:old.product_id,:old.product_name,
:old.supplier_name,:old.unit_price);
END;
/
```

Output:



```
ORA-00903: invalid table name


SQL> CREATE or REPLACE TRIGGER price_history_trigger
     BEFORE UPDATE OF unit_price ON product1
     FOR EACH ROW
     BEGIN
     INSERT INTO product_price_history VALUES
     (:old.product_id,:old.product_name,
     :old.supplier_name,:old.unit_price);
     END;
     /

Trigger created.

SQL> update product1 set unit_price=800 WHERE prod_id=1001;
update product1 set unit_price=800 WHERE prod_id=1001
                                         *
ERROR at line 1:
ORA-00904: "PROD_ID": invalid identifier


SQL> update product1 set unit_price=800 WHERE product_id=1001;

1 row updated.

SQL> select * from product_price_history;

PRODUCT_ID PRODUCT_NAME                   SUPPLIER_NAME
---------- ------------------------------ ------------------------------
UNIT_PRICE
----------
      1001 ipad                           maaz naik
     25500
```

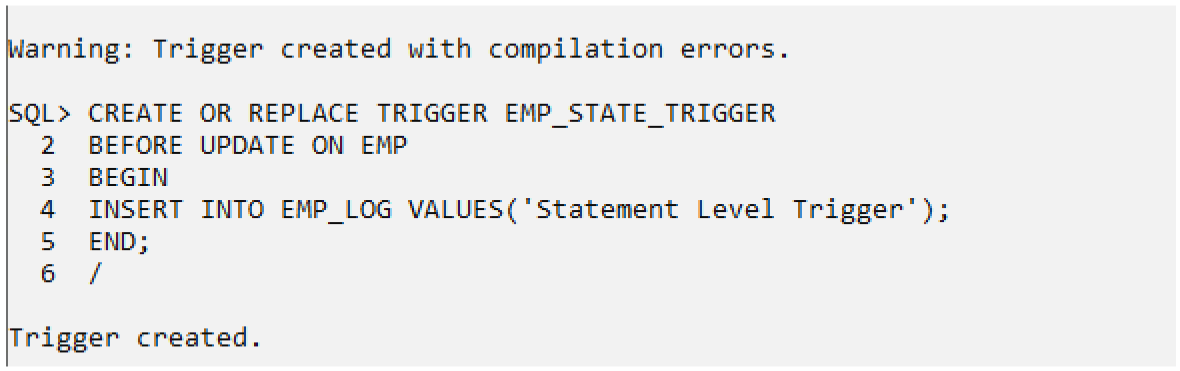
2) Statement Level trigger

create table emp_log (type varchar2(30));

@)Statement level trigger
create or replace trigger emp_state_trigger
before update on emp
begin
insert into emp_log values("Statement lavel trigger");
end;
/

update emp.set salary=salary+1000;

Output:

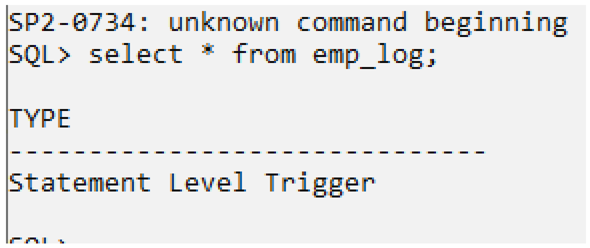


```
Warning: Trigger created with compilation errors.

SQL> CREATE OR REPLACE TRIGGER EMP_STATE_TRIGGER
  2  BEFORE UPDATE ON EMP
  3  BEGIN
  4  INSERT INTO EMP_LOG VALUES('Statement Level Trigger');
  5  END;
  6  /

Trigger created.
```

Update trigger

UPDATE EMP SET SALARY = SALARY + 1000;

Output:

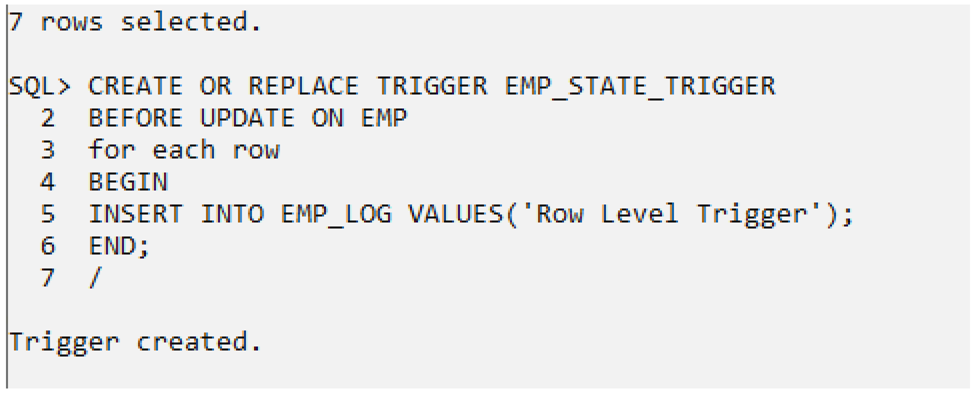


```
SP2-0734: unknown command beginning
SQL> select * from emp_log;

TYPE
------------------------------
Statement Level Trigger
```

2. row level trigger in same code.

CREATE OR REPLACE TRIGGER EMP_STATE_TRIGGER
BEFORE UPDATE ON EMP
for each row
BEGIN
INSERT INTO EMP_LOG VALUES('Row Level Trigger');
END;
/

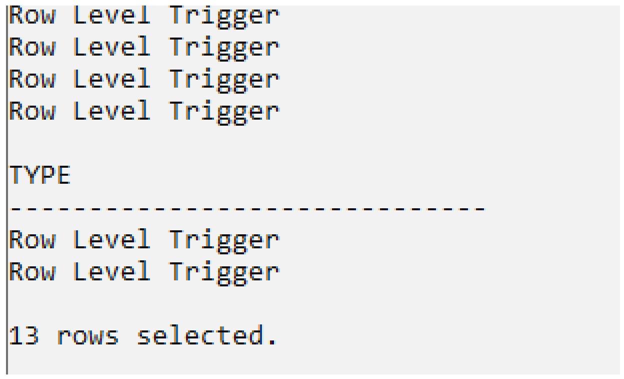
Output:



```
7 rows selected.

SQL> CREATE OR REPLACE TRIGGER EMP_STATE_TRIGGER
BEFORE UPDATE ON EMP
for each row
BEGIN
INSERT INTO EMP_LOG VALUES('Row Level Trigger');
END;
/

Trigger created.
```

Code:
update emp set salary = salary+1000;

Output:



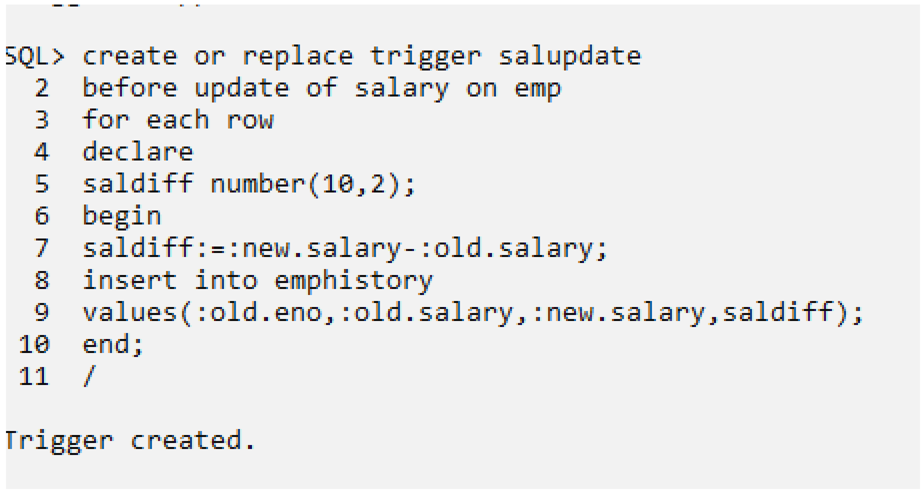
```
Row Level Trigger
Row Level Trigger
Row Level Trigger
Row Level Trigger

TYPE
------------------------------
Row Level Trigger
Row Level Trigger

13 rows selected.
```

Exercise:
Create pl/sql block to maintain database table to store information log on history table.
It should contain eno,oldslary, new salary and difference in salary.

Code creating the new emp history table:
create table emphistory(
eno number(10),
oldsalary number(10),
newsalary number(10),
difference_in_salary number(10)
);
Code of trigger:

```
create or replace trigger salupdate
before update of salary on emp
for each row
declare
saldiff number(10,2);
begin
saldiff:=:new.salary-:old.salary;
insert into emphistory
values(:old.eno,:old.salary,:new.salary,saldiff);
end;
/
```

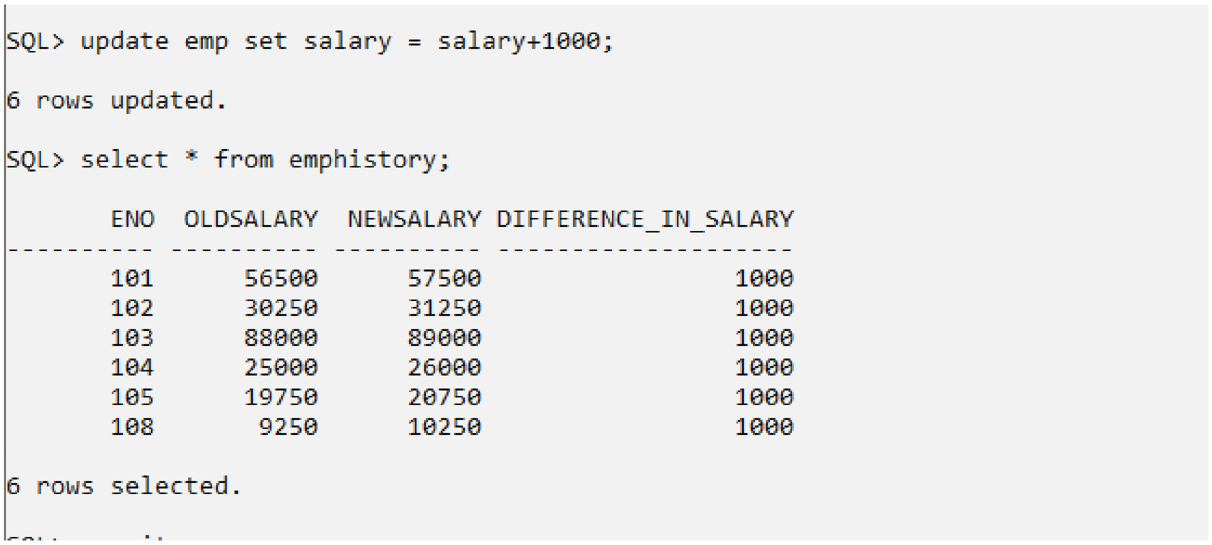
Output:



```
SQL> create or replace trigger salupdate
  2  before update of salary on emp
  3  for each row
  4  declare
  5  saldiff number(10,2);
  6  begin
  7  saldiff:=:new.salary-:old.salary;
  8  insert into emphistory
  9  values(:old.eno,:old.salary,:new.salary,saldiff);
 10  end;
 11  /

Trigger created.
```

Code for update:

```
update emp set salary = salary+1000;
```



```
SQL> update emp set salary = salary+1000;

6 rows updated.

SQL> select * from emphistory;

       ENO  OLDSALARY  NEWSALARY DIFFERENCE_IN_SALARY
---------- ---------- ---------- --------------------
       101      56500      57500                 1000
       102      30250      31250                 1000
       103      88000      89000                 1000
       104      25000      26000                 1000
       105      19750      20750                 1000
       108       9250      10250                 1000

6 rows selected.
```

Practical 18. Functions

Description:
A subprogram is a program unit/module that performs a particular task.
Pl/SQL provides two kinds
Generate Syntax to create a function is:

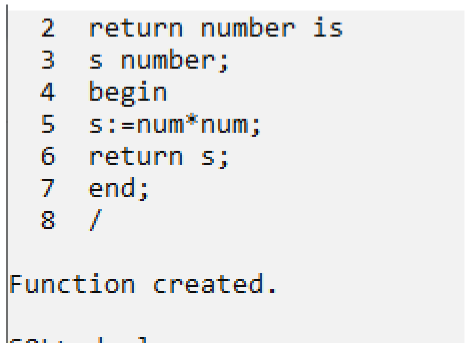
```
CREATE [OR REPLACE]FUNCTION function_name[parameters]
RETURN return_datatype;
IS
Declaration_section
BEGIN
Execution_section
return return_variable;
EXCEPTION
exception section
Return return_variable;
END;
```

Dropping a function

```
DROP  FUNCTION function name;
```

*Functions
to create a function

```
create or replace function sqr(num in number)
return number is
s number;
begin
s:=num*num;
return s;
end;
/
```



```
return number is
s number;
begin
s:=num*num;
return s;
end;
/

Function created.
```

Code square of number.

Code:

declare

n number;

sq number;

begin

n:=&n;

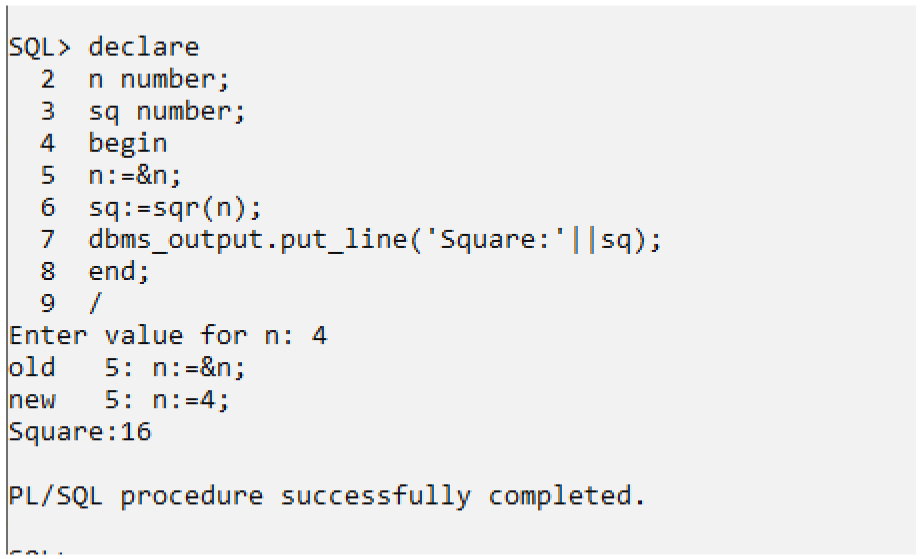
sq:=sqr(n);

dbms_output.put_line('Square:'||sq);

end;

/

Output:



```
SQL> declare
n number;
sq number;
begin
n:=&n;
sq:=sqr(n);
dbms_output.put_line('Square:'||sq);
end;
/
Enter value for n: 4
old   5: n:=&n;
new   5: n:=4;
Square:16

PL/SQL procedure successfully completed.
```

Write a function to accept the side of a square and find the area

declare

Function

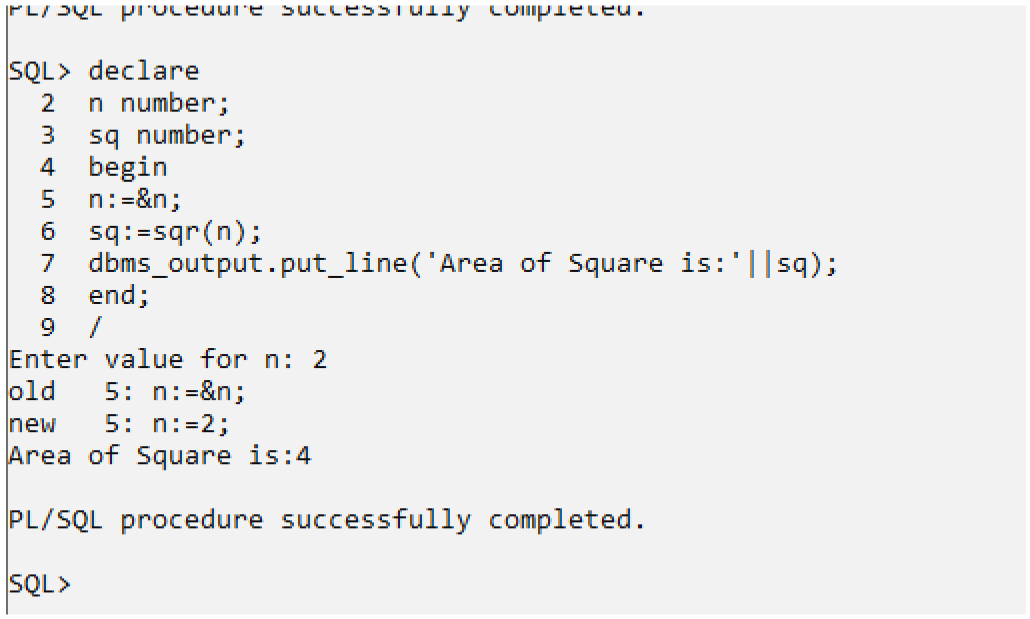
Code:

```
create or replace function sqr(num in number)
return number is
s number;
begin
s:=num*num;
return s;
end;
/
```

Code:

```
declare
n number;
sq number;
begin
n:=&n;
sq:=sqr(n);
dbms_output.put_line('Area of Square is:'||sq);
end;
/
```

Output:

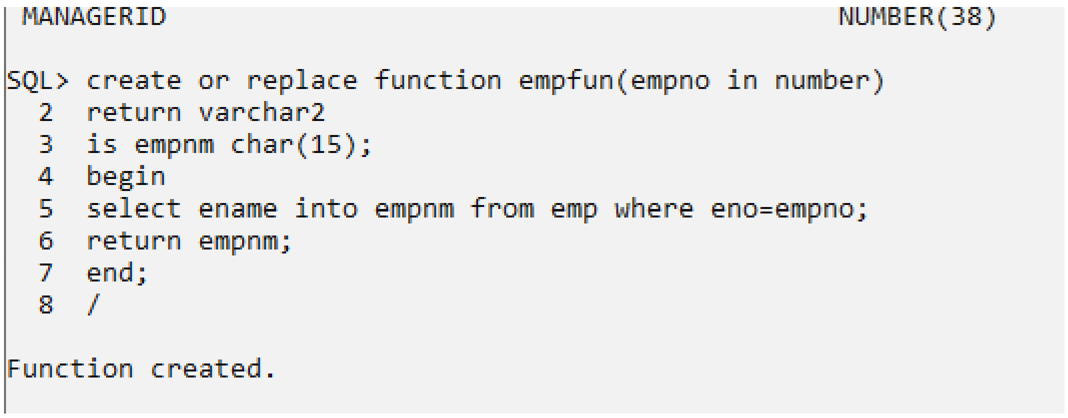


```
PL/SQL procedure successfully completed.

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);
  8  end;
  9  /
Enter value for n: 2
old   5: n:=&n;
new   5: n:=2;
Area of Square is:4

PL/SQL procedure successfully completed.

SQL>
```

*Create a function to accept empno as a aprameter and return empname.

create or replace function empfun(empno in number)
return varchar2
is empnm char(15);
begin
select ename into empnm from emp where eno=empno;
return empnm;
end;
/

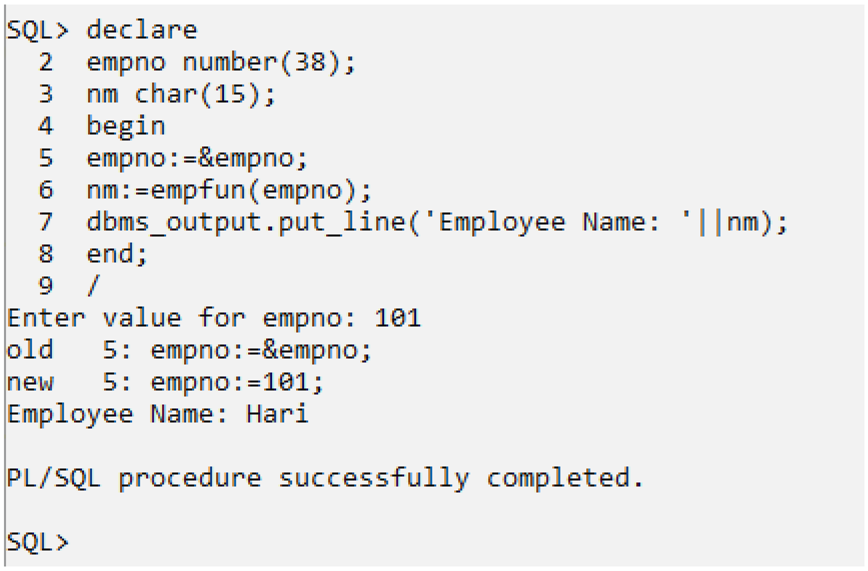
Output:



```
 MANAGERID                                            NUMBER(38)

SQL> create or replace function empfun(empno in number)
  2  return varchar2
  3  is empnm char(15);
  4  begin
  5  select ename into empnm from emp where eno=empno;
  6  return empnm;
  7  end;
  8  /

Function created.
```

to execute a pl/sql function
declare
empno number(38);
nm char(15);
begin
empno:=&empno;
nm:=empfun(empno);
dbms_output.put_line('Employee Name: '||nm);
end;
/

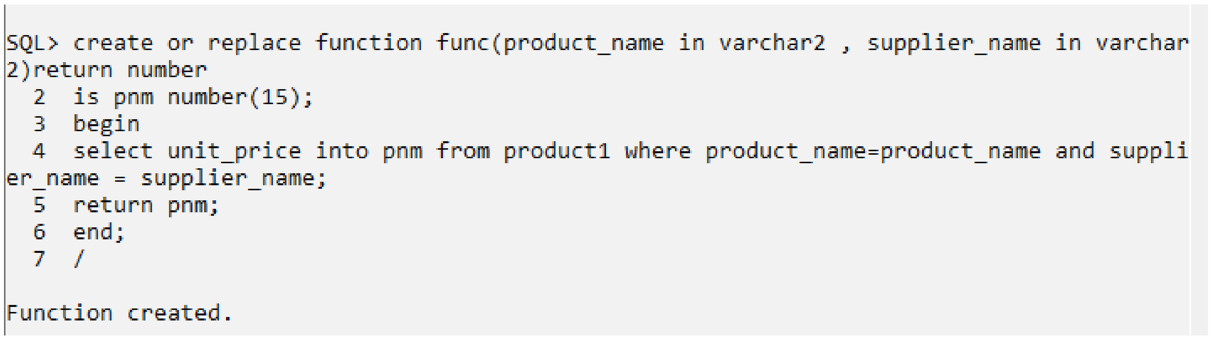
output:


```
SQL> declare
  2  empno number(38);
  3  nm char(15);
  4  begin
  5  empno:=&empno;
  6  nm:=empfun(empno);
  7  dbms_output.put_line('Employee Name: '||nm);
  8  end;
  9  /
Enter value for empno: 101
old   5: empno:=&empno;
new   5: empno:=101;
Employee Name: Hari

PL/SQL procedure successfully completed.

SQL>
```

2. **<u>Q.</u> <u>Create a function to accept product name and supplier name and display the unit price</u>**

create or replace function func(product_name in varchar2 , supplier_name in varchar2)return number
is pnm number(15);
begin
select unit_price into pnm from product1 where product_name=product_name and supplier_name = supplier_name;
return pnm;
end;
/

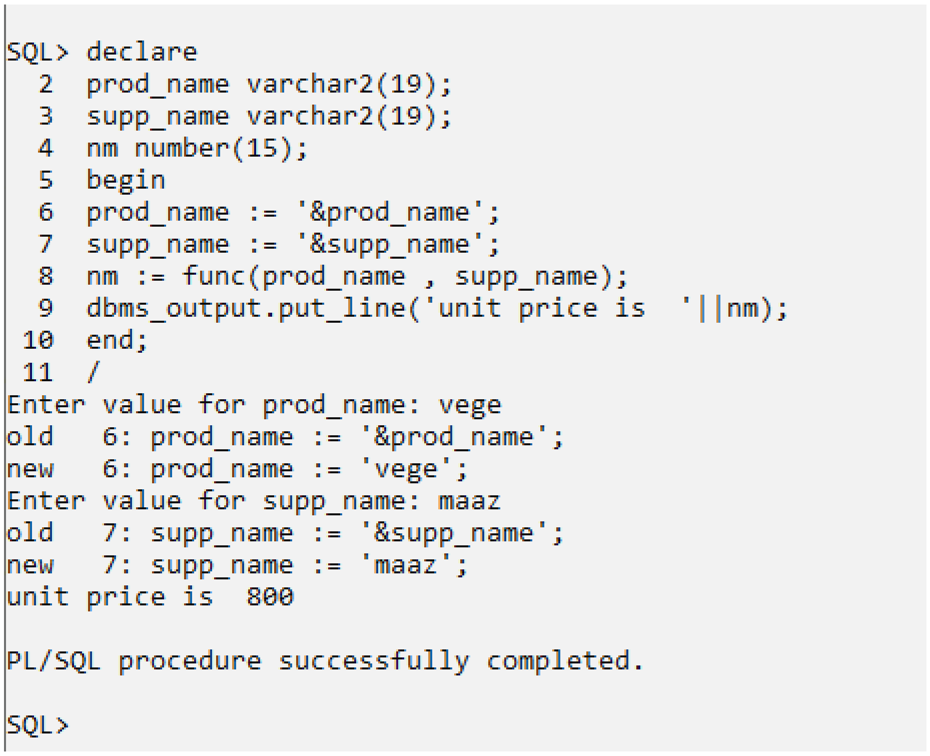
Output:


```
SQL> create or replace function func(product_name in varchar2 , supplier_name in varchar
2)return number
  2  is pnm number(15);
  3  begin
  4  select unit_price into pnm from product1 where product_name=product_name and suppli
er_name = supplier_name;
  5  return pnm;
  6  end;
  7  /

Function created.
```

Code:

```
declare
prod_name varchar2(19);
supp_name varchar2(19);
nm number(15);
begin
prod_name := '&prod_name';
supp_name := '&supp_name';
nm := func(prod_name , supp_name);
dbms_output.put_line('unit price is  '||nm);
end;
/
```

output:



```
SQL> declare
  2  prod_name varchar2(19);
  3  supp_name varchar2(19);
  4  nm number(15);
  5  begin
  6  prod_name := '&prod_name';
  7  supp_name := '&supp_name';
  8  nm := func(prod_name , supp_name);
  9  dbms_output.put_line('unit price is  '||nm);
 10  end;
 11  /
Enter value for prod_name: vege
old   6: prod_name := '&prod_name';
new   6: prod_name := 'vege';
Enter value for supp_name: maaz
old   7: supp_name := '&supp_name';
new   7: supp_name := 'maaz';
unit price is  800

PL/SQL procedure successfully completed.

SQL>
```

Practical 19:Procedure

Description:
It explicitly does not return the values

Syntax:
```
CREATE [OR RPLACE] PROCEDURE procedure_name
[(parameter_name[IN |OUT|IN OUT] type[,...])]
{IS|AS}
BEGIN
<procedure_body>
END procedure_name;
```

procedure-name specifies the name of the procedure.
[OR REPLACE] option allows modifying an existing procedure
IN represents that value will be passed from

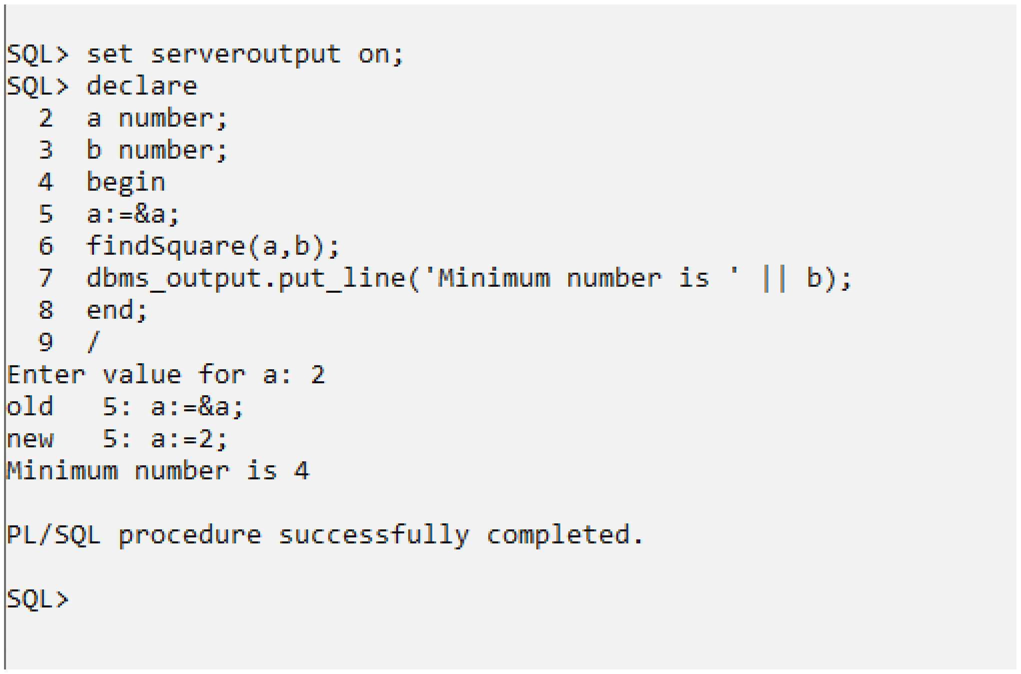
example:
```
CREATE OR REPLACE PROCEDURE greetings
AS
BEGIN
        dbms_output.put_line{'Hello World');
END;

EXECUTE greetings'
OR
BEGIN
greetings;
END;
/
```
2) Create parametric Function:

write a procedure to display square of number
code:
```
create or replace procedure findsquare(x in number, res out number)
```

```
is
begin
res := x*x;
end;
/
```

```
declare
a number;
b number;
begin
a:=&a;
findSquare(a,b);
dbms_output.put_line('Minimum number is ' || b);
end;
/
```

Output:



```
SQL> set serveroutput on;
SQL> declare
  2  a number;
  3  b number;
  4  begin
  5  a:=&a;
  6  findSquare(a,b);
  7  dbms_output.put_line('Minimum number is ' || b);
  8  end;
  9  /
Enter value for a: 2
old   5: a:=&a;
new   5: a:=2;
Minimum number is 4

PL/SQL procedure successfully completed.

SQL>
```

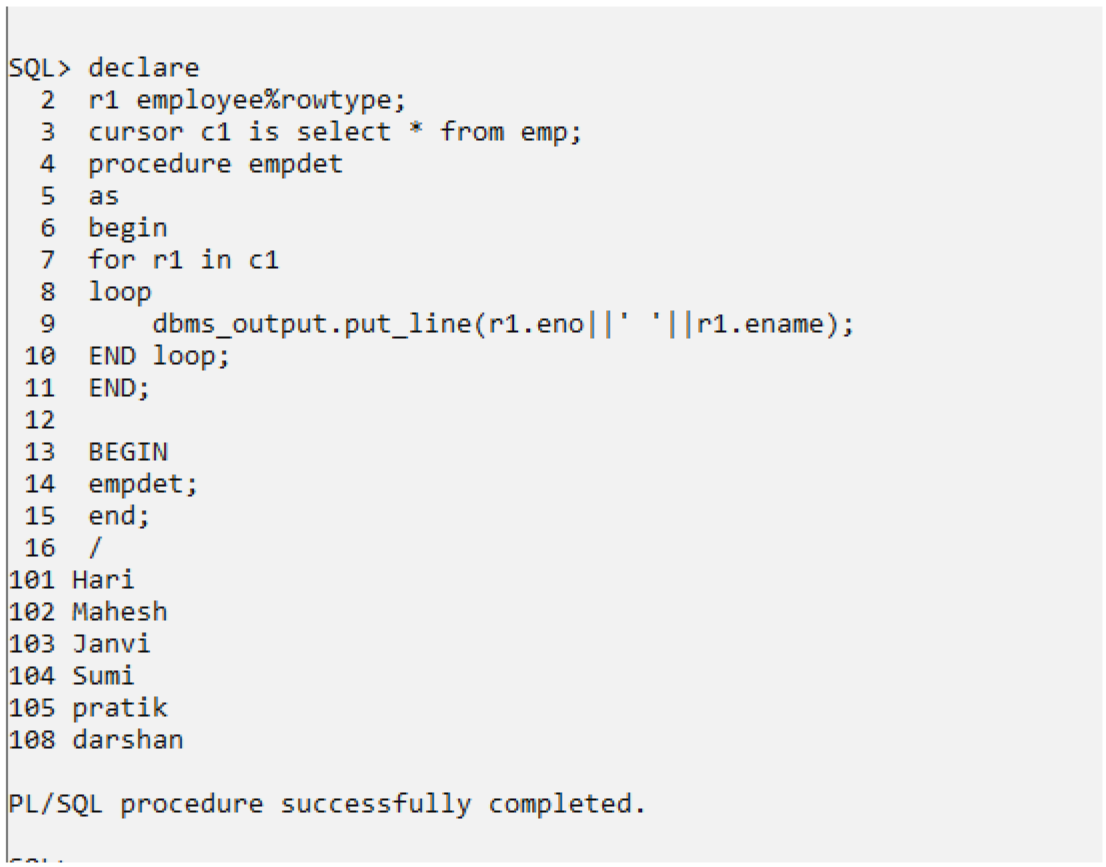
employee code for

Code:

```
declare
r1 employee%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:



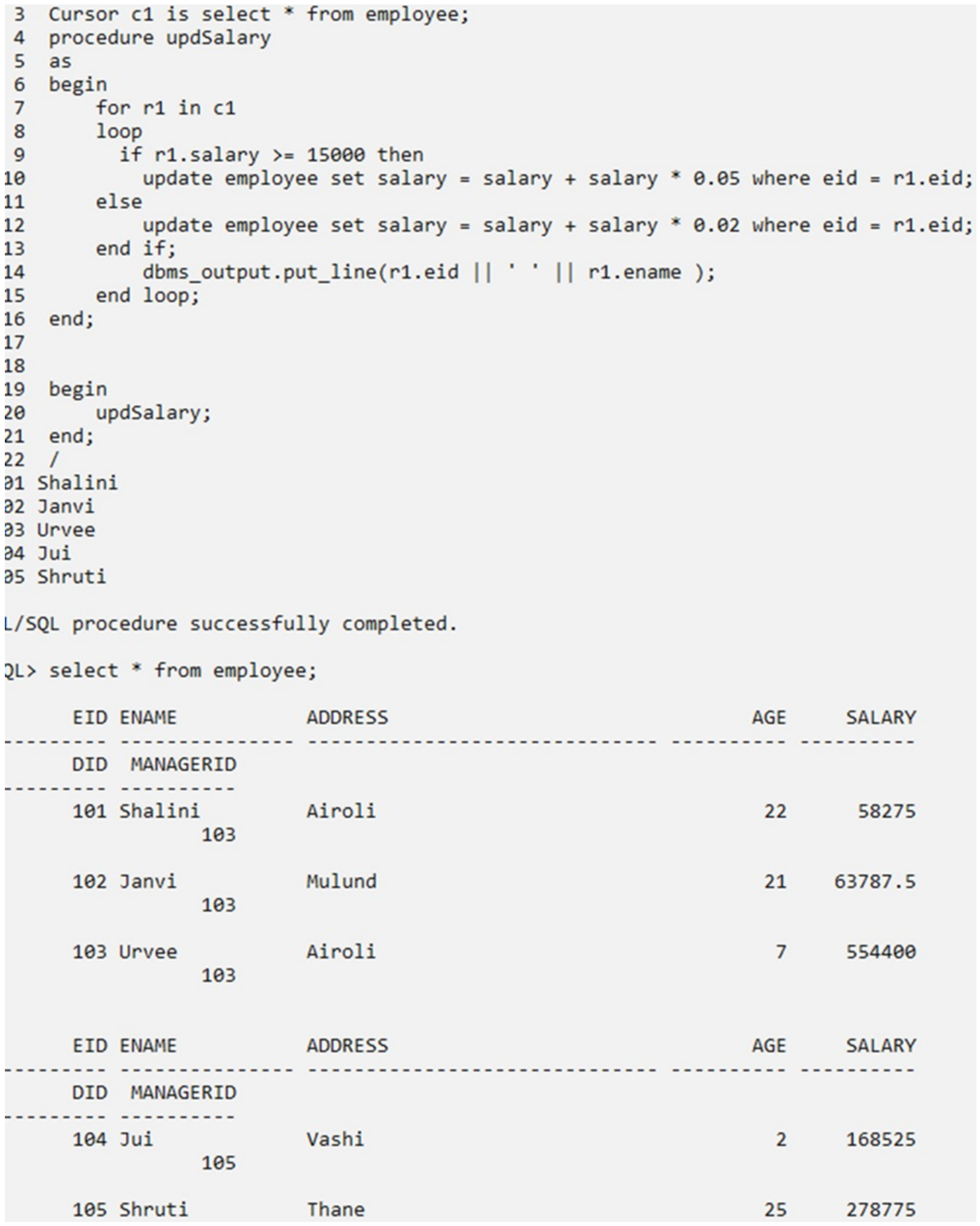
```
SQL> declare
r1 employee%rowtype;
cursor c1 is select * from emp;
procedure empdet
as
begin
for r1 in c1
loop
    dbms_output.put_line(r1.eno||' '||r1.ename);
END loop;
END;

BEGIN
empdet;
end;
/
101 Hari
102 Mahesh
103 Janvi
104 Sumi
105 pratik
108 darshan

PL/SQL procedure successfully completed.
```

4. Create a procedure salary update which acceptsempno and sal>15000 increase by 5% else by 2%

Code:

```
Cursor c1 is select * from employee;
Procedure updsalary
As
Begin
      For r1 in c1
      Loop
      If r1.salary >= 15000 then
            Update employee set salary = salary + salary * 0.05 where eid =
r1.eid;
      Else
            Update employee set salary = salary + salary * 0.02 where eid =
r1.eid;
      End if;
```

```
        dbms_output.put _line(r1.eid||’ ‘ ||r1.ename);
End loop;
end;
```



```
Cursor c1 is select * from employee;
procedure updSalary
as
begin
    for r1 in c1
    loop
      if r1.salary >= 15000 then
        update employee set salary = salary + salary * 0.05 where eid = r1.eid;
    else
        update employee set salary = salary + salary * 0.02 where eid = r1.eid;
    end if;
        dbms_output.put_line(r1.eid || ' ' || r1.ename );
    end loop;
end;


begin
    updSalary;
end;
/
01 Shalini
02 Janvi
03 Urvee
04 Jui
05 Shruti

L/SQL procedure successfully completed.

QL> select * from employee;

      EID ENAME          ADDRESS                               AGE     SALARY
--------- -------------- ------------------------------ ---------- ----------
      DID  MANAGERID
--------- ----------
      101 Shalini        Airoli                                 22      58275
                 103

      102 Janvi          Mulund                                 21    63787.5
                 103

      103 Urvee          Airoli                                  7     554400
                 103


      EID ENAME          ADDRESS                               AGE     SALARY
--------- -------------- ------------------------------ ---------- ----------
      DID  MANAGERID
--------- ----------
      104 Jui            Vashi                                   2     168525
                 105

      105 Shruti         Thane                                  25     278775
```

Practical 20: Packages

Code:

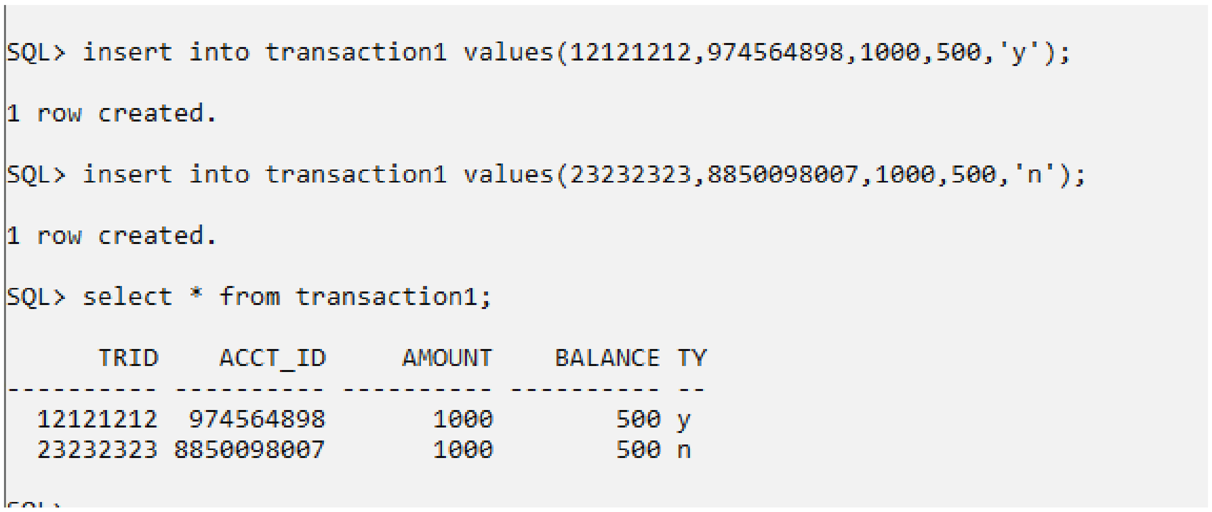
create the relation

create table transaction1(trid int,acct_id number(10),amount number(10,2),balance number(10,2),typ char(2));

insert into transaction1 values(12121212,974564898,1000,500,'y');

insert into transaction1 values(23232323,8850098007,1000,500,'n');

Output:



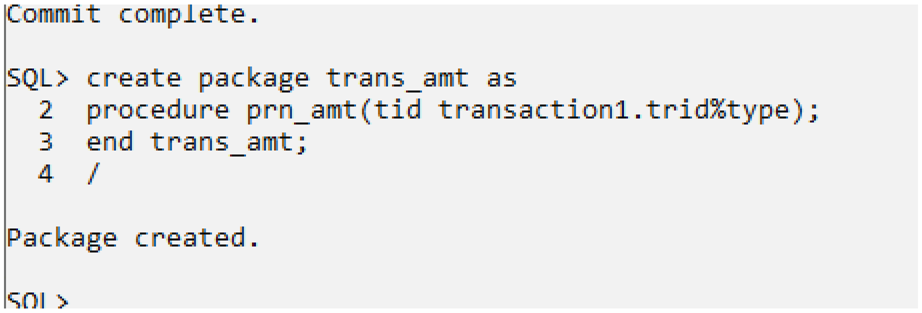
```
SQL> insert into transaction1 values(12121212,974564898,1000,500,'y');

1 row created.

SQL> insert into transaction1 values(23232323,8850098007,1000,500,'n');

1 row created.

SQL> select * from transaction1;

      TRID    ACCT_ID     AMOUNT    BALANCE TY
---------- ---------- ---------- ---------- --
  12121212  974564898       1000        500 y
  23232323 8850098007       1000        500 n
```

code for creating procedure:

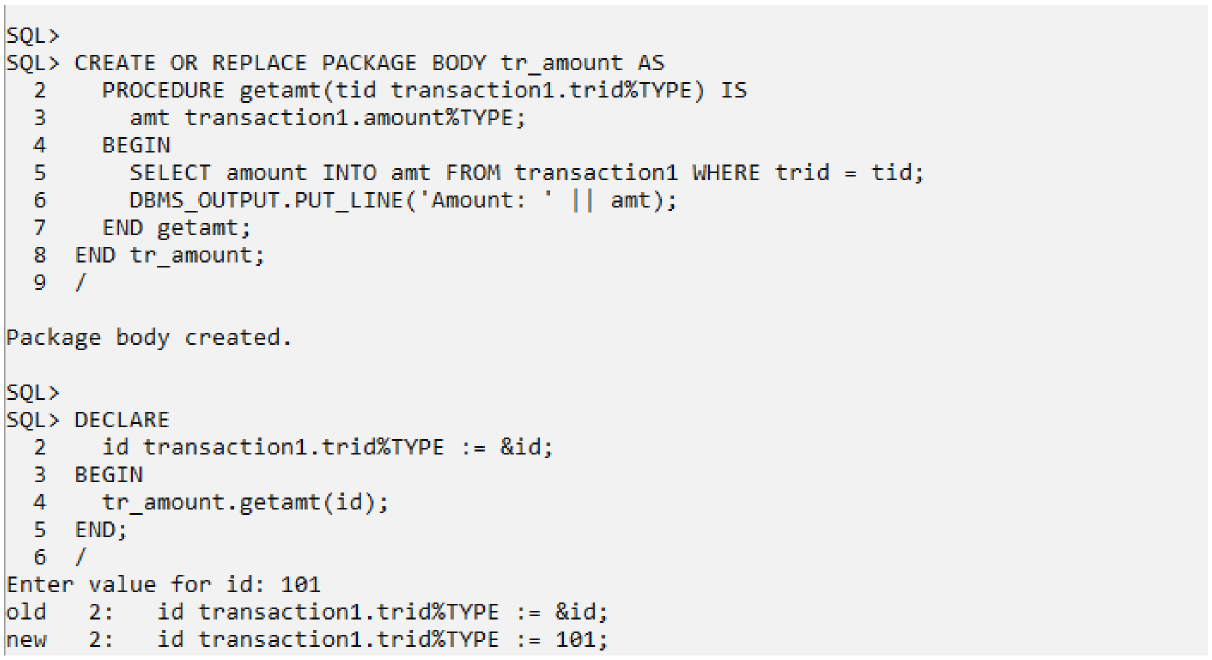
create package trans_amt as

procedure prn_amt(tid transaction1.trid%type);

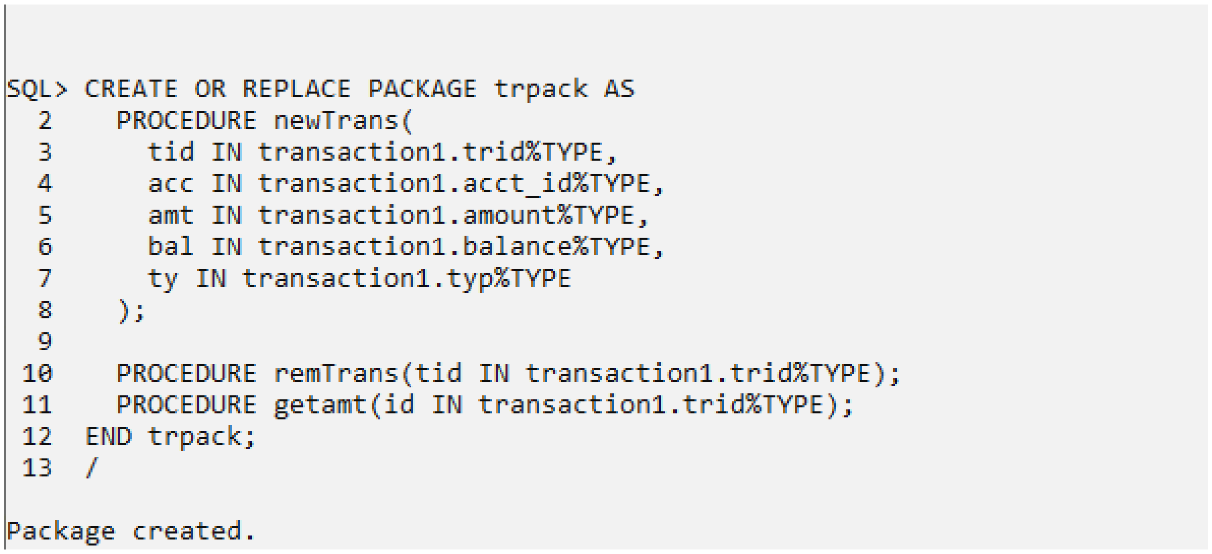
end trans_amt;



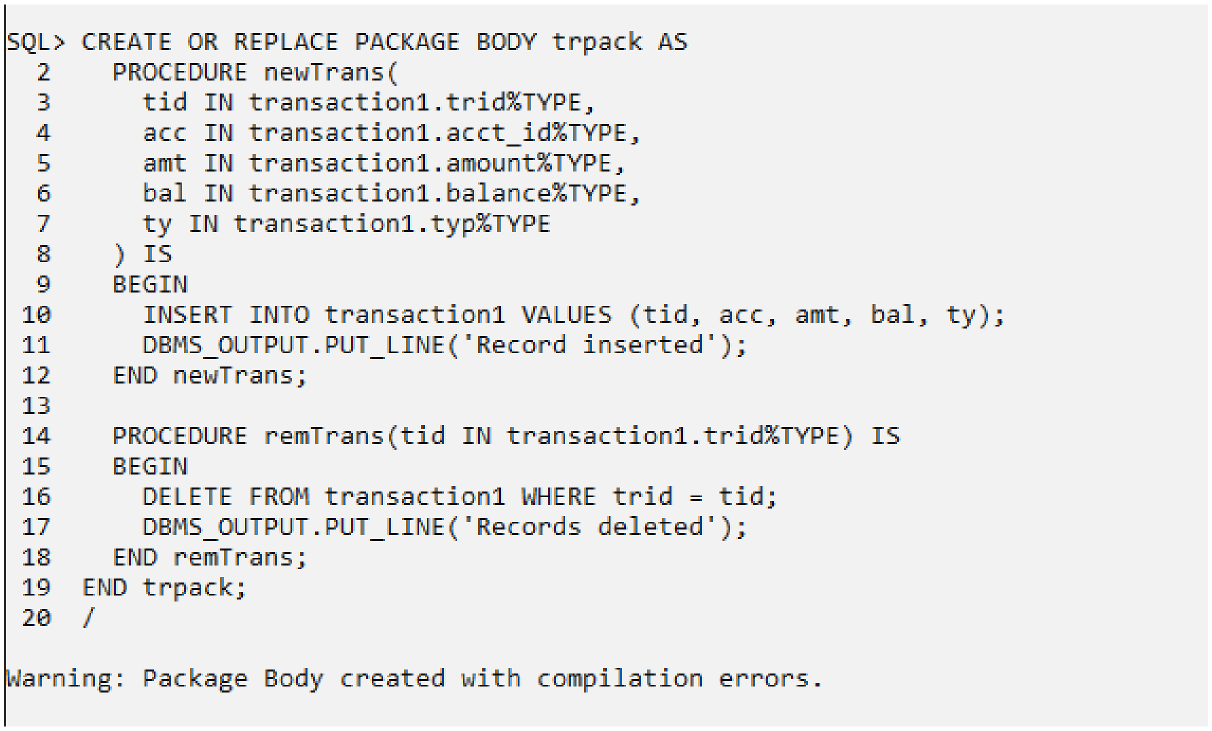
```
Commit complete.

SQL> create package trans_amt as
  2  procedure prn_amt(tid transaction1.trid%type);
  3  end trans_amt;
  4  /

Package created.

SQL>
```



```
SQL>
SQL> CREATE OR REPLACE PACKAGE BODY tr_amount AS
  2    PROCEDURE getamt(tid transaction1.trid%TYPE) IS
  3      amt transaction1.amount%TYPE;
  4    BEGIN
  5      SELECT amount INTO amt FROM transaction1 WHERE trid = tid;
  6      DBMS_OUTPUT.PUT_LINE('Amount: ' || amt);
  7    END getamt;
  8  END tr_amount;
  9  /

Package body created.

SQL>
SQL> DECLARE
  2    id transaction1.trid%TYPE := &id;
  3  BEGIN
  4    tr_amount.getamt(id);
  5  END;
  6  /
Enter value for id: 101
old   2:   id transaction1.trid%TYPE := &id;
new   2:   id transaction1.trid%TYPE := 101;
```



```
SQL> CREATE OR REPLACE PACKAGE trpack AS
  2    PROCEDURE newTrans(
  3      tid IN transaction1.trid%TYPE,
  4      acc IN transaction1.acct_id%TYPE,
  5      amt IN transaction1.amount%TYPE,
  6      bal IN transaction1.balance%TYPE,
  7      ty IN transaction1.typ%TYPE
  8    );
  9
 10    PROCEDURE remTrans(tid IN transaction1.trid%TYPE);
 11    PROCEDURE getamt(id IN transaction1.trid%TYPE);
 12  END trpack;
 13  /

Package created.
```



```
SQL> CREATE OR REPLACE PACKAGE BODY trpack AS
  2    PROCEDURE newTrans(
  3      tid IN transaction1.trid%TYPE,
  4      acc IN transaction1.acct_id%TYPE,
  5      amt IN transaction1.amount%TYPE,
  6      bal IN transaction1.balance%TYPE,
  7      ty IN transaction1.typ%TYPE
  8    ) IS
  9    BEGIN
 10      INSERT INTO transaction1 VALUES (tid, acc, amt, bal, ty);
 11      DBMS_OUTPUT.PUT_LINE('Record inserted');
 12    END newTrans;
 13
 14    PROCEDURE remTrans(tid IN transaction1.trid%TYPE) IS
 15    BEGIN
 16      DELETE FROM transaction1 WHERE trid = tid;
 17      DBMS_OUTPUT.PUT_LINE('Records deleted');
 18    END remTrans;
 19  END trpack;
 20  /

Warning: Package Body created with compilation errors.
```