18CSC303J – DATABASE MANGEMENT SYSTEM LABORATORY RECORD

ACADEMIC YEAR 2021-2022, EVEN SEMESTER
DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

NAME	**:**	**SHREYANSH SACHAN**
REGISTRATION NUMBER	**:**	**RA1911031010070**
YEAR/SEMESTER	**:**	**III / VI**
SECTION	**:**	**L1**

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING
SRM INSTITUTE ODF SCIENCE AND TECHNOLOGY
SRM NAGAR, KATTANKULATHUR – 603203
KANCHEEPURAM DISTRICT
MAY - 2022

LIST OF EXPERIMENTS & SCHEDULE

Course Code: 18CSC303J

Course Title: Database Management System Laboratory

Exp. No.	Title
1	SQL Data Definition Language (DDL)
2	SQL Data Manipulation Language (DML)
3	SQL Data Control Language Commands and Transaction Control Commands
4	Inbuilt functions in SQl
5	ER Diagram
6	Nested Queries
7	Join Queries
8	Set Operations and Views
9	PL/SQL Conditional
10	PL/SQL Triggers

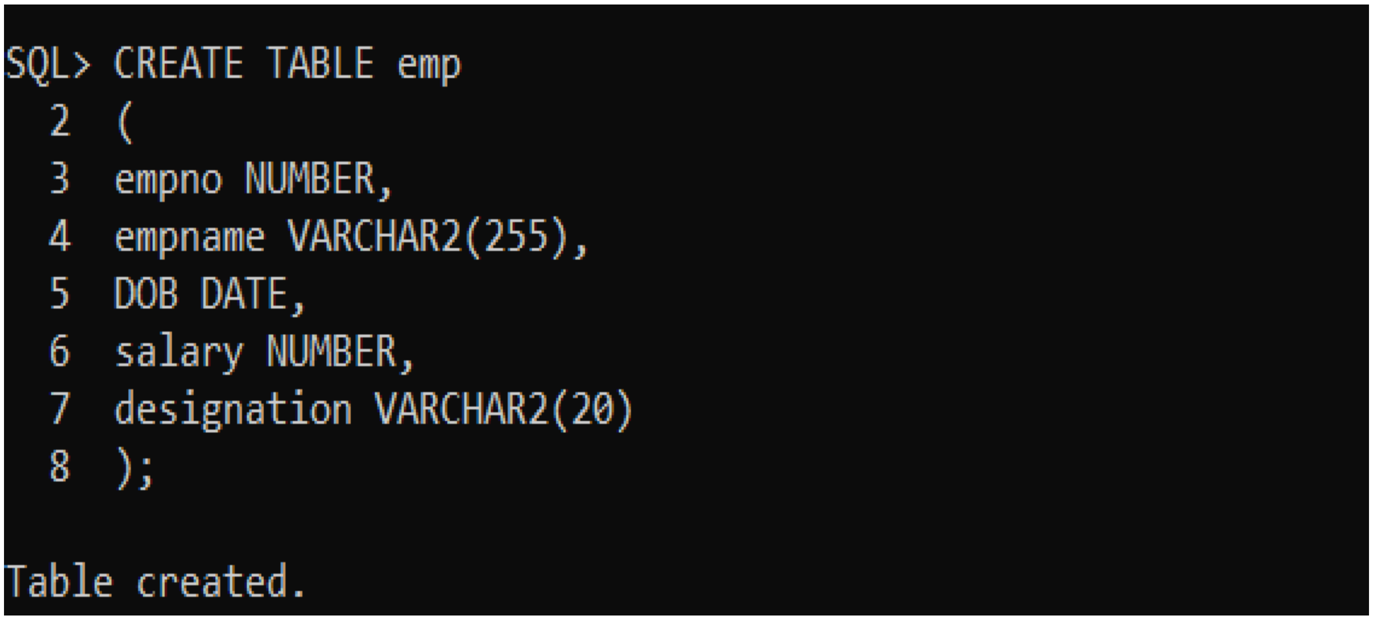
Course Coordinator **Ms.S.Ushasukhanya**

Experiment – 1 SQL BASIC COMMANDS

_

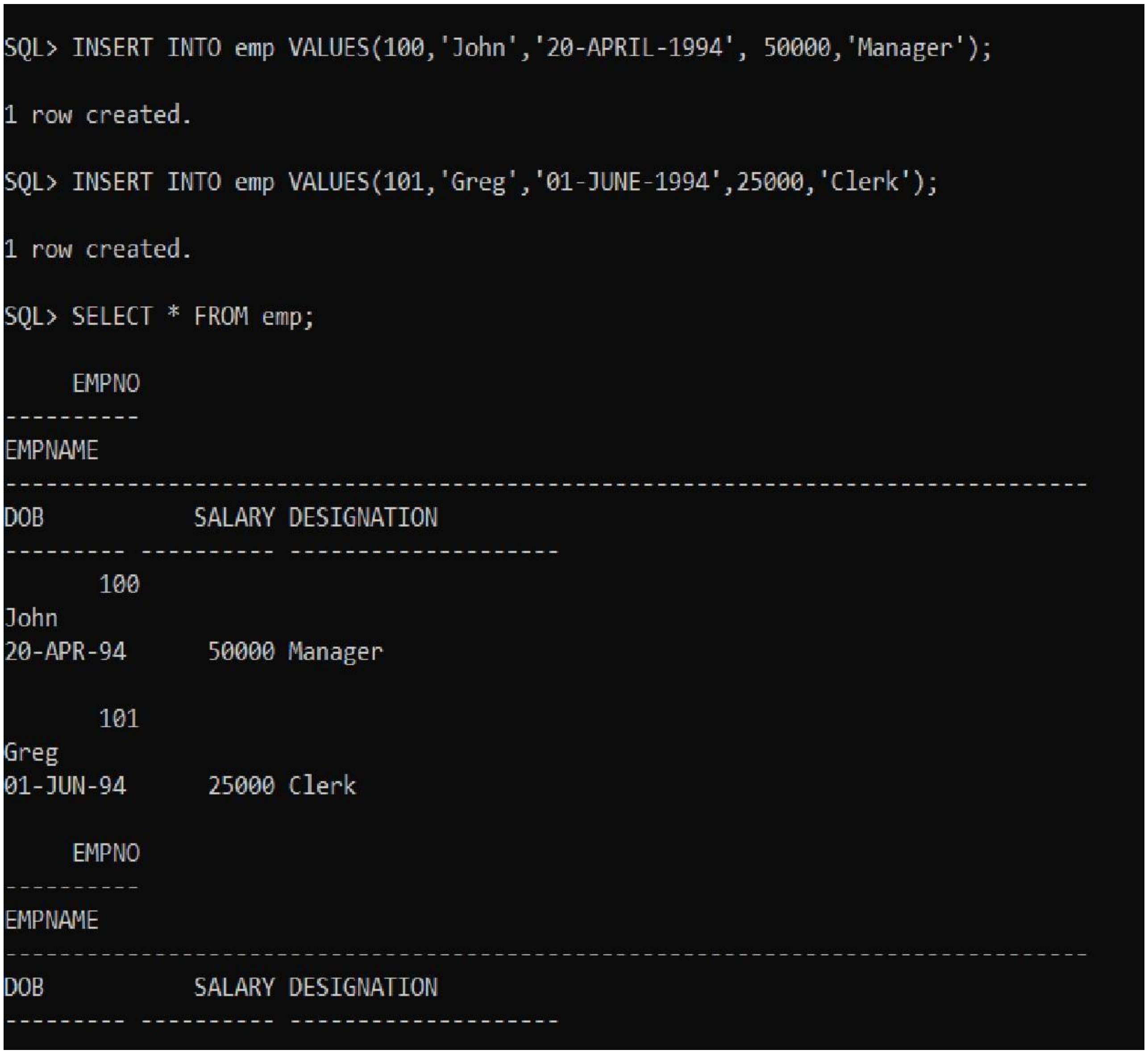
QUERIES:

1. **Create table**



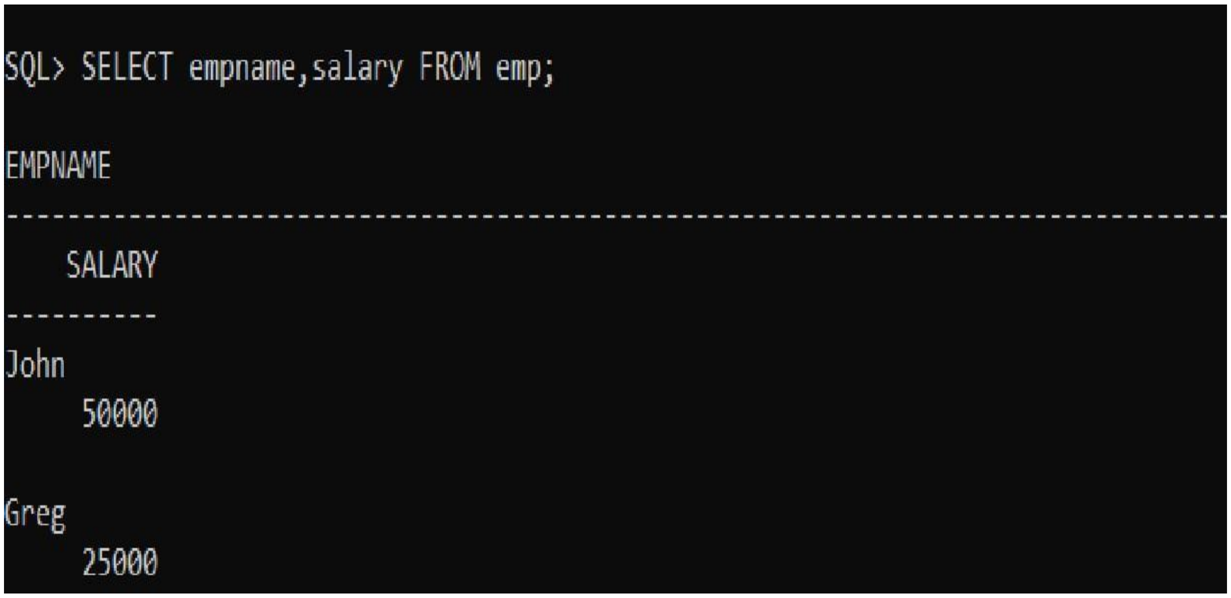
```
SQL> CREATE TABLE emp
  2  (
  3  empno NUMBER,
  4  empname VARCHAR2(255),
  5  DOB DATE,
  6  salary NUMBER,
  7  designation VARCHAR2(20)
  8  );

Table created.
```

2. **Insert values**



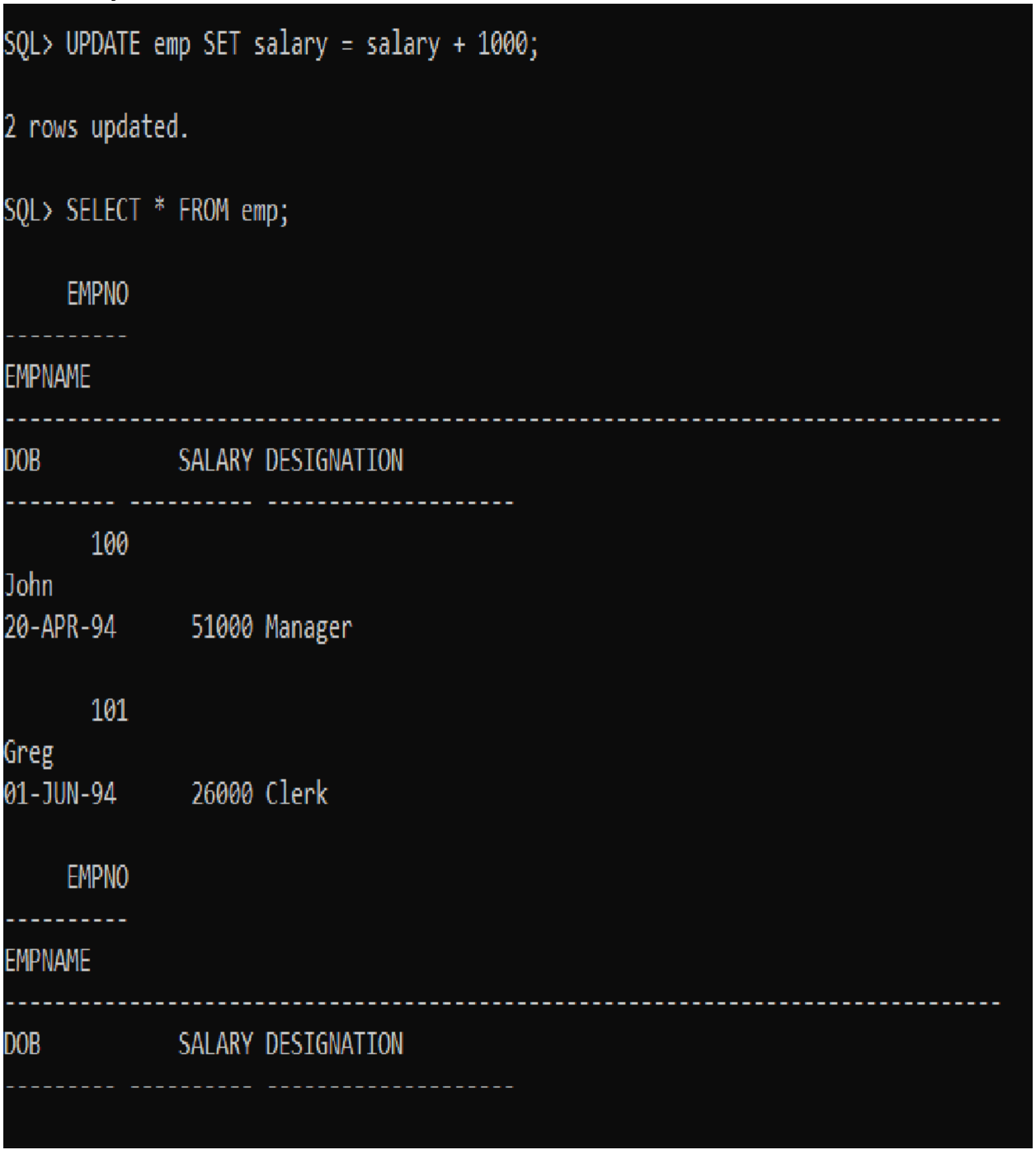
```
SQL> INSERT INTO emp VALUES(100,'John','20-APRIL-1994', 50000,'Manager');

1 row created.

SQL> INSERT INTO emp VALUES(101,'Greg','01-JUNE-1994',25000,'Clerk');

1 row created.

SQL> SELECT * FROM emp;

     EMPNO
----------
EMPNAME
--------------------------------------------------------------------------------
DOB           SALARY DESIGNATION
--------- ---------- --------------------
       100
John
20-APR-94      50000 Manager

       101
Greg
01-JUN-94      25000 Clerk

     EMPNO
----------
EMPNAME
--------------------------------------------------------------------------------
DOB           SALARY DESIGNATION
--------- ---------- --------------------
```

3. **Display values**



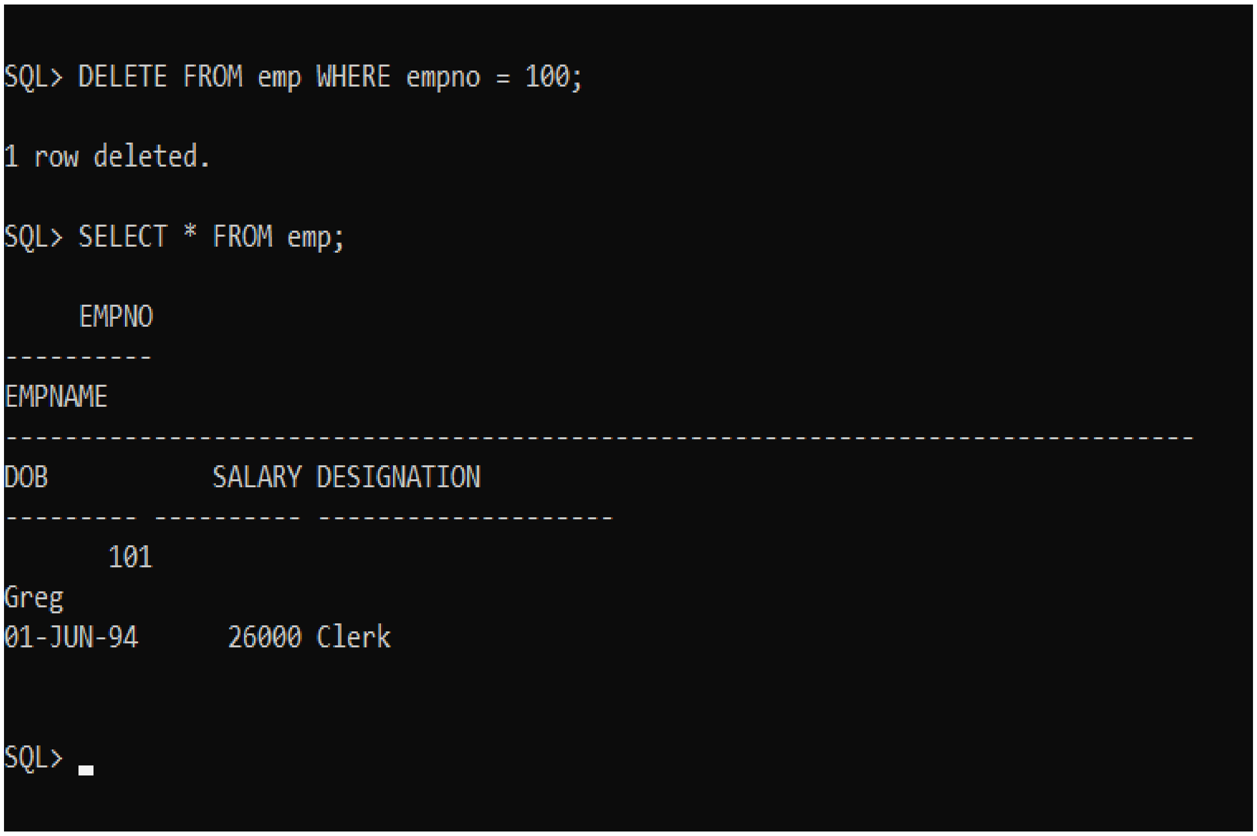
```
SQL> SELECT empname,salary FROM emp;

EMPNAME
--------------------------------------------------------------------------------
    SALARY
----------
John
     50000

Greg
     25000
```

4. Modify values



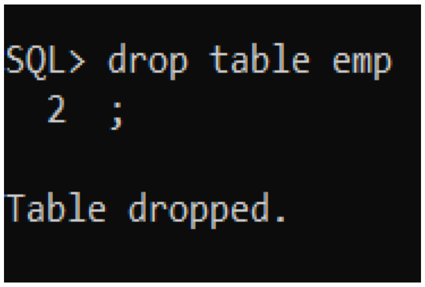
```
SQL> UPDATE emp SET salary = salary + 1000;

2 rows updated.

SQL> SELECT * FROM emp;

     EMPNO
----------
EMPNAME
--------------------------------------------------------------------------------
DOB           SALARY DESIGNATION
--------- ---------- --------------------
       100
John
20-APR-94      51000 Manager

       101
Greg
01-JUN-94      26000 Clerk

     EMPNO
----------
EMPNAME
--------------------------------------------------------------------------------
DOB           SALARY DESIGNATION
--------- ---------- --------------------
```

5. Delete values



```
SQL> DELETE FROM emp WHERE empno = 100;

1 row deleted.

SQL> SELECT * FROM emp;

     EMPNO
----------
EMPNAME
--------------------------------------------------------------------------------
DOB           SALARY DESIGNATION
--------- ---------- --------------------
       101
Greg
01-JUN-94      26000 Clerk


SQL>
```

6. Drop Table

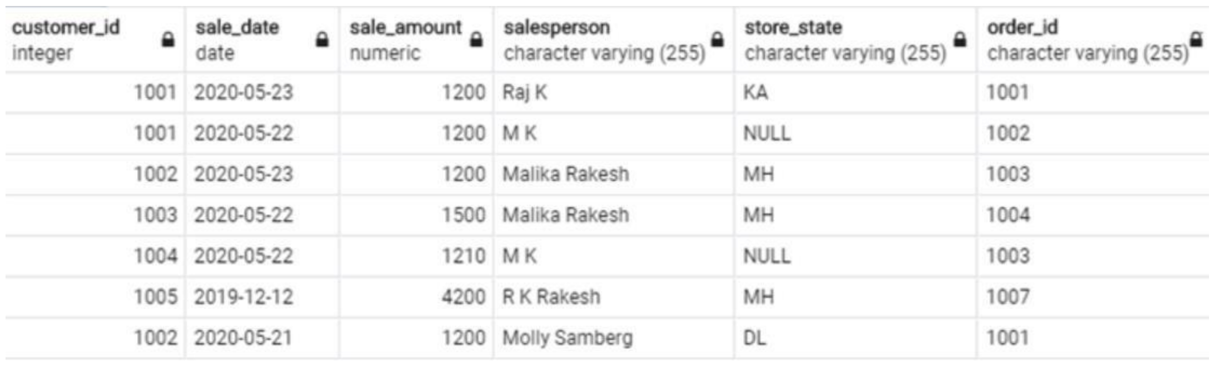


```
SQL> drop table emp
  2  ;

Table dropped.
```

_

Experiment – 2 SQL DML COMMANDS

Data base created for this exercise is:



customer_id integer	sale_date date	sale_amount numeric	salesperson character varying (255)	store_state character varying (255)	order_id character varying (255)
1001	2020-05-23	1200	Raj K	KA	1001
1001	2020-05-22	1200	M K	NULL	1002
1002	2020-05-23	1200	Malika Rakesh	MH	1003
1003	2020-05-22	1500	Malika Rakesh	MH	1004
1004	2020-05-22	1210	M K	NULL	1003
1005	2019-12-12	4200	R K Rakesh	MH	1007
1002	2020-05-21	1200	Molly Samberg	DL	1001

DML Commands:

- **INSERT -** Used to insert new data records or rows in the database table

  Syntax,

  INSERT INTO table_name (column_name_1, column_name_2, column_name_3, ...)

  VALUES (value1, value2, value3, ...)

  Example:

INSERT INTO customers(

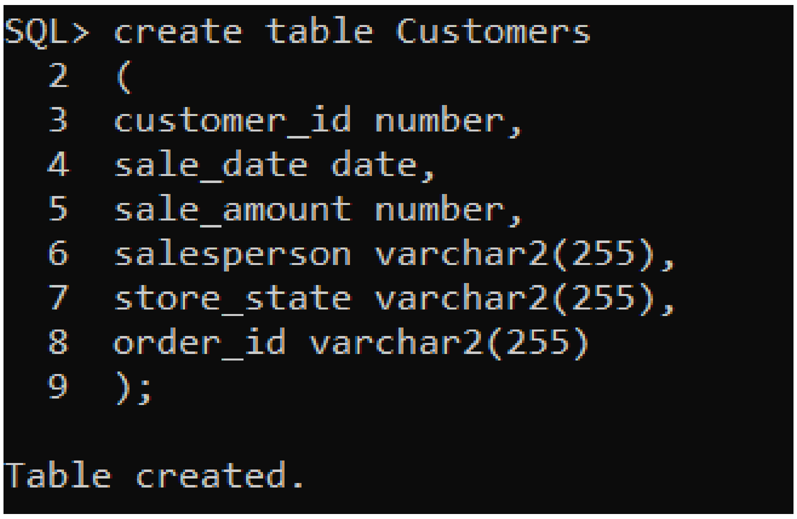
customer_id, sale_date, sale_amount, salesperson, store_state, order_id)

VALUES (1005,'12-DECEMBER-2019',4200,'R K Rakesh','MH','1007');

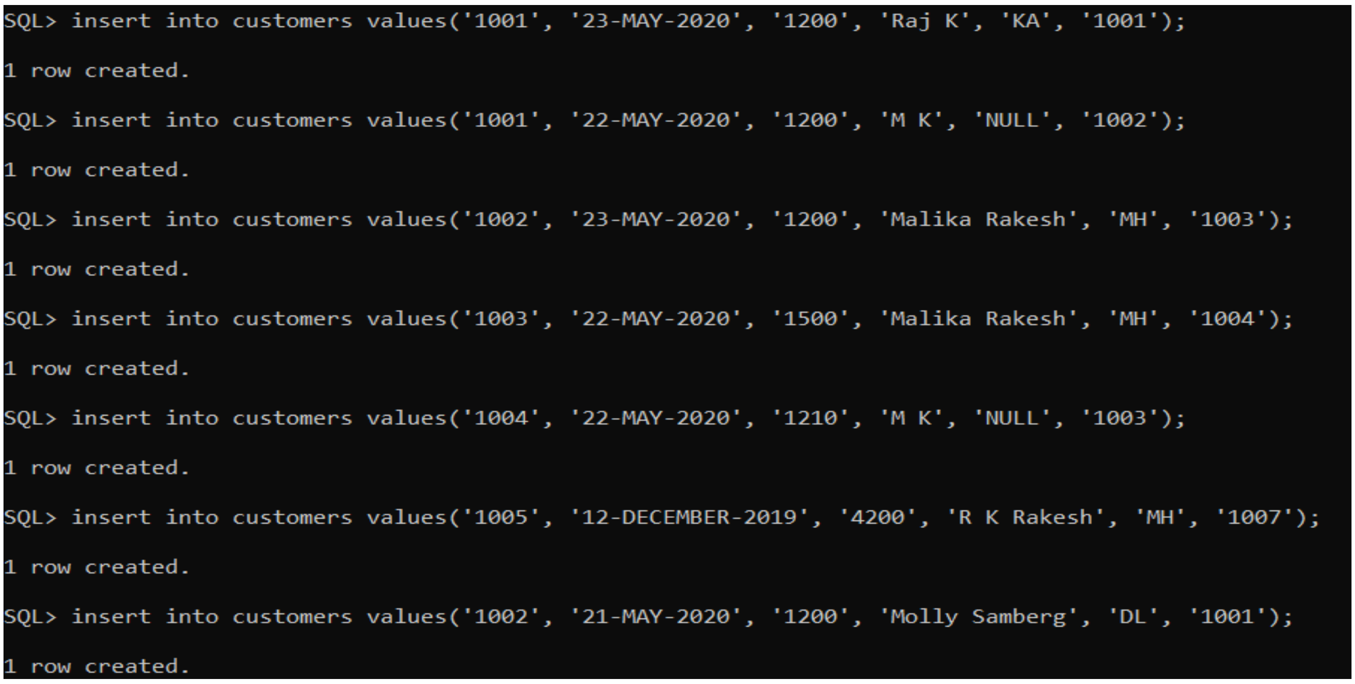
(or)

INSERT INTO customers

VALUES ('1006','4-MARCH-2020',3200,'DL', '1008');



```
SQL> create table Customers
  2  (
  3  customer_id number,
  4  sale_date date,
  5  sale_amount number,
  6  salesperson varchar2(255),
  7  store_state varchar2(255),
  8  order_id varchar2(255)
  9  );

Table created.
```



```
SQL> insert into customers values('1001', '23-MAY-2020', '1200', 'Raj K', 'KA', '1001');

1 row created.

SQL> insert into customers values('1001', '22-MAY-2020', '1200', 'M K', 'NULL', '1002');

1 row created.

SQL> insert into customers values('1002', '23-MAY-2020', '1200', 'Malika Rakesh', 'MH', '1003');

1 row created.

SQL> insert into customers values('1003', '22-MAY-2020', '1500', 'Malika Rakesh', 'MH', '1004');

1 row created.

SQL> insert into customers values('1004', '22-MAY-2020', '1210', 'M K', 'NULL', '1003');

1 row created.

SQL> insert into customers values('1005', '12-DECEMBER-2019', '4200', 'R K Rakesh', 'MH', '1007');

1 row created.

SQL> insert into customers values('1002', '21-MAY-2020', '1200', 'Molly Samberg', 'DL', '1001');

1 row created.
```

- **SELECT -** Used to query or fetch selected fields or columns from a database table

  Syntax:

  SELECT column_name1, column_name2, …

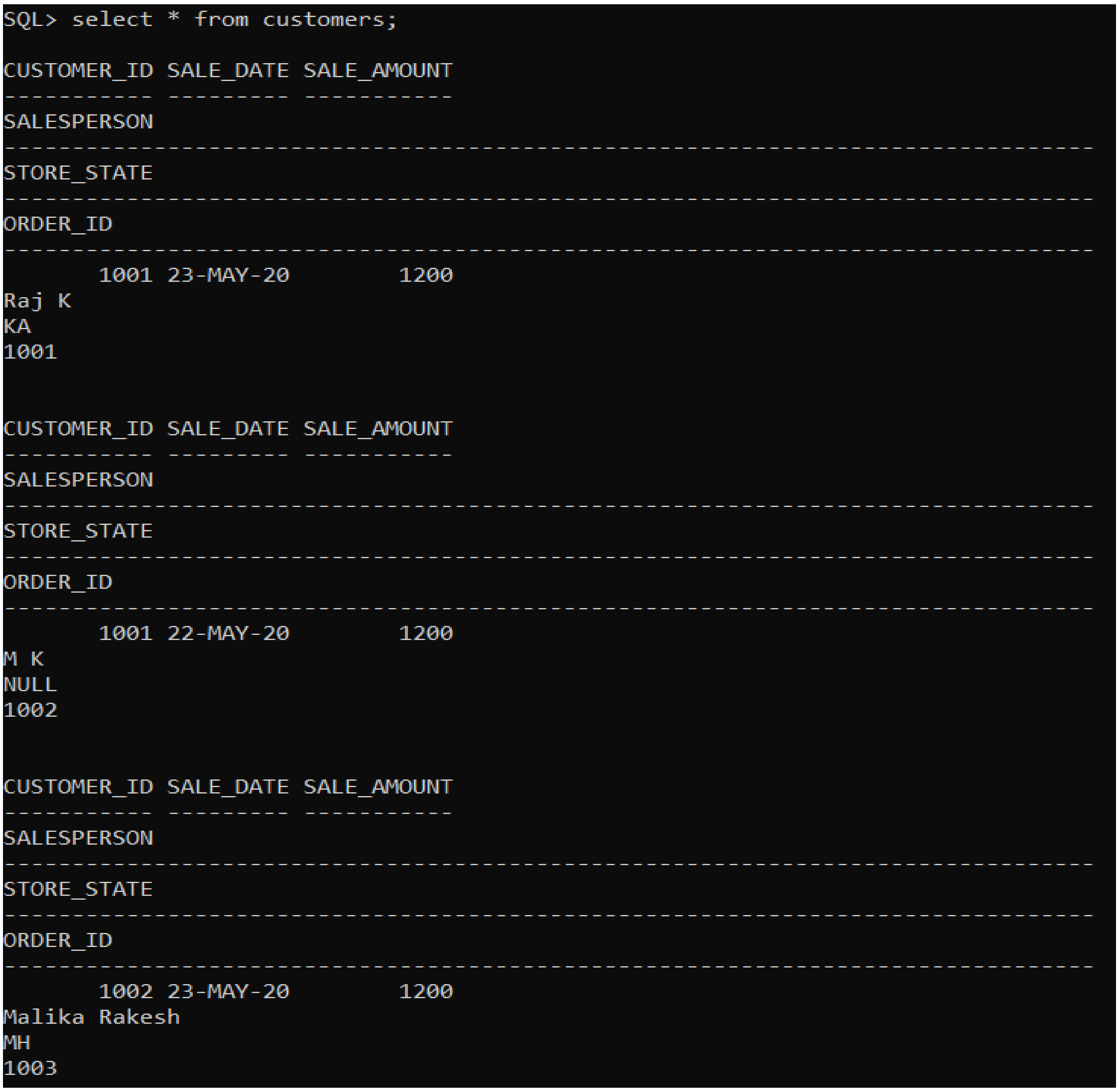
  FROM table_name

  WHERE condition_ expression;

  Example:

  Select customer_id, sale_date, order_id, store_state from customers;

  Select * from customers;



```
SQL> select * from customers;

CUSTOMER_ID SALE_DATE SALE_AMOUNT
----------- --------- -----------
SALESPERSON
--------------------------------------------------------------------------------
STORE_STATE
--------------------------------------------------------------------------------
ORDER_ID
--------------------------------------------------------------------------------
       1001 23-MAY-20        1200
Raj K
KA
1001


CUSTOMER_ID SALE_DATE SALE_AMOUNT
----------- --------- -----------
SALESPERSON
--------------------------------------------------------------------------------
STORE_STATE
--------------------------------------------------------------------------------
ORDER_ID
--------------------------------------------------------------------------------
       1001 22-MAY-20        1200
M K
NULL
1002


CUSTOMER_ID SALE_DATE SALE_AMOUNT
----------- --------- -----------
SALESPERSON
--------------------------------------------------------------------------------
STORE_STATE
--------------------------------------------------------------------------------
ORDER_ID
--------------------------------------------------------------------------------
       1002 23-MAY-20        1200
Malika Rakesh
MH
1003
```

- **UPDATE -** Used to set the value of a field or column for a particular record to a new value

  Syntax:

  UPDATE table_name

  SET column_name_1 = value1, column_name_2 = value2, ...

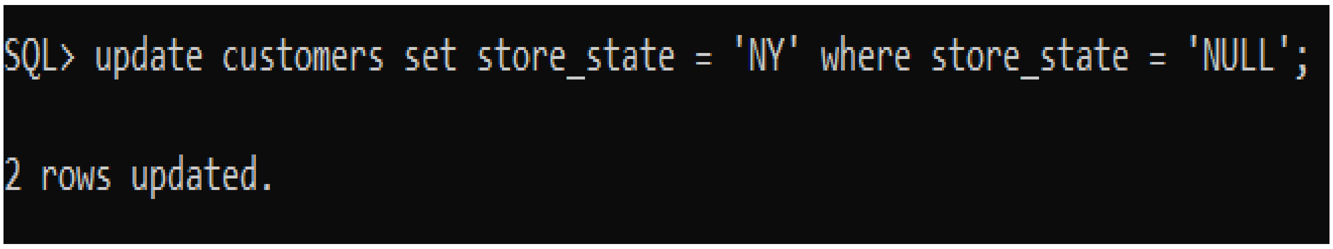
  WHERE condition;

  Example:

  UPDATE customers

  SET store_state = 'DL'

  WHERE store_state = 'NY';



```
SQL> update customers set store_state = 'NY' where store_state = 'NULL';

2 rows updated.
```

- **DELETE -** Used to remove one or more rows from the database table

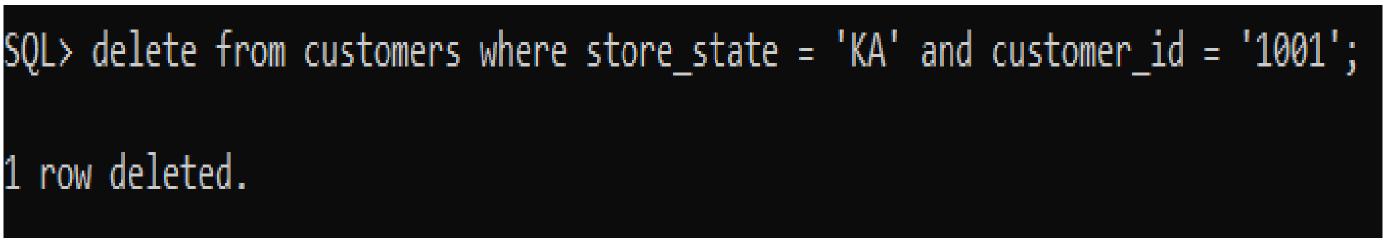
  Syntax:

  DELETE FROM table_name WHERE condition;

  Example:

  DELETE FROM customers

  WHERE store_state = 'MH'



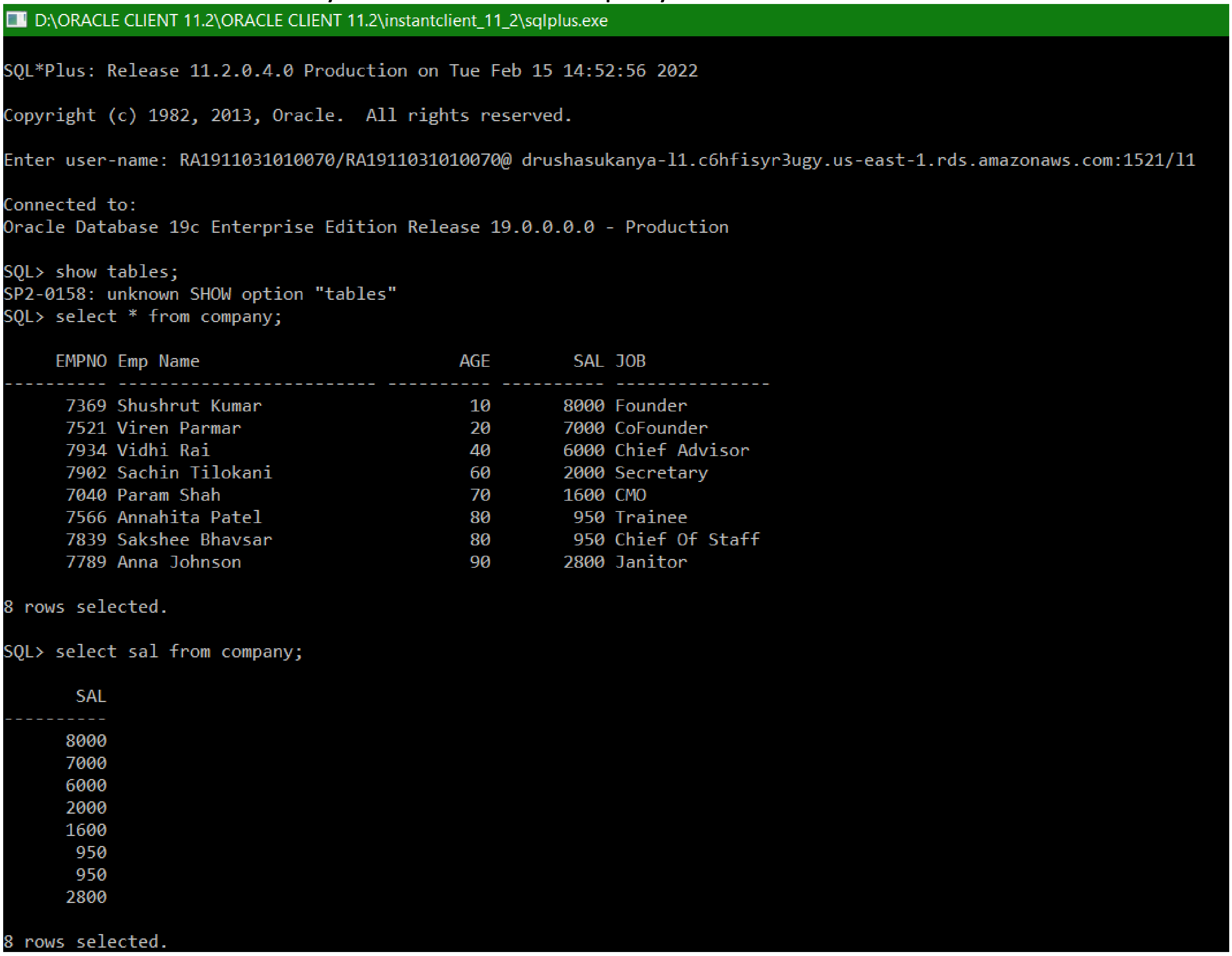
```
SQL> delete from customers where store_state = 'KA' and customer_id = '1001';

1 row deleted.
```

  AND customer_id = '1001';

Experiment – 3 DCL

Queries:

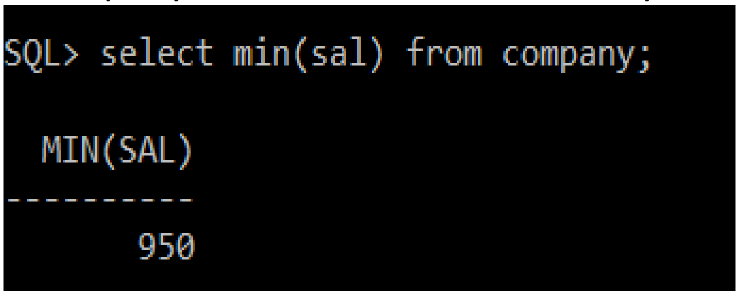
1. List the distinct salary records in the company table.



D:\ORACLE CLIENT 11.2\ORACLE CLIENT 11.2\instantclient_11_2\sqlplus.exe

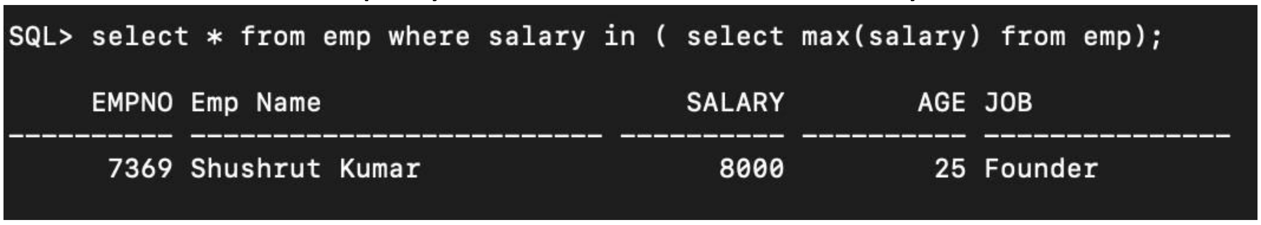
```
SQL*Plus: Release 11.2.0.4.0 Production on Tue Feb 15 14:52:56 2022

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

Enter user-name: RA1911031010070/RA1911031010070@ drushasukanya-l1.c6hfisyr3ugy.us-east-1.rds.amazonaws.com:1521/l1

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

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

     EMPNO Emp Name                         AGE        SAL JOB
---------- ------------------------- ---------- ---------- ---------------
      7369 Shushrut Kumar                    10       8000 Founder
      7521 Viren Parmar                      20       7000 CoFounder
      7934 Vidhi Rai                         40       6000 Chief Advisor
      7902 Sachin Tilokani                   60       2000 Secretary
      7040 Param Shah                        70       1600 CMO
      7566 Annahita Patel                    80        950 Trainee
      7839 Sakshee Bhavsar                   80        950 Chief Of Staff
      7789 Anna Johnson                      90       2800 Janitor

8 rows selected.

SQL> select sal from company;

       SAL
----------
      8000
      7000
      6000
      2000
      1600
       950
       950
      2800

8 rows selected.
```

2. List the records in the company table with minimum salary.



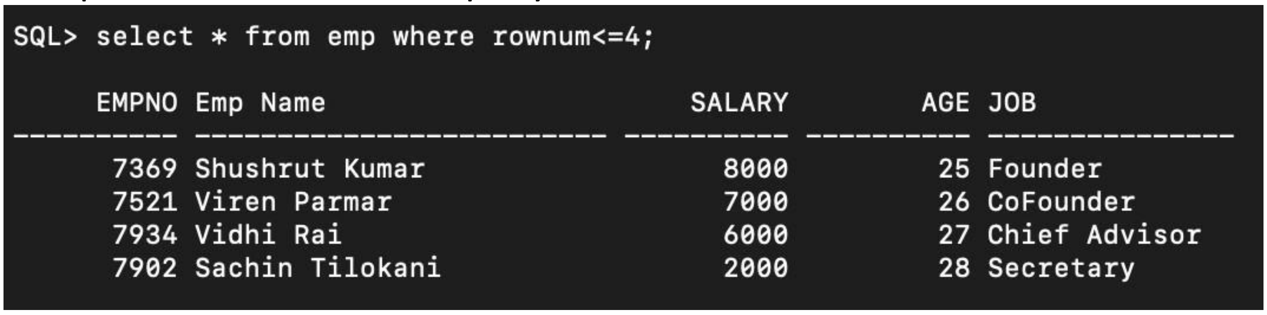
```
SQL> select min(sal) from company;

  MIN(SAL)
----------
       950
```

3. List the records in the company table with maximum salary.



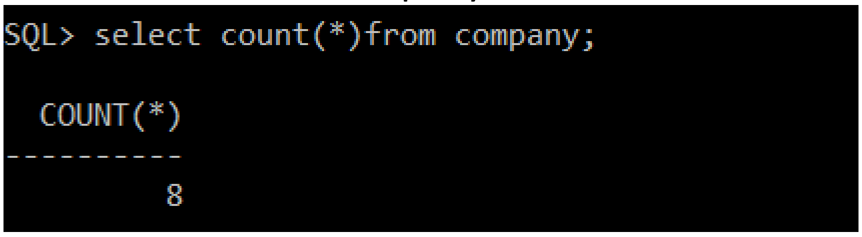
```
SQL> select * from emp where salary in ( select max(salary) from emp);

     EMPNO Emp Name                     SALARY        AGE JOB
---------- ------------------------- ---------- ---------- ---------------
      7369 Shushrut Kumar                 8000         25 Founder
```

4. List the top 4 records in the company table.



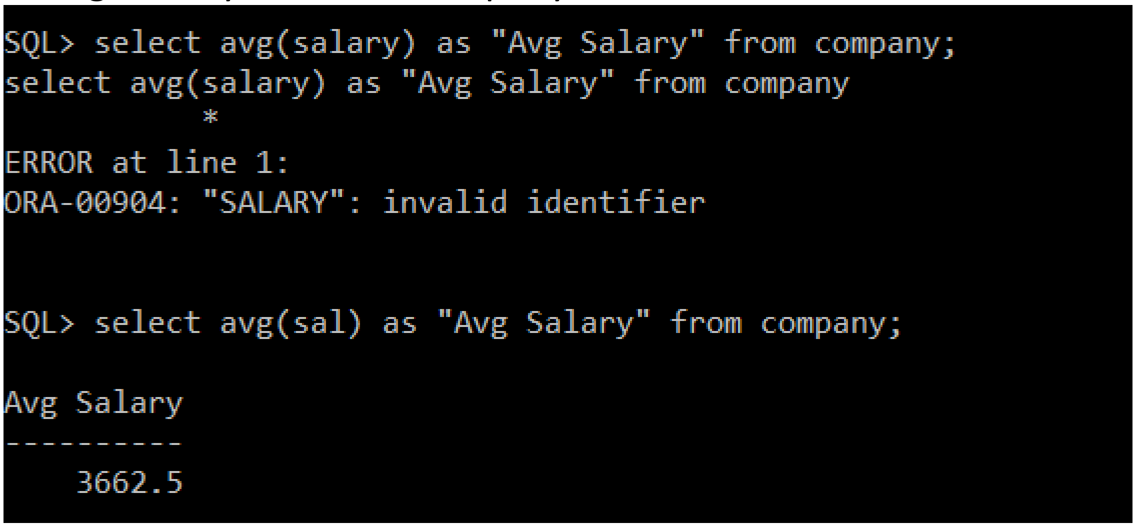
```
SQL> select * from emp where rownum<=4;

     EMPNO Emp Name                     SALARY        AGE JOB
---------- ------------------------- ---------- ---------- ---------------
      7369 Shushrut Kumar                 8000         25 Founder
      7521 Viren Parmar                   7000         26 CoFounder
      7934 Vidhi Rai                      6000         27 Chief Advisor
      7902 Sachin Tilokani                2000         28 Secretary
```

5. Count the number of records in the company table.



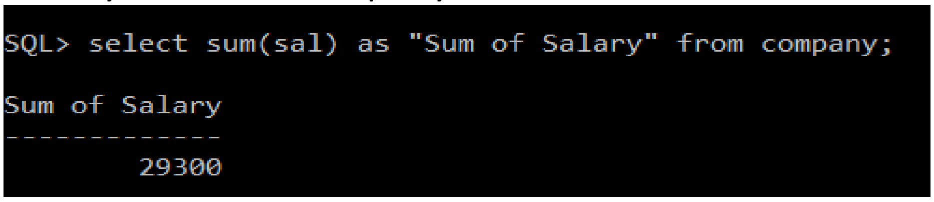
```
SQL> select count(*)from company;

  COUNT(*)
----------
         8
```

6. Find the average salary from the company table.



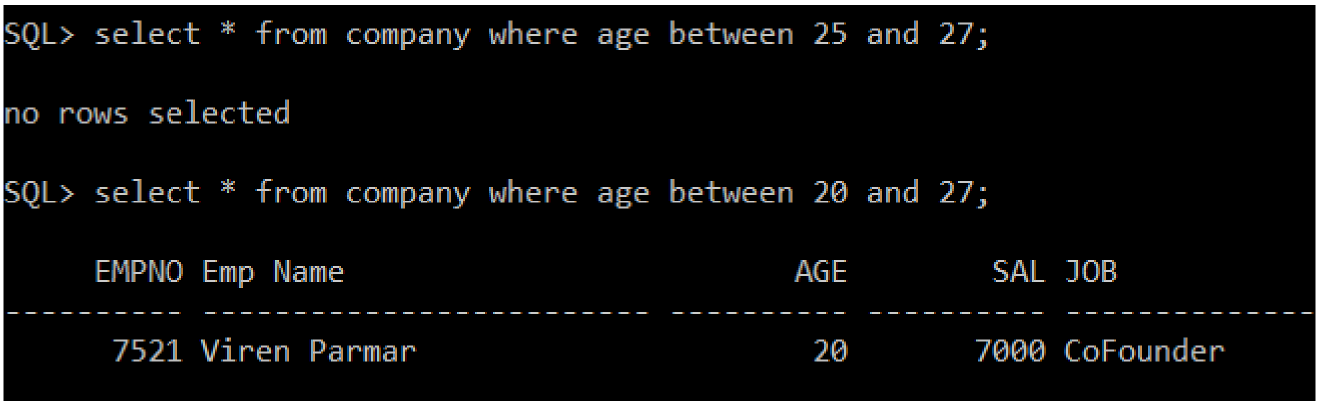
```
SQL> select avg(salary) as "Avg Salary" from company;
select avg(salary) as "Avg Salary" from company
           *
ERROR at line 1:
ORA-00904: "SALARY": invalid identifier


SQL> select avg(sal) as "Avg Salary" from company;

Avg Salary
----------
    3662.5
```

7. Find the sum of salary from the company table.



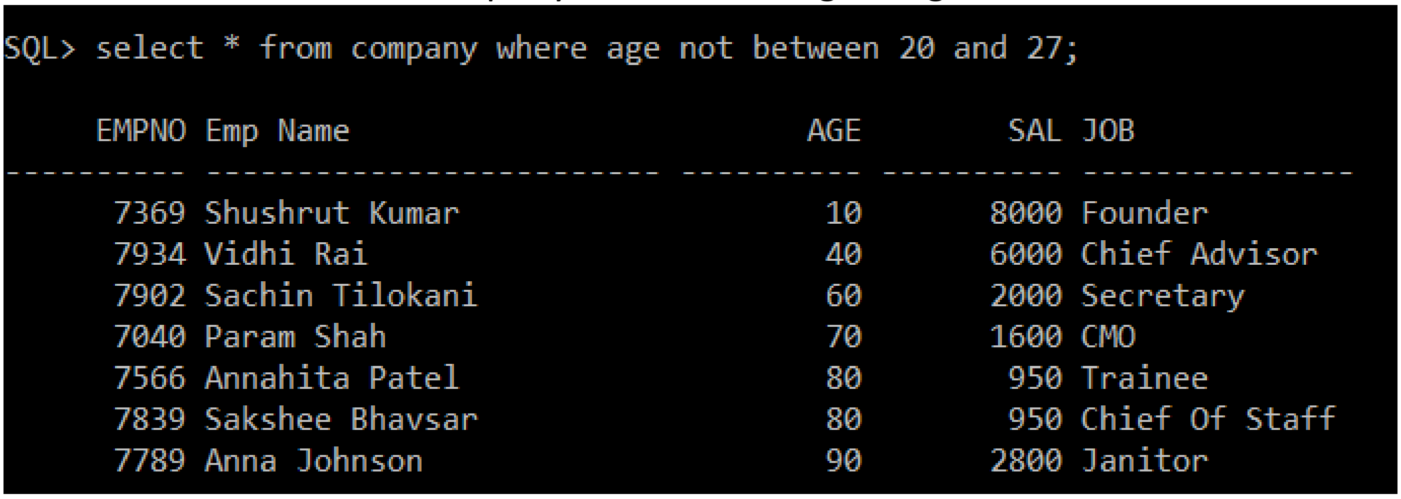
```
SQL> select sum(sal) as "Sum of Salary" from company;

Sum of Salary
-------------
        29300
```

8. List the records from the company table where age ranges between 20 to 27.



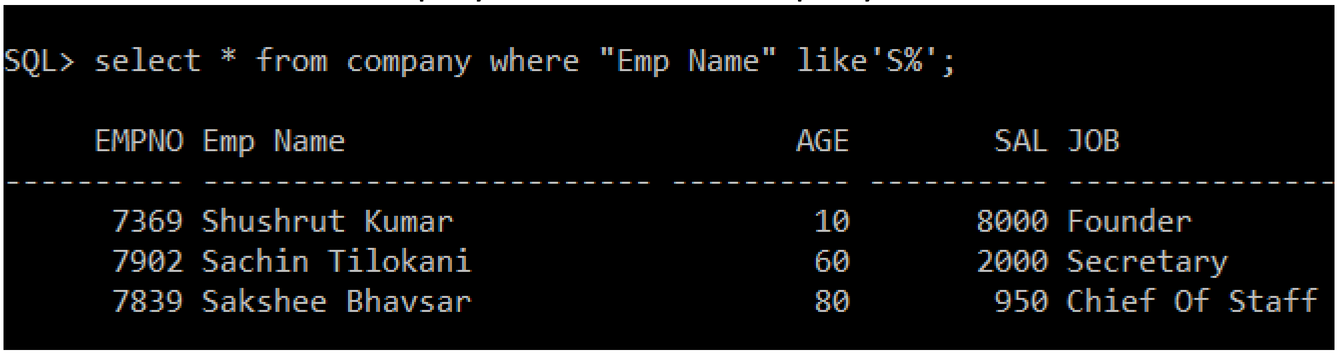
```
SQL> select * from company where age between 25 and 27;

no rows selected

SQL> select * from company where age between 20 and 27;

     EMPNO Emp Name                        AGE        SAL JOB
---------- ------------------------- ---------- ---------- --------------
      7521 Viren Parmar                     20       7000 CoFounder
```

1. List the records from the company table where age ranges not between 25 to 27.



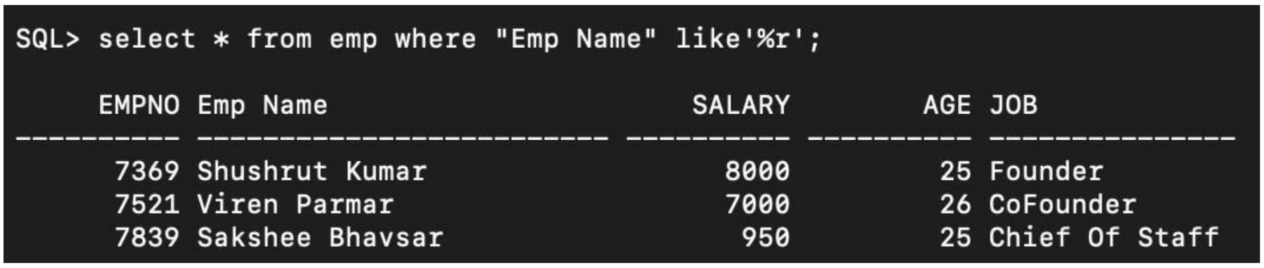
```
SQL> select * from company where age not between 20 and 27;

     EMPNO Emp Name                        AGE        SAL JOB
---------- ------------------------- ---------- ---------- ---------------
      7369 Shushrut Kumar                   10       8000 Founder
      7934 Vidhi Rai                        40       6000 Chief Advisor
      7902 Sachin Tilokani                  60       2000 Secretary
      7040 Param Shah                       70       1600 CMO
      7566 Annahita Patel                   80        950 Trainee
      7839 Sakshee Bhavsar                  80        950 Chief Of Staff
      7789 Anna Johnson                     90       2800 Janitor
```

2. List the names of the employees from the company where name starts with 'S'.



```
SQL> select * from company where "Emp Name" like'S%';

     EMPNO Emp Name                        AGE        SAL JOB
---------- ------------------------- ---------- ---------- ---------------
      7369 Shushrut Kumar                   10       8000 Founder
      7902 Sachin Tilokani                  60       2000 Secretary
      7839 Sakshee Bhavsar                  80        950 Chief Of Staff
```

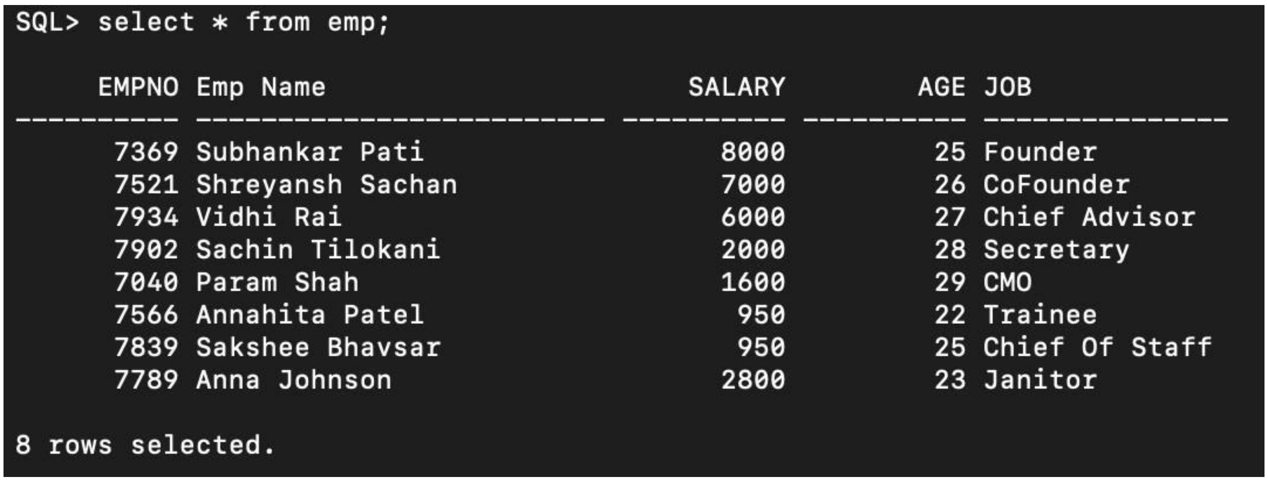
3. List the names of the employees from the company where name ends with 'r'.



```
SQL> select * from emp where "Emp Name" like'%r';

     EMPNO Emp Name                          SALARY        AGE JOB
---------- ------------------------- ---------- ---------- ---------------
      7369 Shushrut Kumar                      8000         25 Founder
      7521 Viren Parmar                        7000         26 CoFounder
      7839 Sakshee Bhavsar                      950         25 Chief Of Staff
```

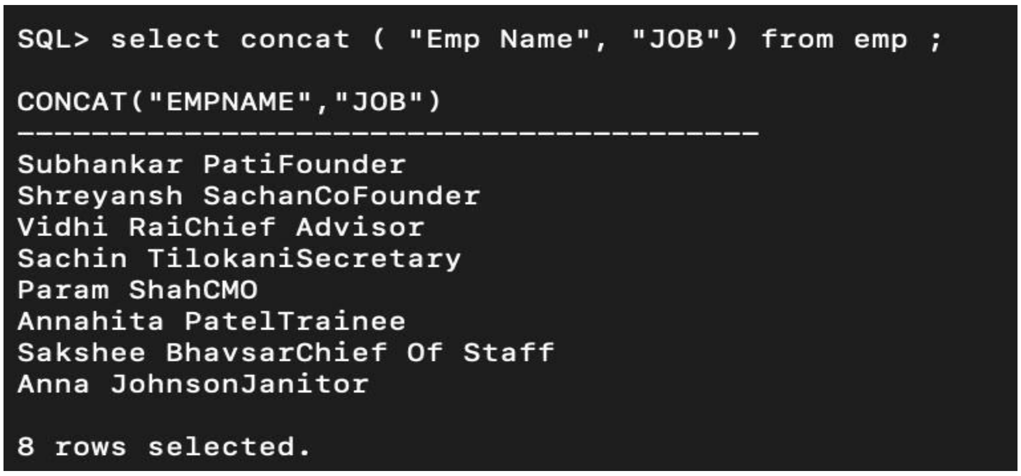
Experiment – 4 Inbuild Functions

1. Display all records



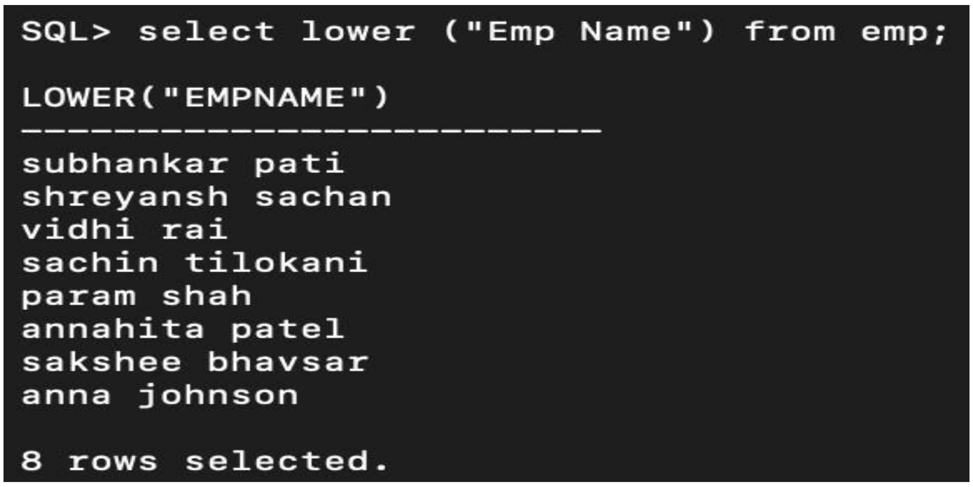
```
SQL> select * from emp;

     EMPNO Emp Name                           SALARY        AGE JOB
---------- ------------------------- ---------- ---------- ---------------
      7369 Subhankar Pati                       8000         25 Founder
      7521 Shreyansh Sachan                     7000         26 CoFounder
      7934 Vidhi Rai                            6000         27 Chief Advisor
      7902 Sachin Tilokani                      2000         28 Secretary
      7040 Param Shah                           1600         29 CMO
      7566 Annahita Patel                        950         22 Trainee
      7839 Sakshee Bhavsar                       950         25 Chief Of Staff
      7789 Anna Johnson                         2800         23 Janitor

8 rows selected.
```

2. Concat



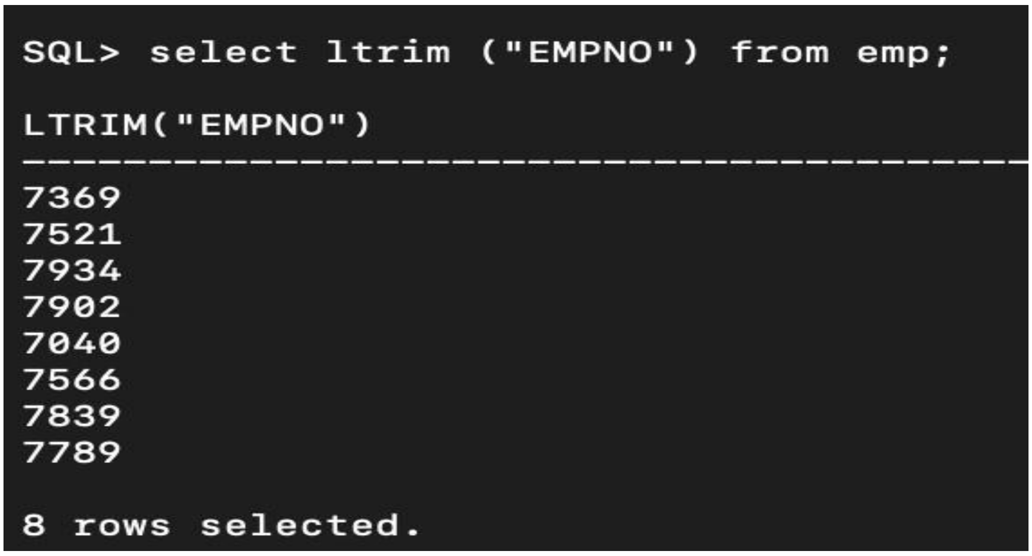
```
SQL> select concat ( "Emp Name", "JOB") from emp ;

CONCAT("EMPNAME","JOB")
----------------------------------------
Subhankar PatiFounder
Shreyansh SachanCoFounder
Vidhi RaiChief Advisor
Sachin TilokaniSecretary
Param ShahCMO
Annahita PatelTrainee
Sakshee BhavsarChief Of Staff
Anna JohnsonJanitor

8 rows selected.
```

3. Lower



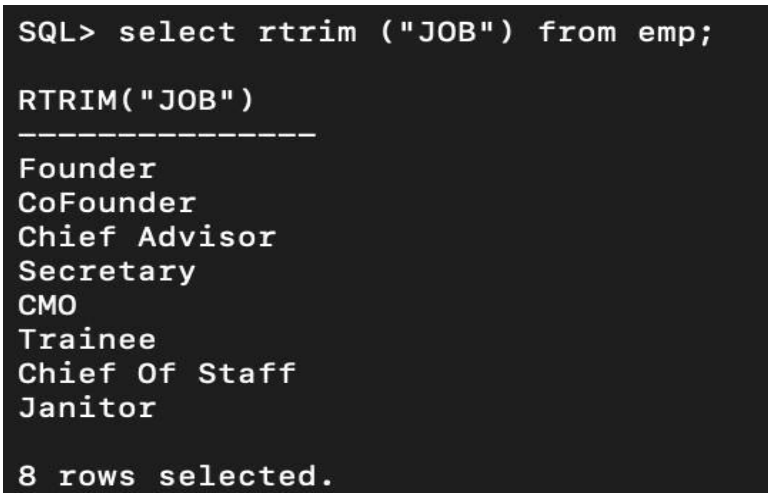
```
SQL> select lower ("Emp Name") from emp;

LOWER("EMPNAME")
-------------------------
subhankar pati
shreyansh sachan
vidhi rai
sachin tilokani
param shah
annahita patel
sakshee bhavsar
anna johnson

8 rows selected.
```

4. LTRIM



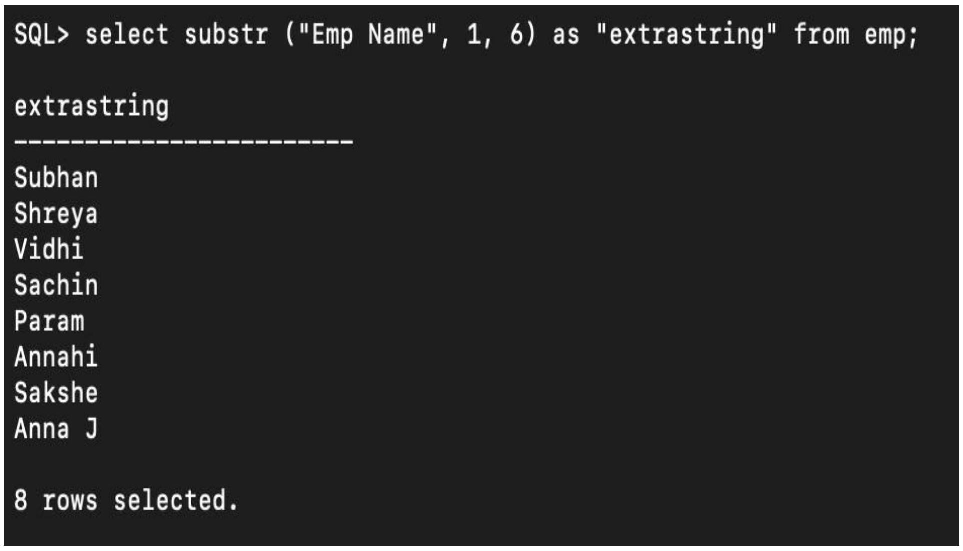
```
SQL> select ltrim ("EMPNO") from emp;

LTRIM("EMPNO")
----------------------------------------
7369
7521
7934
7902
7040
7566
7839
7789

8 rows selected.
```

5. RTRIM



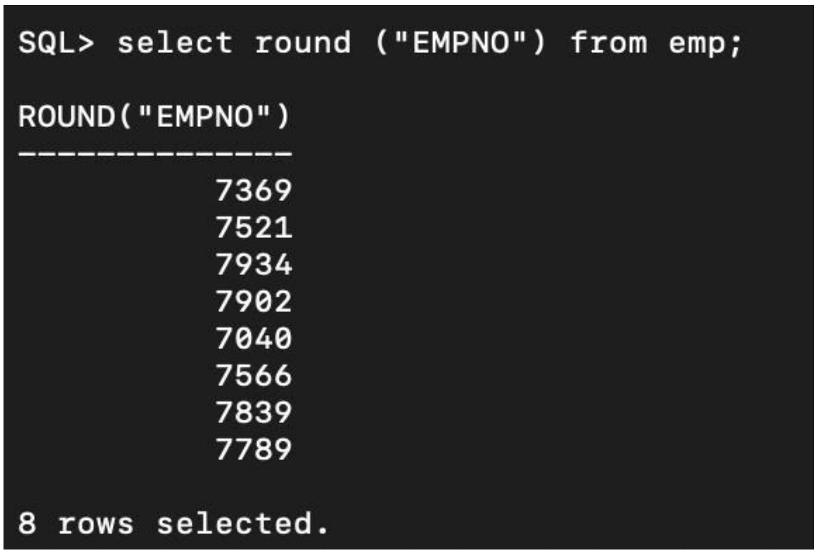
```
SQL> select rtrim ("JOB") from emp;

RTRIM("JOB")
---------------
Founder
CoFounder
Chief Advisor
Secretary
CMO
Trainee
Chief Of Staff
Janitor

8 rows selected.
```

6. Substring



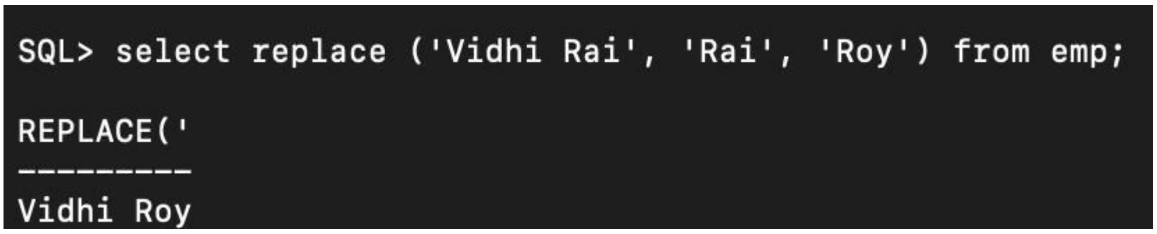
```
SQL> select substr ("Emp Name", 1, 6) as "extrastring" from emp;

extrastring
------------------------
Subhan
Shreya
Vidhi
Sachin
Param
Annahi
Sakshe
Anna J

8 rows selected.
```

7. Round



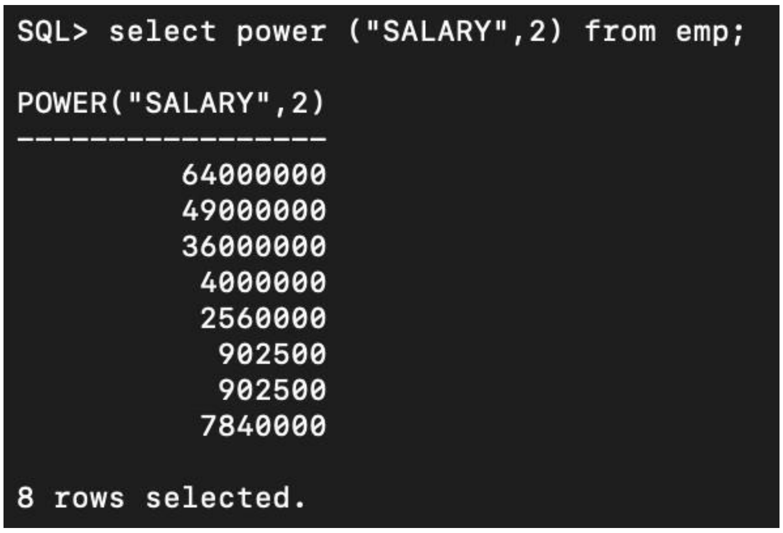
```
SQL> select round ("EMPNO") from emp;

ROUND("EMPNO")
--------------
          7369
          7521
          7934
          7902
          7040
          7566
          7839
          7789

8 rows selected.
```

8. Replace

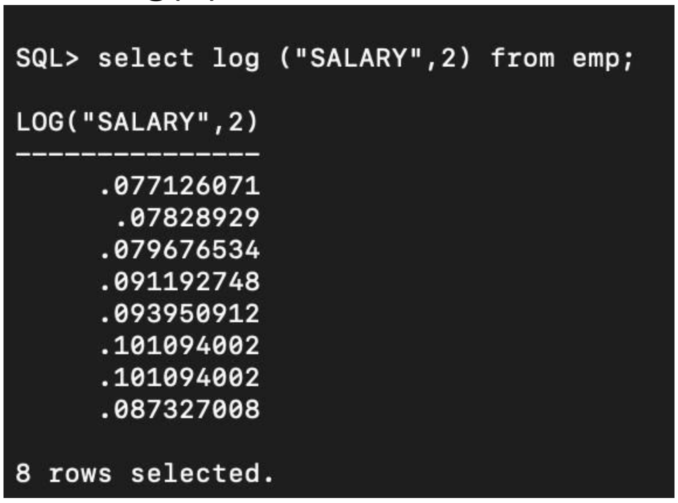


```
SQL> select replace ('Vidhi Rai', 'Rai', 'Roy') from emp;

REPLACE('
---------
Vidhi Roy
```

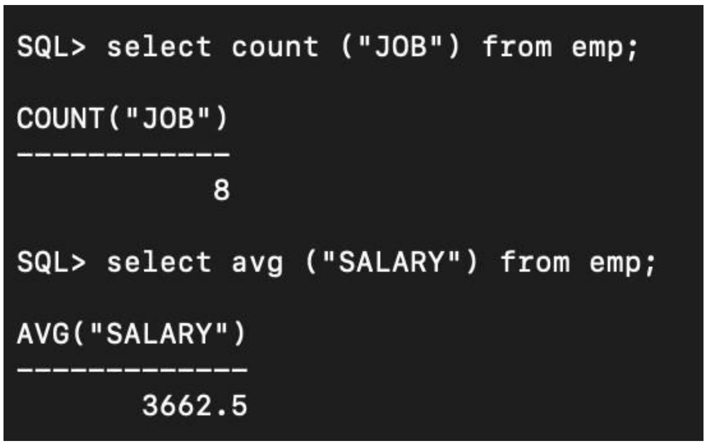
9. Power


```
SQL> select power ("SALARY",2) from emp;

POWER("SALARY",2)
-----------------
         64000000
         49000000
         36000000
          4000000
          2560000
           902500
           902500
          7840000

8 rows selected.
```

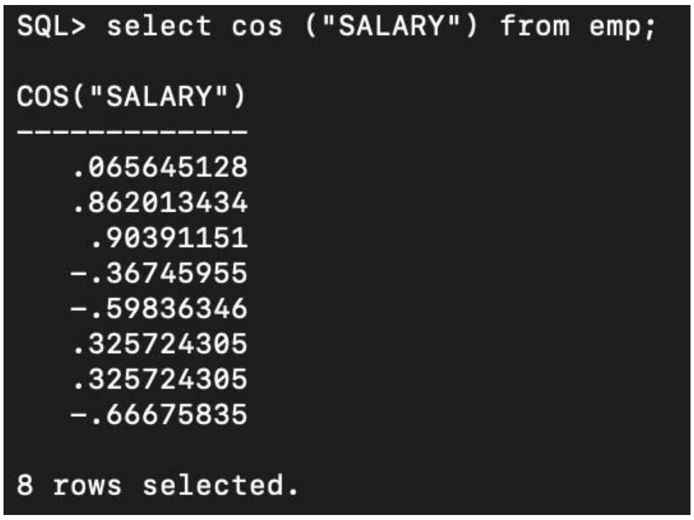
10. Log(2)


```
SQL> select log ("SALARY",2) from emp;

LOG("SALARY",2)
---------------
     .077126071
      .07828929
     .079676534
     .091192748
     .093950912
     .101094002
     .101094002
     .087327008

8 rows selected.
```

11. Count & 12. Avg

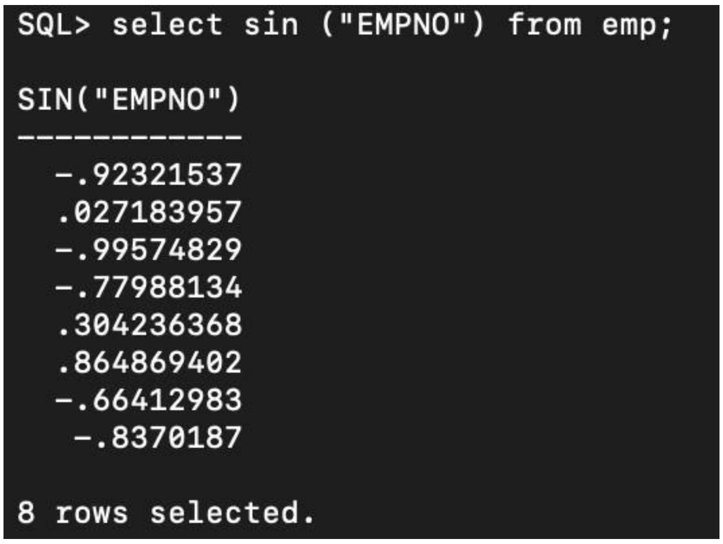

```
SQL> select count ("JOB") from emp;

COUNT("JOB")
------------
           8

SQL> select avg ("SALARY") from emp;

AVG("SALARY")
-------------
       3662.5
```

13. COS



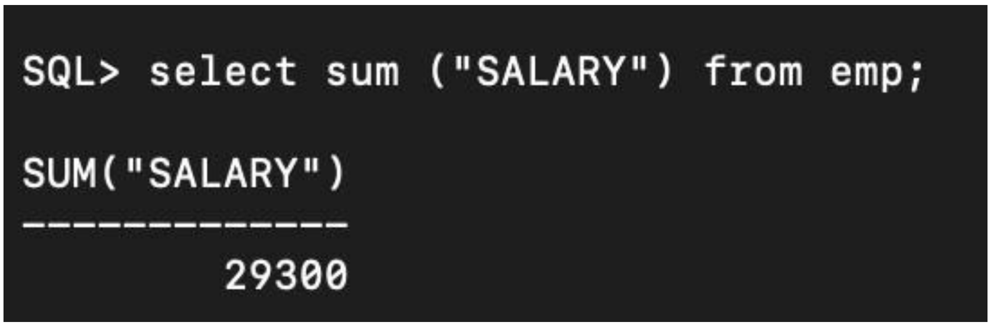
```
SQL> select cos ("SALARY") from emp;

COS("SALARY")
-------------
   .065645128
   .862013434
    .90391151
   -.36745955
   -.59836346
   .325724305
   .325724305
   -.66675835

8 rows selected.
```

14. SIN



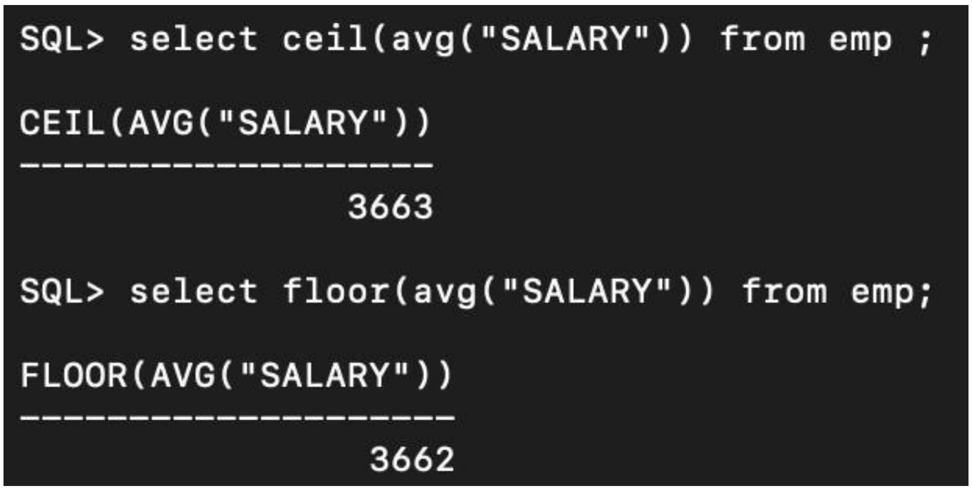
```
SQL> select sin ("EMPNO") from emp;

SIN("EMPNO")
------------
  -.92321537
  .027183957
  -.99574829
  -.77988134
  .304236368
  .864869402
  -.66412983
   -.8370187

8 rows selected.
```

15. Sum



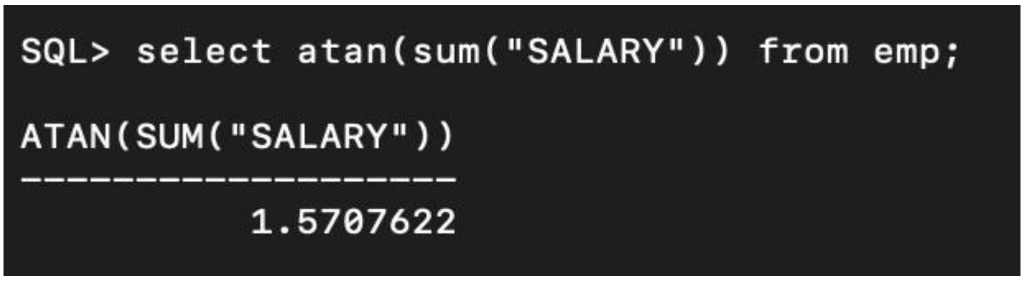
```
SQL> select sum ("SALARY") from emp;

SUM("SALARY")
-------------
        29300
```

16. Ceiling



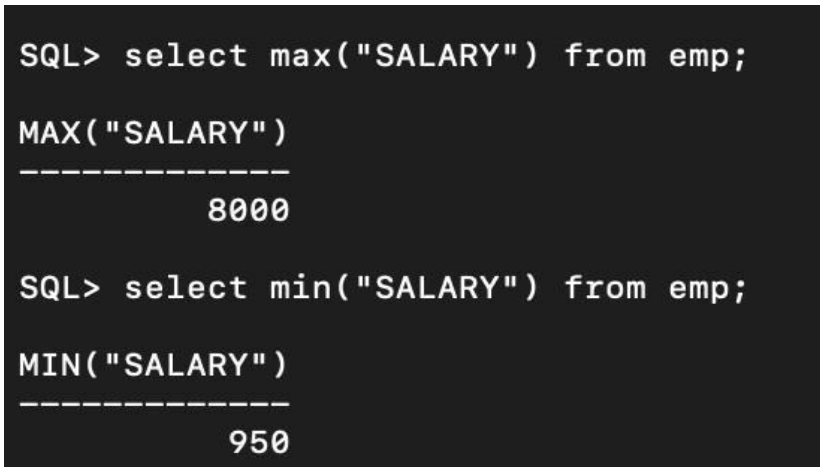
```
SQL> select ceil(avg("SALARY")) from emp ;

CEIL(AVG("SALARY"))
-------------------
               3663

SQL> select floor(avg("SALARY")) from emp;

FLOOR(AVG("SALARY"))
--------------------
                3662
```

17. Atan



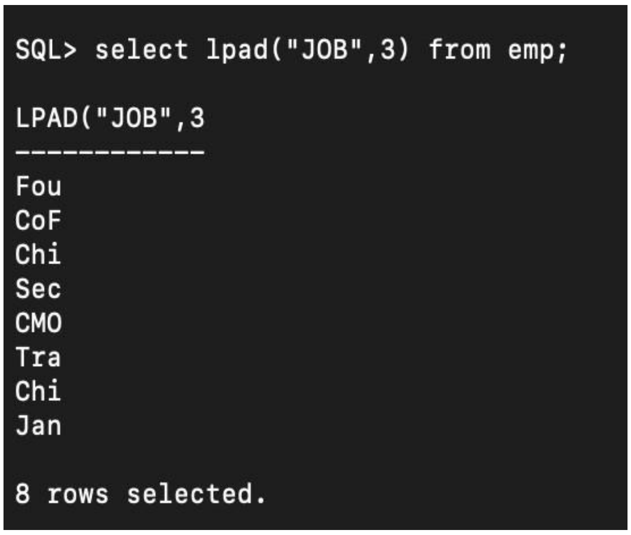
```
SQL> select atan(sum("SALARY")) from emp;

ATAN(SUM("SALARY"))
-------------------
          1.5707622
```

18. Max and 19. Min



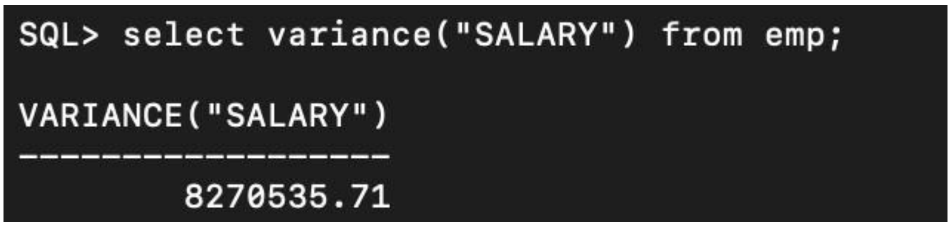
```
SQL> select max("SALARY") from emp;

MAX("SALARY")
-------------
         8000

SQL> select min("SALARY") from emp;

MIN("SALARY")
-------------
          950
```

20 LPAD



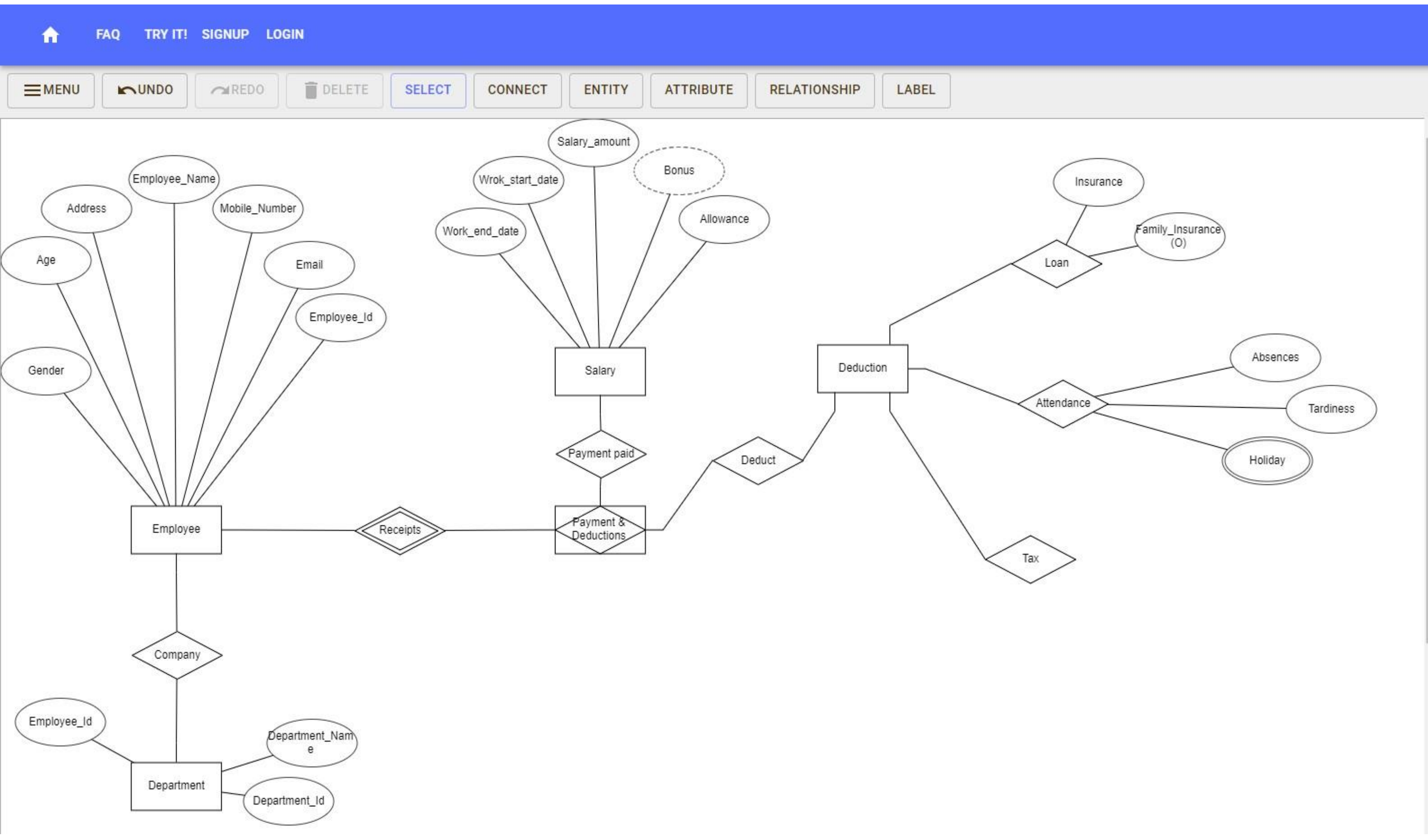
```
SQL> select lpad("JOB",3) from emp;

LPAD("JOB",3
------------
Fou
CoF
Chi
Sec
CMO
Tra
Chi
Jan

8 rows selected.
```

21 Variance



```
SQL> select variance("SALARY") from emp;

VARIANCE("SALARY")
------------------
        8270535.71
```

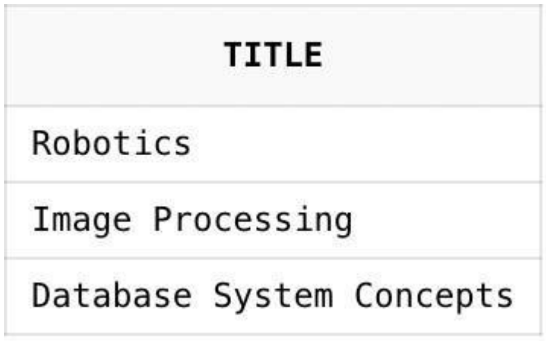
EXPERIMENT 5
Entity Relationship Diagram



EXPERIMENT 6

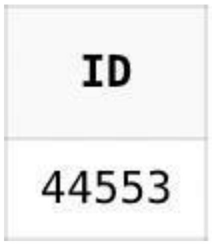
1. Write the following queries in SQL, using the university schema. Create a table with appropriate attributes.
a. Find the titles of courses in the Comp. Sci. department that have 3 credits.
b. Find the IDs of all students who were taught by an instructor named Einstein; make sure there are no duplicates in the result.
c. Find the highest salary of any instructor.
d. Find all instructors earning the highest salary (there may be more than one with the same salary).
e. Find the enrollment of each section that was offered in Fall 2017.
f. Find the maximum enrollment, across all sections, in Fall 2017.
g. Find the sections that had the maximum enrollment in Fall 2017.

A.
Query:
select title from course where dept_name = 'Comp. Sci.' and credits = 3;



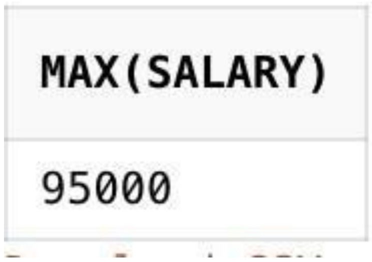
TITLE
Robotics
Image Processing
Database System Concepts

B.
Query:
```
select distinct takes.ID from takes, instructor, teaches where takes.course_id =
teaches.course_id and takes.sec_id = teaches.sec_id and takes.semester =
teaches.semester and takes.year = teaches.year and teaches.id = instructor.id and
instructor.name = 'Einstein';
```



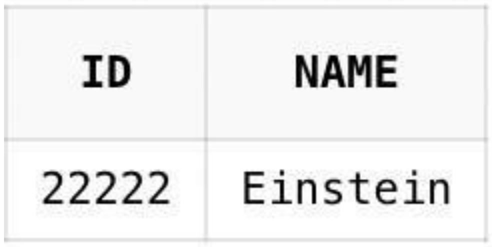
ID
44553

C.
Query:
```
select max(salary) from instructor;
```



MAX(SALARY)
95000

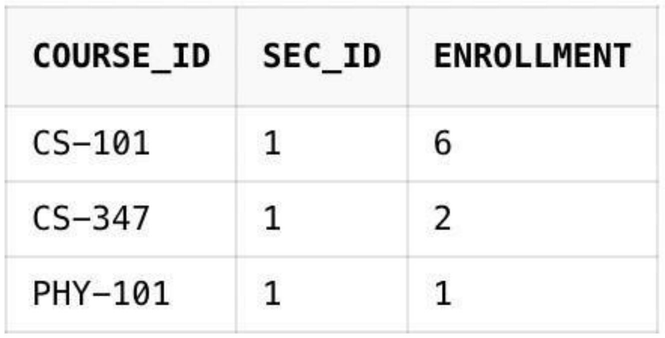
D.
Query:
```
select ID, name from instructor where salary = (select max(salary) from instructor);
```



ID	NAME
22222	Einstein

E.
Query:
```
select course_id, sec_id,
(select count(ID)
from takes
where takes.year = section.year
and takes.semester = section.semester
and takes.course_id = section.course_id
and takes.sec_id = section.sec_id)
as enrollment
```

```
from section
where semester = 'Fall'
and year = 2017;
```

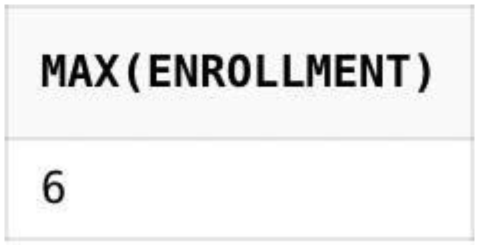


COURSE_ID	SEC_ID	ENROLLMENT
CS−101	1	6
CS−347	1	2
PHY−101	1	1

F.

Query:

```
select max(enrollment)
from (select count(ID) as enrollment
from section, takes
where takes.year = section.year
and takes.semester = section.semester
and takes.course_id = section.course_id
and takes.sec_id = section.sec_id
and takes.semester = 'Fall'
and takes.year = 2017
group by takes.course_id, takes.sec_id);
```



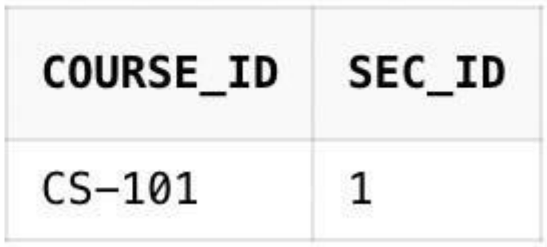
MAX(ENROLLMENT)
6

G.

Query:

```
with sec_enrollment as (
select takes.course_id, takes.sec_id, count(ID) as enrollment
from section, takes
where takes.year = section.year
and takes.semester = section.semester
```

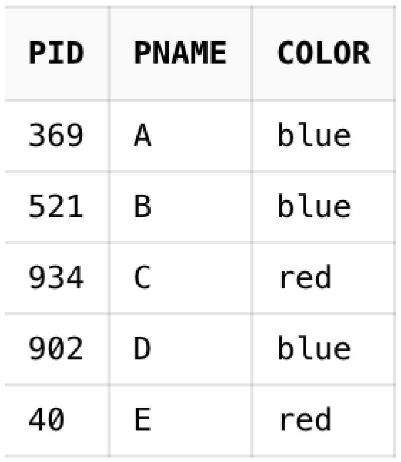
and takes.course_id = section.course_id
and takes.sec_id = section.sec_id
and takes.semester = 'Fall'
and takes.year = 2017
group by takes.course_id, takes.sec_id)
select course_id, sec_id
from sec_enrollment
where enrollment = (select max(enrollment) from sec_enrollment);



COURSE_ID	SEC_ID
CS-101	1

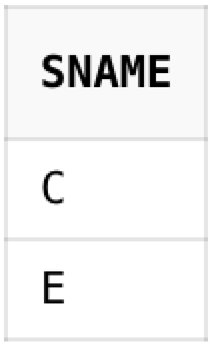
2. Suppliers(sid:integer, sname:string, city:string, street:string)
Parts(pid:integer, pname:string, color:string)
Catalog(sid:integer, pid:integer, cost:real)
Write a query retrieves the name (sname) of suppliers, who have supplied a non-blue part.

Ans:



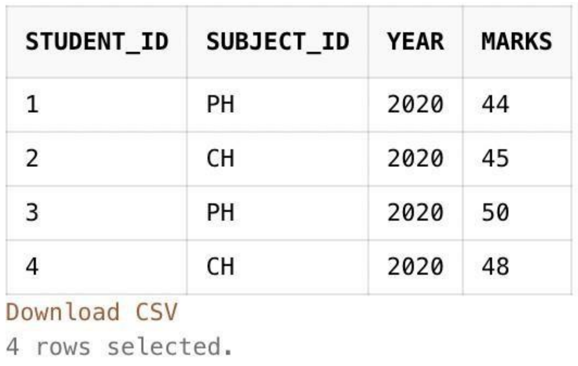
PID	PNAME	COLOR
369	A	blue
521	B	blue
934	C	red
902	D	blue
40	E	red

Query:
SELECT sname FROM suppliers WHERE sid NOT IN (SELECT sid FROM catalog WHERE pid NOT in (SELECT pid FROM parts WHERE color <> 'blue'));



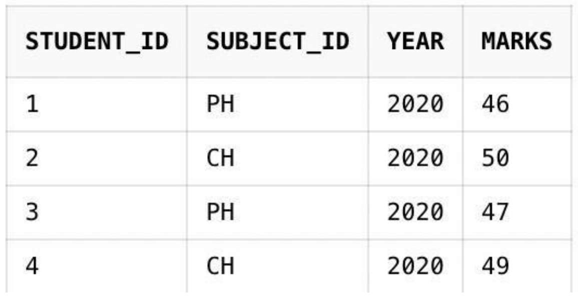
SNAME
C
E

3. Write a query to find the sum of marks for each student from two tables



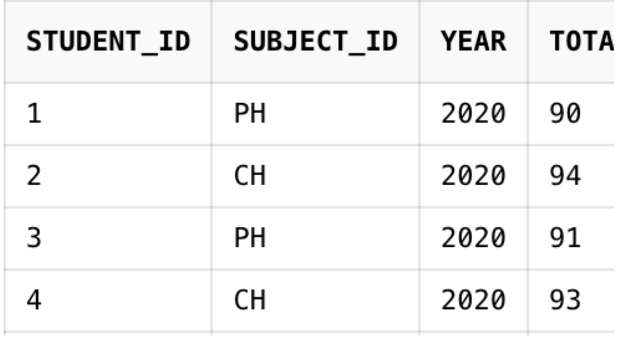
STUDENT_ID	SUBJECT_ID	YEAR	MARKS
1	PH	2020	44
2	CH	2020	45
3	PH	2020	50
4	CH	2020	48

```
Download CSV
4 rows selected.
```



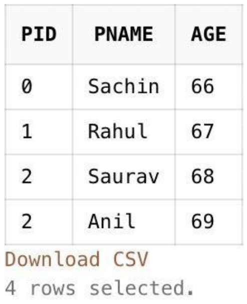
STUDENT_ID	SUBJECT_ID	YEAR	MARKS
1	PH	2020	46
2	CH	2020	50
3	PH	2020	47
4	CH	2020	49

Query:
SELECT finalterm.student_id, finalterm.subject_id, finalterm.year,(midterm.marks+finalterm.marks) AS total FROM midterm, finalterm;



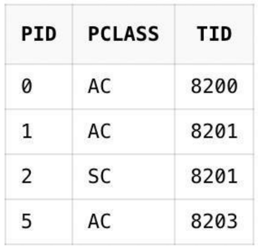
STUDENT_ID	SUBJECT_ID	YEAR	TOTA
1	PH	2020	90
2	CH	2020	94
3	PH	2020	91
4	CH	2020	93

4. Write a query to find the passengers who have done registration and also whohave age greater



PID	PNAME	AGE
0	Sachin	66
1	Rahul	67
2	Saurav	68
2	Anil	69

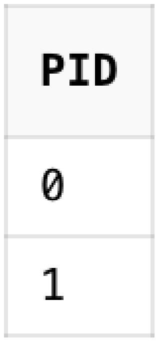
```
Download CSV
4 rows selected.
```



PID	PCLASS	TID
0	AC	8200
1	AC	8201
2	SC	8201
5	AC	8203

than 65 who are travelling in “AC” class from two tables.

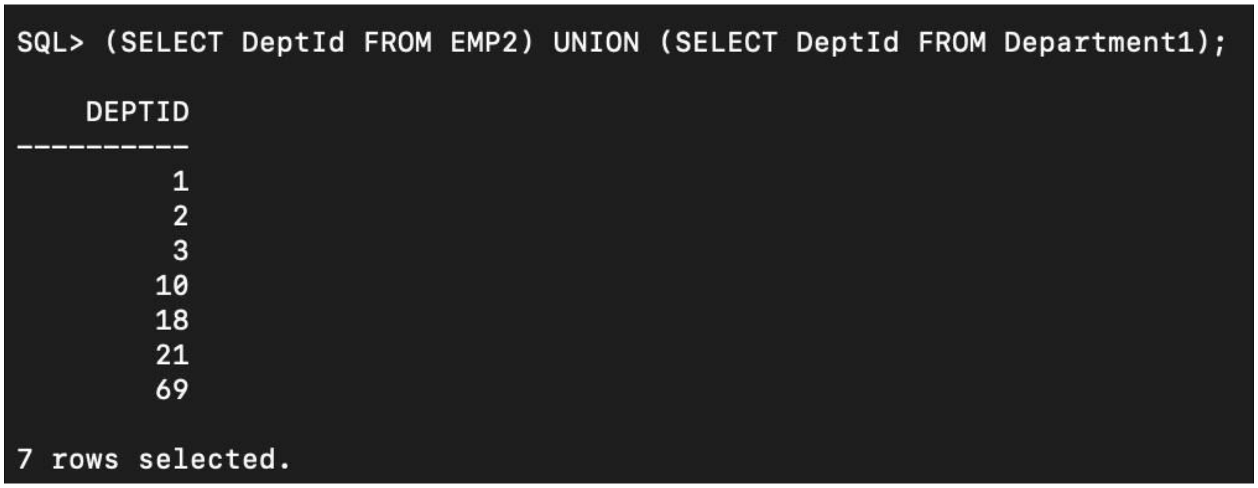
Query:
select pid from reservation where pclass='AC' and
exists (select * from passengerwhere age > '65'
AND passenger.pid = reservation.pid);



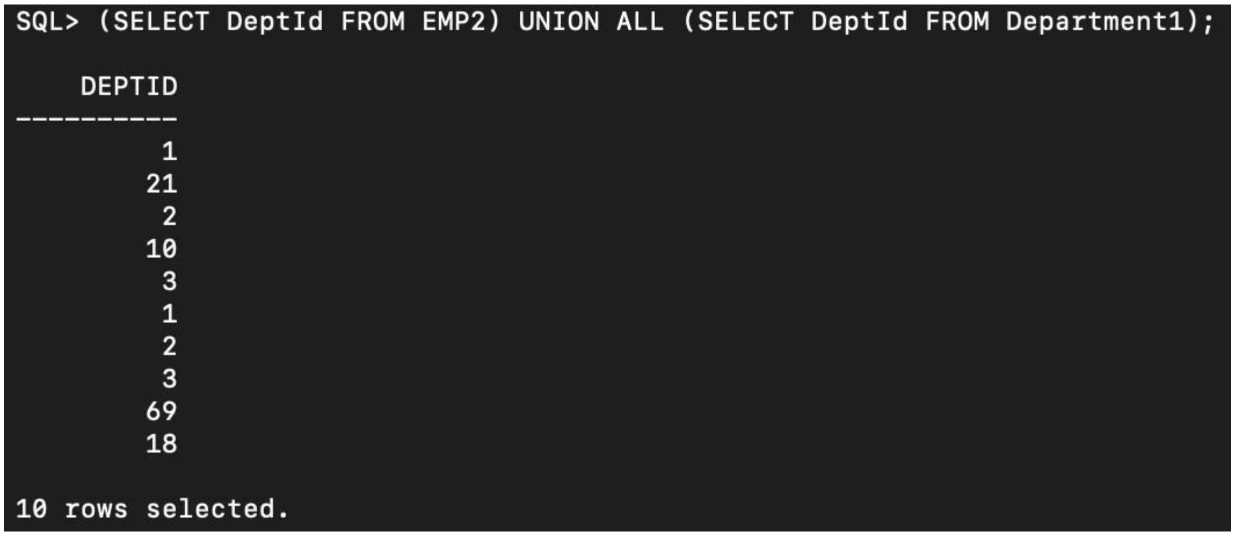
PID
0
1

EXPERIMENT 7

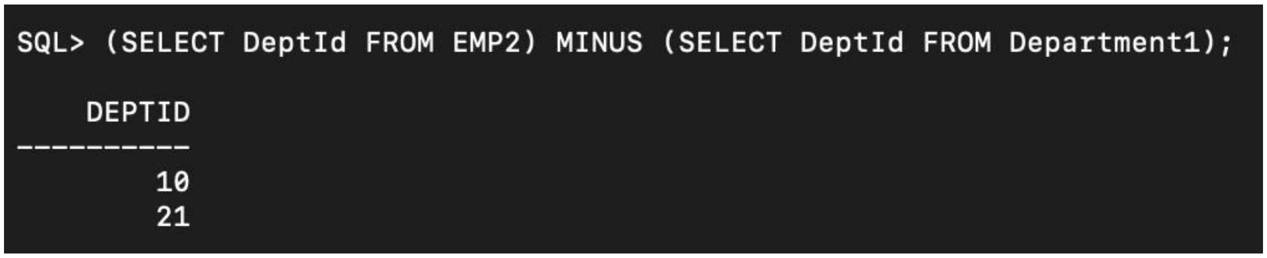
Write the query to demonstrate the various set operators (UNION, UNION ALL, MINUS, INTERSECT)
Write a query using INTERSECT set operator to list the student id and residence location of the students.



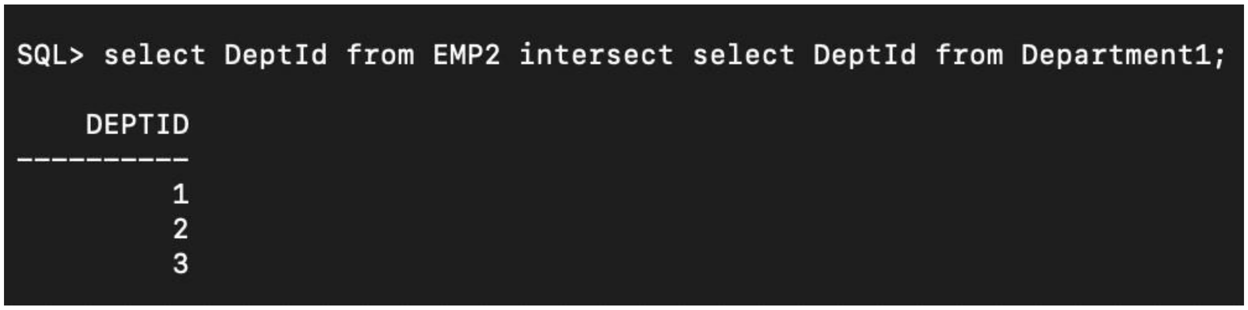
```
SQL> (SELECT DeptId FROM EMP2) UNION (SELECT DeptId FROM Department1);

    DEPTID
----------
         1
         2
         3
        10
        18
        21
        69

7 rows selected.
```



```
SQL> (SELECT DeptId FROM EMP2) UNION ALL (SELECT DeptId FROM Department1);

    DEPTID
----------
         1
        21
         2
        10
         3
         1
         2
         3
        69
        18

10 rows selected.
```

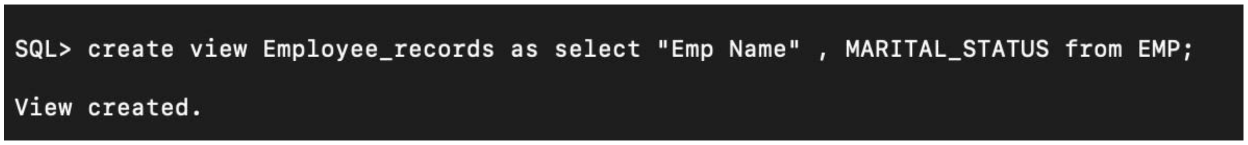


```
SQL> (SELECT DeptId FROM EMP2) MINUS (SELECT DeptId FROM Department1);

    DEPTID
----------
        10
        21
```

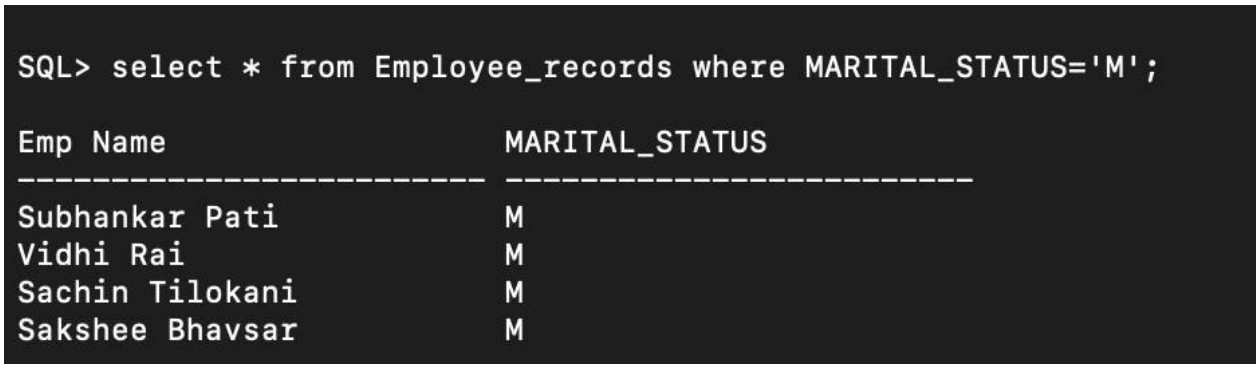


```
SQL> select DeptId from EMP2 intersect select DeptId from Department1;

    DEPTID
----------
         1
         2
         3
```

Write a query for SQL view (view name: Employee_Records) to fetch columns of the table and filter the results using where clause with the martial_status ‘M’.

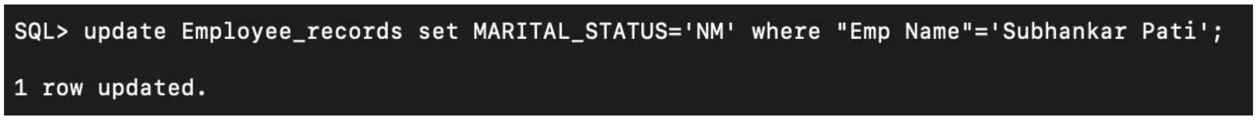


```
SQL> create view Employee_records as select "Emp Name" , MARITAL_STATUS from EMP;

View created.
```

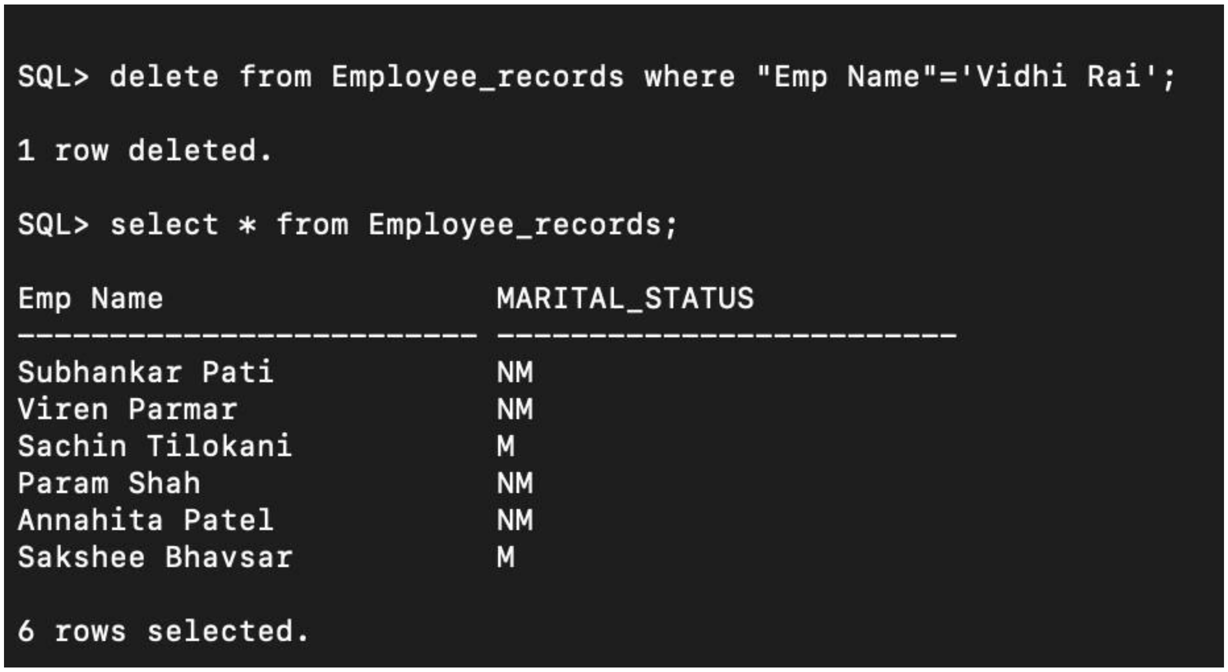


```
SQL> select * from Employee_records where MARITAL_STATUS='M';

Emp Name                  MARITAL_STATUS
------------------------- -------------------------
Subhankar Pati            M
Vidhi Rai                 M
Sachin Tilokani           M
Sakshee Bhavsar           M
```

Q.Write a query to update, delete and insert from SQL view (view name: Employee_Records) table.



```
SQL> update Employee_records set MARITAL_STATUS='NM' where "Emp Name"='Subhankar Pati';

1 row updated.
```

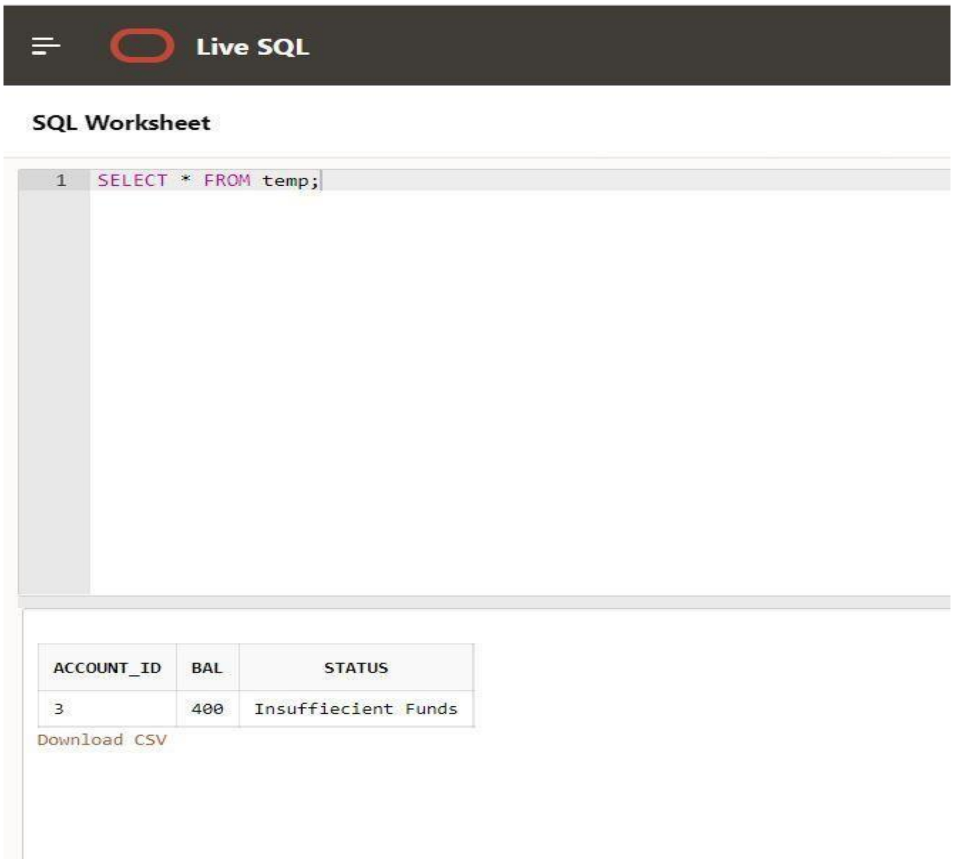


```
SQL> delete from Employee_records where "Emp Name"='Vidhi Rai';

1 row deleted.

SQL> select * from Employee_records;

Emp Name                  MARITAL_STATUS
------------------------- -------------------------
Subhankar Pati            NM
Viren Parmar              NM
Sachin Tilokani           M
Param Shah                NM
Annahita Patel            NM
Sakshee Bhavsar           M

6 rows selected.
```

Experiment – 8

1. **Write a PL/SQL program which processes a bank transaction. Before allowing you to withdraw $500 from account 3, it makes sure the account has sufficient funds to cover the withdrawal. If the funds are available, the program debits the account. Otherwise, the program prints a message "insufficient funds".**

```
CREATE TABLE accounts(account_id number(10), bal number(11,2));
INSERT INTO accounts VALUES('1','1200.00');
INSERT INTO accounts VALUES('2', '600.00');
INSERT INTO accounts VALUES('3', '400.00');

CREATE TABLE temp(account_id number(10), bal number(11,2), status varchar(50));
DECLARE
   acct_balance NUMBER(11,2);
   acct CONSTANT NUMBER(4):=3;
   debit_amt CONSTANT NUMBER(5,2):=500.00;
BEGIN
   SELECT bal INTO acct_balance FROM accounts
      WHERE account_id = acct
      FOR UPDATE OF bal;
   IF acct_balance >= debit_amt THEN
      UPDATE accounts SET bal = bal- debit_amt
      WHERE account_id = acct;
   ELSE
      INSERT INTO temp VALUES
         (acct, acct_balance,'Insuffiecient Funds');
   END IF;
   COMMIT;
END;
```



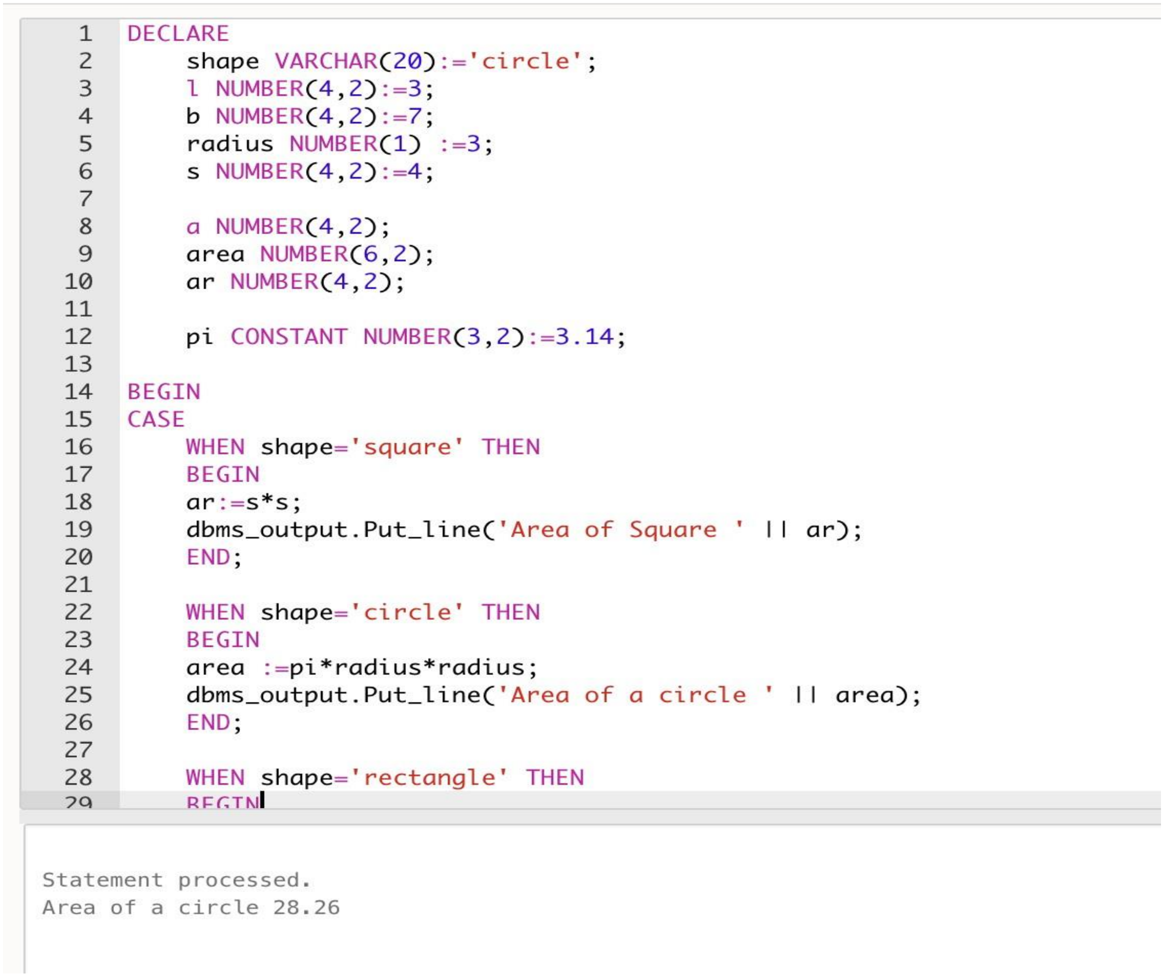
```
SELECT * FROM temp;
```

ACCOUNT_ID	BAL	STATUS
3	400	Insuffiecient Funds

2. Write a PL/SQL program for finding the area of square, circle, and rectangle using switch case.

```
DECLARE
   shape VARCHAR(20):='circle';
   l NUMBER(4,2):=3;
   b NUMBER(4,2):=7;
   radius NUMBER(1) :=3;
   s NUMBER(4,2):=4;

   a NUMBER(4,2);
   area NUMBER(6,2);
   ar NUMBER(4,2);

   pi CONSTANT NUMBER(3,2):=3.14;

BEGIN
CASE
   WHEN shape='square' THEN
   BEGIN
   ar:=s*s;
   dbms_output.Put_line('Area of Square ' || ar);
   END;

   WHEN shape='circle' THEN
   BEGIN
   area :=pi*radius*radius;
   dbms_output.Put_line('Area of a circle ' || area);
   END;

   WHEN shape='rectangle' THEN
   BEGIN
   a:=l*b;
   dbms_output.Put_line('Area of recatangle ' || a);
   END;
```

END CASE;
END;

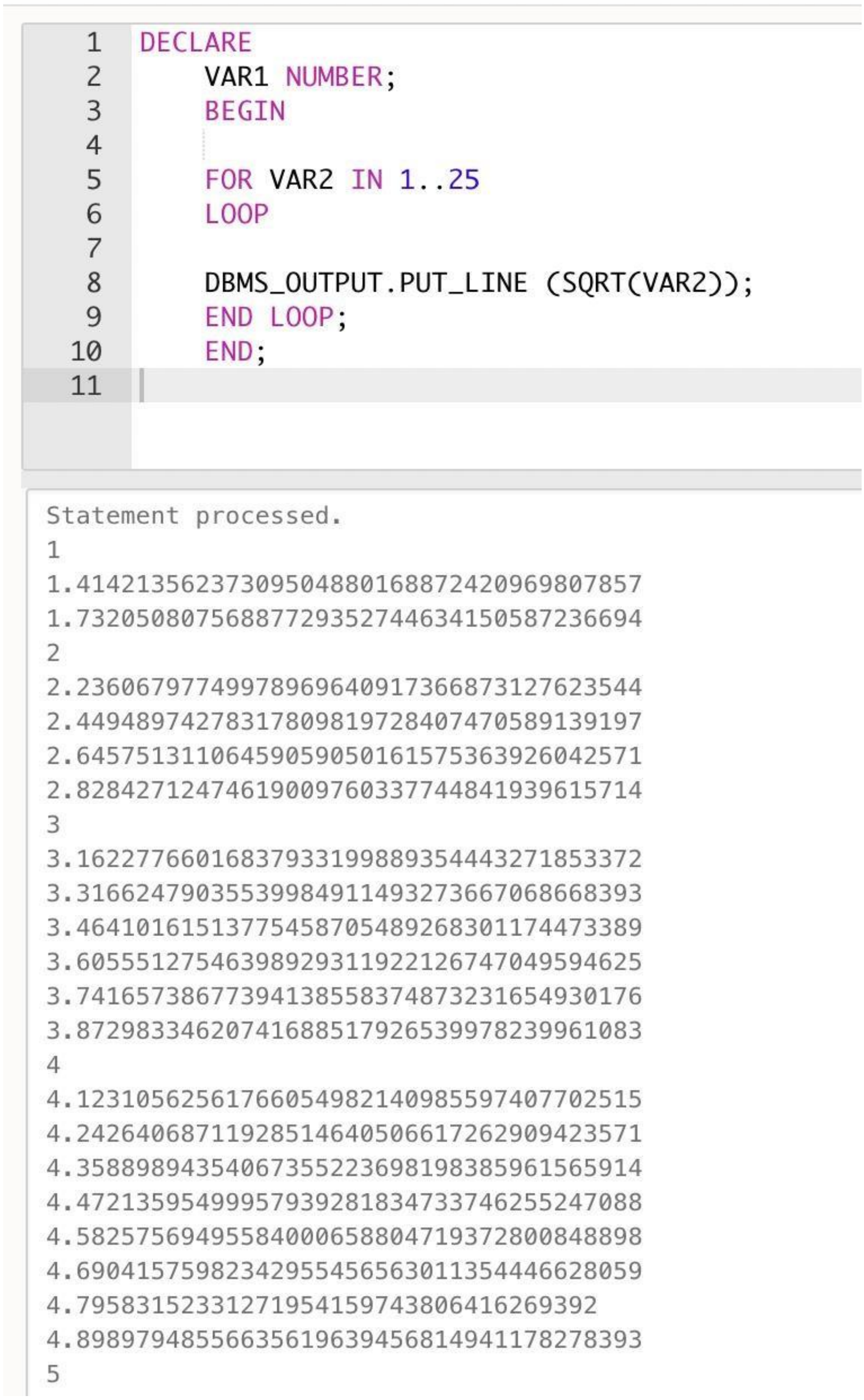


```
DECLARE
    shape VARCHAR(20):='circle';
    l NUMBER(4,2):=3;
    b NUMBER(4,2):=7;
    radius NUMBER(1) :=3;
    s NUMBER(4,2):=4;

    a NUMBER(4,2);
    area NUMBER(6,2);
    ar NUMBER(4,2);

    pi CONSTANT NUMBER(3,2):=3.14;

BEGIN
CASE
    WHEN shape='square' THEN
    BEGIN
    ar:=s*s;
    dbms_output.Put_line('Area of Square ' || ar);
    END;

    WHEN shape='circle' THEN
    BEGIN
    area :=pi*radius*radius;
    dbms_output.Put_line('Area of a circle ' || area);
    END;

    WHEN shape='rectangle' THEN
    BEGIN
```

```
Statement processed.
Area of a circle 28.26
```

3. Write a PL/SQL program for finding the square roots of 1 to 25 using for loop.

```
DECLARE
   VAR1 NUMBER;
   BEGIN

   FOR VAR2 IN 1..25
   LOOP

   DBMS_OUTPUT.PUT_LINE (SQRT(VAR2));
   END LOOP;
   END;
```



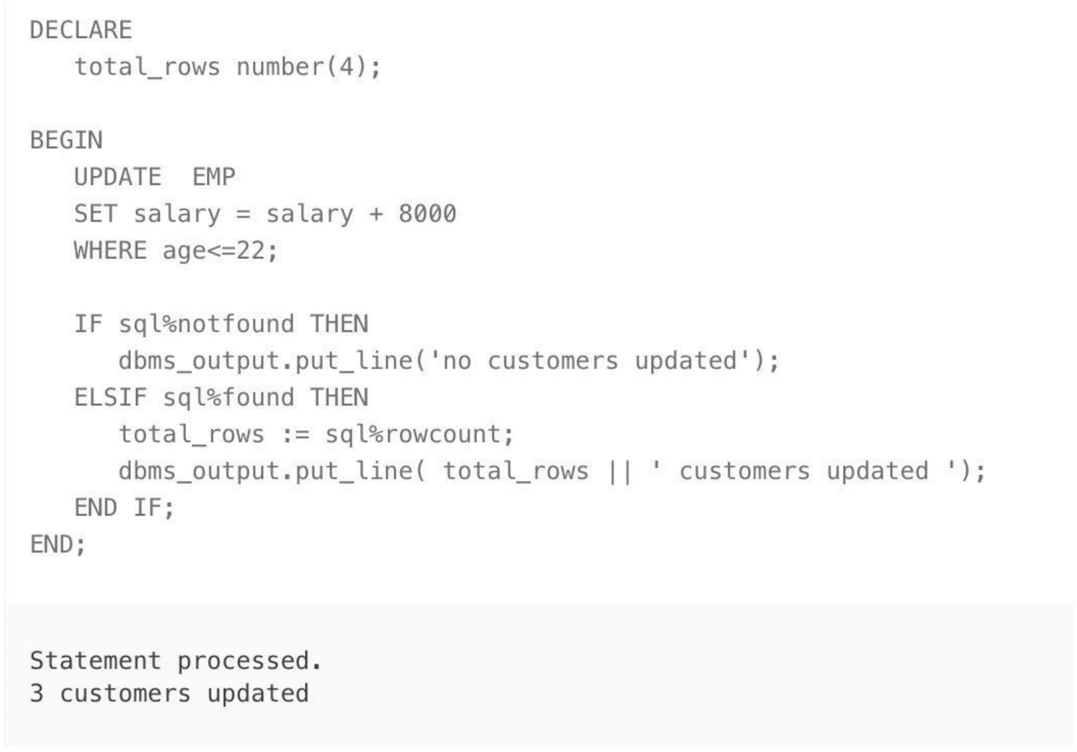
```sql
DECLARE
    VAR1 NUMBER;
    BEGIN

    FOR VAR2 IN 1..25
    LOOP

    DBMS_OUTPUT.PUT_LINE (SQRT(VAR2));
    END LOOP;
    END;

```

```
Statement processed.
1
1.41421356237309504880168872420969807857
1.73205080756887729352744634150587236694
2
2.23606797749978969640917366873127623544
2.44948974278317809819728407470589139197
2.64575131106459059050161575363926042571
2.82842712474619009760337744841939615714
3
3.16227766016837933199889354443271853372
3.31662479035539984911493273667068668393
3.46410161513775458705489268301174473389
3.60555127546398929311922126747049594625
3.74165738677394138558374873231654930176
3.87298334620741688517926539978239961083
4
4.12310562561766054982140985597407702515
4.24264068711928514640506617262909423571
4.35889894354067355223698198385961565914
4.47213595499957939281834733746255247088
4.58257569495584000658804719372800848898
4.69041575982342955456563011354446628059
4.79583152331271954159743806416269392
4.89897948556635619639456814941178278393
5
```

Exp 9 PL/SQL

1. **Write a program to find the age of employees who are <=22 and increase the salary by 8000. Use sql%rowcount attribute to find the rows that got updated after execution. (Hint: implicit cursor)**

```
DROP TABLE emp;
CREATE TABLE emp (
   emp_id number,
   FirstName varchar(255),
   age number,
   salary number);
INSERT INTO emp VALUES('101', 'Adam', '20', 15000);
INSERT INTO emp VALUES('102', 'Ben', '23', 25000);
INSERT INTO emp VALUES('103', 'Chris', '21', 20000);
INSERT INTO emp VALUES('104', 'Dan', '19', 10000);

SELECT * FROM emp WHERE age<=22;

DECLARE
  total_rows number(4);

BEGIN
  UPDATE  EMP
  SET salary = salary + 8000
  WHERE age<=22;

  IF sql%notfound THEN
    dbms_output.put_line('no customers updated');
  ELSIF sql%found THEN
    total_rows := sql%rowcount;
    dbms_output.put_line( total_rows || ' customers updated ');
  END IF;
END;
```



```
DECLARE
   total_rows number(4);

BEGIN
   UPDATE  EMP
   SET salary = salary + 8000
   WHERE age<=22;

   IF sql%notfound THEN
      dbms_output.put_line('no customers updated');
   ELSIF sql%found THEN
      total_rows := sql%rowcount;
      dbms_output.put_line( total_rows || ' customers updated ');
   END IF;
END;
```

```
Statement processed.
3 customers updated
```

2. **Write a sql procedure program to find the factorial of a given number. (Hint: get the value of x in IN parameter and fact in OUT parameter)**

```
declare
x number;
fact number;
i number;

PROCEDURE factorial(x IN number, fact OUT number) IS
begin
fact:=1;
for i in 1..x
loop
fact:=fact*i;
end loop;
end;

begin
x:=5;
fact:=1;
factorial(x,fact);

dbms_output.put_line('factorial='||fact);
end;
```

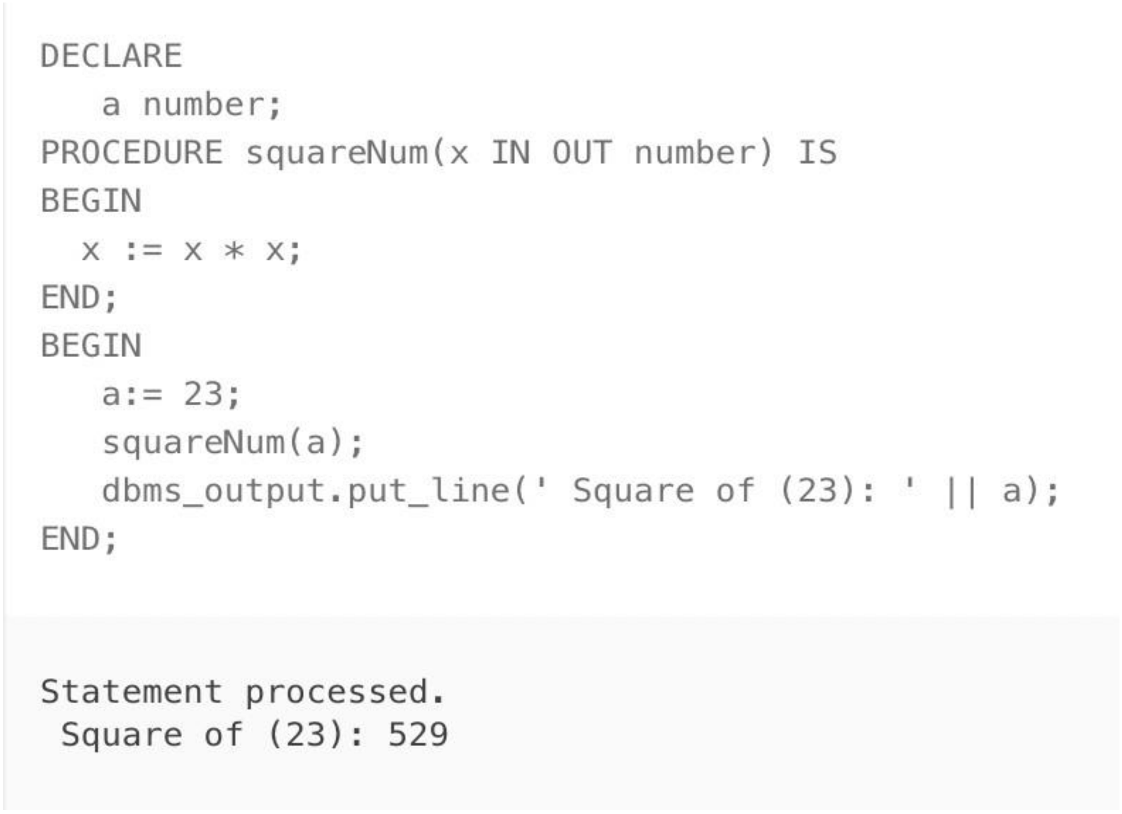
```
declare
x number;
fact number;
i number;

PROCEDURE factorial(x IN number, fact OUT number) IS
begin
fact:=1;
for i in 1..x
loop
fact:=fact*i;
end loop;
end;

begin
x:=5;
fact:=1;
factorial(x,fact);
dbms_output.put_line('factorial='||fact);
end;
```

```
Statement processed.
factorial=120
```

3. Write a sql procedure program to find the square of a given number (Hint: use X as IN OUT parameter)

```
DECLARE
  a number;
PROCEDURE squareNum(x IN OUT number) IS
BEGIN
  x := x * x;
END;
BEGIN
  a:= 23;
  squareNum(a);
  dbms_output.put_line(' Square of (23): ' || a);
```

```
END;
```

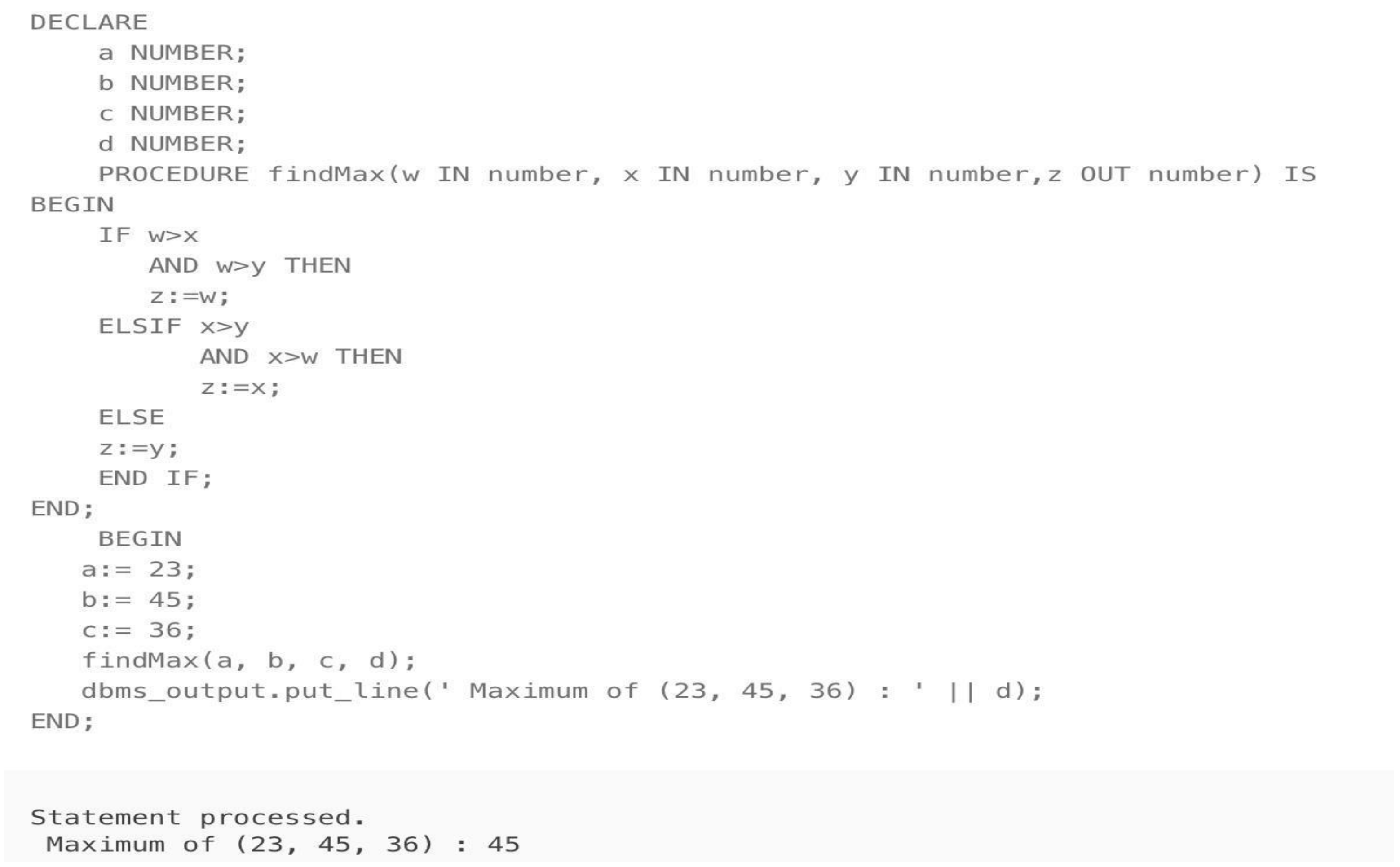


```
DECLARE
   a number;
PROCEDURE squareNum(x IN OUT number) IS
BEGIN
  x := x * x;
END;
BEGIN
   a:= 23;
   squareNum(a);
   dbms_output.put_line(' Square of (23): ' || a);
END;
```

```
Statement processed.
 Square of (23): 529
```

4. Write a sql procedure program to find the largest of given three numbers. (Hint: A, B, C as IN parameter and Large as OUT parameter)

```
DECLARE
   a NUMBER;
   b NUMBER;
   c NUMBER;
   d NUMBER;
   PROCEDURE findMax(w IN number, x IN number, y IN number,z OUT number) IS
BEGIN
   IF w>x
     AND w>y THEN
     z:=w;
   ELSIF x>y
       AND x>w THEN
       z:=x;
   ELSE
   z:=y;
   END IF;
END;
   BEGIN
  a:= 23;
  b:= 45;
  c:= 36;
  findMax(a, b, c, d);
  dbms_output.put_line(' Maximum of (23, 45, 36) : ' || d);
END;
```

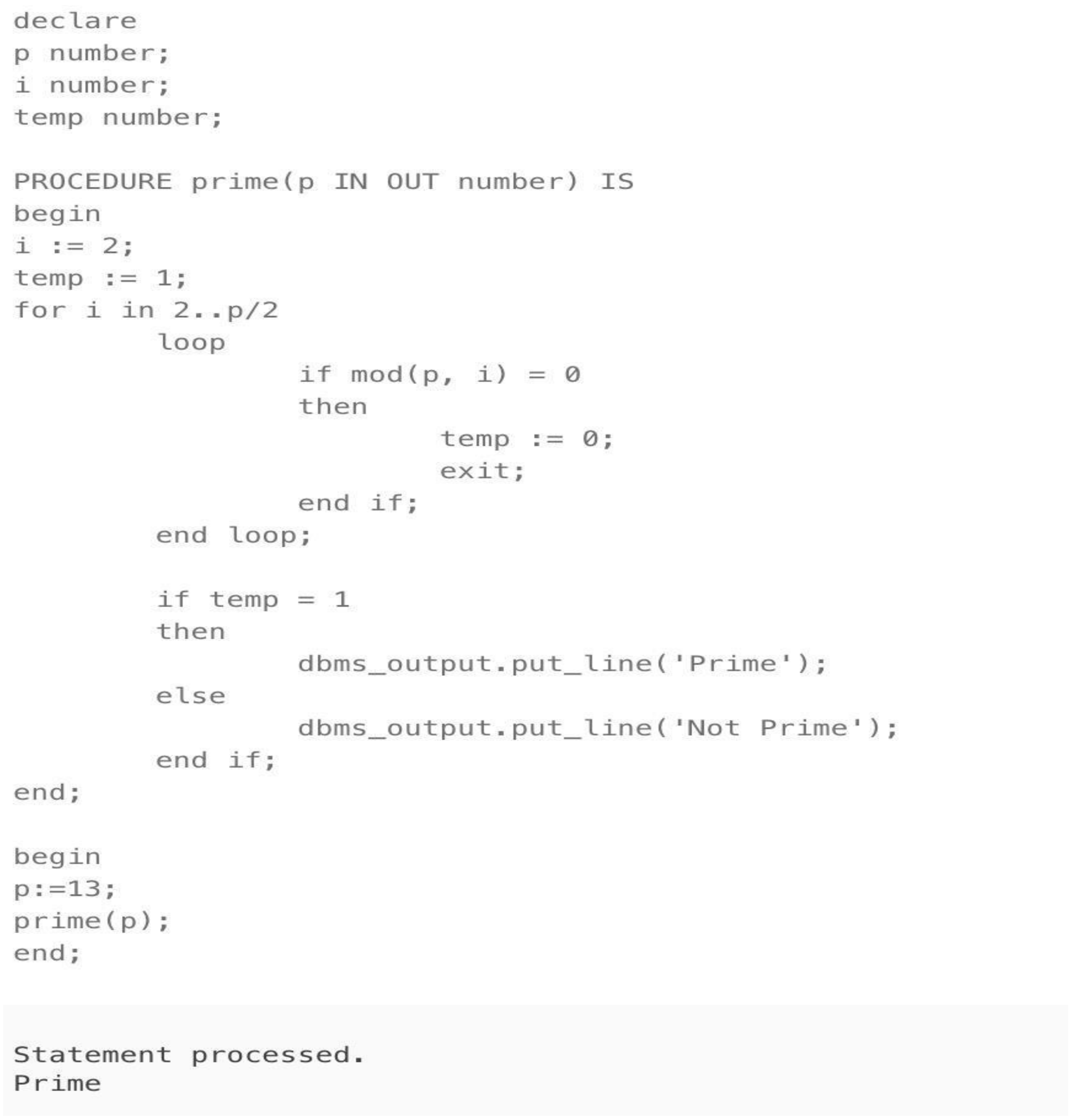


```
DECLARE
     a NUMBER;
     b NUMBER;
     c NUMBER;
     d NUMBER;
     PROCEDURE findMax(w IN number, x IN number, y IN number,z OUT number) IS
BEGIN
     IF w>x
        AND w>y THEN
        z:=w;
     ELSIF x>y
           AND x>w THEN
           z:=x;
     ELSE
     z:=y;
     END IF;
END;
     BEGIN
    a:= 23;
    b:= 45;
    c:= 36;
    findMax(a, b, c, d);
    dbms_output.put_line(' Maximum of (23, 45, 36) : ' || d);
END;
```

```
Statement processed.
 Maximum of (23, 45, 36) : 45
```

5. **Write a sql procedure program to find whether the given number is prime or not. (Hint: use P as IN OUT parameter)**

```
declare
p number;
i number;
temp number;

PROCEDURE prime(p IN OUT number) IS
begin
i := 2;
temp := 1;
for i in 2..p/2
loop
if mod(p, i) = 0
then
temp := 0;
exit;
end if;
end loop;

if temp = 1
then
dbms_output.put_line('Prime');
else
dbms_output.put_line('Not Prime');
end if;
end;

begin
p:=13;
prime(p);
end;
```



```
declare
p number;
i number;
temp number;

PROCEDURE prime(p IN OUT number) IS
begin
i := 2;
temp := 1;
for i in 2..p/2
        loop
                if mod(p, i) = 0
                then
                        temp := 0;
                        exit;
                end if;
        end loop;

        if temp = 1
        then
                dbms_output.put_line('Prime');
        else
                dbms_output.put_line('Not Prime');
        end if;
end;

begin
p:=13;
prime(p);
end;
```

```
Statement processed.
Prime
```

6. Write a sql procedure program to find the even or odd of a given number (Hint: Use A as IN OUT parameter)

```
DECLARE
A NUMBER;
PROCEDURE oddeven(A IN OUT number) IS
BEGIN
IF MOD(A,2) = 0 THEN
DBMS_OUTPUT.PUT_LINE ('The number '||A||' is even number');
ELSE
DBMS_OUTPUT.PUT_LINE ('The number '||A||' is odd number.');
END IF;
END;

begin
A:=6;
oddeven(A);
end;
```

```
DECLARE
A NUMBER;
PROCEDURE oddeven(A IN OUT number) IS
BEGIN
IF MOD(A,2) = 0 THEN
DBMS_OUTPUT.PUT_LINE ('The number '||A||' is even number');
ELSE
DBMS_OUTPUT.PUT_LINE ('The number '||A||' is odd number.');
END IF;
END;

begin
A:=6;
oddeven(A);
end;
```

```
Statement processed.
The number 6 is even number
```

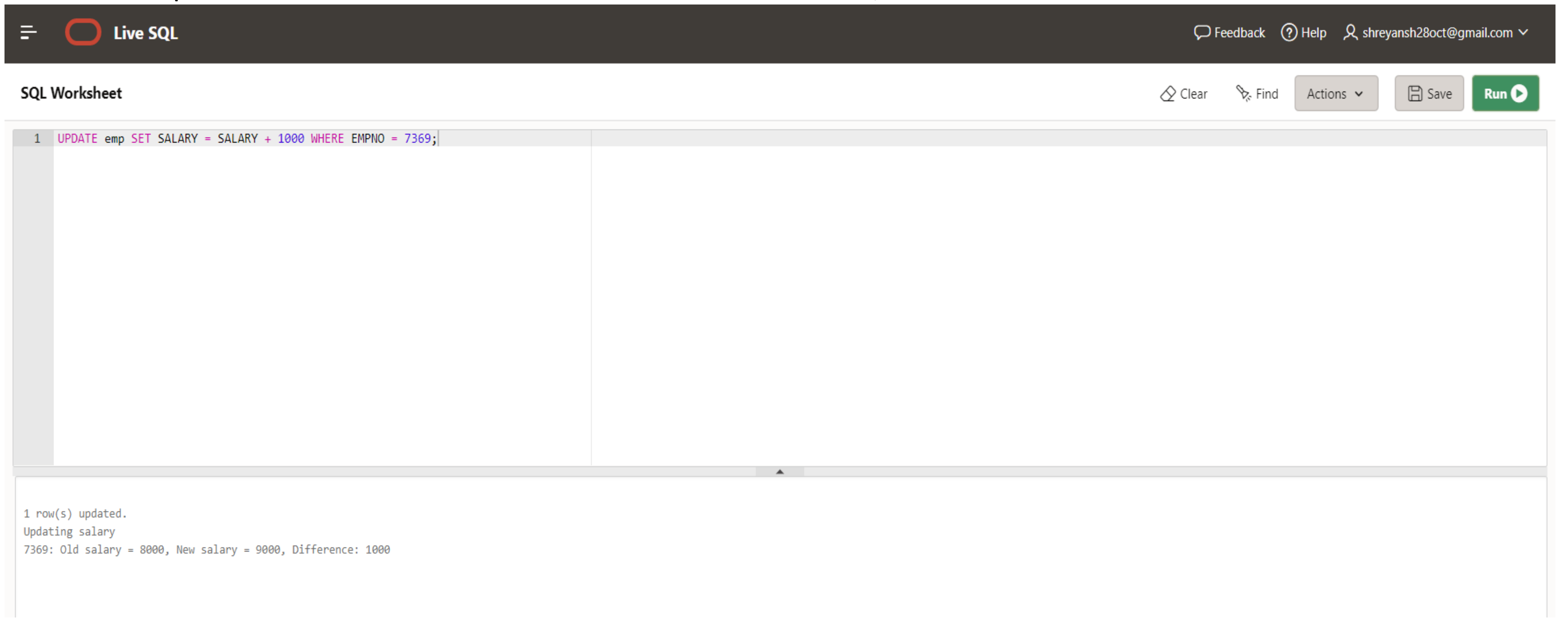
Experiment – 10

1. **Create a row-level trigger for the EMPLOYEE table that would get executed by the DML statement like UPDATE OR INSERT on that table. The trigger should compute and show the SALARY difference between current and previous values. (Hint: previous salary:, current salary:___, salary difference:____)**

```
Create table EMP(EmpNo number(4), "Emp Name" varchar2(25), Salary number(6),Age number(4), Job
varchar2(15));

Alter table EMP add marital_status varchar2(25);

INSERT INTO EMP VALUES (7369, 'Shreyansh Sachan', 8000, 25, 'Founder', 'NM');
INSERT INTO EMP VALUES (7521, 'Ravi Gupta', 7000, 26, 'CoFounder','NM');
INSERT INTO EMP VALUES (7934, 'Rahul Tripathi', 6000, 27, 'Chief Advisor','M');
INSERT INTO EMP VALUES (7902, 'Sachin Tilokani', 2000, 28, 'Secretary','M');
INSERT INTO EMP VALUES (7040, 'Param Shah', 1600, 29, 'CMO','NM');
INSERT INTO EMP VALUES (7566, 'Annahita Patel', 950, 22, 'Trainee','NM');
INSERT INTO EMP VALUES (7839, 'Yash Singh', 950, 25, 'Chief Of Staff','M');
INSERT INTO EMP VALUES (7789, 'Mayank Agarwal', 2800, 23, 'Janitor','M');


CREATE OR REPLACE TRIGGER tBEFORE
    INSERT OR
    UPDATE OF SALARY, EMPNO ORDELETE
```

```
  ON empBEGIN CASE
    WHEN INSERTING THEN DBMS_OUTPUT.PUT_LINE('Inserting');
    WHEN UPDATING('SALARY') THEN
      DBMS_OUTPUT.PUT_LINE('Updating salary');WHEN UPDATING('EMPNO')
    THEN
      DBMS_OUTPUT.PUT_LINE('Updating empno');WHEN DELETING THEN
    DBMS_OUTPUT.PUT_LINE('Deleting');
  END CASE;
 END;


CREATE OR REPLACE TRIGGER print_salary_changesBEFORE DELETE OR INSERT OR UPDATE ON emp FOR EACH ROW
 DECLARE
  sal_diff  NUMBER;
 BEGIN
  sal_diff := :NEW.SALARY - :OLD.SALARY; DBMS_OUTPUT.PUT(:NEW.EMPNO || ': ');
  DBMS_OUTPUT.PUT('Old salary = ' || :OLD.SALARY || ', '); DBMS_OUTPUT.PUT('New
  salary = ' || :NEW.SALARY || ', ');
  DBMS_OUTPUT.PUT_LINE('Difference: ' || sal_diff);END;
```

UPDATE emp SET SALARY = SALARY + 1000 WHERE EMPNO = 7369;



```
UPDATE emp SET SALARY = SALARY + 1000 WHERE EMPNO = 7369;
```

```
1 row(s) updated.
Updating salary
7369: Old salary = 8000, New salary = 9000, Difference: 1000
```

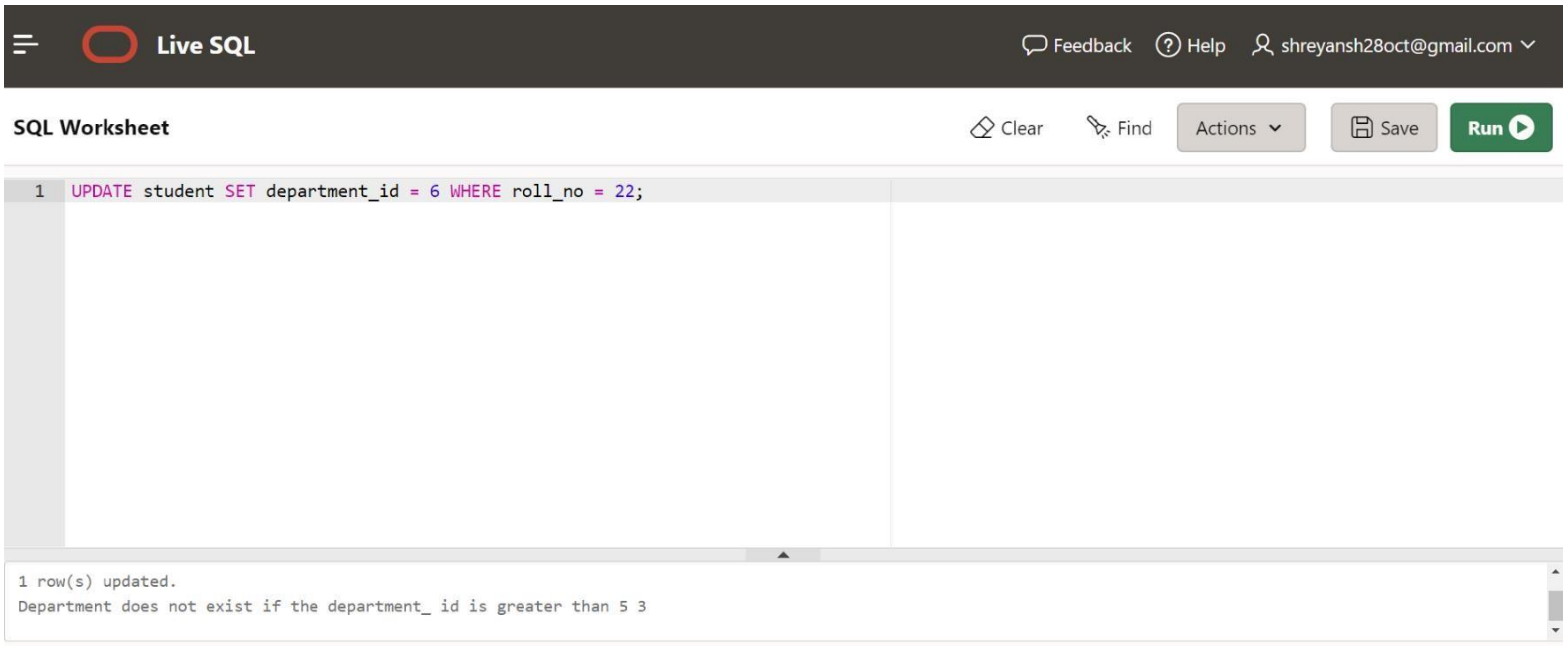
2. **Create a trigger for the STUDENT table that would get executed by the DML statement like UPDATE OR INSERT on that table. The trigger will compute and show the message "Department does not exist if the department_ id is greater than 5".**

create table student (department_id number(4), student_name varchar2(25), roll_nonumber(4), grade varchar2(15))

```
INSERT INTO student VALUES (1, 'Shreyansh Sachan',25,'9th') ;
INSERT INTO student VALUES (2, 'Ravi Gupta', 26,'10th');
INSERT INTO student VALUES (3, 'Rahul Tripathi', 27,'11th') ;
INSERT INTO student VALUES (4, 'Sachin Tilokani', 28,'10th') ;
INSERT INTO student VALUES (5, 'Param Shah', 29,'12th') ;
INSERT INTO student VALUES (3, 'Annahita Patel', 22,'9th') ;
INSERT INTO student VALUES (1, 'Yash Singh',25,'11th') ;
INSERT INTO student VALUES (2, 'Mayank Agarwal', 23,'10th');
```

```
CREATE OR REPLACE TRIGGER display_department_changesBEFORE DELETE OR INSERT OR
UPDATE ON student
FOR EACH ROW
WHEN (NEW.department_id > 5)BEGIN
  dbms_output.put_line('Department does not exist if the department_ id is greater than5 ' || :OLD.department_id);
END;
```

UPDATE student SET department_id = 6 WHERE roll_no = 22;

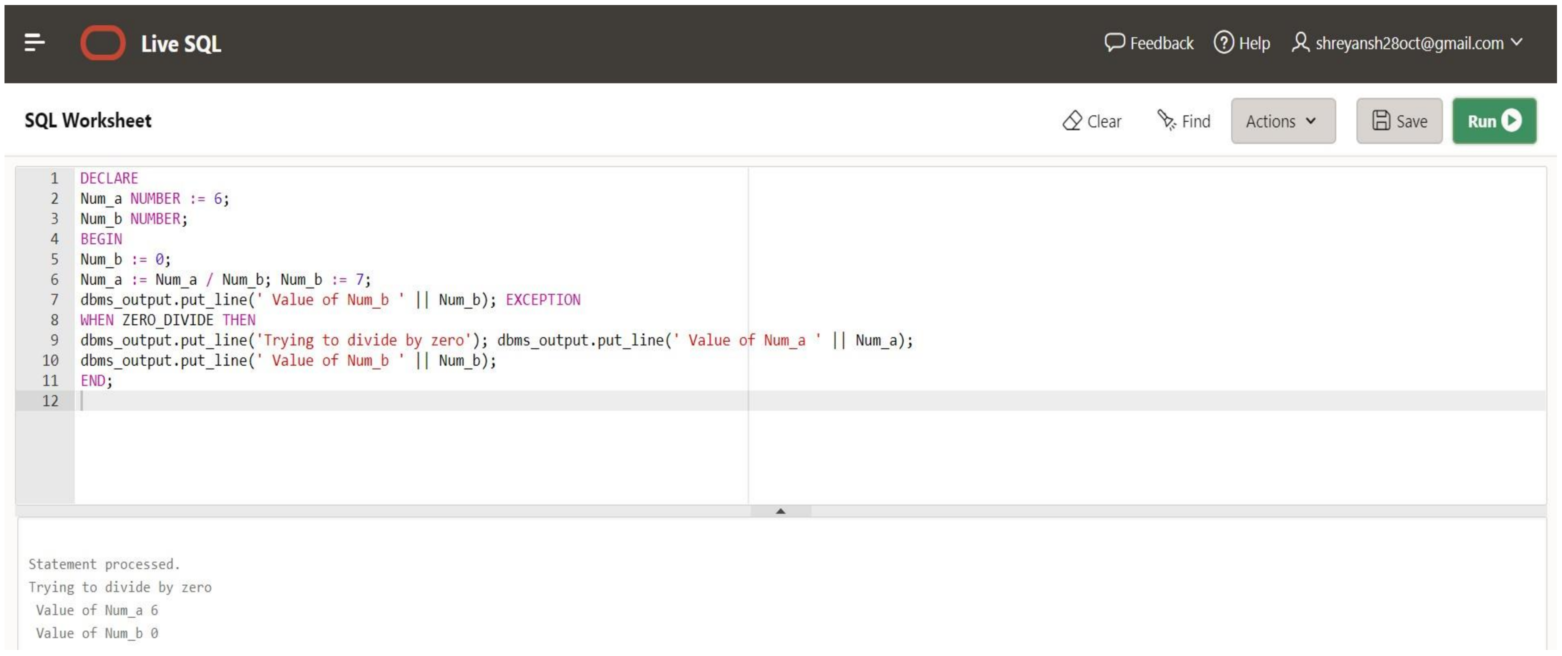


```
UPDATE student SET department_id = 6 WHERE roll_no = 22;
```

```
1 row(s) updated.
Department does not exist if the department_ id is greater than 5 3
```

3. Write a program to raise exception WHEN dividing with zero.

```
DECLARE
   Num_a NUMBER := 6;
   Num_b NUMBER;
BEGIN
   Num_b := 0;
   Num_a := Num_a / Num_b;Num_b := 7;
   dbms_output.put_line(' Value of Num_b ' || Num_b);EXCEPTION
   WHEN ZERO_DIVIDETHEN
         dbms_output.put_line('Trying to divide by zero'); dbms_output.put_line('
         Value of Num_a ' || Num_a);
         dbms_output.put_line(' Value of Num_b ' || Num_b);
END;
```

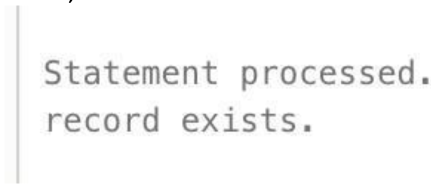


```
DECLARE
Num_a NUMBER := 6;
Num_b NUMBER;
BEGIN
Num_b := 0;
Num_a := Num_a / Num_b; Num_b := 7;
dbms_output.put_line(' Value of Num_b ' || Num_b); EXCEPTION
WHEN ZERO_DIVIDE THEN
dbms_output.put_line('Trying to divide by zero'); dbms_output.put_line(' Value of Num_a ' || Num_a);
dbms_output.put_line(' Value of Num_b ' || Num_b);
END;
```

```
Statement processed.
Trying to divide by zero
 Value of Num_a 6
 Value of Num_b 0
```

4. Write a program to check whether the name entered is existing in database entered or not.

```
Declare
  n_count number;
Begin
  Select count(1) into n_countfrom emp
   Where job = 'CMO';

  if n_count > 0 then
    -- do something here if exists dbms_output.put_line('record
  exists.'); else
    -- do something here if not exists dbms_output.put_line('record does not
  exists.'); end if;
End;
```



```
Statement processed.
record exists.
```