**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

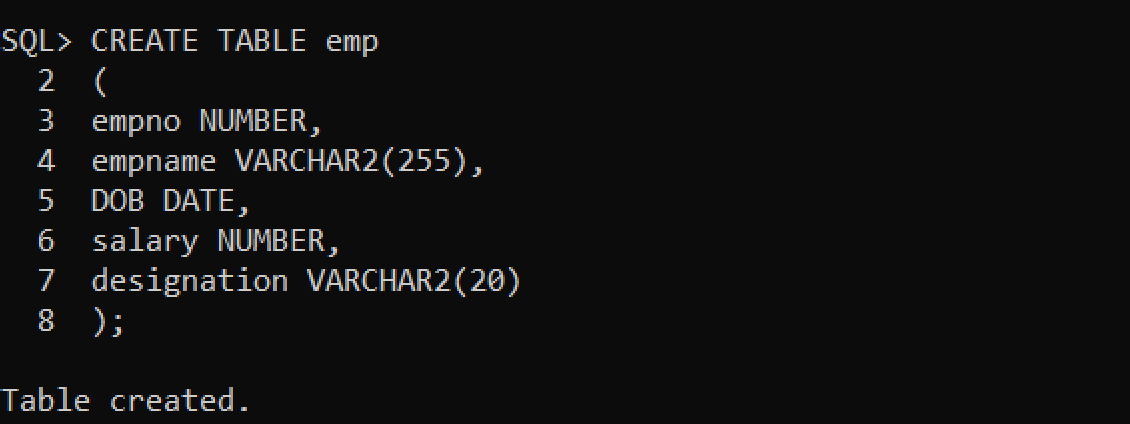
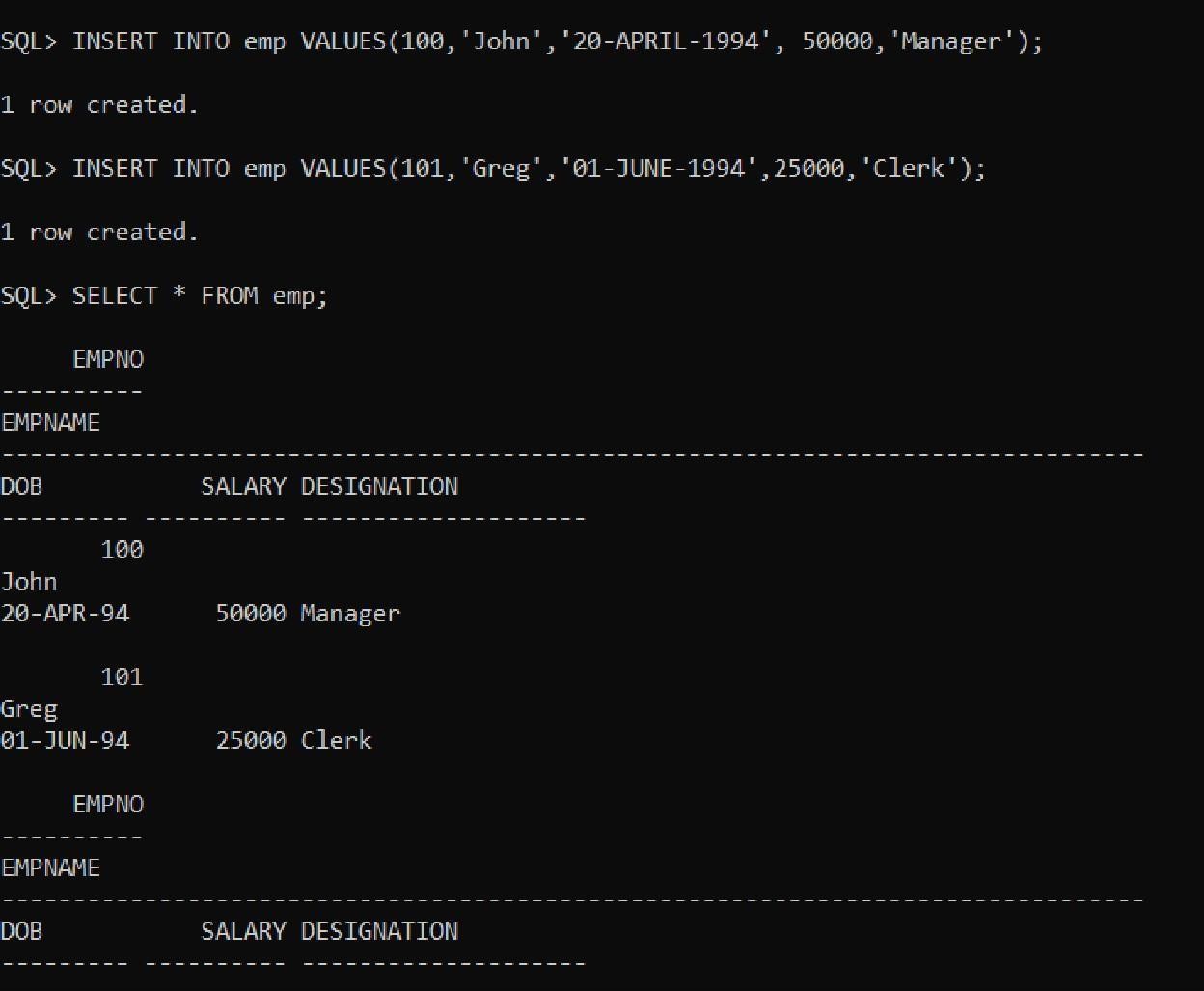
##### Course Title: Database Management System Laboratory

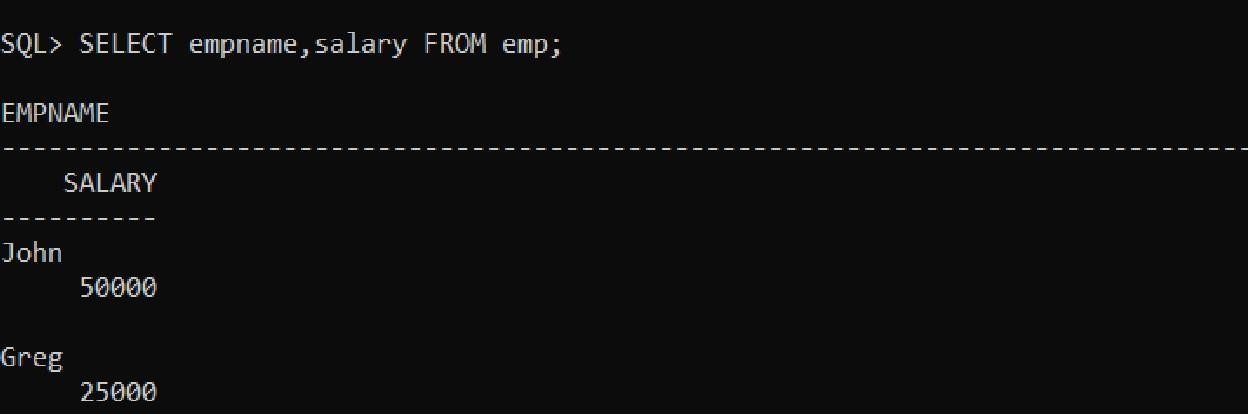
|  |  |
| --- | --- |
| **Exp. No.** | **Title** |
| 1 | SQL Data Definition Language (DDL) |
| 2 | SQL Data Manipulation Language (DML) |
| 3 | SQL Data Control Language Commands and Transaction Control Commands |
| 4 | Inbuilt functions in SQl |
| 5 | ER Diagram |
| 6 | Nested Queries |
| 7 | Join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

**Course Coordinator Ms.S.Ushasukhanya**

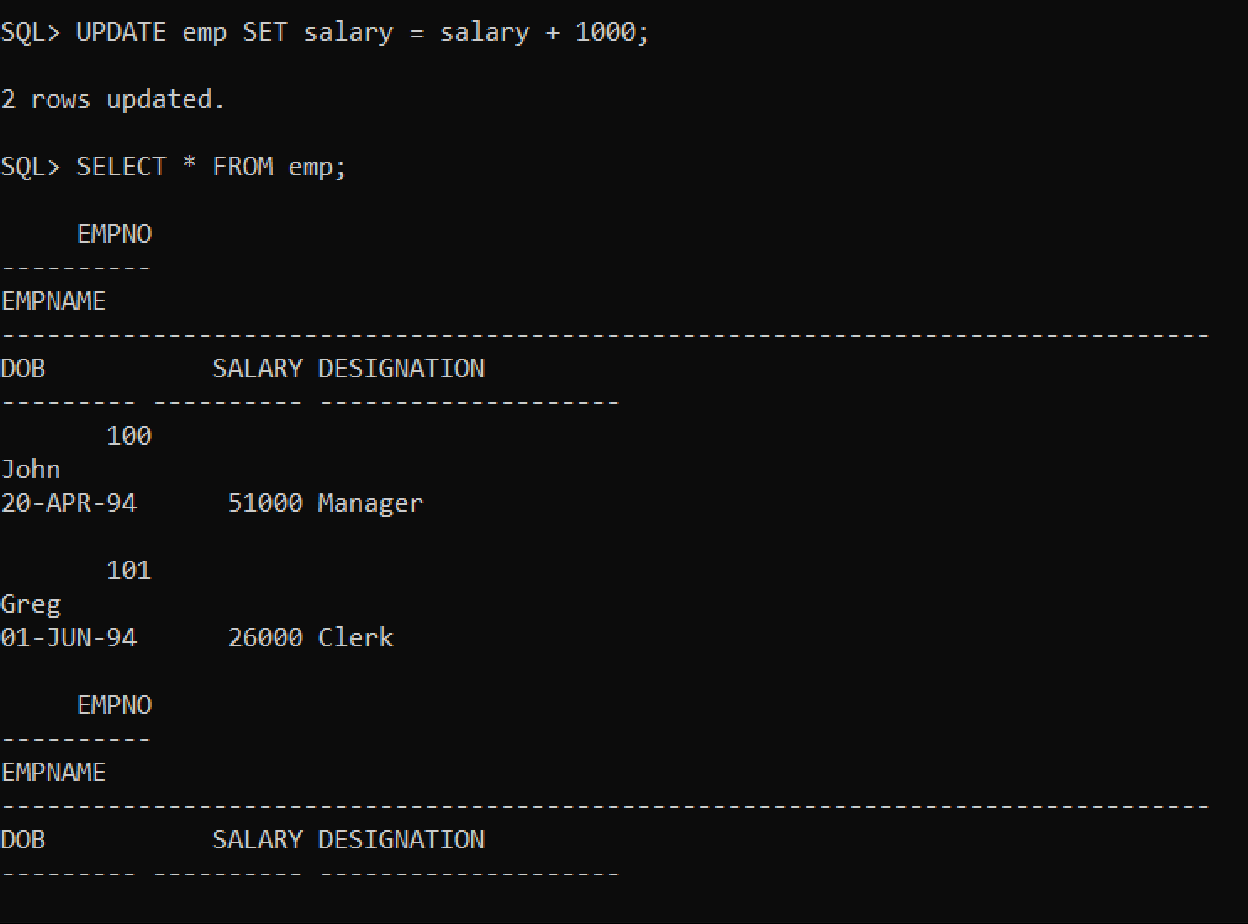
### Experiment – 1 SQL BASIC COMMANDS

#### QUERIES:

1. **Create table**
2. **Insert values**
3. **Display values**



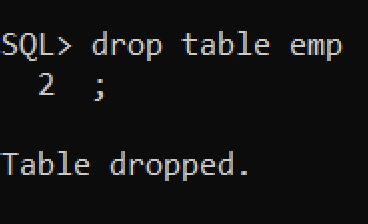
#### Modify values



1. **Delete values**

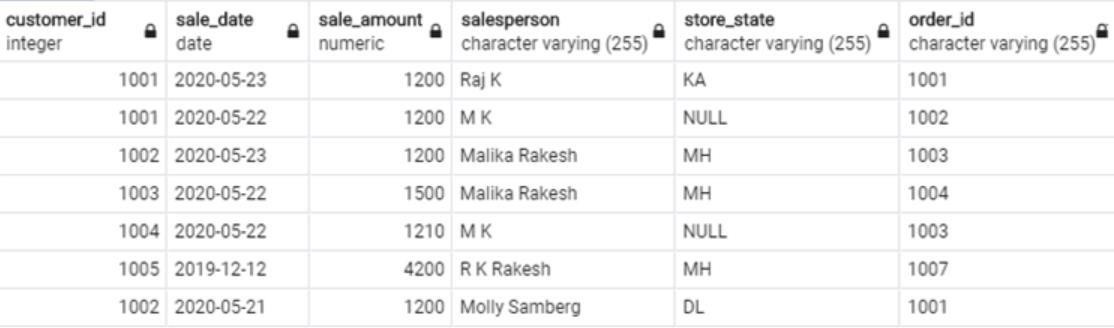


1. **Drop Table**



# Experiment – 2 SQL DML COMMANDS

Data base created for this exercise is:



**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:

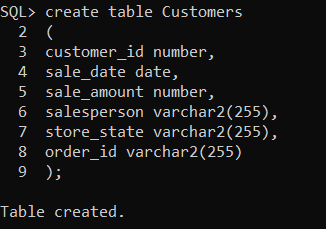
INSERT INTO customers(

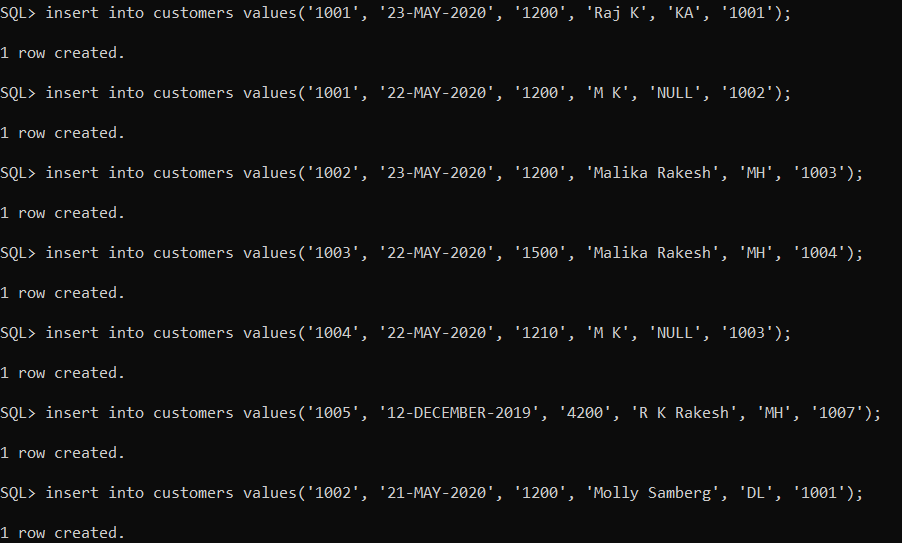
customer\_id, sale\_date, sale\_amount, salesperson, store\_state, order\_id) VALUES (1005,'12-DECEMBER-2019',4200,'R K Rakesh','MH','1007');

(or)

INSERT INTO customers

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





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

**Syntax:**

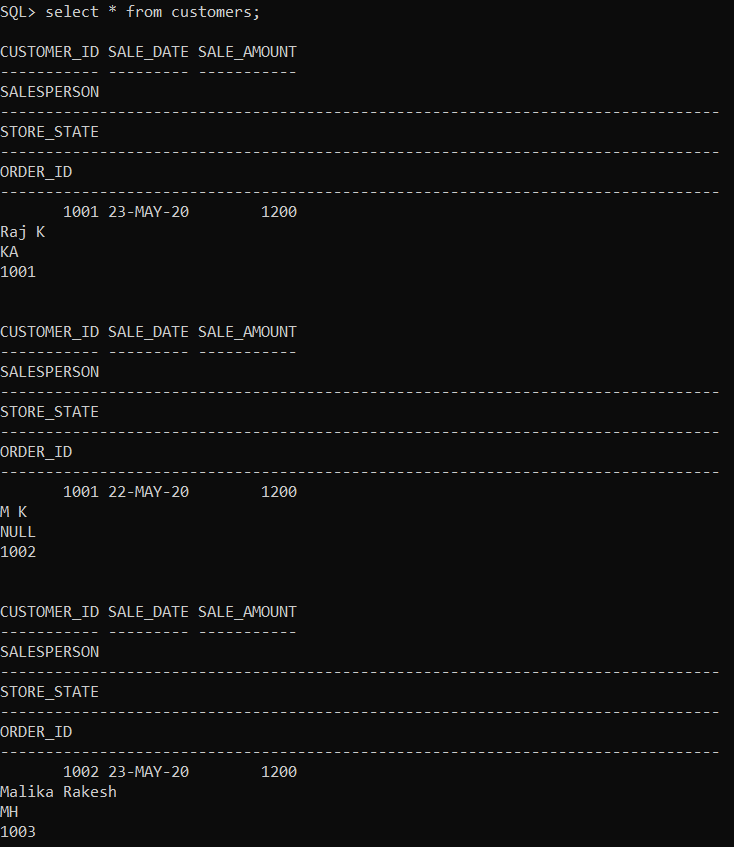
SELECT column\_name1, column\_name2, …

FROM table\_name

WHERE condition\_ expression;

#### Example:

Select customer\_id, sale\_date, order\_id, store\_state from customers; Select \* from customers;



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

#### Syntax:

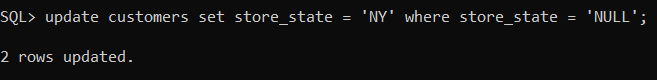
UPDATE table\_name

SET column\_name\_1 = value1, column\_name\_2 = value2, ... WHERE condition;

#### Example:

UPDATE customers SET store\_state = 'DL'

WHERE store\_state = 'NY';



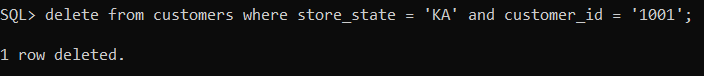
* **DELETE -** Used to remove one or more rows from the database table

#### Syntax:

DELETE FROM table\_name WHERE condition;

#### Example:

DELETE FROM customers WHERE store\_state = 'MH'

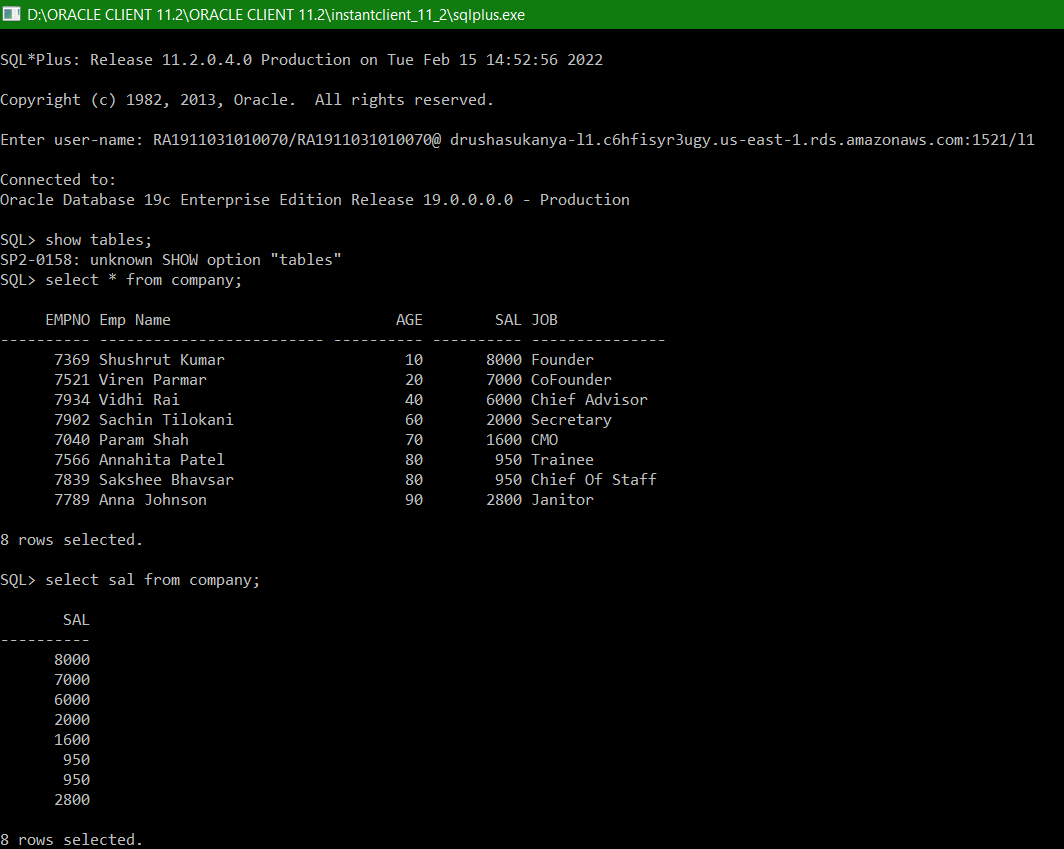


AND customer\_id = '1001';

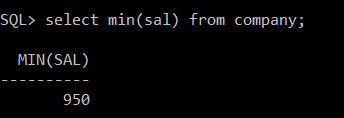
# Experiment – 3 DCL

**Queries:**

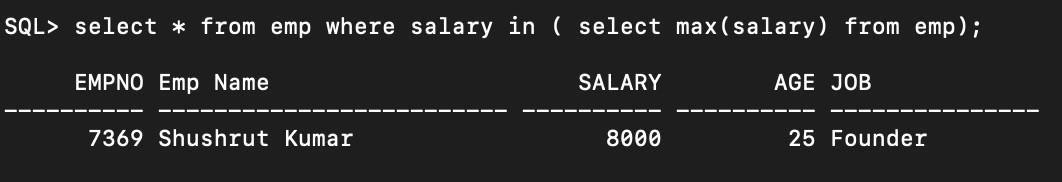
1. List the distinct salary records in the company table.



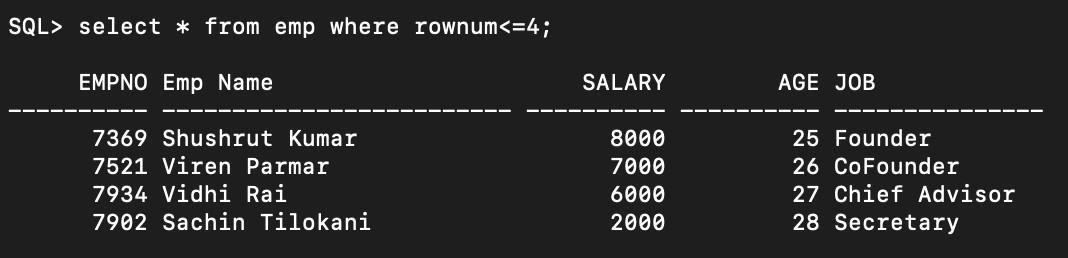
1. List the records in the company table with minimum salary.



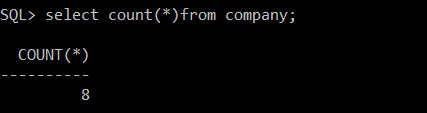
1. List the records in the company table with maximum salary.



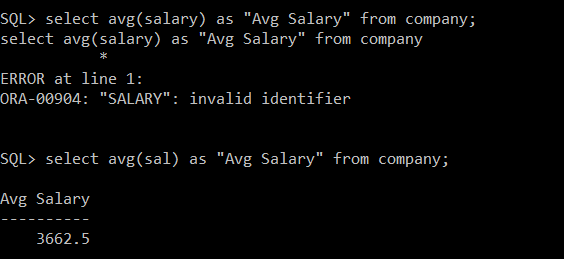
1. List the top 4 records in the company table.



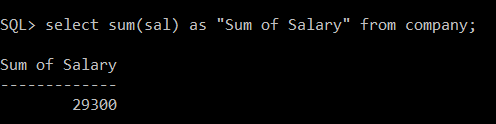
1. Count the number of records in the company table.



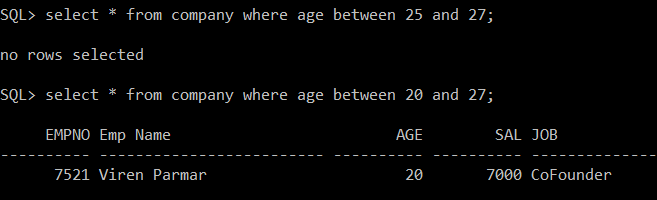
1. Find the average salary from the company table.



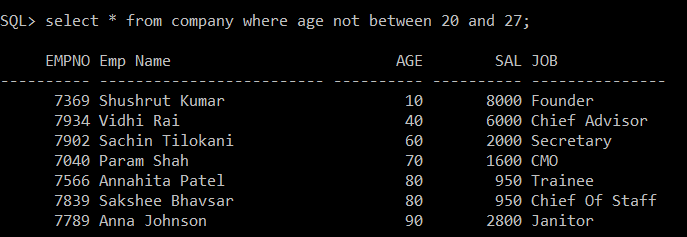
1. Find the sum of salary from the company table.



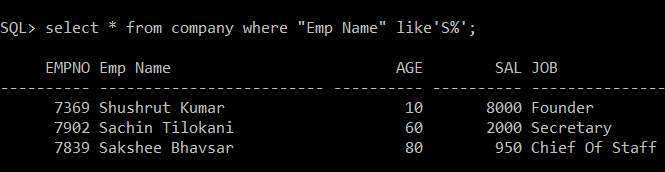
1. List the records from the company table where age ranges between 20 to 27.



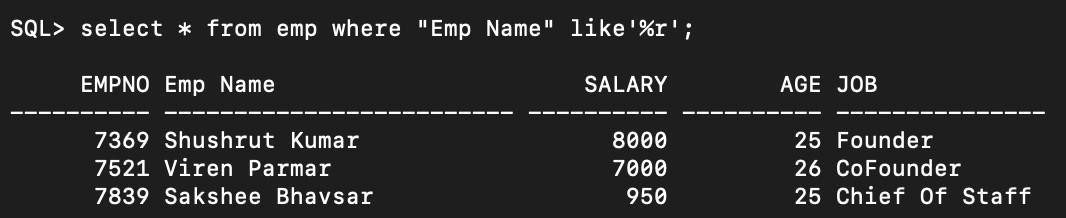
1. List the records from the company table where age ranges not between 25 to 27.



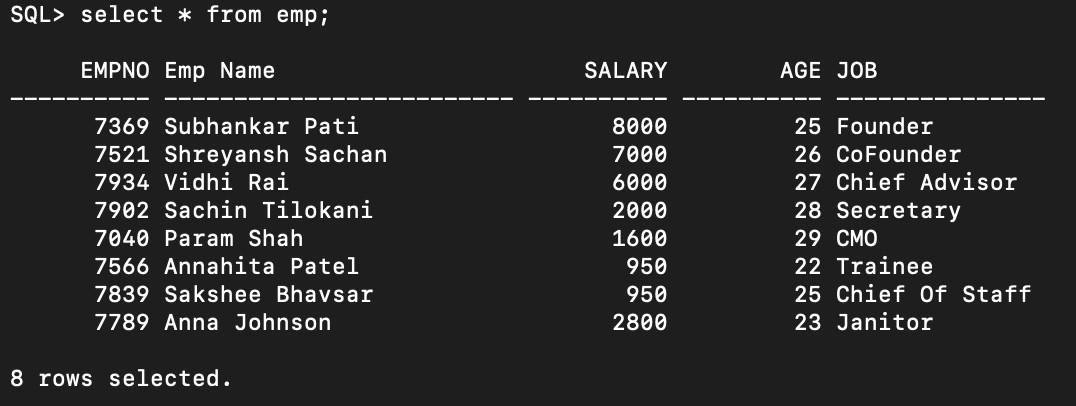
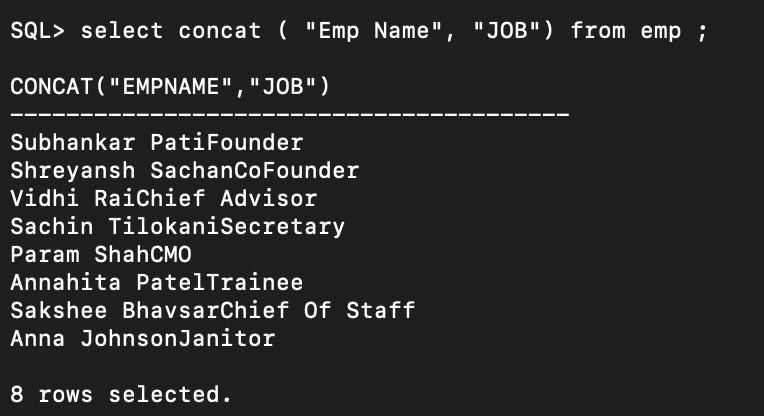
1. List the names of the employees from the company where name starts with 'S'.

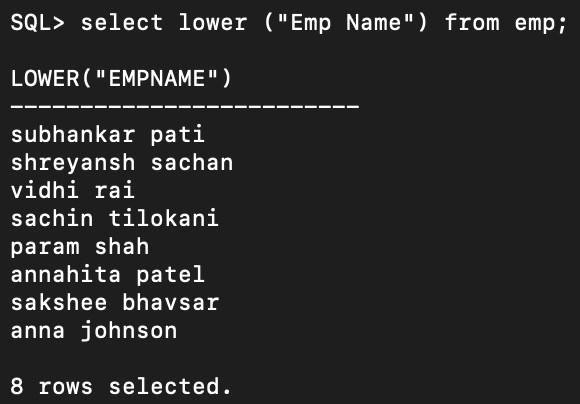


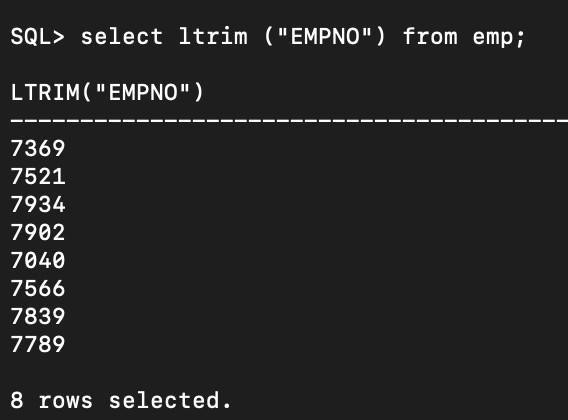
1. List the names of the employees from the company where name ends with 'r'.

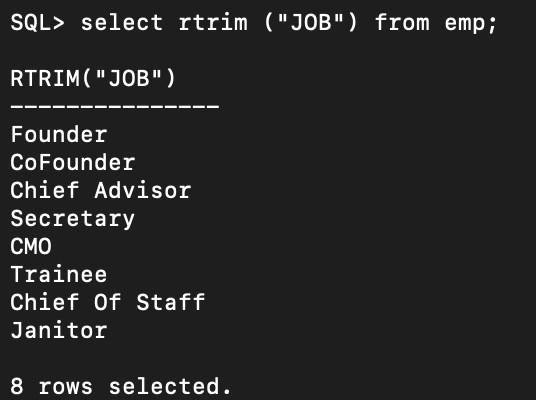


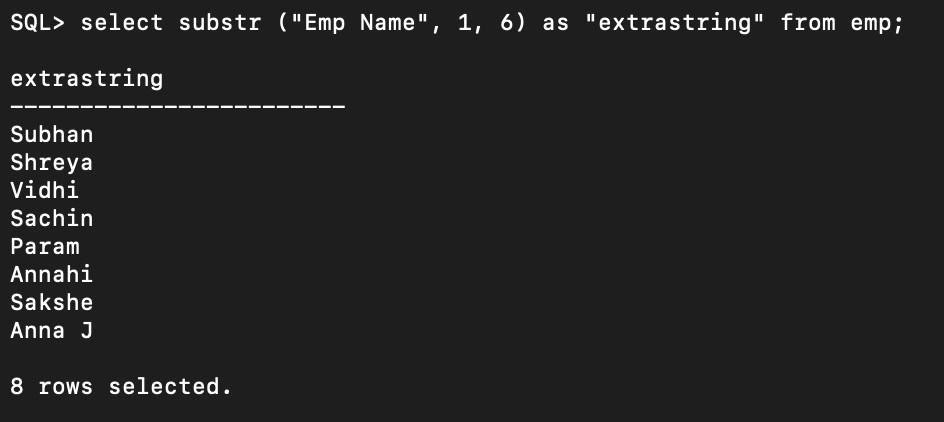
**Experiment – 4 Inbuild Functions**

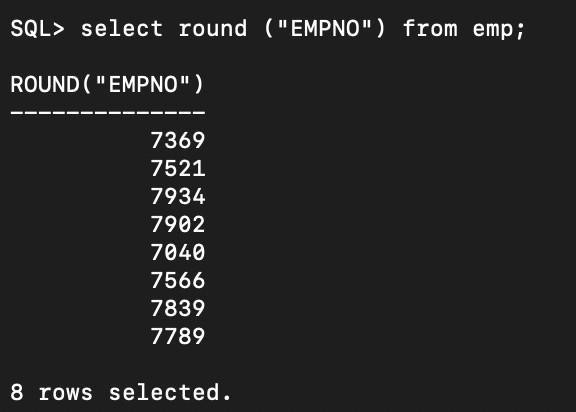
1. Display all records
2. Concat
3. Lower

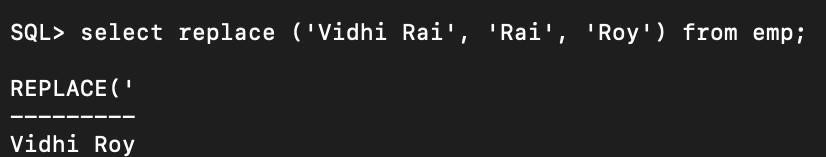
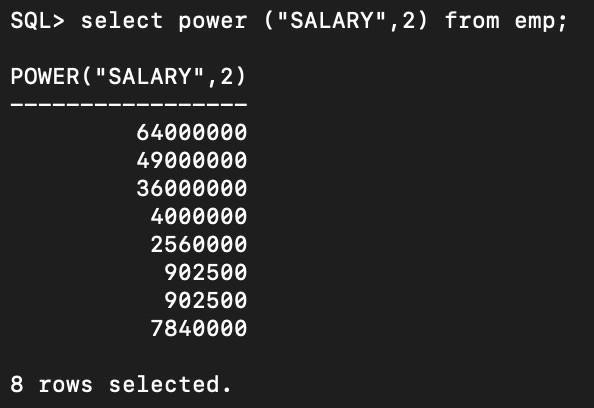


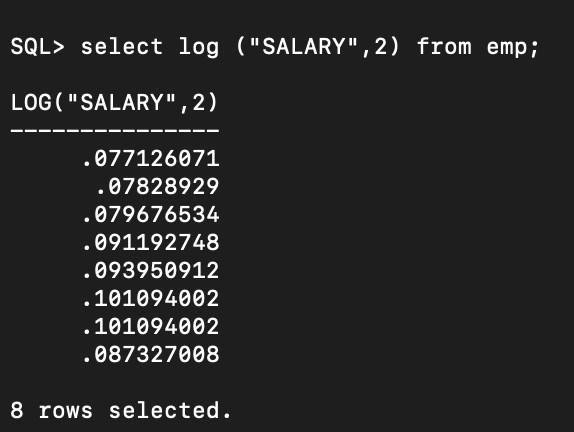
1. LTRIM
2. RTRIM

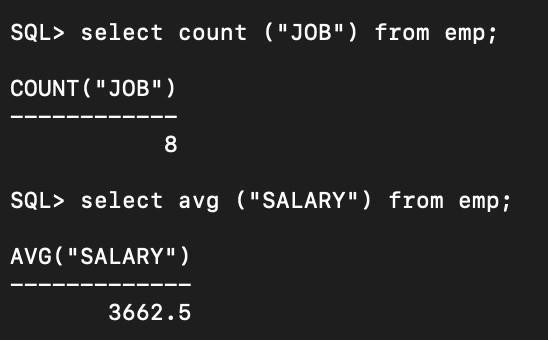
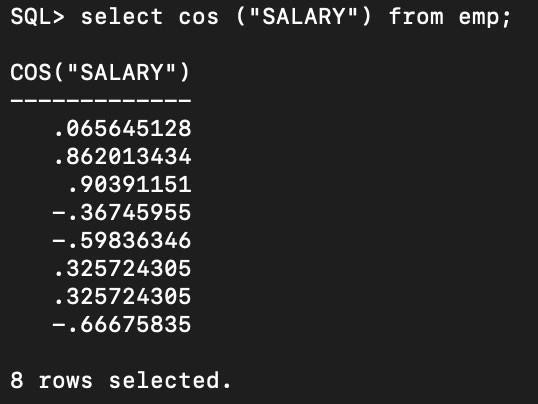


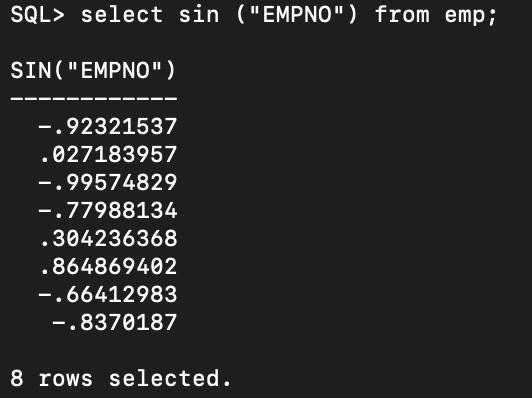
1. Substring
2. Round



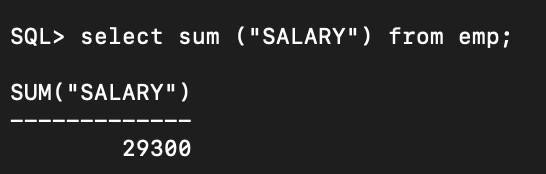
1. Replace
2. Power
3. Log(2)

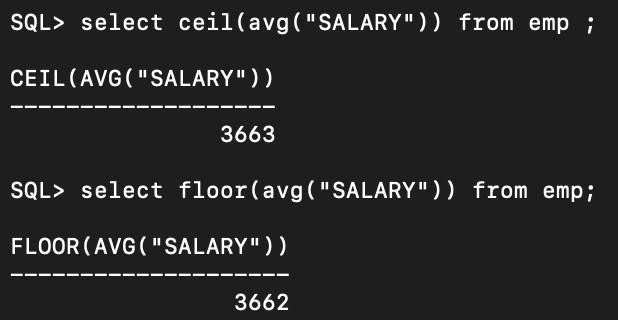


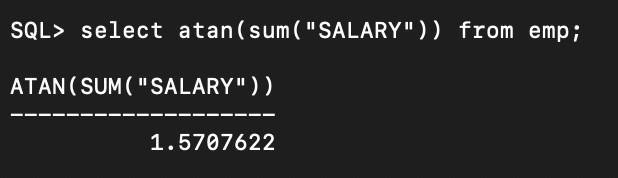
1. Count & 12. Avg
2. COS
3. SIN



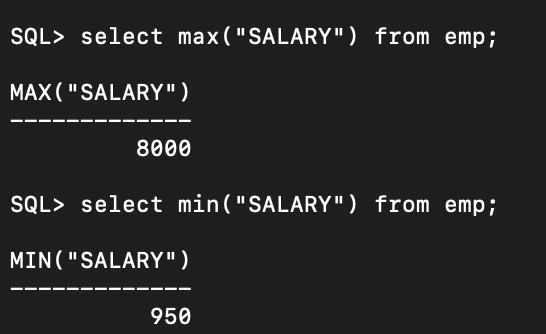
1. Sum



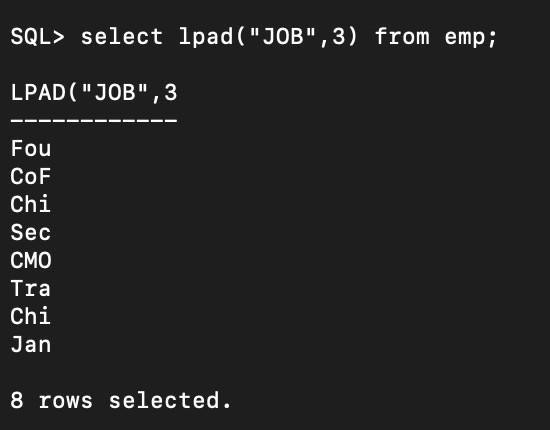
1. Ceiling
2. Atan



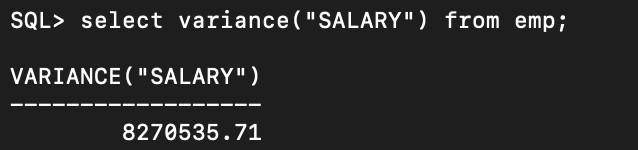
1. Max and 19. Min

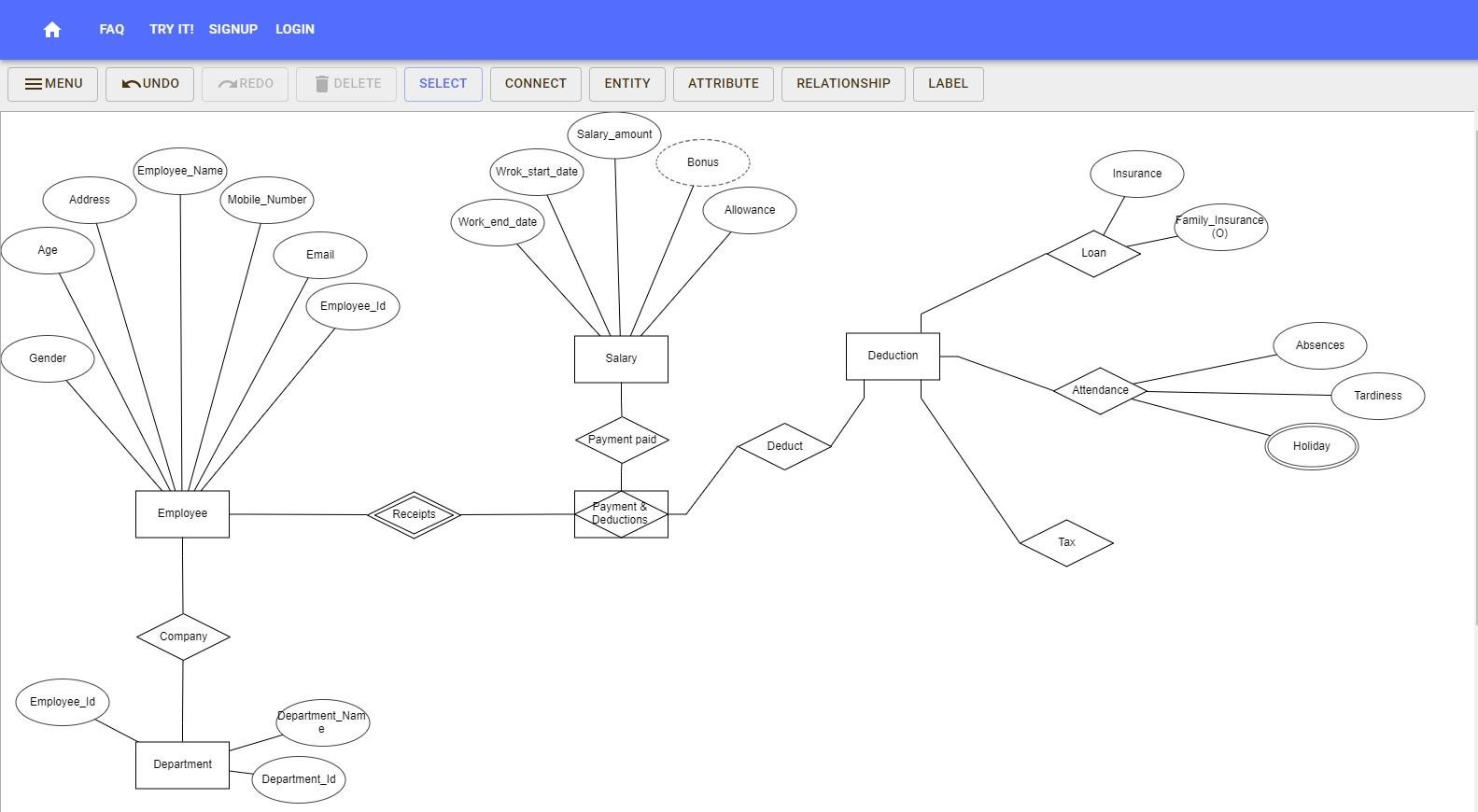


1. LPAD



1. Variance



**EXPERIMENT 5**

**Entity Relationship Diagram**

#### EXPERIMENT 6

##### Write the following queries in SQL, using the university schema. Create a table with appropriate attributes.

* + 1. **Find the titles of courses in the Comp. Sci. department that have 3 credits.**

##### Find the IDs of all students who were taught by an instructor named Einstein; make sure there are no duplicates in the result.

* + 1. **Find the highest salary of any instructor.**

##### Find all instructors earning the highest salary (there may be more than one with the same salary).

* + 1. **Find the enrollment of each section that was offered in Fall 2017.**

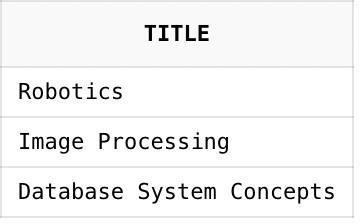
##### Find the maximum enrollment, across all sections, in Fall 2017.

* + 1. **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;



##### 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';



##### C.

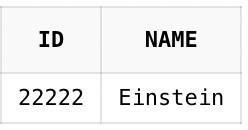
Query:

select max(salary) from instructor;

##### D.

Query:

select ID, name from instructor where salary = (select max(salary) from instructor);



##### E.

Query:

select course\_id, sec\_id, (select count(ID)

from takes

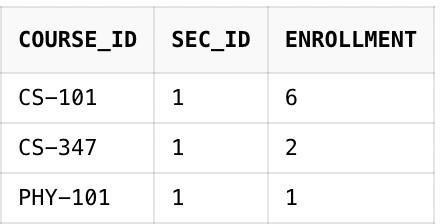
where takes.year = section.year

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;



##### 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);



##### 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

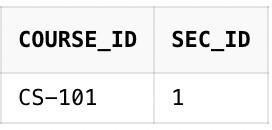
and takes.semester = section.semester

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

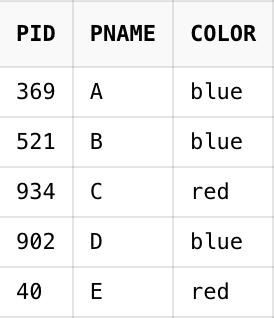
from sec\_enrollment

where enrollment = (select max(enrollment) from sec\_enrollment);

##### 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:

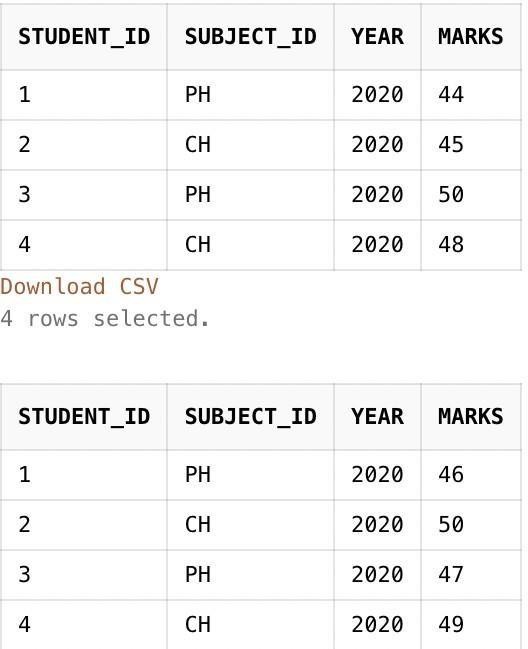


Query:

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

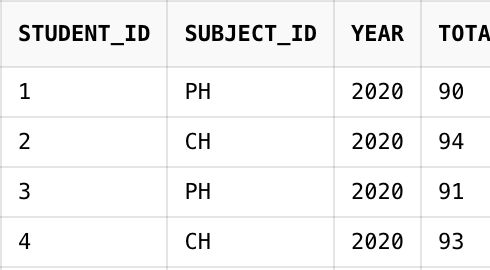


##### Write a query to find the sum of marks for each student from two tables



Query:

SELECT finalterm.student\_id, finalterm.subject\_id, finalterm.year,( midterm.marks+finalterm.marks) AS total FROM midterm, finalterm;



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

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

Query:

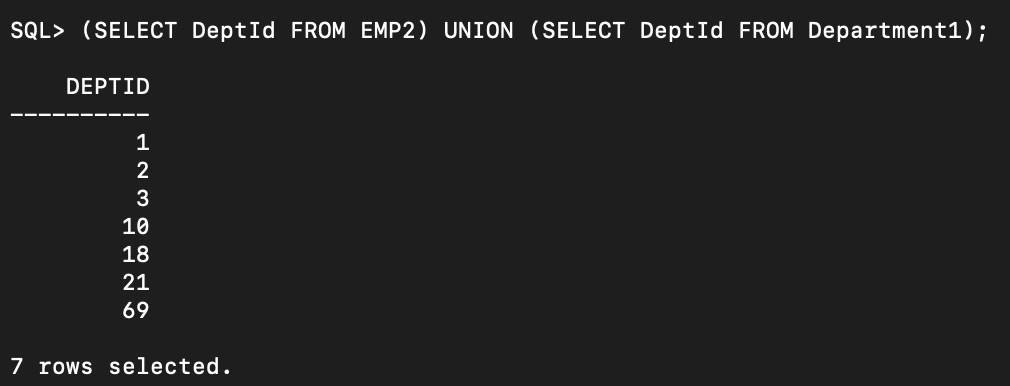
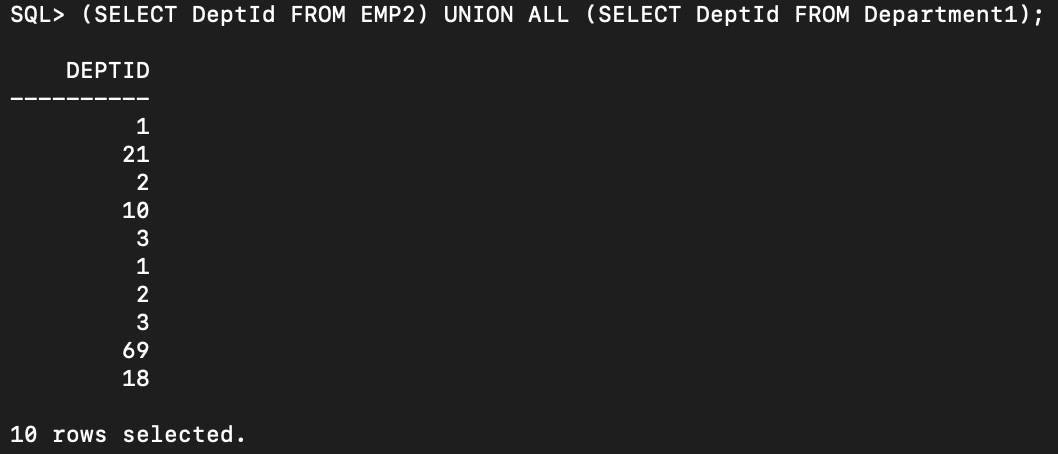
select pid from reservation where pclass='AC' and exists (select \* from passengerwhere age > '65' AND passenger.pid = reservation.pid);

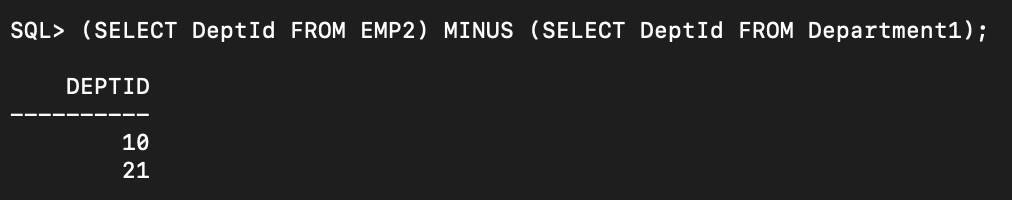


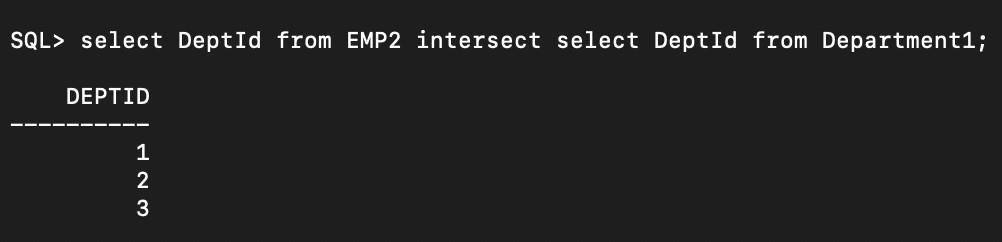
#### EXPERIMENT 7

##### Write the query to demonstrate the various set operators (UNION, UNION ALL, MINUS, INTERSECT)

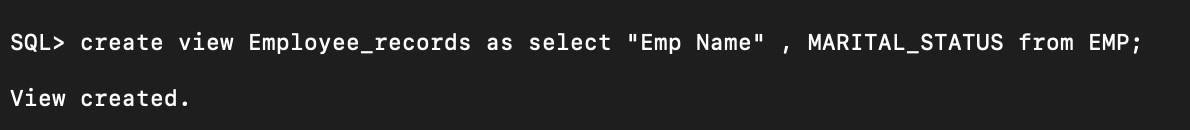
**Write a query using INTERSECT set operator to list the student id and residence location of the students.**

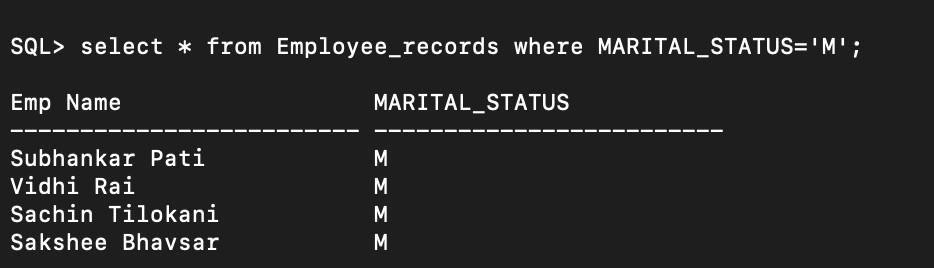




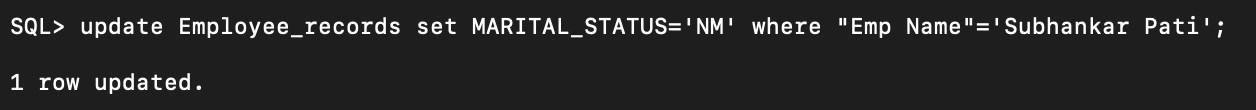
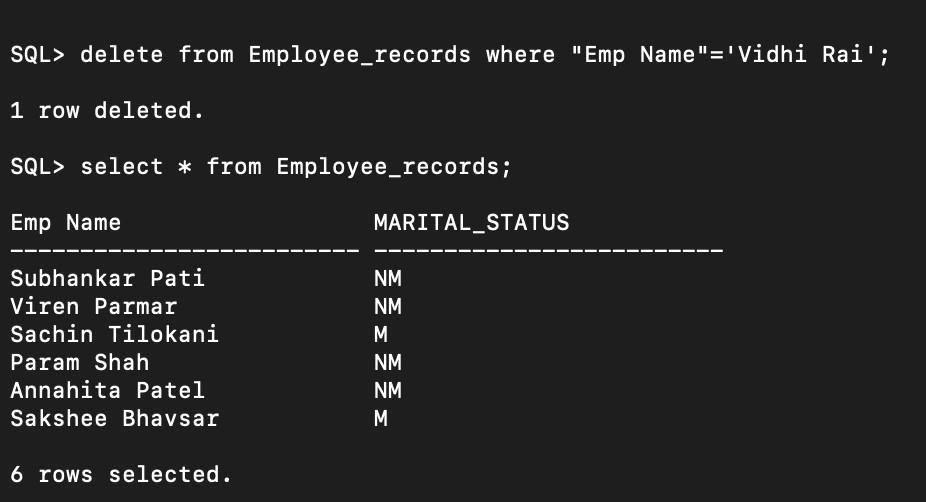


##### Write a query for SQL view (view name: Employee\_Records) to fetch columns of the table and filter the results using where clause with the martial\_status ‘M’.





**Q.Write a query to update, delete and insert from SQL view (view name: Employee\_Records) table.**



## Experiment – 8

##### 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

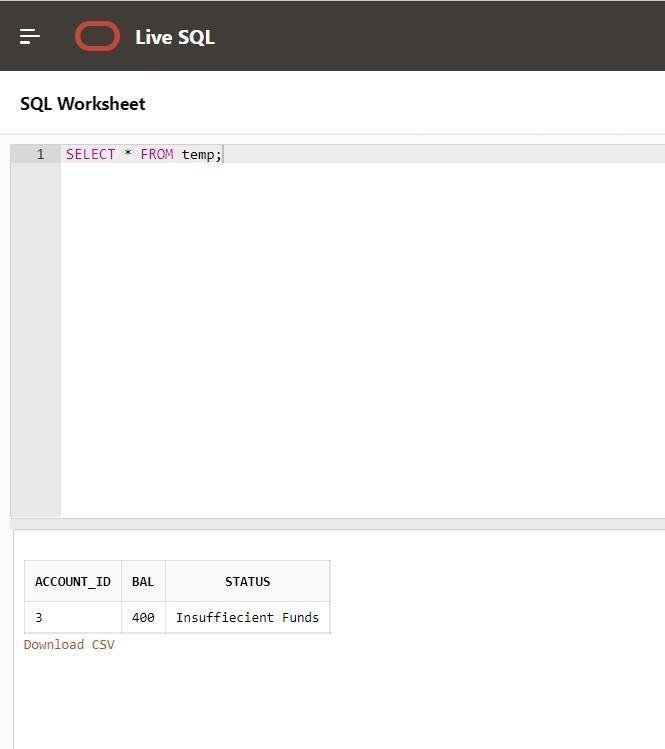
UPDATE accounts SET bal = bal- debit\_amt WHERE account\_id = acct;

ELSE

INSERT INTO temp VALUES

(acct, acct\_balance,'Insuffiecient Funds'); END IF;

COMMIT; END;



##### 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;

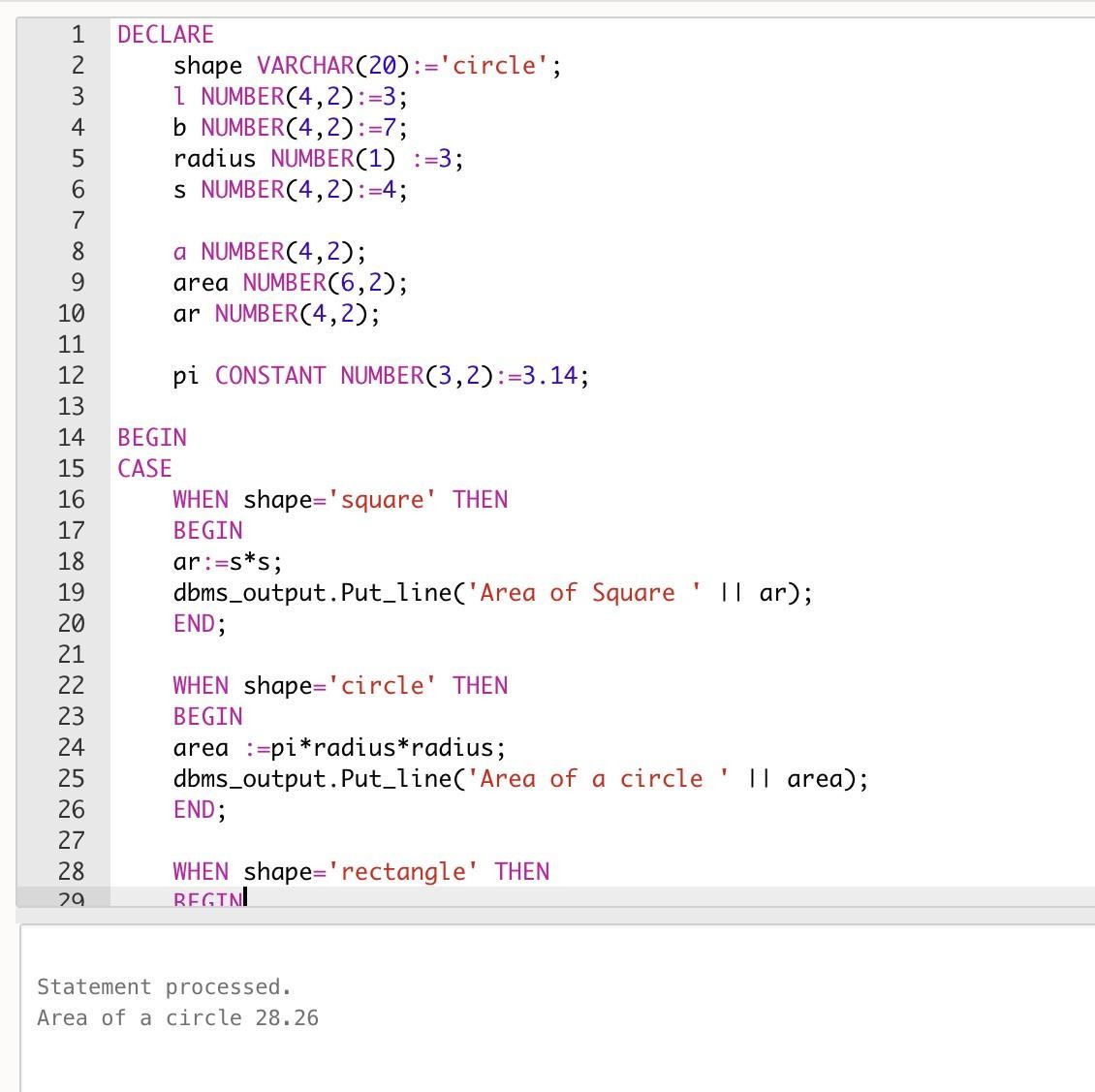
WHEN shape='circle' THEN BEGIN

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;

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

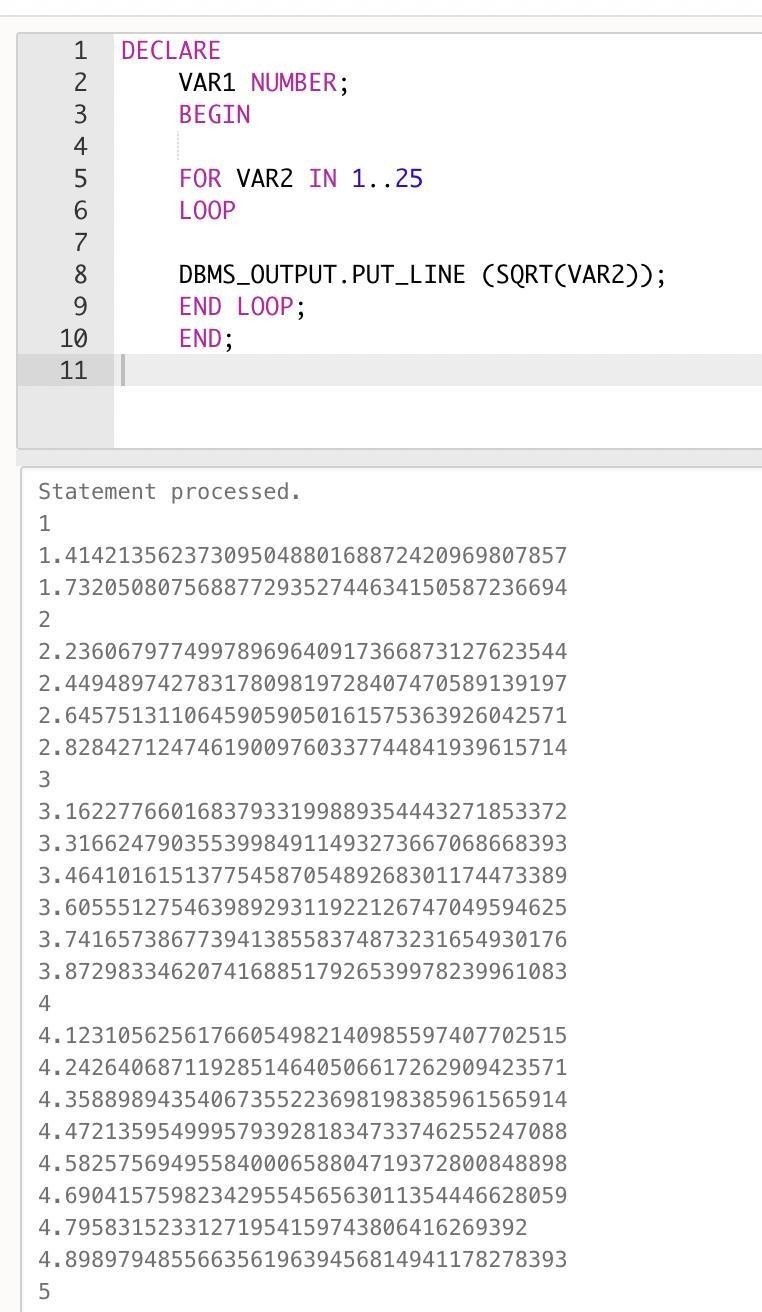
DECLARE

VAR1 NUMBER; BEGIN

FOR VAR2 IN 1..25 LOOP

DBMS\_OUTPUT.PUT\_LINE (SQRT(VAR2)); END LOOP;

END;



**Exp 9 PL/SQL**

#### 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;

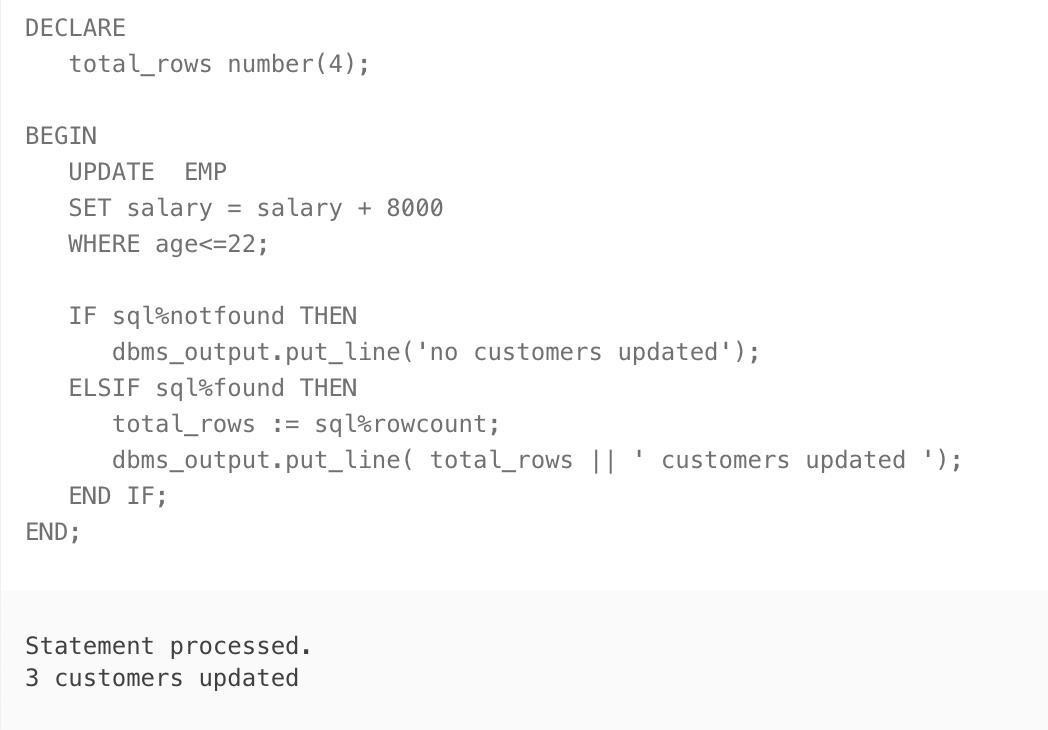
IF sql%notfound THEN

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;



#### 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

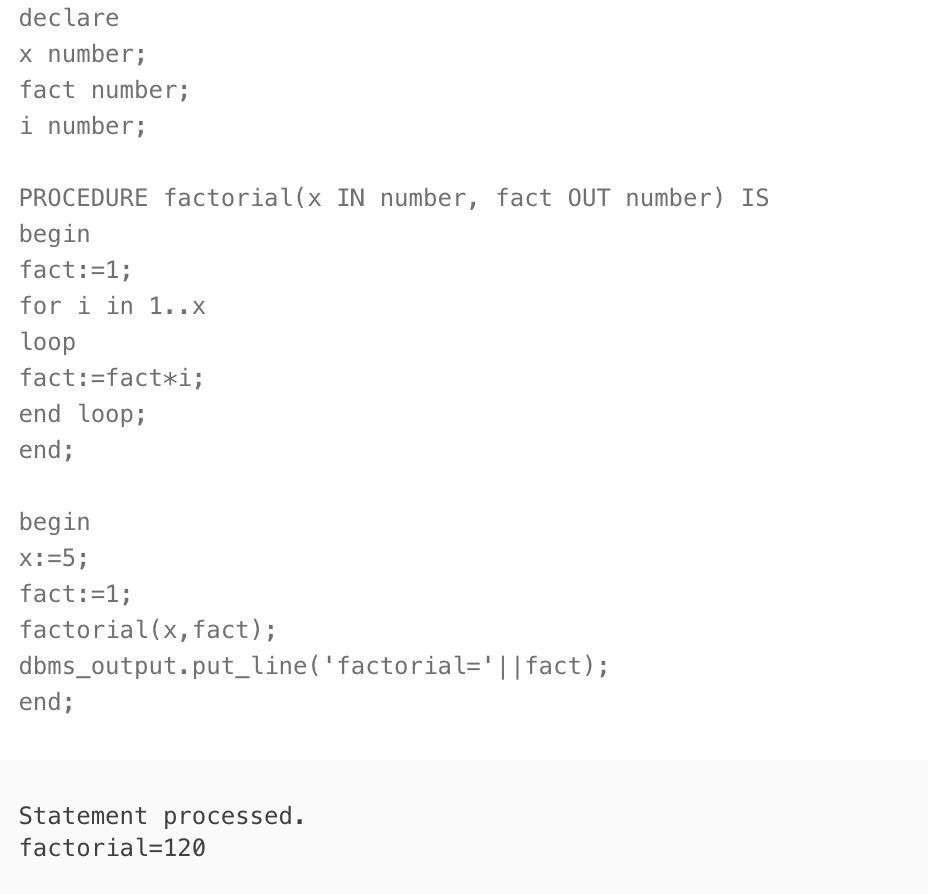
fact:=1; for i in 1..x loop

fact:=fact\*i; end loop; end;

begin x:=5;

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

dbms\_output.put\_line('factorial='||fact); end;



#### 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;

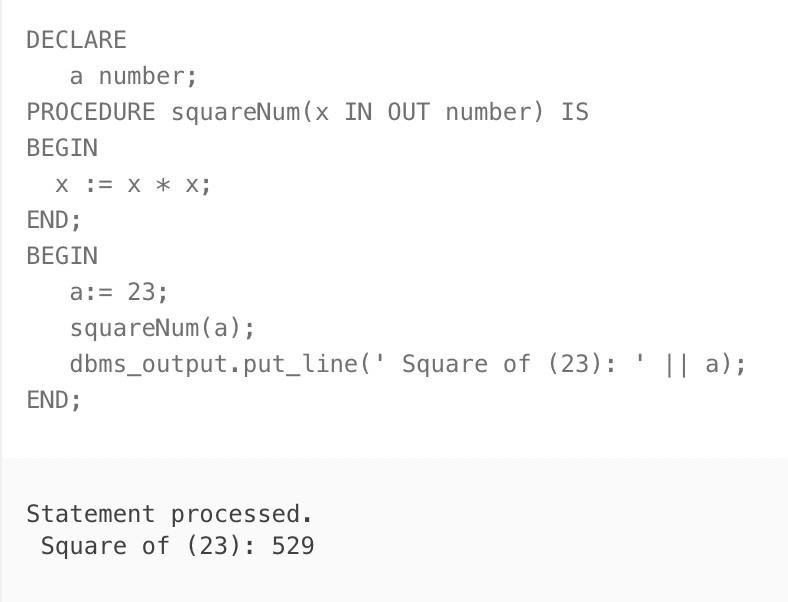
BEGIN

a:= 23;

squareNum(a);

dbms\_output.put\_line(' Square of (23): ' || a);

END;



#### 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;

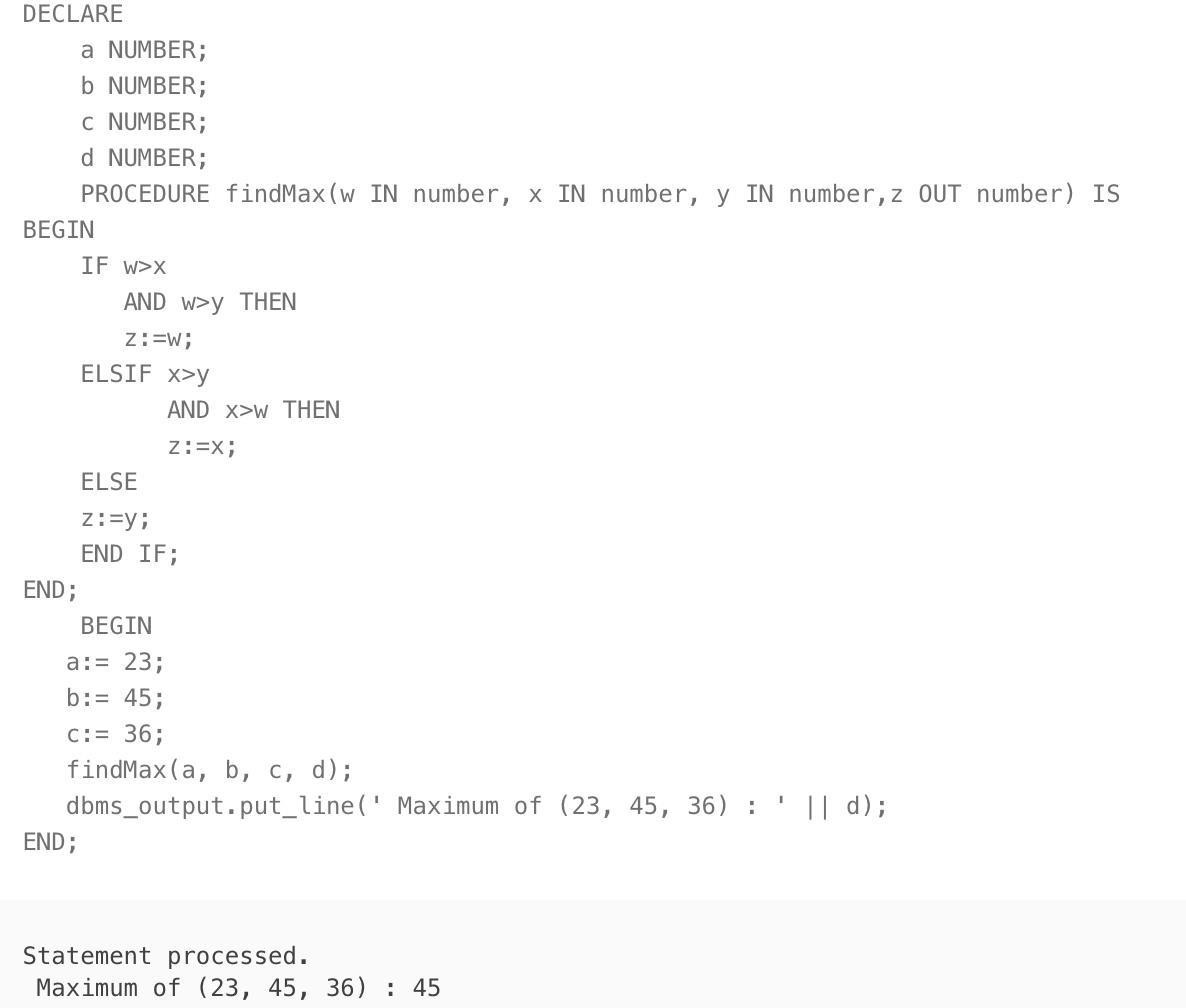
BEGIN a:= 23;

b:= 45;

c:= 36;

findMax(a, b, c, d);

dbms\_output.put\_line(' Maximum of (23, 45, 36) : ' || d); END;



#### 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

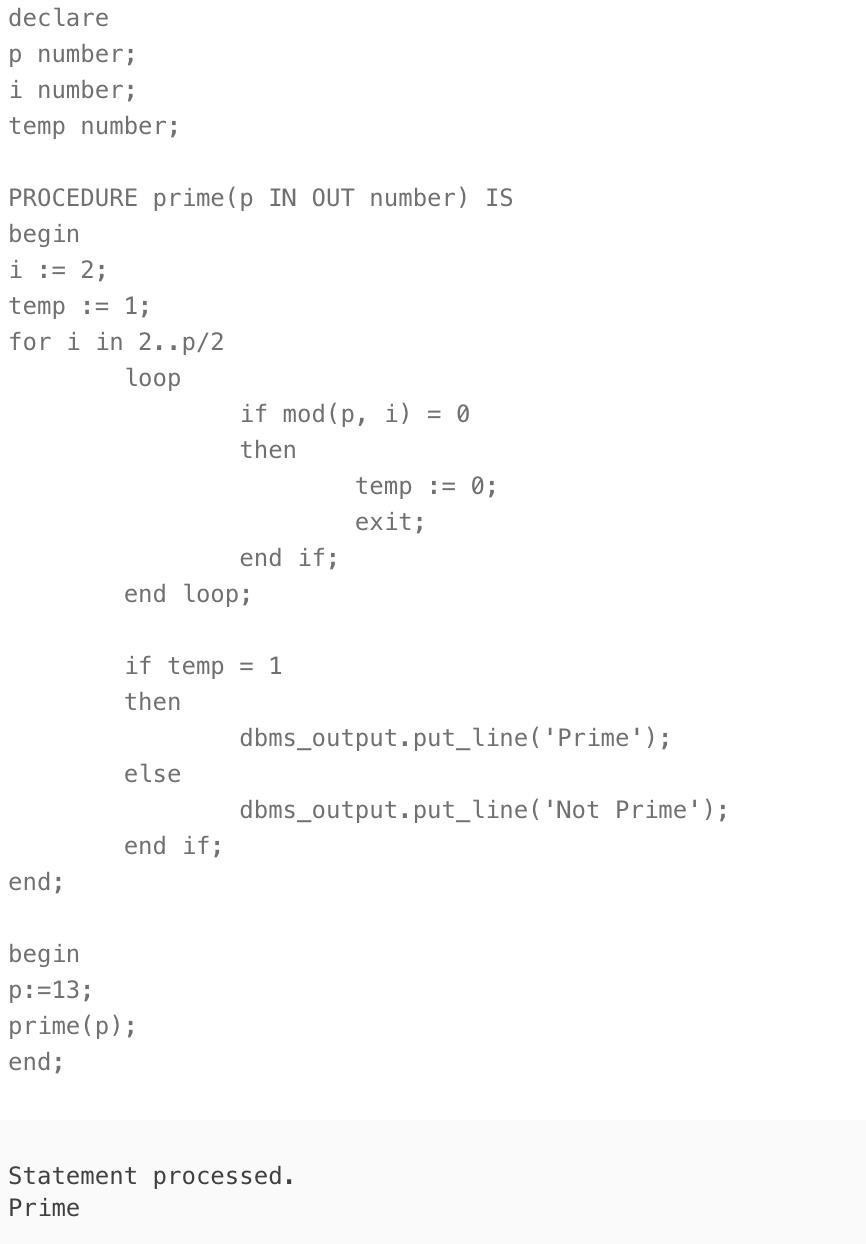
dbms\_output.put\_line('Prime'); else

dbms\_output.put\_line('Not Prime'); end if;

end;

begin p:=13;

prime(p); end;



#### 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

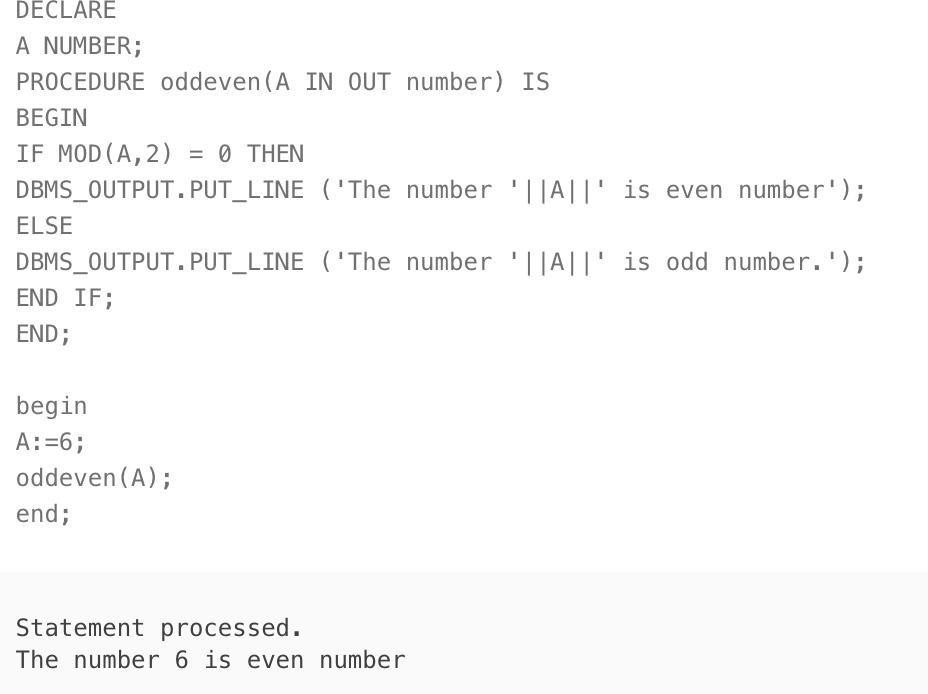
DBMS\_OUTPUT.PUT\_LINE ('The number '||A||' is even number'); ELSE

DBMS\_OUTPUT.PUT\_LINE ('The number '||A||' is odd number.'); END IF;

END;

begin A:=6;

oddeven(A); end;



## Experiment – 10

#### 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

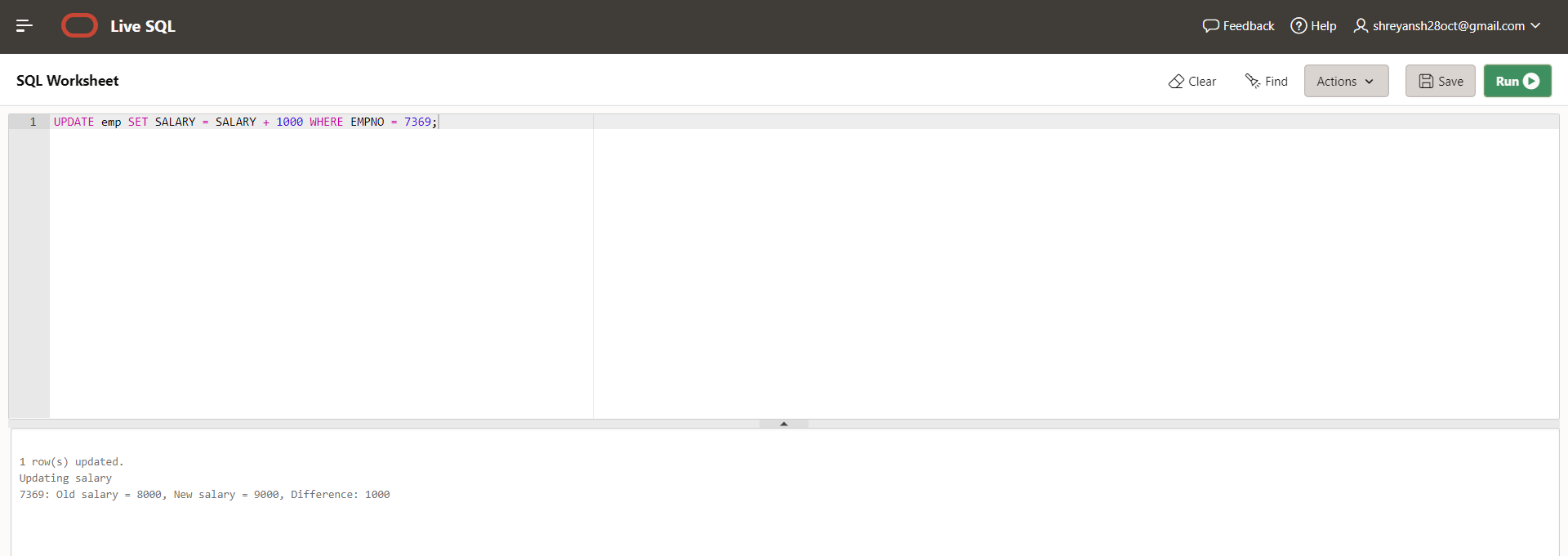
sal\_diff NUMBER;

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;



#### 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');

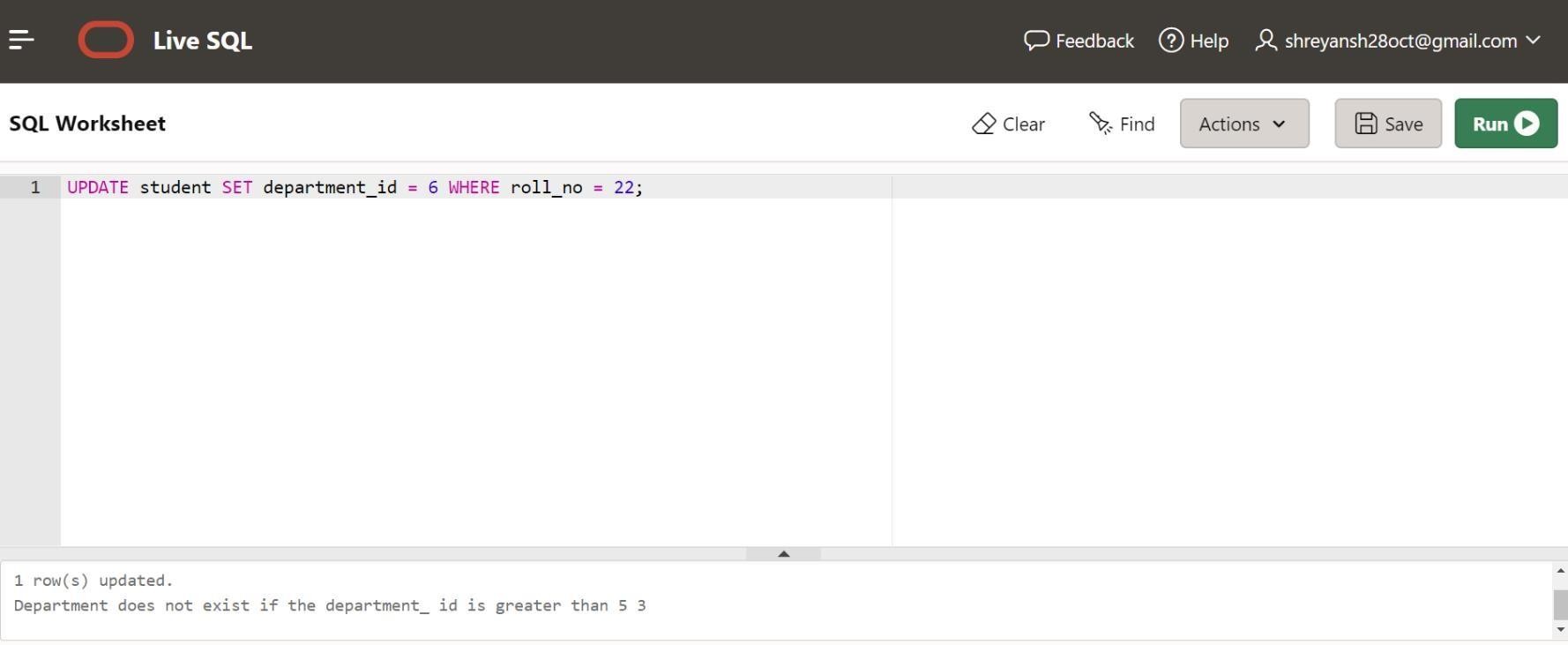
CREATE OR REPLACE TRIGGER display\_department\_changesBEFORE DELETE OR INSERT OR UPDATE ON student

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;



#### 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;



#### 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;

