**Assignment 2**

Q1-

create table book( mcode varchar(5),

mname varchar(12),

bcode varchar(5),

idate date,rdate date,

cost int(5), cname varchar(10));

**Table created.**

Q2-

insert into book values('m001','priyanshu',130,'2023-05-01','2023-09-02',70,'bio');

1 row(s) inserted.

insert into book values('m041','amit',111,'2023-02-01','2023-09-02',60,'computer');

1 row(s) inserted.

insert into book values('m021','abir',222,'2023-05-06','2023-09-02',130,'chemistry');

1 row(s) inserted.

insert into book values('m111','anant',320,'2023-06-30','2023-09-02',170,'maths');

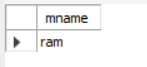
1 row(s) inserted.

insert into book values('n701','ram',450,'2023-02-24','2023-09-02',270,'science');

1 row(s) inserted.

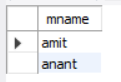
Q3-

select mname from book where mname like ‘r%’;



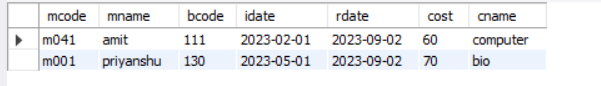
Q4-

select mname from book where mname like ‘a%t’;



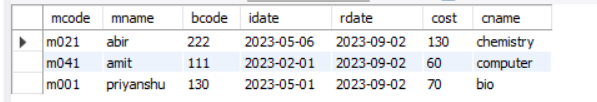
Q5-

select \* from book where cname = 'computer' or cname = 'bio';



**2 rows selected.**

Q6-  
 select \*from book where cost>50 and cost<150;



**3 rows selected.**

Q7-

UPDATE book

SET idate = DATE\_ADD(idate, INTERVAL 2 MONTH)

WHERE mcode = 'm001';

**1 row(s) affected**

Q8-

 select \* from book where cost=(select min(cost) from book);

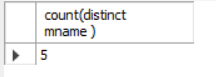
Q9-

select \* from book where cost=(select max(cost) from book);



Q10-

select count(distinct mname ) from book;



Q11-

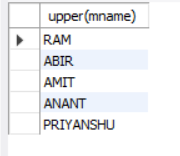
select avg(cost) from book

where cname='computer';



Q12-

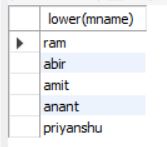
select upper(mname) from book;



**5 rows selected.**

Q13-

select lower(mname) from book;

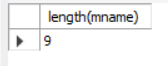
****

**5 rows selected.**

Q14-

select length(mname) from book

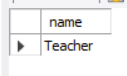
where mcode='m001';



**Q15-Using Dual table-**

Q15.1-

select concat(upper(substr('teacher',1,1)),lower(substr('teacher',2,6))) as name from dual;



Q15.2-

select substr('bombaydelhipune',7,5) as city from dual;



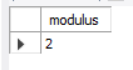
Q15.3-

select abs(-25) as num from dual;



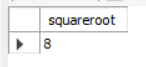
Q15.4-

select mod(20,3)as modulus from dual;



Q15.5-

select sqrt(64) as squareroot from dual;



**Assignment 3 IINTEGRITY CONSTRAINTS**

Q1-

CREATE TABLE PROJECT(

PID int(6) PRIMARY KEY,

PSDATE DATE NOT NULL,

PEDATE DATE NOT NULL,

BAMT int(5) CHECK (BAMT<=9000),

MSTAFF int NOT NULL

);

**Table created.**

Q2-

CREATE TABLE ASSIGN

(

PID int(10),

ENO int(10) PRIMARY KEY, ENAME VARCHAR(255),

CONSTRAINT FK\_PID FOREIGN KEY(PID) REFERENCES PROJECT(PID),

CONSTRAINT CHK\_UPPERCASE CHECK (ENAME=UPPER(ENAME))

);

**Table created.**

Q3-

CREATE TABLE EMP

(

ENO INT,

ENAME VARCHAR(255),

ADDS VARCHAR(255) NOT NULL,

TNO VARCHAR(10) NOT NULL,

CONSTRAINT FK\_ENO\_EMP FOREIGN KEY(ENO) REFERENCES ASSIGN(ENO),

CONSTRAINT PK\_EMP PRIMARY KEY(ENO)

);

**Table created.**

Q4-

INSERT INTO PROJECT VALUES(1, '2024-01-12', '2024-04-22', 4500, 10);

1 row(s) inserted.

INSERT INTO PROJECT VALUES (2, '2024-02-05', '2024-06-05', 5500, 7);

1 row(s) inserted.

INSERT INTO PROJECT VALUES (3, '2022-03-12', '2023-05-18', 6500, 4);

1 row(s) inserted.

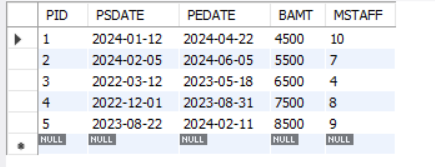
INSERT INTO PROJECT VALUES (4, '2022-12-01', '2023-08-31', 7500, 8);

1 row(s) inserted.

INSERT INTO PROJECT VALUES (5, '2023-08-22', '2024-02-11', 8500, 9);

1 row(s) inserted.

SELECT \* FROM PROJECT;



INSERT INTO ASSIGN VALUES (1, 105, 'ARJUN');

1 row(s) inserted.

INSERT INTO ASSIGN VALUES (2, 107, 'SHREYA');

1 row(s) inserted.

INSERT INTO ASSIGN VALUES (3, 111, 'KRISHNA');

1 row(s) inserted.

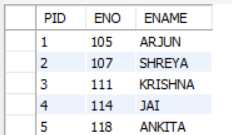
INSERT INTO ASSIGN VALUES (4, 114, 'JAI');

1 row(s) inserted.

INSERT INTO ASSIGN VALUES (5, 118, 'ANKITA');

1 row(s) inserted.

SELECT \* FROM ASSIGN;



INSERT INTO EMP VALUES(105, 'ARJUN', 'NOIDA', '8897965432');

1 row(s) inserted.

INSERT INTO EMP VALUES(107, 'SHREYA', 'GURUGRAM', '7897453671');

1 row(s) inserted.

INSERT INTO EMP VALUES(111, 'KRISHNA', 'LUCKNOW', '7979776562');

1 row(s) inserted.

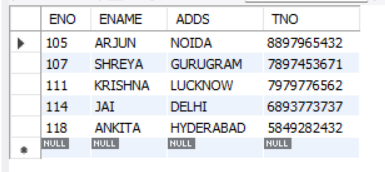
INSERT INTO EMP VALUES(114, 'JAI', 'DELHI', '6893773737');

1 row(s) inserted.

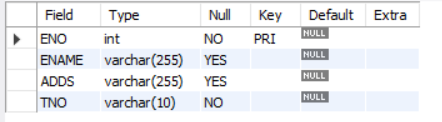
INSERT INTO EMP VALUES(118, 'ANKITA', 'HYDERABAD', '5849282432');

1 row(s) inserted.

SELECT\*FROM EMP;



Q5- ALTER TABLE EMP MODIFY ADDS VARCHAR(255) NULL;



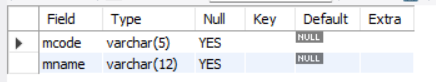
**Assignment 4 SQL JOINS**

Q1 (A)- CREATE TABLE Member AS

SELECT mcode,mname FROM book;

**Table created.**

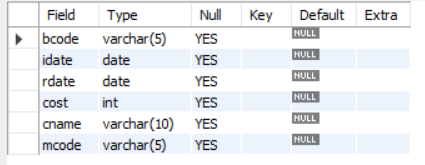
DESC Member;



Q1 (B)- CREATE TABLE Issue AS

SELECT bcode,idate,rdate,cost,cname,mcode FROM book;

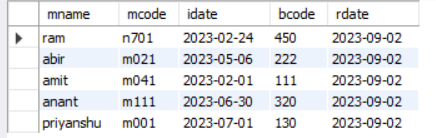
**Table created.**

**** DESC Issue;

Q2 SELECT Member.mname, Member.mcode, Issue.idate, Issue.bcode,Issue.rdate

FROM Member, Issue

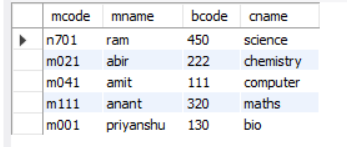
WHERE Member.mcode = Issue.mcode;



Q3 SELECT m.mcode, m.mname, i.bcode, i.cname

FROM member m

LEFT JOIN issue i ON m.mcode = i.mcode;



Q4 SELECT i.bcode, i.idate, i.rdate, i.cname

FROM issue i

where i.cost < 200 AND i.cname = 'computer'

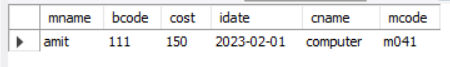


Q5 select member.mname,issue.bcode, issue.cost, issue.idate, issue.cname, member.mcode

from issue

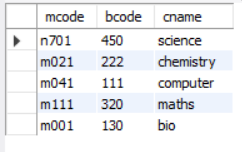
join member on member.mcode=issue.mcode

where member.mname = 'amit' AND issue.cost = 150;



Q6 select member.mcode,issue.bcode,issue.cname from

issue join member on issue.mcode=member.mcode;



**SET OPERATIONS**

create table member\_b(mcode varchar(5),mname varchar(12));

**Table created.**

insert into member\_b values('m041','amit');

**1 row(s) inserted.**

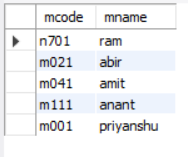
insert into member\_b values('m001','Priyanshu');

**1 row(s) inserted.**

Q1 select distinct \* from member

union

select distinct \* from member\_b;

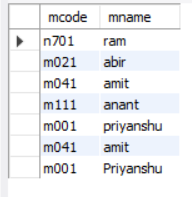


**5 rows selected.**

Q2 select \* from member

union all

select \* from member\_b;

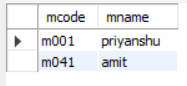


**7 rows selected.**

Q3 select \* from member

intersect

select \* from member\_b;

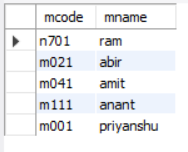


**2 rows selected.**

Q4 select \* from member

union

select \* from member\_b;

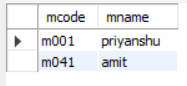


**5 rows selected.**

Q5 select \* from member

intersect

select \* from member\_b;



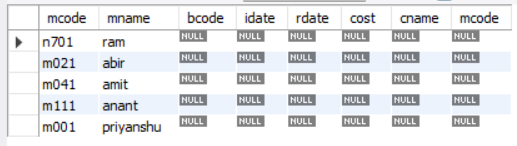
**2 rows selected.**

Q6 select \* from member

left join issue

on member.mcode

where issue.mcode is null;



**Assignment 5 Sub Queries**

Q1 CREATE TABLE CLIENT (

client\_no INT PRIMARY KEY,

client\_name VARCHAR(100),

address VARCHAR(255),

pincode VARCHAR(20),

city VARCHAR(100),

client\_state VARCHAR(100),

bill\_due DECIMAL(10, 2)

);

**Table created.**

INSERT INTO CLIENT VALUES (1, 'Ram', 'jaipur', '302010', 'GZB', 'UP', 500.00);

**1 row(s) inserted.**

INSERT INTO CLIENT VALUES (2, 'Divya', 'Delhi', '110001', 'RAIPUR', 'Bihar', 200.00);

**1 row(s) inserted.**

INSERT INTO CLIENT VALUES(3, 'Mayank', 'kanpur','208001', 'MASSORIE', 'UK', 0.00);

**1 row(s) inserted.**

Q2 CREATE TABLE PRODUCT (

product\_no INT PRIMARY KEY,

description VARCHAR(255),

profit\_percent DECIMAL(5, 2),

unit\_measures VARCHAR(50),

quantity INT,

re\_order\_level INT,

selling\_price DECIMAL(10, 2),

cost\_price DECIMAL(10, 2)

);

**Table created.**

INSERT INTO PRODUCT VALUES (1, 'Battery', 20.00, 'Piece', 100, 20, 50.00, 40.00);

**1 row(s) inserted.**

INSERT INTO PRODUCT VALUES (2, 'tyre', 15.00, 'Piece', 50, 10, 75.00, 60.00);

**1 row(s) inserted.**

INSERT INTO PRODUCT VALUES (3, 'Soap', 25.00, 'Piece', 200, 30, 30.00, 20.00);

**1 row(s) inserted.**

Q3 CREATE TABLE SALES\_ORDER\_DETAIL (

order\_no INT,

product\_no INT,

quantity\_ordered INT,

quantity\_dispatch INT,

product\_rate DECIMAL(10, 2),

PRIMARY KEY (order\_no, product\_no),

FOREIGN KEY (order\_no) REFERENCES SALES(order\_no),

FOREIGN KEY (product\_no) REFERENCES PRODUCT(product\_no)

);

**Table created.**

INSERT INTO SALES\_ORDER\_DETAIL VALUES (100, 1, 10, 10, 50.00);

**1 row(s) inserted.**

INSERT INTO SALES\_ORDER\_DETAIL VALUES (100, 2, 5, 5, 75.00);

**1 row(s) inserted.**

INSERT INTO SALES\_ORDER\_DETAIL VALUES (101, 3, 20, 20, 30.00);

**1 row(s) inserted.**

INSERT INTO SALES\_ORDER\_DETAIL VALUES (102, 1, 15, 15, 50.00);

**1 row(s) inserted.**

Q4 CREATE TABLE SALES (

order\_no INT PRIMARY KEY,

client\_no INT,

delivery\_address VARCHAR(255),

salesman\_no INT,

delivery\_type VARCHAR(50),

delivery\_date DATE,

FOREIGN KEY (client\_no) REFERENCES CLIENT(client\_no),

FOREIGN KEY (salesman\_no) REFERENCES SALESMAN(salesman\_no)

);

**Table created.**

INSERT INTO SALES VALUES (100, 1, 'Ghaziabad', 1, 'Offline','2024-03-15');

**1 row(s) inserted.**

INSERT INTO SALES VALUES (101, 2, 'Lucknow', 2, 'Online','2024-05-10');

**1 row(s) inserted.**

INSERT INTO SALES VALUES (102, 3, 'Meerut', 3, 'Online', '2024-01-05');

**1 row(s) inserted.**

Q5 CREATE TABLE SALESMAN (

salesman\_no INT PRIMARY KEY,

salesman\_name VARCHAR(100),

address VARCHAR(255),

city VARCHAR(100),

state VARCHAR(100),

sales\_amount DECIMAL(10, 2),

target\_to\_get DECIMAL(10, 2),

YTD\_sales DECIMAL(10, 2),

remark VARCHAR(255)

);

**Table created.**

INSERT INTO SALESMAN VALUES (1, 'Rahul', 'Meerut', 'Meerut', 'up', 500.00, 1000.00, 200.00, 'Fresher');

**1 row(s) inserted.**

INSERT INTO SALESMAN VALUES (2, 'Smith', 'Seemapur', 'Seemapur', 'Manipur', 400.00, 800.00, 1800.00, 'Experienced');

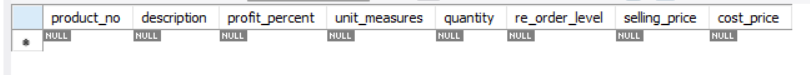
**1 row(s) inserted.**

INSERT INTO SALESMAN VALUES (3, 'Karan', 'Ramgarh', 'Ramgarh', 'Kerala', 600.00, 1200.00, 250.00, 'Senior');

**1 row(s) inserted.**

Q6 select \* from product

where product\_no not in (select distinct product\_no from sales\_order\_detail);



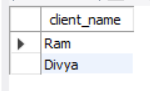
Q7 SELECT client\_name, address FROM CLIENT

where client\_no IN(select client\_no from sales WHERE order\_no = 101);



Q8 SELECT client\_name FROM CLIENT where client\_no IN

(select client\_no from sales where order\_no<=101 AND delivery\_date<'2024-06-20');



**Assignment 5 SQL Views**

Q1 CREATE view view1 AS

SELECT PID,ENO,ENAME,atype FROM assign;

**View created.**

Q2 CREATE view view2 AS

SELECT assign.pid,pedate,psdate,mstaff,eno,ename,atype

FROM assign

Join project where assign.pid=project.pid;

**View created.**

Q3 insert into view1 values(12,201,’jai’,’B3’);

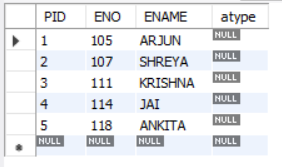
**1 row(s) inserted.**

insert into view1 values(14,121,’Abhi’,’A2’);

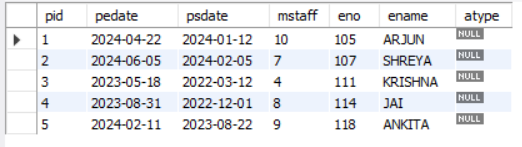
**1 row(s) inserted.**

insert into view1 values(21,231,’Tiya’,’C1’);

**1 row(s) inserted.**

 select \* from assign;

Q4 select \* from view2;



Q5 Drop view view1;

**View Dropped.**

**Assignment 6 PL/SQL**

**Q.1**

DECLARE

num1 NUMBER;

num2 NUMBER;

result NUMBER;

BEGIN

num1:=7;

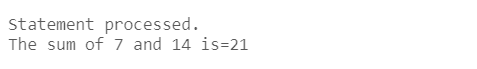
num2:=14;

result := num1 + num2;

DBMS\_OUTPUT.PUT\_LINE('The sum of ' || num1 || ' and ' || num2 || ' is=' || result);

END;

**OUTPUT:**



**Q.2**

DECLARE

quantity NUMBER;

rate NUMBER(10,2);

amount NUMBER(10,2);

BEGIN

quantity :=10;

rate := 25;

DBMS\_OUTPUT.PUT\_LINE('Quantity:-'||quantity);

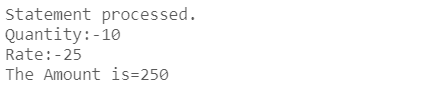
DBMS\_OUTPUT.PUT\_LINE('Rate:-'||rate);

amount := quantity\*rate;

DBMS\_OUTPUT.PUT\_LINE('The Amount is='|| amount );

END;

**OUTPUT:**

****

**Q.3**

SELECT \* FROM employee;

DECLARE

v\_empno number(10);

v\_salary number(10);

totalsalary number(10);

BEGIN

v\_empno:=101;

SELECT SALARY INTO v\_salary FROM employee

WHERE EMPNO=v\_empno;

totalsalary:=v\_salary\*12;

dbms\_output.put\_line('monthly salary='||v\_salary);

dbms\_output.put\_line('total salary='||totalsalary);

EXCEPTION

WHEN no\_data\_found THEN

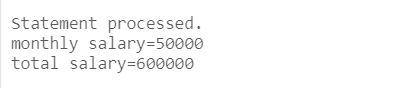
dbms\_output.put\_line('no employee found with empno='||v\_empno);

WHEN others THEN

dbms\_output.put\_line('An error occured');

END;

**OUTPUT:**

****

**Q.4**

create table employees(

empno number(10) unique ,

salary number(10)

);

insert into employees values(1002,2800);

insert into employees values(1003,3500);

insert into employees values(1004,5000);

insert into employees values(1012,2500);

insert into employees values(1013,11000);

DECLARE

v\_empno varchar(10);

v\_salary NUMBER(10);

grade char(1);

BEGIN

v\_empno:=1004;

SELECT salary INTO v\_salary FROM employees

where v\_empno=empno;

IF v\_salary<=3000 THEN

grade := 'C';

ELSIF v\_salary> 3000 AND v\_salary<= 4500 THEN

grade := 'B';

ELSE

grade := 'A';

END IF;

DBMS\_OUTPUT.PUT\_LINE('Salary='||v\_salary||CHR(10)||'Grade:'||grade);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

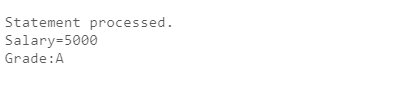
DBMS\_OUTPUT.PUT\_LINE('NO Employee found with empno'||v\_empno);

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error Occurred');

END;

**OUTPUT:**

****

**Q.5**

CREATE table area(

radius number(5,2),

area number(5,2)

);

DECLARE

r number(5,2);

res number(5,2);

BEGIN

FOR r IN 3..7 LOOP

res:=3.14159\*r\*r;

INSERT into area values(r,res);

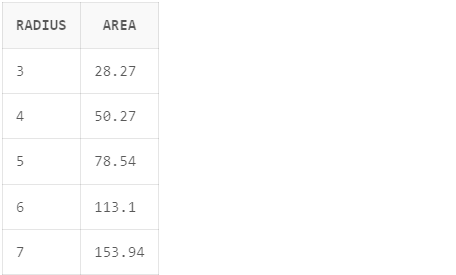
END LOOP;

COMMIT;

END;

SELECT \* from area;

**OUTPUT:**



**Assignment 7 CURSOR**

**Q1** create table Employee(

Emp\_Code Varchar2(5) primary key,

Ename Varchar2 (12),

Deptno Number (5),

Job Varchar2(10),

Salary Number(11,2));



**Q2** create table emp\_raise(

Emp\_Code Varchar2(5),

Raise\_Date Date,

Raise\_Amt Number(11,2),

FOREIGN KEY (Emp\_code) REFERENCES Employee(Emp\_code)

);



**Q3** declare

ecod employee.Emp\_code%type;

nam employee.ename%type;

dno employee.deptno%type;

sal employee.salary%type;

cursor c1 is

select Emp\_code,ename,deptno,salary from employee

where deptno=10;

Begin

open c1;

Loop

Fetch c1 into ecod,nam,dno,sal;

Exit when c1%notfound;

dbms\_output.put\_line( 'employee no -'|| ecod||' department no-'||dno||' name- '||nam||' salary-'||sal);

End loop;

Close c1;

End;



**Q4** DECLARE

Str\_emp\_code employee.emp\_code%type;

Num\_salary employee.salary%type;

CURSOR c\_cum IS

SELECT emp\_code, salary FROM employee WHERE deptno = 20;

BEGIN

OPEN c\_cum;

LOOP

FETCH c\_cum INTO str\_emp\_code, num\_salary;

IF c\_cum%FOUND THEN

UPDATE employee SET salary = num\_salary + (num\_salary \* .05)

WHERE emp\_code = str\_emp\_code ;

INSERT INTO emp\_raise VALUES(str\_emp\_code, sysdate, num\_salary \* 0.05);

ELSE exit;

END IF;

END LOOP;

COMMIT;

CLOSE c\_cum;

END;



**Q5** create table AcctMast (

accno Varchar2(6) check (accno LIKE 'C%') Primary Key,

Name Varchar2(12) Not Null,

Balance Number(10),

type Char(2)

);



**Q6** create table Acctran (

transno number(6) primary key,

accno Varchar2(6),

Tran\_date Date,

type Char(2),

Deb\_cre Char(1) Check ( deb\_cre='C' or deb\_cre='D' ),

Amount Number(10,2),

Proc Char(1) Check (proc='Y' or proc= 'N'),

FOREIGN KEY (accno) REFERENCES AcctMast(accno)

);



**Q7** declare cursor c\_acc is select accno,deb\_cre,amount from acctran where proc='N';

acno acctran.accno%type;

dc acctran.deb\_cre%type;

amt acctran.amount%type;

begin

open c\_acc;

loop

fetch c\_acc into acno,dc,amt;

if dc='d' then

update acctmast set balance=(balance-amt) where accno=acno;

else

update acctmast set balance=(balance+amt) where accno=acno;

end if;

update acctran set proc='y' where accno=acno;

exit when c\_acc%notfound;

end loop;

close c\_acc;

end;



**Assignment 8 TRIGGERS**

**Q1** CREATE TABLE employee (

employee\_id NUMBER,

employee\_name VARCHAR2(100),

salary NUMBER

);

Image

**Q2**  CREATE TABLE temp\_employee (

operation VARCHAR2(10),

employee\_id NUMBER,

employee\_name VARCHAR2(100),

salary NUMBER

);

Image

**Q3** CREATE OR REPLACE TRIGGER check\_salary\_update

BEFORE UPDATE ON employee

FOR EACH ROW

BEGIN

IF :OLD.salary > :NEW.salary THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Updation not possible. Old salary is greater than new salary.');

END IF;

END;

Image

**Q4** CREATE OR REPLACE TRIGGER copy\_deleted\_employee

AFTER DELETE ON employee

FOR EACH ROW

BEGIN

INSERT INTO temp\_employee (operation, employee\_id, employee\_name, salary)

VALUES ('delete', :OLD.employee\_id, :OLD.employee\_name, :OLD.salary);

END;

Image

**Q5** CREATE OR REPLACE TRIGGER capture\_modified\_employee

AFTER UPDATE OR DELETE ON employee

FOR EACH ROW

DECLARE

operation VARCHAR2(10);

BEGIN

IF UPDATING THEN

operation := 'update';

ELSIF DELETING THEN

operation := 'delete';

ELSE

operation := 'unknown';

END IF;

INSERT INTO temp\_employee (operation, employee\_id, employee\_name, salary)

VALUES (operation, :OLD.employee\_id, :OLD.employee\_name, :OLD.salary);

END;

Image

**Q6** INSERT INTO employee (employee\_id, employee\_name, salary)

VALUES (1, 'NEHA TYAGI ', 5000);



**Q7** UPDATE employee

SET salary = 6000

WHERE employee\_id = 1;

Image

**Q8** DELETE FROM employee WHERE employee\_id = 1;

Image

**Q9** SELECT \* FROM temp\_employee;

|  |  |  |  |
| --- | --- | --- | --- |
| **OPERATION** | **EMPLOYEE\_ID** | **EMPLOYEE\_NAME** | **SALARY** |
| update | *1* | *Arjun Singh* | *5000* |
| *delete* | *1* | *Arjun Singh* | *6000* |
| *delete* | *1* | *Arjun Singh* | *6000* |

**Assignment 9 PROCEDURES**

**Q1** CREATE TABLE employee (

eno NUMBER,

ename VARCHAR2(50),

salary NUMBER,

dept VARCHAR2(50),

hiredate DATE

);



**Q2** INSERT INTO employee (eno, ename, salary, dept, hiredate)

VALUES (1, 'Jai sethi', 50000, 'HR', TO\_DATE('2022-01-01', 'YYYY-MM-DD'));

**1 row inserted**

INSERT INTO employee (eno, ename, salary, dept, hiredate)

VALUES (2, 'Amerita das', 85000, 'software developer', TO\_DATE('2020-06-01', 'YYYY-MM-DD'));

**1 row inserted**

INSERT INTO employee (eno, ename, salary, dept, hiredate)

VALUES (3, 'rahul singh', 150000, 'AI engineer', TO\_DATE('2022-11-16', 'YYYY-MM-DD'));

**1 row inserted**

INSERT INTO employee (eno, ename, salary, dept, hiredate)

VALUES (4, 'jaspreet kaur', 25000, 'Data operator', TO\_DATE('2018-03-11', 'YYYY-MM-DD'));

**1 row inserted**

INSERT INTO employee (eno, ename, salary, dept, hiredate)

VALUES (5, 'Raj', 5000, 'security', TO\_DATE('2022-04-03', 'YYYY-MM-DD'));

**1 row inserted**

**Q3** CREATE OR REPLACE PROCEDURE increase\_salary(

p\_ename IN employee.ename%TYPE,

p\_increment IN NUMBER)

IS

BEGIN

UPDATE employee

SET salary = salary + p\_increment

WHERE ename = p\_ename;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Salary for employee ' || p\_ename || ' increased by ' || p\_increment || ' Rs.');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Employee ' || p\_ename || ' not found.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred while updating the salary.');

END;



**Q4** BEGIN

increase\_salary('Raj', 1000);

END;

**Statement Processed.**

**Salary for employee Raj increased by 1000 rs.**

**Q5.** CREATE OR REPLACE PROCEDURE give\_bonus

IS

BEGIN

FOR emp IN (SELECT \* FROM employee)

LOOP

DECLARE

v\_total\_months NUMBER;

v\_bonus\_amount NUMBER;

BEGIN

SELECT MONTHS\_BETWEEN(SYSDATE, emp.hiredate) INTO v\_total\_months FROM DUAL;

IF v\_total\_months > 5 THEN

v\_bonus\_amount := emp.salary \* 0.1 + 500;

ELSE

v\_bonus\_amount := emp.salary \* 0.1;

END IF;

UPDATE employee

SET salary = emp.salary + v\_bonus\_amount

WHERE eno = emp.eno;

END;

END LOOP;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Bonus given to all eligible employees.');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred while giving the bonus.');

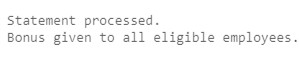
END;



**Q6** BEGIN

give\_bonus;

END;



**Assignment 10 FUNCTIONS**

**Q1.**

CREATE OR REPLACE FUNCTION cal\_bonus(p\_empno IN NUMBER, p\_jdate IN DATE)

RETURN NUMBER IS

v\_salary NUMBER(8,2);

BEGIN

-- Assume we have a SELECT statement here to get the salary of the employee

-- For this example, let's assume the salary is 5000

v\_salary := 5000;

-- Calculate the bonus

-- For this example, let's assume the bonus is 10% of the salary

RETURN v\_salary \* 0.1;

END cal\_bonus;

**Function created.**

**Q2.**

DECLARE

v\_bonus NUMBER(8,2);

BEGIN

-- Call the cal\_bonus function

v\_bonus := cal\_bonus(12345, TO\_DATE('2022-01-01', 'YYYY-MM-DD'));

-- Display the bonus

DBMS\_OUTPUT.PUT\_LINE('The bonus is: ' || v\_bonus);

END;

**Statement processed.**

**The bonus is: 500**

**Q3.**

CREATE OR REPLACE FUNCTION f\_chkaccno(p\_accno IN NUMBER)

RETURN NUMBER IS

v\_count NUMBER(1);

BEGIN

-- Assume we have a SELECT statement here to count the number of records for the account no

-- For this example, let's assume the SELECT statement is

-- SELECT COUNT(\*) INTO v\_count FROM accounts WHERE accno = p\_accno

-- Check if the account no exists

IF v\_count > 0 THEN

RETURN 1;

ELSE

RETURN 0;

END IF;

END f\_chkaccno;

**Function created.**

**Q4.**

DECLARE

v\_result NUMBER(1);

BEGIN

-- Call the f\_chkaccno function

v\_result := f\_chkaccno(123456);

-- Display the result

IF v\_result = 1 THEN

DBMS\_OUTPUT.PUT\_LINE('The account no exists');

ELSE

DBMS\_OUTPUT.PUT\_LINE('The account no does not exist');

END IF;

END;

**Statement processed.**

**The account no does not exist**

**Q5.**

CREATE OR REPLACE PROCEDURE p\_chkjoiningdate(p\_empno IN NUMBER) IS

v\_jdate DATE;

BEGIN

-- Assume we have a SELECT statement here to get the joining date of the employee

-- For this example, let's assume the joining date is 2022-01-01

v\_jdate := TO\_DATE('2022-01-01', 'YYYY-MM-DD');

-- Check the joining month

IF EXTRACT(MONTH FROM v\_jdate) > 6 THEN

DBMS\_OUTPUT.PUT\_LINE('Eligible for bonus');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Not Eligible for bonus');

END IF;

END p\_chkjoiningdate;

**Procedure created.**

**Q6.**

CREATE OR REPLACE PROCEDURE p\_dispsalary(p\_empno IN NUMBER) IS

v\_salary NUMBER(8,2);

BEGIN

-- Assume we have a SELECT statement here to get the salary of the employee

-- For this example, let's assume the salary is 5000

v\_salary := 5000;

-- Display the total salary

DBMS\_OUTPUT.PUT\_LINE('The total salary of the employee is: ' || v\_salary);

END p\_dispsalary;

**Procedure created.**