4. Create a row level trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old & new Salary.

CUSTOMERS (ID, NAME, AGE, ADDRESS, SALARY)

```
SET SERVEROUTPUT ON;

CREATE TABLE CUSTOMER (
 ID INT PRIMARY KEY,
 NAME VARCHAR(10),
 AGE INT,
 ADDRESS VARCHAR(10),
 SALARY number(10, 2)
);

INSERT INTO CUSTOMER values(1,'Ramesh',25,'Mysore',200000);
INSERT INTO CUSTOMER values(2,'Komal',35,'Bangalore',800000);
INSERT INTO CUSTOMER values(3,'Mala',45,'Mangalore',56000);

CREATE OR REPLACE TRIGGER sal_difference_trigger
BEFORE INSERT OR UPDATE OR DELETE ON CUSTOMER
FOR EACH ROW
DECLARE
old_salary NUMBER;
new_salary NUMBER;
BEGIN
  IF INSERTING OR UPDATING THEN
old_salary := NVL(:OLD.SALARY, 0);
new_salary := NVL(:NEW.SALARY, 0);
    DBMS_OUTPUT.PUT_LINE('Salary difference: ' || (new_salary - old_salary));
  ELSIF DELETING THEN
old_salary := NVL(:OLD.SALARY, 0);
    DBMS_OUTPUT.PUT_LINE('Salary before deletion: ' || old_salary);
  END IF;
END;
```

RESULT:

```
SQL> SET SERVEROUTPUT ON;
SQL> CREATE TABLE CUSTOMER (
  2      ID INT PRIMARY KEY,
  3      NAME VARCHAR(10),
  4      AGE INT,
  5      ADDRESS VARCHAR(10),
  6      SALARY number(10, 2)
  7  );

Table created.

SQL> INSERT INTO CUSTOMER values(1,'Ramesh',25,'Mysore',200000);

1 row created.

SQL> INSERT INTO CUSTOMER values(2,'Komal',35,'Bangalore',800000);

1 row created.

SQL> INSERT INTO CUSTOMER values(3,'Mala',45,'Mangalore',56000);

1 row created.

SQL>
SQL> select * from customer;

        ID NAME              AGE ADDRESS        SALARY
---------- ---------- ---------- ---------- ----------
         1 Ramesh             25 Mysore         200000
         2 Komal              35 Bangalore      800000
         3 Mala               45 Mangalore       56000
```

```
SQL> CREATE OR REPLACE TRIGGER sal_difference_trigger
  2  BEFORE INSERT OR UPDATE OR DELETE ON CUSTOMER
  3  FOR EACH ROW
  4  DECLARE
  5  old_salary NUMBER;
  6  new_salary NUMBER;
  7  BEGIN
  8       IF INSERTING OR UPDATING THEN
  9  old_salary := NVL(:OLD.SALARY, 0);
 10  new_salary := NVL(:NEW.SALARY, 0);
 11           DBMS_OUTPUT.PUT_LINE('Salary difference: ' || (new_salary - old_salary));
 12       ELSIF DELETING THEN
 13  old_salary := NVL(:OLD.SALARY, 0);
 14           DBMS_OUTPUT.PUT_LINE('Salary before deletion: ' || old_salary);
 15       END IF;
 16  END;
 17  /

Trigger created.

SQL> select * from customer;

        ID NAME               AGE ADDRESS        SALARY
---------- ---------- ---------- ---------- ----------
         1 Ramesh             25 Mysore         200000
         2 Komal              35 Bangalore      800000
         3 Mala               45 Mangalore       56000

SQL> INSERT INTO CUSTOMER values(6,'Jamal',30,'Mumbai',70000);
Salary difference: 70000

1 row created.

SQL> select * from customer;

        ID NAME               AGE ADDRESS        SALARY
---------- ---------- ---------- ---------- ----------
         1 Ramesh             25 Mysore         200000
         2 Komal              35 Bangalore      800000
         3 Mala               45 Mangalore       56000
         6 Jamal              30 Mumbai          70000

SQL> UPDATE customer SET salary = salary + 5000   WHERE id = 2;
Salary difference: 5000

SQL> select * from customer;

        ID NAME               AGE ADDRESS        SALARY
---------- ---------- ---------- ---------- ----------
         1 Ramesh             25 Mysore         200000
         2 Komal              35 Bangalore      805000
         3 Mala               45 Mangalore       56000
         6 Jamal              30 Mumbai          70000

SQL> delete from customer where id=2;
Salary before deletion: 805000

1 row deleted.
```

5. Create cursor for Employee table & extract the values from the table. Declare the variables,Open the cursor & extract the values from the cursor. Close the cursor. Employee(E_id, E_name, Age, Salary).

```
CREATE TABLE EMPLOYEE5
(
E_ID INT PRIMARY KEY,
E_NAME VARCHAR (15),
AGE INT,
SALARY DECIMAL (10, 2)
);
```

```
INSERT INTO EMPLOYEE5 VALUES (1, 'Ramesh', 32, 2000.00);
INSERT INTO EMPLOYEE5 VALUES (2, 'Khilan', 25,1500.00);
INSERT INTO EMPLOYEE5 VALUES (3, 'Kaushik', 23,2000.00);
INSERT INTO EMPLOYEE5 VALUES (4, 'Chaitali', 25,6500.00);
```

```
DECLARE
 E_id Employee5.E_id%TYPE;
 E_name Employee5.E_name%TYPE;
Age Employee5.Age%TYPE;
Salary Employee5.Salary%TYPE;

-- Declare cursor
CURSOR employee_cursor
 SELECT E_id, E_name, Age, Salary
 FROM Employee5;

-- Open the cursor
 BEGIN
 OPEN employee_cursor;
 -- Fetch data from cursor
 LOOP
 FETCH employee_cursor INTO E_id, E_name, Age, Salary;
 EXIT WHEN employee_cursor%NOTFOUND;
```

```
-- Output or use the fetched values
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || E_id || ', Name: ' || E_name || ', Age: ' || Age || ',
Salary: ' || Salary);
END LOOP;

CLOSE employee_cursor;
END;
```

Result:

```
SQL> CREATE TABLE EMPLOYEE5
  2   (
  3  E_ID INT PRIMARY KEY,
  4  E_NAME VARCHAR (15),
  5   AGE INT,
  6   SALARY DECIMAL (10, 2)
  7  );

Table created.

SQL>
SQL> INSERT INTO EMPLOYEE5 VALUES (1, 'Ramesh', 32, 2000.00);

1 row created.

SQL> INSERT INTO EMPLOYEE5 VALUES (2, 'Khilan', 25,1500.00);

1 row created.

SQL> INSERT INTO EMPLOYEE5VALUES (3, 'Kaushik', 23,2000.00);
INSERT INTO EMPLOYEE5VALUES (3, 'Kaushik', 23,2000.00)
                            *
ERROR at line 1:
ORA-00928: missing SELECT keyword


SQL> INSERT INTO EMPLOYEE5 VALUES (4, 'Chaitali', 25,6500.00);

1 row created.

SQL> INSERT INTO EMPLOYEE5 VALUES (3, 'Kaushik', 23,2000.00);

1 row created.
```

```
SQL> select * from employee5;

      E_ID E_NAME                    AGE     SALARY
---------- --------------- ---------- ----------
         1 Ramesh                     32       2000
         2 Khilan                     25       1500
         4 Chaitali                   25       6500
         3 Kaushik                    23       2000
```

```
SQL> DECLARE
  2  E_id Employee5.E_id%TYPE;
  3  E_name Employee5.E_name%TYPE;
  4      Age Employee5.Age%TYPE;
  5      Salary Employee5.Salary%TYPE;
  6
  7  -- Declare cursor
  8  CURSOR employee_cursor IS
  9      SELECT E_id, E_name, Age, Salary
 10      FROM Employee5;
 11
 12  -- Open the cursor
 13  BEGIN
 14      OPEN employee_cursor;
 15
 16      -- Fetch data from cursor
 17      LOOP
 18          FETCH employee_cursor INTO E_id, E_name, Age, Salary;
 19          EXIT WHEN employee_cursor%NOTFOUND;
 20
 21          -- Output or use the fetched values
 22          DBMS_OUTPUT.PUT_LINE('Employee ID: ' || E_id || ', Name: ' || E_name || ', Age: ' || Ag
e || ', Salary: ' || Salary);
 23      END LOOP;
 24
 25      -- Close the cursor
 26      CLOSE employee_cursor;
 27  END;
 28  /
Employee ID: 1, Name: Ramesh, Age: 32, Salary: 2000
Employee ID: 2, Name: Khilan, Age: 25, Salary: 1500
Employee ID: 4, Name: Chaitali, Age: 25, Salary: 6500
Employee ID: 3, Name: Kaushik, Age: 23, Salary: 2000

PL/SQL procedure successfully completed.
```

6. Write a PL/SQL block of code using parameterized Cursor, that will merge the data Available in the newly created table N_RollCall with the data available in the table O_RollCall. If the data in the first table already exist in the second table then that data should be skipped.

```
create table O_RollCall (roll int,name varchar(10));
create table N_RollCall (roll int,name varchar(10));

insert into O_RollCall values(1,'bc');
insert into O_RollCall values(3,'bcd');
insert into O_RollCall values(4,'d');
insert into O_RollCall values(5,'bch');

insert into N_RollCall values(1,'bc');
insert into N_RollCall values(2,'b');
insert into N_RollCall values(5,'bch');

DECLARE
v_count NUMBER;
   CURSOR c_new_rollcall IS
      SELECT roll, name
      FROM N_RollCall;
BEGIN
   FOR new_rec IN c_new_rollcall LOOP
      -- Check if the record already exists in O_RollCall
      SELECT COUNT(*)
      INTO   v_count
      FROM   O_RollCall
WHERE  roll = new_rec. roll;

      -- If record doesn't exist, insert it
      IF v_count = 0 THEN
         INSERT INTO O_RollCall (roll, name)
         VALUES (new_rec. roll, new_rec.name);
         DBMS_OUTPUT.PUT_LINE('Record inserted: ' || new_rec. roll);
```

```
      ELSE
         DBMS_OUTPUT.PUT_LINE('Record skipped: ' || new_rec. roll);
      END IF;
   END LOOP;
   COMMIT;
END;
select * from N_RollCall;
select * from O_RollCall;
```

RESULT:

```
SQL>  select * from O_RollCall;

      ROLL NAME
---------- ----------
         1 bc
         3 bcd
         4 d
         5 bch

SQL> select * from N_RollCall;

      ROLL NAME
---------- ----------
         1 bc
         2 b
         5 bch
```

```
SQL> DECLARE
  2  v_count NUMBER;
  3      CURSOR c_new_rollcall IS
  4          SELECT roll, name
  5          FROM N_RollCall;
  6  BEGIN
  7      FOR new_rec IN c_new_rollcall LOOP
  8          -- Check if the record already exists in O_RollCall
  9          SELECT COUNT(*)
 10          INTO   v_count
 11          FROM   O_RollCall
 12  WHERE  roll = new_rec. roll;
 13
 14          -- If record doesn't exist, insert it
 15          IF v_count = 0 THEN
 16              INSERT INTO O_RollCall (roll, name)
 17              VALUES (new_rec. roll, new_rec.name);
 18              DBMS_OUTPUT.PUT_LINE('Record inserted: ' || new_rec. roll);
 19          ELSE
 20              DBMS_OUTPUT.PUT_LINE('Record skipped: ' || new_rec. roll);
 21          END IF;
 22      END LOOP;
 23      COMMIT;
 24  END;
 25  /
Record skipped: 1
Record inserted: 2
Record skipped: 5

PL/SQL procedure successfully completed.
```

```
SQL> select * from N_RollCall;

      ROLL NAME
---------- ----------
         1 bc
         2 b
         5 bch

SQL> select * from O_RollCall;

      ROLL NAME
---------- ----------
         1 bc
         3 bcd
         4 d
         5 bch
         2 b
```

7. Install an Open Source NoSQL Data base MangoDB & perform basic CRUD(Create, Read, Update & Delete) operations. Execute MangoDB basic Queries using CRUD operations.

https://www.mongodb.com/try/download/community

https://www.mongodb.com/try/download/shell

Set Environmental Path

Mongod –version
Mongosh
Use dbname
db.createCollection("collectionname")
db.sana.insert({ "name": "Alice", "age": 30 })

db.sana.find()
db.sana.update({ "name": "Alice" }, { $set: { "age": 31 } })