1. **CREATION OF TABLE**

SQL> CREATE TABLE students ("student\_id" VARCHAR(15),Name VARCHAR(50),Age INT,Branch VARCHAR(6));

Table created.

SQL> ALTER TABLE students ADD address VARCHAR(100);

Table altered.

SQL> INSERT INTO students ("student\_id", Name, Age, Branch, address) VALUES ('24F45A0501', 'ROOP KUMAR, 18, 'CSE', 'kuppam');

1 row created.

SQL> INSERT INTO students ("student\_id", Name, Age, Branch, address) VALUES ('23F41A05c7', 'PAWAN', 19, 'CSE', 'v kota');

1 row created.

SQL> INSERT INTO students ("student\_id", Name, Age, Branch, address) VALUES ('23F41A0506', 'MADAN', 18, 'ECE', 'CG PALLI);

1 row created.

SQL> INSERT INTO students ("student\_id", Name, Age, Branch, address) VALUES ('23F41A0562', 'tharun', 18, 'HAS', 'temple road');

1 row created.

SQL> SELECT \* from students;

STUDENT\_ID NAME AGE BRANCH ADDRESS

----------------------------------------------------------------------------------------------------------------

23F41A0597 ROOPKUMAR 18 CSE kuppam

23F41A0584 PAWAN 19 CSE v kota

23F41A0505 AKIB 18 ECE palace road

23F41A0562 HANEEF 18 HAS temple road

SQL> UPDATE students SET address='dk.palli kuppam' WHERE "student\_id"='23F41A0597';

1 row updated.

SQL> SELECT \* from students;

STUDENT\_ID NAME AGE BRANCH ADDRESS

----------------------------------------------------------------------------------------------------------------

23F41A0597 NIKHILKUMAR 18 CSE dk.palli kuppam

23F41A0584 DURGA 19 CSE gudupalli

23F41A0505 AKIB 18 ECE palace road

23F41A0562 HANEEF 18 HAS temple road

SQL> ALTER TABLE students drop column address;

Table altered.

SQL> SELECT \* from students;

STUDENT\_ID NAME AGE BRANCH

----------------------------------------------------------------------------------------------------------------

23F41A0597 NIKHILKUMAR 18 CSE

23F41A0584 DURGA 19 CSE

23F41A0505 AKIB 18 ECE

23F41A0562 HANEEF 18 HAS

SQL> drop table students;

Table dropped.

SQL> desc students

ERROR:

ORA-04043: object students does not exist

**2.** **SQL Queries with Subqueries and Set Operations**

SQL> CREATE TABLE Student (roll\_no INT PRIMARY KEY,name VARCHAR(100) NOT NULL,marks INT NOT NULL,rank INT UNIQUE);

Table created.

SQL> INSERT INTO Student (roll\_no, name, marks, rank) VALUES

2 (1, 'Ankit Sharma', 450, 1);

1 row created.

SQL> INSERT INTO Student (roll\_no, name, marks, rank) VALUES (2, 'Pooja Patel', 430, 2);

1 row created.

SQL> INSERT INTO Student (roll\_no, name, marks, rank) VALUES (3, 'Rahul Verma', 420, 3);

1 row created.

SQL> INSERT INTO Student (roll\_no, name, marks, rank) VALUES (4, 'Neha Singh', 410, 4);

1 row created.

SQL> INSERT INTO Student (roll\_no, name, marks, rank) VALUES (5, 'Vikram Yadav', 400, 5);

1 row created.

SQL> SELECT \* FROM student;

ROLL\_NO NAME MARKS RANK

------------------------------------------------------------------------------------

1 Ankit Sharma 450 1

2 Pooja Patel 430 2

3 Rahul Verma 420 3

4 Neha Singh 410 4

5 Vikram Yadav 400 5

SQL> SELECT name, marks FROM Student WHERE marks > ANY (SELECT marks FROM Student WHERE rank >= 3);

NAME MARKS

---------------------------------------------------

Ankit Sharma 450

Pooja Patel 430

Rahul Verma 420

Neha Singh 410

SQL> SELECT name, marks FROM Student WHERE marks > ALL (SELECT marks FROM Student WHERE rank > 4);

NAME MARKS

----------------------------------------------

Ankit Sharma 450

Pooja Patel 430

Rahul Verma 420

Neha Singh 410

SQL> SELECT name FROM Student WHERE rank IN (2, 4);

NAME

--------------------------------------------------------------------------------

Pooja Patel

Neha Singh

SQL> SELECT name FROM Student WHERE NOT EXISTS (SELECT 1 FROM Student WHERE rank IS NULL);

NAME

--------------------------------------------------------------------------------

Ankit Sharma

Pooja Patel

Rahul Verma

Neha Singh

Vikram Yadav

SQL> SELECT name FROM Student WHERE rank IN (1, 2) UNION SELECT name FROM Student WHERE rank IN (4, 5);

NAME

--------------------------------------------------------------------------------

Ankit Sharma

Pooja Patel

Neha Singh

Vikram Yadav

SQL> SELECT name FROM Student WHERE rank <= 3 INTERSECT SELECT name FROM Student WHERE marks > 420;

NAME

--------------------------------------------------------------------------------

Ankit Sharma

Pooja Patel

SQL> SELECT 'Exists' AS Result FROM dual WHERE EXISTS (SELECT 1 FROM Student WHERE marks > 450);

no rows selected

**3.SQL QUERIES USING AGGREGATE FUNCTION AND VIEWS**

SQL> CREATE TABLE Sales (ProductID INT,ProductName VARCHAR(50),Quantity INT,Price DECIMAL(10, 2),Category VARCHAR(20));

Table created.

SQL> INSERT INTO Sales VALUES (1, 'Product A', 10, 15.50, 'Category 1');

1 row created.

SQL> INSERT INTO Sales VALUES (2, 'Product B', 20, 25.75, 'Category 2');

1 row created.

SQL> INSERT INTO Sales VALUES (3, 'Product C', 15, 10.00, 'Category 1');

1 row created.

SQL> INSERT INTO Sales VALUES (4, 'Product D', 8, 50.00, 'Category 3');

1 row created.

SQL> INSERT INTO Sales VALUES (5, 'Product E', 12, 40.00, 'Category 2');

1 row created.

SQL> SELECT \* FROM sales;

PRODUCTID PRODUCTNAME QUANTITY PRICE CATEGORY

-----------------------------------------------------------------------------------------------------

1 Product A 10 15.5 Category 1

2 Product B 20 25.75 Category 2

3 Product C 15 10 Category 1

4 Product D 8 50 Category 3

5 Product E 12 40 Category 2

SQL> SELECT COUNT(\*) AS TotalRows FROM Sales;

TOTALROWS

--------------------------

5

SQL> SELECT SUM(Quantity) AS TotalQuantity FROM Sales;

TOTALQUANTITY

--------------------------------

65

SQL> SELECT MAX(Price) AS MaximumPrice FROM Sales;

MAXIMUMPRICE

--------------------------------

50

SQL> SELECT MIN(Price) AS MinimumPrice FROM Sales;

MINIMUMPRICE

---------------------------------------------

10

SQL> SELECT Category, SUM(Quantity) AS TotalQuantity FROM Sales GROUP BY Category HAVING SUM(Quantity) > 15;

CATEGORY TOTALQUANTITY

----------------------------------------------------------------

Category 1 25

Category 2 32

SQL> SELECT AVG(Price) AS AveragePrice FROM sales;

AVERAGEPRICE

------------------------------------------

28.25

SQL> CREATE VIEW salessummary as select productname,quantity \* price as totalrLokeshnue from sales;

View created.

SQL> SELECT \* FROM SalesSummary;

PRODUCTNAME TOTALRLOKESHNUE

-------------------------------------------------------------------------------------------

Product A 155

Product B 515

Product C 150

Product D 400

Product E 480

SQL> DROP VIEW SalesSummary;

View dropped.

4.**CONVERSION, STRING AND DATE FUNCTIONS**

SQL> CREATE TABLE Employee ( EmpID INT, EmpName VARCHAR(50), Salary DECIMAL(10, 2), JoinDate DATE );

Table created.

SQL> INSERT INTO Employee VALUES (1, 'John Doe', 50000, TO\_DATE('2022-01-15', 'YYYY-MM-DD'));

1 row created.

SQL> INSERT INTO Employee VALUES (2, 'Jane Smith', 65000, TO\_DATE('2021-06-01', 'YYYY-MM-DD'));

1 row created.

SQL> INSERT INTO Employee VALUES (3, 'Nikhil Johnson', 45000, TO\_DATE('2023-03-10', 'YYYY-MM-DD'));

1 row created.

SQL> SELECT \* FROM Employee;

EMPID EMPNAME SALARY JOINDATE

-----------------------------------------------------------------------------

1 John Doe 50000 15-JAN-22

2 Jane Smith 65000 01-JUN-21

3 Nikhil Johnson 45000 10-MAR-23

SQL> SELECT TO\_CHAR(SYSDATE, 'DD-MON-YYYY') AS CurrentDate FROM DUAL;

CURRENTDATE

------------------------------------------

10-JAN-2025

SQL> SELECT TO\_NUMBER('12345.67') AS NumberValue FROM DUAL;

NUMBERVALUE

-----------------------------------

12345.67

SQL> SELECT TO\_DATE('31-DEC-2024', 'DD-MON-YYYY') AS DateValue FROM DUAL;

DATEVALUE

--------------------------

31-DEC-24

SQL> SELECT CONCAT('Hello', ' World') AS ConcatenatedString FROM DUAL;

CONCATENATESTRING

------------------------------------------

Hello World

SQL> SELECT LPAD('SQL', 10, '\*') AS LeftPadded FROM DUAL;

LEFTPADDED

-------------------------------

\*\*\*\*\*\*\*SQL

SQL> SELECT RPAD('SQL', 10, '-') AS RightPadded FROM DUAL;

RIGHTPADDED

--------------------------------------

SQL-------

SQL> SELECT LTRIM(' TrimLeft') AS LeftTrimmed FROM DUAL;

LEFTTRIMMED

----------------------------

TrimLeft

SQL> SELECT RTRIM('TrimRight ') AS RightTrimmed FROM DUAL;

RIGHTTRIMMED

-----------------------------------

TrimRight

SQL> SELECT LOWER('LOWERCASE') AS LowerCase FROM DUAL;

LOWERCASE

-------------------------------

lowercase

SQL> SELECT UPPER('uppercase') AS UpperCase FROM DUAL;

UPPERCASE

-----------------------------

UPPERCASE

SQL> SELECT INITCAP('sql language') AS InitCap FROM DUAL;

INITCAP

------------------------------

Sql Language

SQL> SELECT LENGTH('Measure') AS LengthOfString FROM DUAL;

LENGTHOFSTRING

--------------------------------------

7

SQL> SELECT SUBSTR('Substring', 1, 3) AS SubString FROM DUAL;

SUBSTRING

---------------------------------

Sub

SQL> SELECT INSTR('FindPosition', 'i') AS PositionInString FROM DUAL;

POSITIONINSTRING

---------------------------------------

2

SQL> SELECT SYSDATE AS CurrentDate FROM DUAL;

CURRENTDATE

-------------------------------------

10-JAN-25

SQL> SELECT NEXT\_DAY(SYSDATE, 'MONDAY') AS NextMonday FROM DUAL;

NEXTMONDAY

---------------------------------------

13-JAN-25

SQL> SELECT ADD\_MONTHS(SYSDATE, 6) AS DateAfterSixMonths FROM DUAL;

DATEAFTERSIXMONTHS

---------------------------------------

10-JUL-25

SQL> SELECT LAST\_DAY(SYSDATE) AS LastDayOfMonth FROM DUAL;

LASTDAYOFMONTH

-----------------------------------

31-JAN-25

SQL> SELECT MONTHS\_BETWEEN(TO\_DATE('2024-12-31', 'YYYY-MM-DD'), SYSDATE) AS MonthsDifference FROM DUAL;

MONTHSDIFFERENCE

-----------------------------------------------

-0.23

SQL> SELECT LEAST(SYSDATE, TO\_DATE('2025-01-01', 'YYYY-MM-DD')) AS EarlierDate FROM DUAL;

EARLIERDATE

--------------------------

01-JAN-25

SQL> SELECT GREATEST(SYSDATE, TO\_DATE('2025-01-01', 'YYYY-MM-DD')) AS LaterDate FROM DUAL;

LATERDATE

--------------------------

10-JAN-25

SQL> SELECT TRUNC(SYSDATE, 'MM') AS FirstDayOfMonth FROM DUAL;

FIRSTDAYOFMONTH

------------------------------------

01-JAN-25

SQL> SELECT ROUND(SYSDATE, 'MM') AS RoundedDate FROM DUAL;

ROUNDEDDATE

---------------------------------

01-JAN-25

**5(i) Declaration Execution Exception Handling**

SQL> CREATE TABLE Students (StudentID INT, StudentName VARCHAR(50), Marks INT );

Table created.

SQL> INSERT INTO Students VALUES (1, 'Nikhil', 85);

1 row created.

SQL> INSERT INTO Students VALUES (2, 'Fareed', 72);

1 row created.

SQL> INSERT INTO Students VALUES (3, 'Charan', 58);

1 row created.

SQL> INSERT INTO Students VALUES (4, 'Naveen', 90);

1 row created.

SQL> SET SERVEROUTPUT ON

SQL> DECLARE

2 v\_StudentName VARCHAR(50);

3 v\_Marks INT;

4 CURSOR FirstClassCursor IS

5 SELECT StudentName, Marks FROM Students WHERE Marks >= 60;

6 NoRecordsFound EXCEPTION;

7 v\_RecordCount INT := 0;

8 BEGIN

9 OPEN FirstClassCursor;

10 LOOP

11 FETCH FirstClassCursor INTO v\_StudentName, v\_Marks;

12 EXIT WHEN FirstClassCursor%NOTFOUND;

13 v\_RecordCount := v\_RecordCount + 1;

14 DBMS\_OUTPUT.PUT\_LINE('Student: ' || v\_StudentName || ', Marks: ' || v\_Marks);

15 END LOOP;

16 CLOSE FirstClassCursor;

17 IF v\_RecordCount = 0 THEN

18 RAISE NoRecordsFound;

19 END IF;

20 EXCEPTION

21 WHEN NoRecordsFound THEN

22 DBMS\_OUTPUT.PUT\_LINE('No records found for students with first-class marks.');

23 END;

24 /

Student: Nikhil, Marks: 85

Student: Fareed, Marks: 72

Student: Naveen, Marks: 90

PL/SQL procedure successfully completed.

**5(ii) Commit Rollback and Savepoint in SQL**

SQL> CREATE TABLE student (id NUMBER PRIMARY KEY, name VARCHAR2(50), age NUMBER, department VARCHAR2(50) );

Table created.

SQL> DECLARE

2 BEGIN

3 INSERT INTO student (id, name, age, department) VALUES (1, 'Nikhil', 20, 'Physics');

4 INSERT INTO student (id, name, age, department) VALUES (2, 'Dugra', 22, 'Mathematics');

5 SAVEPOINT savepoint1; -- Create a savepoint named savepoint1

6 INSERT INTO student (id, name, age, department) VALUES (3, 'Charan ', 23, 'Chemistry');

7 ROLLBACK TO savepoint1; -- Rollback changes after the savepoint

8 COMMIT; -- Commit the changes up to the savepoint

9 END;

10 /

PL/SQL procedure successfully completed.

SQL> SELECT \* FROM student;

ID NAME AGE DEPARTMENT

----------------------------------------------------------------

1 Nikhil 20 Physics

2 Dugra 22 Mathematics

**7. LOOPS AND ERROR HANDLING**

SQL> DECLARE

2 i NUMBER := 1;

3 total\_sum NUMBER := 0;

4 ex\_negative\_value EXCEPTION;

5

6 BEGIN

7 WHILE i <= 5 LOOP

8 total\_sum := total\_sum + i;

9 i := i + 1;

10 END LOOP;

11 DBMS\_OUTPUT.PUT\_LINE('Total sum (WHILE loop): ' || total\_sum);

12

13 FOR j IN 1..5 LOOP

14 DBMS\_OUTPUT.PUT\_LINE('Iteration (FOR loop): ' || j);

15 END LOOP;

16

17 FOR x IN 1..3 LOOP

18 FOR y IN 1..3 LOOP

19 DBMS\_OUTPUT.PUT\_LINE('Nested loop: x=' || x || ', y=' || y);

20 END LOOP;

21 END LOOP;

22

23 BEGIN

24 total\_sum := total\_sum / 0;

25 EXCEPTION

26 WHEN ZERO\_DIVIDE THEN

27 DBMS\_OUTPUT.PUT\_LINE('Error: Division by zero!');

28 END;

29

30 IF total\_sum < 0 THEN

31 RAISE ex\_negative\_value;

32 END IF;

33

34 EXCEPTION

35 WHEN ex\_negative\_value THEN

36 DBMS\_OUTPUT.PUT\_LINE('Error: Total sum is negative!');

37 WHEN OTHERS THEN

38 RAISE\_APPLICATION\_ERROR(-20001, 'An unexpected error occurred.');

39 END;

40 /

Total sum (WHILE loop): 15

Iteration (FOR loop): 1

Iteration (FOR loop): 2

Iteration (FOR loop): 3

Iteration (FOR loop): 4

Iteration (FOR loop): 5

Nested loop: x=1, y=1

Nested loop: x=1, y=2

Nested loop: x=1, y=3

Nested loop: x=2, y=1

Nested loop: x=2, y=2

Nested loop: x=2, y=3

Nested loop: x=3, y=1

Nested loop: x=3, y=2

Nested loop: x=3, y=3

Error: Division by zero

**6. Nested IF Case Null IF and COALESCE**

SQL> CREATE TABLE employee ( id NUMBER PRIMARY KEY,name VARCHAR2(50), salary NUMBER,department VARCHAR2(50));

Table created.

SQL> DECLARE

2 salary\_category VARCHAR2(20);

3 tax\_slab VARCHAR2(20);

4 status VARCHAR2(20);

5 adjusted\_salary NUMBER;

6 BEGIN

7 INSERT INTO employee (id, name, salary, department) VALUES (1, 'John', 70000, 'IT');

8 INSERT INTO employee (id, name, salary, department) VALUES (2, 'Doe', 40000, 'HR');

9 INSERT INTO employee (id, name, salary, department) VALUES (3, 'Jane', NULL, 'Charan');

10 FOR emp IN (SELECT \* FROM employee) LOOP

11 IF emp.salary IS NULL THEN

12 salary\_category := 'No Salary';

13 ELSE

14 IF emp.salary > 50000 THEN

15 salary\_category := 'High';

16 ELSIF emp.salary BETWEEN 30000 AND 50000 THEN

17 salary\_category := 'Medium';

18 ELSE

19 salary\_category := 'Low';

20 END IF;

21 END IF;

22 tax\_slab := CASE

23 WHEN emp.salary > 60000 THEN '20%'

24 WHEN emp.salary BETWEEN 40000 AND 60000 THEN '15%'

25 ELSE '10%'

26 END;

27 status := CASE emp.department

28 WHEN 'IT' THEN 'Tech'

29 WHEN 'HR' THEN 'People Ops'

30 ELSE 'Others'

31 END;

32 adjusted\_salary := COALESCE(NULLIF(emp.salary, 0), 30000); -- Default salary is 30000 if NULL or 0

33 DBMS\_OUTPUT.PUT\_LINE('Employee: ' || emp.name);

34 DBMS\_OUTPUT.PUT\_LINE('Salary Category: ' || salary\_category);

35 DBMS\_OUTPUT.PUT\_LINE('Tax Slab: ' || tax\_slab);

36 DBMS\_OUTPUT.PUT\_LINE('Status: ' || status);

37 DBMS\_OUTPUT.PUT\_LINE('Adjusted Salary: ' || adjusted\_salary);

38 END LOOP;

39 END;

40 /

Employee: John

Salary Category: High

Tax Slab: 20%

Status: Tech

Adjusted Salary: 70000

Employee: Doe

Salary Category: Medium

Tax Slab: 15%

Status: People Ops

Adjusted Salary: 40000

Employee: Jane

Salary Category: No Salary

Tax Slab: 10%

Status: Others

Adjusted Salary: 30000

PL/SQL procedure successfully completed.

**8. Procedures With In and Out Parameter**

SQL> CREATE TABLE Students ( StudentID INT PRIMARY KEY, Name VARCHAR(50), Marks INT );

Table created.

SQL> CREATE OR REPLACE PROCEDURE GetStudentMarks (

2 stu\_id IN NUMBER,

3 stu\_marks OUT NUMBER

4 ) AS

5 BEGIN

6 SELECT Marks INTO stu\_marks

7 FROM Students

8 WHERE StudentID = stu\_id;

9 END;

10 /

Procedure created.

SQL> INSERT INTO Students VALUES (1, 'Nikhil', 85);

1 row created.

SQL> INSERT INTO Students VALUES(2, 'Dugra', 90);

1 row created.

SQL> INSERT INTO Students VALUES (3, 'Charan ', 78);

1 row created.

SQL> SELECT \* FROM Students;

STUDENTID NAME MARKS

---------- -------------------------------------------------- ----------

1 Nikhil 85

2 Dugra 90

3 Charan 78

SQL> VARIABLE marks NUMBER;

SQL> EXEC GetStudentMarks(2, :marks);

PL/SQL procedure successfully completed.

SQL> PRINT marks;

MARKS

-------------------

90

**9. SQL Stored Functions**

SQL> CREATE SEQUENCE EmployeeSeq START WITH 1 INCREMENT BY 1 NOCACHE;

Sequence created.

SQL> CREATE TABLE Employees(EmployeeID INT PRIMARY KEY, Name VARCHAR2(100), BasicSalary DECIMAL(10, 2), Bonus DECIMAL(10, 2), Deductions DECIMAL(10, 2));

Table created.

SQL> CREATE OR REPLACE TRIGGER trg\_EmployeeID

2 BEFORE INSERT ON Employees

3 FOR EACH ROW

4 BEGIN

5 :NEW.EmployeeID := EmployeeSeq.NEXTVAL;

6 END;

7 /

Trigger created.

SQL> INSERT INTO Employees (Name, BasicSalary, Bonus, Deductions) VALUES ('Nikhil', 50000, 5000, 2000);

1 row created.

SQL> INSERT INTO Employees (Name, BasicSalary, Bonus, Deductions) VALUES ('Dugra', 60000, 7000, 3000);

1 row created.

SQL> INSERT INTO Employees (Name, BasicSalary, Bonus, Deductions) VALUES ('Charan ', 55000, 4000, 2500);

1 row created.

SQL> CREATE OR REPLACE FUNCTION calculate\_net\_salary(

2 basic\_salary DECIMAL,

3 bonus DECIMAL,

4 deductions DECIMAL

5 ) RETURN DECIMAL IS

6 net\_salary DECIMAL(10, 2);

7 BEGIN

8 net\_salary := (basic\_salary + bonus) - deductions;

9 RETURN net\_salary;

10 END;

11 /

Function created.

SQL> SELECT EmployeeID,Name,BasicSalary,Bonus,Deductions,calculate\_net\_salary(BasicSalary, Bonus, Deductions) AS NetSalary FROM Employees;

EMPID EMPNAME BASICSALARY BONUS DEDUCTIONS NETSALARY

----------------------------------------------------------------------------------------------------------------------

1 Nikhil 50000 5000 2000 53000

2 Dugra 60000 7000 3000 64000

3 Charan 55000 4000 2500 56500

SQL> SELECT EmployeeID,Name,calculate\_net\_salary(BasicSalary, Bonus, Deductions) AS NetSalary FROM Employees WHERE calculate\_net\_salary(BasicSalary, Bonus, Deductions) > 55000;

EMPLOYEEID NAME NETSALARY

-----------------------------------------------------------

2 Dugra 64000

3 Charan 56500

SQL> INSERT INTO Employees (Name, BasicSalary, Bonus, Deductions) VALUES ('Rajesh', 0, 0,0);

1 row created.

SQL> INSERT INTO Employees (Name, BasicSalary, Bonus, Deductions) VALUES ('Lokesh', 45000, 5000, 60000);

1 row created.

SQL> SELECT EmployeeID,Name,calculate\_net\_salary(BasicSalary, Bonus, Deductions) AS NetSalary FROM Employees;

EMPLOYEEID NAME NETSALARY

---------------------------------------------------------------------

1 Nikhil 53000

2 Dugra 64000

3 Charan 56500

4 Rajesh 0

5 Lokesh -10000

**10. Cursor**

SQL> CREATE TABLE Employee ( EmpID INT PRIMARY KEY, Name VARCHAR(50), Salary DECIMAL(10, 2) );

Table created.

SQL> INSERT INTO Employee VALUES (1, 'Nikhil', 30000);

1 row created.

SQL> INSERT INTO Employee VALUES (2, 'Charan', 25000);

1 row created.

SQL> INSERT INTO Employee VALUES(3, 'Naveen', 40000);

1 row created.

SQL> CREATE OR REPLACE PROCEDURE UpdateLowSalary(

2 salary\_limit IN DECIMAL,

3 increment IN DECIMAL

4 )

5 IS

6 CURSOR emp\_cursor IS

7 SELECT EmpID FROM Employee WHERE Salary < salary\_limit FOR UPDATE;

8 emp\_id INT;

9 BEGIN

10 FOR emp\_record IN emp\_cursor LOOP

11 UPDATE Employee

12 SET Salary = Salary + increment

13 WHERE EmpID = emp\_record.EmpID;

14 END LOOP;

15 END;

16 /

Procedure created.

SQL> EXEC UpdateLowSalary(35000, 5000);

PL/SQL procedure successfully completed.

SQL> SELECT \* FROM Employee;

EMPID NAME SALARY

1 Nikhil 35000

2 Charan 30000

3 Naveen 40000

**11. Triggers**

SQL> CREATE TABLE employees ( employee\_id INT PRIMARY KEY, name VARCHAR2(50), salary DECIMAL(10, 2), created\_at DATE );

Table created.

SQL> INSERT INTO employees (employee\_id, name, salary, created\_at) VALUES (1,'Nikhil', 30000, SYSDATE);

1 row created.

SQL> INSERT INTO employees (employee\_id, name, salary, created\_at) VALUES (2, 'Charan', 25000, SYSDATE);

1 row created.

SQL> INSERT INTO employees (employee\_id, name, salary, created\_at) VALUES (3, 'Naveen', 40000, SYSDATE);

1 row created.

SQL> CREATE VIEW employee\_view AS

2 SELECT employee\_id, name, salary

3 FROM employees;

View created.

SQL> CREATE OR REPLACE TRIGGER instead\_of\_trigger

2 INSTEAD OF INSERT ON employee\_view

3 BEGIN

4 INSERT INTO employees (employee\_id, name, salary)

5 VALUES (:NEW.employee\_id, :NEW.name, :NEW.salary);

6 END;

7 /

Trigger created.

SQL> CREATE OR REPLACE TRIGGER before\_insert\_trigger

2 BEFORE INSERT ON employees

3 FOR EACH ROW

4 BEGIN

5 :NEW.created\_at := SYSDATE;

6 END;

7 /

Trigger created.

SQL> set sAkashroutput on

SP2-0735: unknown SET option beginning "sAkashroutpu..."

SQL> CREATE OR REPLACE TRIGGER after\_delete\_trigger

2 AFTER DELETE ON employees

3 FOR EACH ROW

4 BEGIN

5 DBMS\_OUTPUT.PUT\_LINE('Employee deleted: ' || :OLD.employee\_id);

6 END;

7 /

Trigger created.

SQL> CREATE OR REPLACE TRIGGER row\_update\_trigger

2 BEFORE UPDATE ON employees

3 FOR EACH ROW

4 BEGIN

5 IF :NEW.salary > :OLD.salary THEN

6 DBMS\_OUTPUT.PUT\_LINE('Salary increased for Employee ID: ' || :NEW.employee\_id);

7 END IF;

8 END;

9 /

Trigger created..

SQL> CREATE OR REPLACE TRIGGER statement\_update\_trigger

2 AFTER UPDATE ON employees

3 BEGIN

4 DBMS\_OUTPUT.PUT\_LINE('Employees table updated.');

5 END;

6 /

Trigger created.

SQL> INSERT INTO employee\_view (employee\_id, name, salary)

2 VALUES (4, 'John Doe', 35000);

1 row created.

SQL> UPDATE employees

2 SET salary = salary + 5000

3 WHERE employee\_id = 1;

Salary increased for Employee ID: 1

Employees table updated.

1 row updated.

SQL> DELETE FROM employees WHERE employee\_id = 2;

Employee deleted: 2

1 row deleted.

SQL> SELECT \* FROM employees;

EMPLOYEE ID NAME SALARY CREATED\_A

-------------------------------------------------------------------------------------------------

1 Nikhil 35000 17-JAN-25

3 Naveen 40000 17-JAN-25

4 John Doe 35000 17-JAN-25

**12. Indexing and Non-Indexing Search in SQL**

SQL> CREATE TABLE EMPLOYEES (EMP\_ID NUMBER PRIMARY KEY,NAME VARCHAR2(50), DEPARTMENT VARCHAR2(50), SALARY NUMBER );

Table created.

SQL>

SQL> INSERT INTO EMPLOYEES VALUES (1, 'Nikhil', 'HR', 50000);

1 row created.

SQL> INSERT INTO EMPLOYEES VALUES (2, 'Dugra', 'IT', 60000);

1 row created.

SQL> INSERT INTO EMPLOYEES VALUES (3, 'Charan',’ Finance’, 55000);

1 row created.

SQL> INSERT INTO EMPLOYEES VALUES (4, 'Lokesh', 'IT', 70000);

1 row created.

SQL> INSERT INTO EMPLOYEES VALUES (5, 'Akash', 'HR', 45000);

1 row created.

SQL> COMMIT;

Commit complete.

SQL> SELECT \* FROM EMPLOYEES WHERE DEPARTMENT = 'IT';

EMP\_ID NAME DEPARTMENT SALARY

----------------------------------------------------------------------------

2 Dugra IT 60000

4 Lokesh IT 70000

SQL> CREATE INDEX idx\_department ON EMPLOYEES (DEPARTMENT);

Index created.

SQL> SELECT \* FROM EMPLOYEES WHERE DEPARTMENT = 'IT';

EMP\_ID NAME DEPARTMENT SALARY

-------------------------------------------------------------------------------

2 Dugra IT 60000

4 Lokesh IT 70000

SQL> EXPLAIN PLAN FOR

2 SELECT \* FROM EMPLOYEES WHERE DEPARTMENT = 'IT';

Explained.

SQL> SELECT \* FROM TABLE(DBMS\_XPLAN.DISPLAY);

Plan hash value: 1234567890

------------------------------------------------------------------------------------------------------------------------

| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |

-------------------------------------------------------------------------------------------------------------------------

| 0 | SELECT STATEMENT | | 2 | 160 | 1 (0) | 00:00:01 |

| 1 | INDEX RANGE SCAN | IDX\_DEPARTMENT | 2 | 160 | 1 (0) | 00:00:01 |

------------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):

---------------------------------------------------

1 - access("DEPARTMENT"='IT')

SQL> DROP INDEX idx\_department;

Index dropped.