1. **Query using LIKE and ORDER BY:**

SELECT \* FROM student WHERE s\_name LIKE 'J%';

1. **Query using BETWEEN:**

SELECT \* FROM test WHERE test\_date BETWEEN '2023-01-01' AND '2023-12-31';

1. **Query using IN:**

SELECT \* FROM results WHERE ob\_marks IN (85, 90, 95);

1. **Query using GROUP BY:**

SELECT sid, AVG(ob\_marks) AS average\_marks FROM results GROUP BY sid;

1. **Query using ORDER BY and LIMIT:**

SELECT \* FROM feedback ORDER BY sid ;

1. **Inner Join Query:**

SELECT s.s\_name, t.name FROM student s INNER JOIN test t ON s.sid = t.sid;

1. **Left Join Query:**

SELECT s.s\_name, r.ob\_marks FROM student s LEFT JOIN results r ON s.sid = r.sid;

1. **Right Join Query:**

SELECT t.name, r.ob\_marks FROM test t RIGHT JOIN results r ON t.tid = r.tid;

1. **Outer Join Query:**

SELECT s.s\_name, c.cid FROM student s FULL OUTER JOIN course c ON s.sid = c.sid;

1. **Subquery Example:**

SELECT s\_name FROM student WHERE sid IN (SELECT sid FROM results WHERE ob\_marks > 90);

1. **Query with Aggregate Function:**

SELECT MAX(amount) AS max\_payment FROM payment;

1. **Query with Scalar Function:**

SELECT UPPER(s\_name) AS upper\_name FROM student;

1. **Cursor Example:**

DECLARE

v\_sid student.sid%TYPE;

v\_sname student.s\_name%TYPE;

CURSOR student\_cursor IS SELECT sid, s\_name FROM student;

BEGIN

OPEN student\_cursor;

LOOP

FETCH student\_cursor INTO v\_sid, v\_sname;

EXIT WHEN student\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Student ID: ' || v\_sid || ', Student Name: ' || v\_sname);

END LOOP;

CLOSE student\_cursor;

END;

1. **Trigger Example:**

CREATE OR REPLACE TRIGGER update\_rating

AFTER INSERT ON results

FOR EACH ROW

BEGIN

UPDATE course SET rating = (SELECT AVG(ob\_marks) FROM results WHERE cid = :new.cid) WHERE cid = :new.cid;

END;

1. **Trigger Example 2:**

CREATE OR REPLACE TRIGGER log\_payment

AFTER INSERT ON payment

FOR EACH ROW

BEGIN

INSERT INTO payment\_log (p\_id, amount, payment\_date) VALUES (:new.p\_id, :new.amount, :new.payment\_date);

END;

* 1. SELECT s\_name

FROM STUDENT

WHERE sid IN (SELECT sid FROM PAID\_COURSE);

* 1. SELECT sid, COUNT(DISTINCT cid) AS num\_courses\_enrolled

FROM COURSE

GROUP BY sid;

* 1. SELECT r.\*, t.name AS test\_name, c.cid AS course\_id

FROM RESULTS r

INNER JOIN TEST t ON r.tid = t.tid

INNER JOIN COURSE c ON t.sid = c.sid

WHERE r.ob\_marks >= 80; -- Example condition: filtering results where the obtained marks are greater than or equal to 80