Set 4

1. Display all the departments where department does not have any employees.

```
SELECT d.deptno, d.dname
 FROM dept d
 LEFT JOIN emp e ON d.deptno = e.deptno
 WHERE e.empno IS NULL;
```

2. Display all the departments where department does have at least one employee.

```
SELECT d.deptno, d.dname
FROM dept d
JOIN emp e ON d.deptno = e.deptno
GROUP BY d.deptno, d.dname;
```

3. Display all employees those who are not managers?

```
SELECT empno, ename, job
FROM emp
 WHERE empno NOT IN (SELECT DISTINCT mgr FROM emp WHERE mgr IS NOT
NULL);
```

4. Display ename, deptno from emp table with format of (ename) belongs to (deptno)

```
 SELECT ename || ' belongs to ' || deptno AS "Employee Department Info"
 FROM emp;
```

5. Display total number of employees hired for 1980, 1981, 1982. The output should be in one record

```
 SELECT
 SUM(CASE WHEN TO_CHAR(hiredate, 'YYYY') = '1980' THEN 1 ELSE 0 END) AS "1980",
SUM(CASE WHEN TO_CHAR(hiredate, 'YYYY') = '1981' THEN 1 ELSE 0 END) AS "1981",
SUM(CASE WHEN TO_CHAR(hiredate, 'YYYY') = '1982' THEN 1 ELSE 0 END) AS "1982"
FROM emp;
```

1. Write a query to get 4th max salary from EMP table

```
SELECT sal
FROM (
    SELECT sal, DENSE_RANK() OVER (ORDER BY sal DESC) rank
    FROM emp
)
WHERE rank = 4;
```

2. Write a query to get 2nd & 6th max salary from EMP table

```
SELECT sal
FROM (
   SELECT sal, DENSE_RANK() OVER (ORDER BY sal DESC) rank
   FROM emp
)
WHERE rank IN (2, 6);
```

3. Write a query to get first 3 salaries from the EMP table

```
SELECT sal
FROM (
   SELECT sal, DENSE_RANK() OVER (ORDER BY sal DESC) rank
   FROM emp
)
WHERE rank <= 3;
```

1. Write a program to accept a mgr and display who are working under that mgr?

```
SET SERVEROUTPUT ON;
DECLARE
   v_mgr NUMBER :=  &enter_manager_number;

   CURSOR emp_cursor IS
      SELECT empno, ename, job
      FROM emp
      WHERE mgr = v_mgr;   -- now v_mgr has value
BEGIN
   FOR emp_rec IN emp_cursor LOOP
      DBMS_OUTPUT.PUT_LINE('Emp No: ' || emp_rec.empno ||
                     ', Name: ' || emp_rec.ename ||
                     ', Job: ' || emp_rec.job);
   END LOOP;
END;
```

2. Write a program to accept the grade and display emps belongs to that grade?

```
SET SERVEROUTPUT ON;

DECLARE
   v_grade NUMBER := &enter_grade;

   CURSOR emp_cursor IS
      SELECT e.empno, e.ename, g.grade
      FROM emp e
      JOIN salgrade g
          ON e.sal BETWEEN g.losal AND g.hisal
      WHERE g.grade = v_grade;
BEGIN
   FOR emp_rec IN emp_cursor LOOP
      DBMS_OUTPUT.PUT_LINE('Emp No: ' || emp_rec.empno ||
                      ', Name: ' || emp_rec.ename ||
                      ', Grade: ' || emp_rec.grade);
   END LOOP;
END;
```

3. Write a program to accept a deptno and display who are working in that dept?

```
SET SERVEROUTPUT ON;

DECLARE
   v_deptno NUMBER := &Enter_Deptno;   -- Accept deptno
   CURSOR emp_cursor IS
      SELECT empno, ename, job
      FROM emp
      WHERE deptno = v_deptno;
BEGIN
   FOR emp_rec IN emp_cursor LOOP
      DBMS_OUTPUT.PUT_LINE('Emp No: ' || emp_rec.empno ||
                           ', Name: ' || emp_rec.ename ||
                           ', Job: ' || emp_rec.job);
   END LOOP;
END;
/
```

1. Write a database trigger to store the deleted data of EMP table in EMPDEL table.

```
CREATE OR REPLACE TRIGGER emp_delete_trigger
AFTER DELETE ON emp
FOR EACH ROW
BEGIN
   INSERT INTO empdel (empno, ename, job, sal, deptno)
   VALUES (:OLD.empno, :OLD.ename, :OLD.job, :OLD.sal,
:OLD.deptno);
END;
/
```

2. Write a database trigger display the message when the inserting hiredate is greater than system date

```
CREATE OR REPLACE TRIGGER hiredate_check_trigger
BEFORE INSERT ON emp
FOR EACH ROW
BEGIN
   IF :NEW.hiredate > SYSDATE THEN
      RAISE_APPLICATION_ERROR(-20001,'Hire date cannot be in the
future.') ;
```

```
    END IF;
END;
/
```

3. Write a database trigger add Rs 500 if the inserting salary is less than Rs 1000

```
CREATE OR REPLACE TRIGGER salary_check_trigger
BEFORE INSERT OR UPDATE ON emp
FOR EACH ROW
BEGIN
    IF :NEW.sal < 1000 THEN
        :NEW.sal := :NEW.sal + 500;
    END IF;
END;
/
```

Set 5

1. Write a program to accept the location and display empno, name, sal, date of join and also display the total salary, avg salary and no of emps?

```
SET SERVEROUTPUT ON;

DECLARE
   CURSOR emp_cursor (loc_name VARCHAR2) IS
      SELECT e.empno, e.ename, e.sal, e.hiredate
      FROM emp e
      JOIN dept d ON e.deptno = d.deptno
      WHERE d.loc = loc_name;

   total_salary NUMBER := 0;
   emp_count    NUMBER := 0;
   avg_salary   NUMBER := 0;
BEGIN
   FOR emp_rec IN emp_cursor('NEW YORK') LOOP
      total_salary := total_salary + emp_rec.sal;
      emp_count    := emp_count + 1;

      DBMS_OUTPUT.PUT_LINE(
         'Emp No: ' || emp_rec.empno ||
         ', Name: ' || emp_rec.ename ||
         ', Salary: ' || emp_rec.sal ||
         ', Hire Date: ' || emp_rec.hiredate
      );
   END LOOP;

   IF emp_count > 0 THEN
      avg_salary := total_salary / emp_count;
   END IF;

   DBMS_OUTPUT.PUT_LINE('Total Salary: ' || total_salary);
   DBMS_OUTPUT.PUT_LINE('Average Salary: ' || avg_salary);
   DBMS_OUTPUT.PUT_LINE('Number of Employees: ' || emp_count);
END;
/
```

2. Write a program to accept a range of salary (that is lower boundary and higher boundary) and print the details of emps along with loc,grade and exp

```
SET SERVEROUTPUT ON;

DECLARE
   CURSOR salary_cursor (low_sal NUMBER, high_sal NUMBER) IS
      SELECT e.empno, e.ename, e.sal, d.loc, g.grade
       FROM emp e
       JOIN dept d ON e.deptno = d.deptno
       JOIN salgrade g ON e.sal BETWEEN g.losal AND g.hisal
       WHERE e.sal BETWEEN low_sal AND high_sal;

BEGIN
   FOR emp_rec IN salary_cursor(2000, 5000) LOOP
      DBMS_OUTPUT.PUT_LINE(
         'Emp No: ' || emp_rec.empno ||
         ', Name: ' || emp_rec.ename ||
         ', Salary: ' || emp_rec.sal ||
         ', Location: ' || emp_rec.loc ||
         ', Grade: ' || emp_rec.grade ||
          );
   END LOOP;
END;
/
```

1. Write a database trigger halt the transaction on Sunday on EMP table

```
    CREATE OR REPLACE TRIGGER halt_transaction_on_sunday
         BEFORE INSERT OR UPDATE OR DELETE ON emp
         BEGIN
            IF TO_CHAR(SYSDATE, 'DY') = 'SUN' THEN
               RAISE_APPLICATION_ERROR(-20001, 'Transactions are not allowed on
         Sundays');

            END IF;
         END;
         /
```

2. Write a database trigger halt the transaction of USER SCOTT on table EMP

```
CREATE OR REPLACE TRIGGER halt_transaction_for_scott
BEFORE INSERT OR UPDATE OR DELETE ON emp
BEGIN
   IF USER = 'SCOTT' THEN
      RAISE_APPLICATION_ERROR(-20001,
         'User SCOTT is not allowed to perform DML on the EMP table');
   END IF;
   END;
```

1. List the employees working in research department.

```
SELECT e.empno, e.ename, e.job, e.sal
FROM emp e
JOIN dept d ON e.deptno = d.deptno
WHERE d.dname = 'RESEARCH';
```

2. List employees who are located in New York and Chicago.

```
SELECT e.empno, e.ename, e.job, e.sal, d.loc
FROM emp e
JOIN dept d ON e.deptno = d.deptno
WHERE d.loc IN ('NEW YORK', 'CHICAGO');
```

3. Display the department name in which ANALYSTS are working

```
SELECT DISTINCT d.dname
FROM emp e
JOIN dept d ON e.deptno = d.deptno
WHERE e.job = 'ANALYST';
```

1. Create table empl and copy the emp table for deptno 10 while creating the table

```
CREATE TABLE empl AS
SELECT *
FROM emp
WHERE deptno = 10;
```

2. Insert new record in emp1 table, Merge the emp1 table on emp table.

```
INSERT INTO emp1 (empno, ename, job, mgr, hiredate, sal, comm, deptno)
VALUES (9999, 'NEW_EMP', 'CLERK', 7839, SYSDATE, 3000, NULL, 10);

MERGE INTO emp e
USING emp1 e1
ON (e.empno = e1.empno)
WHEN NOT MATCHED THEN
   INSERT (empno, ename, job, mgr, hiredate, sal, comm, deptno)
   VALUES (e1.empno, e1.ename, e1.job, e1.mgr, e1.hiredate, e1.sal, e1.comm, e1.deptno);
/
```

3. Create table emp2 with same structure of emp table. Do not copy the data

```
CREATE TABLE emp2 AS
SELECT *
FROM emp
WHERE 1 = 0;
```

SET - 7

1. Display all the records in EMP table
   SELECT * FROM emp;

2. Display all employees who do not have any reportees
   SELECT empno, ename, job
   FROM emp
   WHERE empno NOT IN (SELECT mgr FROM emp WHERE mgr IS NOT NULL);

3. Write a program accepted string and check whether it is palindrome or not.

```
SET SERVEROUTPUT ON;

DECLARE
   v_str VARCHAR2(50) := '&Enter_String';
   v_rev VARCHAR2(50) := '';
BEGIN
   -- Reverse the string
   FOR i IN REVERSE 1 .. LENGTH(v_str) LOOP
      v_rev := v_rev || SUBSTR(v_str, i, 1);
   END LOOP;

   -- Compare
   IF LOWER(v_str) = LOWER(v_rev) THEN
      DBMS_OUTPUT.PUT_LINE(v_str || ' is a palindrome.');
   ELSE
      DBMS_OUTPUT.PUT_LINE(v_str || ' is NOT a palindrome.');
   END IF;
END;
/
```

4. TRIGGERS IN WRITE A database Trigger hacked the transaction of emp table if the depart number is does not exist in the dept table

```
CREATE OR REPLACE TRIGGER check_dept_exists
BEFORE INSERT OR UPDATE ON emp
FOR EACH ROW
DECLARE
   v_count NUMBER;
BEGIN
   -- Check whether dept number exists
   SELECT COUNT(*) INTO v_count
   FROM dept
   WHERE deptno = :NEW.deptno;
```

```
    IF v_count = 0 THEN
       RAISE_APPLICATION_ERROR(-20050,
          'Department number does not exist in DEPT table.');
    END IF;
END;
/
```

SET - 9

1. Write a database trigger to halt the transaction of USER SCOTT on table EMP.

```
CREATE OR REPLACE TRIGGER halt_scott_transaction
BEFORE INSERT OR UPDATE OR DELETE ON emp
FOR EACH ROW
BEGIN
   IF USER = 'SCOTT' THEN
      RAISE_APPLICATION_ERROR(-20010,
         'User SCOTT is not allowed to perform transactions on EMP table.');
   END IF;
END;
/
```

2. Write a database trigger to halt the transaction between the time 6pm to 10pm on table EMP.

```
CREATE OR REPLACE TRIGGER halt_transaction_time
BEFORE INSERT OR UPDATE OR DELETE ON emp
FOR EACH ROW
BEGIN
   IF TO_NUMBER(TO_CHAR(SYSDATE, 'HH24')) BETWEEN 18 AND 22 THEN
      RAISE_APPLICATION_ERROR(-20002,
         'Transaction halted: Transactions on EMP table are not allowed between 6
         PM to 10 PM.'
      );
   END IF;
END;
/
```

1. Write a procedure to accept the deptno as parameter and display the details of that dept. Also display the total salary, no of employees, max sal and avg sal?

```
CREATE OR REPLACE PROCEDURE display_dept_details(p_deptno IN
emp.deptno%TYPE) IS
   v_total_sal NUMBER;
   v_num_emps NUMBER;
   v_max_sal   NUMBER;
   v_avg_sal   NUMBER;

   CURSOR dept_cursor IS
      SELECT SUM(sal), COUNT(*), MAX(sal), AVG(sal)
      FROM emp
      WHERE deptno = p_deptno;
BEGIN
   OPEN dept_cursor;
   FETCH dept_cursor INTO v_total_sal, v_num_emps, v_max_sal, v_avg_sal;

   IF v_num_emps > 0 THEN
      DBMS_OUTPUT.PUT_LINE('Total Salary: ' || v_total_sal);
      DBMS_OUTPUT.PUT_LINE('Number of Employees: ' || v_num_emps);
      DBMS_OUTPUT.PUT_LINE('Max Salary: ' || v_max_sal);
      DBMS_OUTPUT.PUT_LINE('Average Salary: ' || ROUND(v_avg_sal,2));
   ELSE
      DBMS_OUTPUT.PUT_LINE('No employees found for deptno: ' || p_deptno);
   END IF;

   CLOSE dept_cursor;
END;
/
```

2. Write a program to accept empno and print all the details along with loc and grade.

```
SET SERVEROUTPUT ON;

DECLARE
   v_empno NUMBER := &Enter_Empno;

     CURSOR emp_cursor IS
            SELECT e.empno, e.ename, e.job, e.mgr, e.hiredate,
                  e.sal, e.comm, e.deptno, d.dname, d.loc, g.grade
            FROM emp e
            JOIN dept d ON e.deptno = d.deptno
            JOIN salgrade g ON e.sal BETWEEN g.losal AND g.hisal
            WHERE e.empno = v_empno;

BEGIN
```

```
    FOR emp_rec IN emp_cursor LOOP
      DBMS_OUTPUT.PUT_LINE('Emp No: ' || emp_rec.empno ||
                           ', Name: ' || emp_rec.ename ||
                           ', Job: ' || emp_rec.job ||
                           ', Manager: ' || emp_rec.mgr ||
                           ', Hire Date: ' || TO_CHAR(emp_rec.hiredate, 'DD-MON-YYYY') ||
                           ', Salary: ' || emp_rec.sal ||
                           ', Commission: ' || NVL(emp_rec.comm, 0) ||
                           ', Dept No: ' || emp_rec.deptno ||
                           ', Dept Name: ' || emp_rec.dname ||
                           ', Location: ' || emp_rec.loc ||
                           ', Grade: ' || emp_rec.grade);
    END LOOP;

    -- Optional: If no record found
    IF SQL%ROWCOUNT = 0 THEN
      DBMS_OUTPUT.PUT_LINE('No employee found with empno: ' || v_empno);
    END IF;
  END;
  /
```

1. Display employees who are reporting to JONES.

```
SELECT *
FROM emp
WHERE mgr = (SELECT empno FROM emp WHERE ename = 'JONES');
```

2. Display all the employees who are reporting to Jones Manager.

```
SELECT *
FROM emp
WHERE mgr = (SELECT empno FROM emp WHERE ename = 'JONES');
```

3. Display all the managers in SALES and ACCOUNTING department.

```
SELECT DISTINCT e.ename AS manager
FROM emp e
JOIN dept d ON e.deptno = d.deptno
WHERE d.dname IN ('SALES', 'ACCOUNTING')
  AND e.job = 'MANAGER';
```

4. Display all the employee names in Research and Sales Department who are having at least 1 person reporting to them.

```
SELECT DISTINCT m.ename
FROM emp e
JOIN emp m ON e.mgr = m.empno
JOIN dept d ON m.deptno = d.deptno
WHERE d.dname IN ('RESEARCH', 'SALES');
```

1. Display all jobs that are in department 10. Include the location of department in the output.

```
SELECT DISTINCT e.job, d.loc
FROM emp e
JOIN dept d ON e.deptno = d.deptno
WHERE e.deptno = 10;
```

2. Write a query that displays the employee's names with the first letter capitalized and all other letters lowercase for all employees whose name starts with J, A, or M.

```
SELECT INITCAP(ename) AS formatted_name
FROM emp
WHERE SUBSTR(ename, 1, 1) IN ('J', 'A', 'M');
```

3. Display all the records in emp table where employee hired after 28-SEP-81 and before 03-DEC81.

```
SELECT *
FROM emp
WHERE hiredate > TO_DATE('28-SEP-1981', 'DD-MON-YYYY')
  AND hiredate < TO_DATE('03-DEC-1981', 'DD-MON-YYYY');
```

4. Write a query to display the employee name, department name of all employees who earn a commission.

```
SELECT e.ename, d.dname
FROM emp e
JOIN dept d ON e.deptno = d.deptno
WHERE e.comm IS NOT NULL;
```

1. Write a program to accept a deptno and display who are working in that dept?

```
SET SERVEROUTPUT ON;
```

```
DECLARE
   v_deptno NUMBER := &Enter_Deptno;

   CURSOR emp_cursor IS
      SELECT ename
      FROM emp
      WHERE deptno = v_deptno;

BEGIN
   FOR emp_rec IN emp_cursor LOOP
      DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_rec.ename);
   END LOOP;
```

2. Write a program to display all the information of emp table.

```
SET SERVEROUTPUT ON;

DECLARE
   v_deptno NUMBER := &Enter_Deptno;  -- Accept department number as input

   CURSOR emp_cursor IS
      SELECT ename
      FROM emp
      WHERE deptno = v_deptno;

BEGIN
   FOR emp_rec IN emp_cursor LOOP
      DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_rec.ename);
   END LOOP;

   -- Optional: if no employees found
   IF SQL%ROWCOUNT = 0 THEN
      DBMS_OUTPUT.PUT_LINE('No employees found in deptno: ' || v_deptno);
   END IF;
END;
/
```

SET -10

1) display all the employees who do not have any reportees

```
SELECT ename
FROM emp
WHERE empno NOT IN (SELECT DISTINCT mgr FROM emp WHERE mgr IS NOT NULL);
```

2) list employees who are having at least two reportings

```
SELECT e.ename, COUNT(r.empno) AS reportees_count
FROM emp e
JOIN emp r ON r.mgr = e.empno
GROUP BY e.ename
HAVING COUNT(r.empno) >= 2;
```

3) list the department names which are having more than 5 employees

```
SELECT d.dname, COUNT(e.empno) AS emp_count
FROM dept d
JOIN emp e ON e.deptno = d.deptno
GROUP BY d.dname
HAVING COUNT(e.empno) > 5;
```

4) list department name having at least three salesman

```
SELECT d.dname, COUNT(e.empno) AS sales_count
FROM dept d
JOIN emp e ON e.deptno = d.deptno
WHERE e.job = 'SALESMAN'
GROUP BY d.dname
HAVING COUNT(e.empno) >= 3;
```

5) list employees from research and accounting having at least two reportings

```
SELECT e.ename, COUNT(r.empno) AS reportees_count
FROM emp e
JOIN dept d ON e.deptno = d.deptno
JOIN emp r ON r.mgr = e.empno
WHERE d.dname IN ('RESEARCH', 'ACCOUNTING')
GROUP BY e.ename
HAVING COUNT(r.empno) >= 2;
```

1) DISPLAY THE EMPNO, ENAME ,SAL AND SALARY INCREASED BY 15%

```
SELECT empno, ename, sal, sal * 1.15 AS increased_salary
FROM emp;
```

2) DISPLAY ENAME,SAL,GRADE USE EMP SAL GRADE TABLE

```
. SELECT e.ename, e.sal, g.grade
  FROM emp e
 JOIN salgrade g ON e.sal BETWEEN g.losal AND g.hisal;
```

3) DISPLAY ALL EMPLOYEES AND CORRESPONDING MANAGERS

```
 SELECT e.empno, e.ename, e.job, m.ename AS manager
FROM emp e
LEFT JOIN emp m ON e.mgr = m.empno;
```

4) DISPLAY ALL THE DEPARTMENTS WHERE EMPLOYEES SALARY IS GREATER THAN AVERAGE SALARY OF THE DEPARTMENT

```
SELECT e.deptno, e.ename, e.sal
FROM emp e
WHERE e.sal > (SELECT AVG(sal) FROM emp WHERE deptno = e.deptno);
```

5) DISPLAY ALL EMPLOYEES WHO SALARY GREATER THAN THE MANAGER SALARY

```
SELECT e.empno, e.ename, e.sal
FROM emp e
WHERE e.sal > (SELECT sal FROM emp WHERE empno = e.mgr);
```

1. WRITE A PROGRAM TO ACCEPT A YEAR AND DISPLAY THE EMPS BELONGS TO THAT YEAR

```
SET SERVEROUTPUT ON;

DECLARE
   v_year NUMBER := &Enter_Year;  nput

   CURSOR emp_cursor IS
      SELECT empno, ename, hiredate
      FROM emp
      WHERE TO_CHAR(hiredate, 'YYYY') = v_year;
BEGIN
   FOR emp_rec IN emp_cursor LOOP
      DBMS_OUTPUT.PUT_LINE('Emp No: ' || emp_rec.empno ||
                           ', Name: ' || emp_rec.ename ||
                           ', Hire Date: ' || TO_CHAR(emp_rec.hiredate, 'DD-MON-YYYY'));
   END LOOP;

  END;
/
```

2. WRITE A PROGRAM TO ACCEPT THE GRADE AND DISPLAY EMPS BELONGS TO THE GRADE

```
SET SERVEROUTPUT ON;

DECLARE
   v_grade NUMBER := &Enter_Grade;
   CURSOR emp_cursor IS
      SELECT e.empno, e.ename, e.sal, g.grade
      FROM emp e
      JOIN salgrade g ON e.sal BETWEEN g.losal AND g.hisal
      WHERE g.grade = v_grade;

BEGIN
   FOR emp_rec IN emp_cursor LOOP
      DBMS_OUTPUT.PUT_LINE('Emp No: ' || emp_rec.empno ||
                           ', Name: ' || emp_rec.ename ||
                           ', Salary: ' || emp_rec.sal ||
                           ', Grade: ' || emp_rec.grade);
   END LOOP;

    END;
/
```