1. Consider a relational database as given below
Person (driver-id, name, address)
Car (license, model, year)
Accident (report-number, date, location)
Owns (driver-id, license)
Participated (driver-id, car, report-number, damage-amount)

Solve the following query.
i. Find the total number of people who owned cars that were involved in accidents in1989

```
select   count (distinct name)
from     accident, participated, person
where    accident.report-number = participated.report-number
and      participated.driver-id = person.driver-id
and      date between date '1989-00-00' and date '1989-12-31'
```

ii. Find the number of accidents in which the cars belonging to "John Smith" were Involved.

```
select   count (distinct *)
from     accident
where    exists
         (select *
          from participated, person
          where participated.driver-id = person.driver-id
                and person.name = 'John Smith'
                and accident.report-number = participated.report-number)
```

iii. Update the damage amount for the car with license number "AABB2000"in the accident with report number "AR2197" to $3000

```
update participated
set damage-amount = 3000
where report-number = "AR2197" and driver-id in
      (select driver-id
       from owns
       where license = "AABB2000")
```

2. Employee (EMP name, street, city)
Works (emp name, company name, salary)
Company (company-name, city)

Manager (emp name, manager-name)
Give an expression in sql for each of the following:

i. Find those companies whose employees earn a higher salary, on average than an average salary at ABC ltd.

```
select company-name
from works
group by company-name
having avg (salary) > (select avg (salary)
from works
where company-name= 'ABC ltd');
```

ii. Find all the employees who live in same street and in the same city as their managers.

```
select P.employee-name
from employee P, employee R, manages M
where P.employee-name = M.employee-name and
        M.manager-name = R.employee-name and
        P.street = R.street and P.city = R.city
```

iii. Find the name of employees who do not work for ABC Ltd.

```
SELECT employee_name
FROM Employee
WHERE company_name != 'ABC Ltd.';
```

iv. Find the company with the most employees.

```
select company-name
from works
group by company-name
having count (distinct employee-name) >= all
    (select count (distinct employee-name)
     from works
     group by company-name)
```

Consider relation schema Employee(Empno, ename, Deptno, Salary)
i List Employee names of Computer Department

```
Select ename
from Employee
where Deptno='Computer';
```

ii Find average salary of each department

```
SELECT Deptno, AVG(Salary) AS avg_salary
FROM Employee
GROUP BY Deptno;
```

iii Find Department name of employee named Amit.

```
select Deptno
from Employee
where ename='Amit';
```

5)Consider following relation
Branch(Bno,Bname,Bcity,Asset)

Customer(Cno,Cname,CCity,Street)
Loan(Loanno,Bname,Amount)
Account(Accno,Bname,Balance)
Borrower(Cno,LoanNO)
Depositor(Cno,Accno)
Write SQL Queries

i. Find loan data,ordered by decreasing amounts ,then increasing loan numbers.

```
SELECT Loanno, Bname, Amount
FROM Loan
ORDER BY Amount DESC, Loanno ASC;
```

ii. Find the pairs of names of different customers who live at the same address but have accounts at different branches.

```
SELECT C1.Cname AS Customer1, C2.Cname AS Customer2
FROM Customer C1, Customer C2, Depositor D1, Depositor D2, Account A1, Account A2
WHERE C1.Cno = D1.Cno
  AND C2.Cno = D2.Cno
  AND D1.Accno = A1.Accno
```

```
  AND D2.Accno = A2.Accno
  AND C1.CCity = C2.CCity
  AND C1.Street = C2.Street
  AND C1.Cno != C2.Cno
  AND A1.Bname != A2.Bname;
```

iii. Find the names and address of customers who have a loan for an amount exceeding 3 times their current balance.

```
SELECT C.Cname, C.CCity, C.Street
FROM Customer C
JOIN Borrower B ON C.Cno = B.Cno
JOIN Loan L ON B.Loanno = L.Loanno
JOIN Depositor D ON C.Cno = D.Cno
JOIN Account A ON D.Accno = A.Accno
WHERE L.Amount > 3 * A.Balance;
```

6. Consider the following relarion table
person(pname,street,city)
works_for(pname,cname,salary)
company(cname,city)
Manager(pname,mname)
Solve the following queries in SQL
i. Find the street and cities of all employee who work for Apollo Live in pune and earn more than 50000

```
SELECT P.street, P.city
FROM person P, works_for W
WHERE P.pname = W.pname
  AND W.cname = 'Apollo'
  AND P.city = 'Pune'
  AND W.salary > 50000;
```

ii. Create a view consisting of manager name and average salary of all employees who work for that manager.

```
CREATE VIEW Manager_Avg_Salary AS
```

```
SELECT M.mname, AVG(W.salary) AS avg_salary
FROM Manager M, works_for W
WHERE M.pname = W.pname
GROUP BY M.mname;
```

7. Consider the relational database
supplier(SID,Sname,Saddress)
Parts(pid,pname,color)
Catalog(sid,pid)
Solve the following queries in SQL
(i) Find the name of all parts whose color is green

```
SELECT pname
FROM Parts
WHERE color = 'green';
```

(ii) Find the name of supplier who supply red parts

```
SELECT S.Sname
FROM Supplier S
JOIN Catalog C ON S.SID = C.sid
JOIN Parts P ON C.pid = P.pid
WHERE P.color = 'red';
```

(iii) Find name of all parts whose cost is more than Rs.25

```
SELECT pname
FROM Parts
WHERE cost > 25;
```

8. Consider the following database
Cricket_Player(p_id,name,address)
Matches(Match_code,match_date,match_place)

Score(p_id,match_code,score)
Write SQL Queries
(i) List player name, match_date, match_place and score of each player

```
SELECT C.name, M.match_date, M.match_place, S.score
FROM Cricket_Player C
JOIN Score S ON C.p_id = S.p_id
```

```
JOIN Matches M ON S.match_code = M.Match_code;
```

(ii) List all those player whose maximum score is higher than 50

```
SELECT C.name
FROM Cricket_Player C
JOIN Score S ON C.p_id = S.p_id
GROUP BY C.name
HAVING MAX(S.score) > 50;
```

9. Create a database schema with the following table
Marks)
Stud-Marks(Roll, name, total_marks)
Result(Roll,Name,Class).
Write a PLSQL function to calculate Class of the student and to enter into the Result table

```
CREATE OR REPLACE FUNCTION calculate_class (roll_num NUMBER)
RETURN VARCHAR2 IS
   total_marks NUMBER;
   student_class VARCHAR2(10);
BEGIN
   -- Get the total marks from the Stud_Marks table
   SELECT total_marks INTO total_marks
   FROM Stud_Marks
   WHERE Roll = roll_num;

   -- Determine the class based on total_marks
   IF total_marks >= 90 THEN
      student_class := 'A';
   ELSIF total_marks >= 75 THEN
      student_class := 'B';
   ELSIF total_marks >= 60 THEN
      student_class := 'C';
   ELSE
      student_class := 'D';
   END IF;

   -- Insert the result into the Result table
```

```
    INSERT INTO Result (Roll, Name, Class)
    SELECT Roll, Name, student_class
    FROM Stud_Marks
    WHERE Roll = roll_num;

    RETURN student_class;
END;
```

10. Create two tables O_Roll (Rollno,Name,DOB,Phone,address)
N_Roll (Rollno,Name,DOB,Phone,address)
Write a PLSQL block using cursor to merge records from O_Roll table with that of N_Roll in such a way duplicate records are to be eliminated.

```
DECLARE
    CURSOR roll_cursor IS
        SELECT * FROM O_Roll;
    v_rollno O_Roll.Rollno%TYPE;
    v_name O_Roll.Name%TYPE;
    v_dob O_Roll.DOB%TYPE;
    v_phone O_Roll.Phone%TYPE;
    v_address O_Roll.address%TYPE;
BEGIN
    FOR record IN roll_cursor LOOP
        -- Check if the record already exists in N_Roll
        SELECT COUNT(*) INTO v_rollno
        FROM N_Roll
        WHERE Rollno = record.Rollno;

        -- If the record doesn't exist, insert it
        IF v_rollno = 0 THEN
            INSERT INTO N_Roll (Rollno, Name, DOB, Phone, address)
            VALUES (record.Rollno, record.Name, record.DOB, record.Phone, record.address);
        END IF;
    END LOOP;
END;
```

11. Consider Library database with the following schema
Books (AccNo, Title, Author, Publisher, Count).
Library_Audit (AccNo, Title, Author, Publisher, Count).

Create a before trigger to insert records into Library_Audit table if there is deletion in the Books table.

```
CREATE OR REPLACE TRIGGER audit_books_before_delete
BEFORE DELETE ON Books
FOR EACH ROW
BEGIN
   INSERT INTO Library_Audit (AccNo, Title, Author, Publisher, Count)
   VALUES (:OLD.AccNo, :OLD.Title, :OLD.Author, :OLD.Publisher, :OLD.Count);
END;
```

12. What is PL/SQL? Explain the advantages of using PL/SQL.

PL/SQL (Procedural Language/SQL) is Oracle's procedural extension to SQL. It integrates SQL with procedural programming features like loops, conditions, and variables, allowing for more complex and efficient data manipulation and control.

Advantages of using PL/SQL:

1. **Enhanced Performance**: PL/SQL reduces network traffic by allowing multiple SQL statements to be executed in a single block.
2. **Improved Maintainability**: Code can be organized into procedures and functions, making it easier to maintain and debug.
3. **Error Handling**: PL/SQL provides robust error handling with exceptions, which can catch and manage runtime errors.
4. **Modularity**: It supports modular programming, enabling the reuse of code through stored procedures and functions.

13. What is the difference between implicit and explicit cursor?

Implicit Cursor:

- Automatically created by Oracle for single SQL statements that return only one row.
- Cannot be explicitly controlled or named.

Example: SELECT emp_name INTO v_name FROM employees WHERE emp_id = 101;

Explicit Cursor:

- Defined by the programmer for queries that return multiple rows.
- Provides more control with operations like OPEN, FETCH, and CLOSE.

Example:
plsql
Copy code

```
DECLARE
   CURSOR emp_cursor IS
      SELECT emp_name FROM employees;
   v_name employees.emp_name%TYPE;
BEGIN
   OPEN emp_cursor;
   FETCH emp_cursor INTO v_name;
   CLOSE emp_cursor;
END;
```

14. What is the use of stored procedures and functions in PL/SQL?

Stored Procedures:

- Stored procedures are named PL/SQL blocks that perform a specific task and can be executed by applications or other PL/SQL code.
- **Uses**: Encapsulate business logic, perform batch processing, and manage transactions.

Functions:

- Functions are similar to procedures but return a single value.
- **Uses**: Compute and return values based on input parameters, used within SQL queries and PL/SQL blocks.

15. What are different cursor attributes?

Cursor Attributes are used to get information about the state of the cursor:

1. **%FOUND**: Returns TRUE if the last fetch returned a row; otherwise FALSE.
2. **%NOTFOUND**: Returns TRUE if the last fetch did not return a row; otherwise FALSE.
3. **%ISOPEN**: Returns TRUE if the cursor is open; otherwise FALSE.
4. **%ROWCOUNT**: Returns the number of rows fetched so far.

16. What are various parameters used in PL/SQL Procedure?

Parameters in PL/SQL Procedures:

1. **IN**: The parameter is passed into the procedure. It is read-only.
2. **OUT**: The parameter is used to return values from the procedure. It is write-only.
3. **IN OUT**: The parameter is used for both input and output. It is read-write.

Example:

```
PROCEDURE example_proc (p_in IN NUMBER, p_out OUT NUMBER, p_inout IN OUT
NUMBER) IS
BEGIN
   p_out := p_in * 2;
   p_inout := p_inout + 10;
END;
```

17. What is the difference between statement and row-level trigger?

Statement-Level Trigger:

- Executes once for each triggering statement, regardless of the number of rows affected.
- Example: A trigger that logs changes whenever a table is updated.

Row-Level Trigger:

- Executes once for each row affected by the triggering statement.
- Example: A trigger that updates a history table for each row updated in the target table.

18. What is the advantage of using Embedded SQL?

Embedded SQL:

- Embedded SQL is SQL code written within a host programming language (e.g., C, Java).
- **Advantages**:
    - **Seamless Integration**: Directly integrates SQL operations within host programs.
    - **Improved Performance**: Reduces network traffic by combining SQL and application logic.
    - **Enhanced Functionality**: Allows complex processing and interaction with databases from within applications.

19. What are different types of joins in SQL? Explain with examples.

1. **INNER JOIN**: Returns records with matching values in both tables.

```sql
SELECT A.name, B.department
FROM Employees A
INNER JOIN Departments B ON A.dept_id = B.dept_id;
```

2. **LEFT JOIN (or LEFT OUTER JOIN)**: Returns all records from the left table, and matched records from the right table. Unmatched records in the right table will have NULLs.

```sql
SELECT A.name, B.department
FROM Employees A
LEFT JOIN Departments B ON A.dept_id = B.dept_id;
```

3. **RIGHT JOIN (or RIGHT OUTER JOIN)**: Returns all records from the right table, and matched records from the left table. Unmatched records in the left table will have NULLs.

```sql
SELECT A.name, B.department
FROM Employees A
RIGHT JOIN Departments B ON A.dept_id = B.dept_id;
```

4. **FULL JOIN (or FULL OUTER JOIN)**: Returns all records when there is a match in either left or right table. Unmatched records will have NULLs.

```sql
SELECT A.name, B.department
FROM Employees A
FULL JOIN Departments B ON A.dept_id = B.dept_id;
```

20. What is the difference between SQL and PL/SQL?

Aspect	SQL	PL/SQL
Definition	SQL (Structured Query Language) is a standard language for querying and manipulating relational databases.	PL/SQL (Procedural Language/SQL) is an extension of SQL that adds procedural programming capabilities.
Type	Declarative language used for defining and manipulating data.	Procedural language used for writing complex procedures, functions, and triggers.
Functionality	Executes single SQL statements to perform operations like SELECT, INSERT, UPDATE, DELETE.	Combines SQL with procedural constructs (like loops, conditions) to handle complex logic and batch operations.
Error Handling	Basic error handling using SQL exceptions.	Advanced error handling with detailed exception handling and custom error messages.

21. Write with example aggregate functions in SQL.

Aggregate Functions:

1. COUNT(): Returns the number of rows.

```sql
SELECT COUNT(*) FROM Employees;
```

2. SUM(): Returns the sum of a numeric column.

```sql
SELECT SUM(salary) FROM Employees;
```

3. AVG(): Returns the average value of a numeric column.

```sql
SELECT AVG(salary) FROM Employees;
```

4. MAX(): Returns the maximum value of a column.

```sql
SELECT MAX(salary) FROM Employees;
```

5. MIN(): Returns the minimum value of a column.

```sql
SELECT MIN(salary) FROM Employees;
```

22. Illustrate GROUP BY and HAVING clauses with examples.

GROUP BY:

- Used to group rows that have the same values in specified columns into summary rows.

HAVING:

- Used to filter groups based on a condition (similar to WHERE but for groups).

Example:

```
SELECT department, AVG(salary) AS avg_salary
FROM Employees
GROUP BY department
HAVING AVG(salary) > 50000;
```

In this example:

- **GROUP BY** groups employees by department.
- **HAVING** filters out departments where the average salary is 50,000 or less.