**1..Consider the employee database:**

**Employee (employee\_name, street, city,Id,Mobile no,)**

**Works (employee\_name, company\_name, salary)**

**Company (company\_name, city)**

**Manages (employee\_name, manager\_name).**

**Give an expression in SQL for each of the following: Table,**

**View, Index, Sequence, Synonym, Drop, commit**

A:

CREATE TABLE Employee (

employee\_name VARCHAR(50),

street VARCHAR(100),

city VARCHAR(50),

Id INT,

Mobile\_no VARCHAR(20)

);

CREATE TABLE Works (

employee\_name VARCHAR(50),

company\_name VARCHAR(50),

salary DECIMAL(10, 2)

);

CREATE TABLE Company (

company\_name VARCHAR(50),

city VARCHAR(50)

);

CREATE TABLE Manages (

employee\_name VARCHAR(50),

manager\_name VARCHAR(50)

);

**View:**

CREATE VIEW EmployeeView AS

SELECT employee\_name, street, city, Id, Mobile\_no

FROM Employee;

Select \*from EmployeeView; (to show the view)

**Index creation:**

CREATE INDEX idx\_employee\_name ON Employee (employee\_name);

Show index from Employee;(to show)

**Sequence Creation:**

CREATE SEQUENCE employee\_id\_seq

START WITH 1

INCREMENT BY 1;

**Synonym creation:**

CREATE SYNONYM emp FOR Employee;

**Drop:**

DROP TABLE Employee;

DROP VIEW EmployeeView;

DROP INDEX idx\_employee\_name;

DROP SEQUENCE employee\_id\_seq;

DROP SYNONYM emp;

**Commit:**

COMMIT;

**2. Consider the employee database:**

**Employee (employee\_name, street, city,Id,Mobile no,)**

**Works (employee\_name, company\_name, salary)**

**Company (company\_name, city)**

**Manages (employee\_name, manager\_name).**

**Give an expression in SQL for each of the following:Insert,**

**Select, Update, Delete with operators, functions, and set**

**Operator**.

**A:**

**Insert:**

INSERT INTO Employee (employee\_name, street, city, Id, Mobile\_no)

VALUES ('John Doe', '123 Main St', 'New York', 1, '123-456-7890');

**Select:**

SELECT employee\_name, city

FROM Employee

WHERE Id = 1;

**Update:**

UPDATE Employee

SET city = 'London'

WHERE employee\_name = 'John Doe';

**Delete with Operators, Functions, and Set Operator:**

DELETE FROM Employee

WHERE Id = (SELECT MAX(Id) FROM Employee)

AND city = 'London';

3. Consider the employee database:

Employee (employee\_name, street, city,Id,Mobile no,)

Works (employee\_name, company\_name, salary)

Company (company\_name, city)

Manages (employee\_name, manager\_name).

Give an expression in SQL for each of the following using SQL

DML statements: Join, Sub-Query and View.

**A:**

**Join:**

SELECT Employee.employee\_name, Works.company\_name, Works.salary

FROM Employee

JOIN Works ON Employee.employee\_name = Works.employee\_name;

**Sub-Query:**

SELECT employee\_name

FROM Employee

WHERE employee\_name IN (

SELECT employee\_name

FROM Works

WHERE company\_name = 'Company XYZ');

**View:**

CREATE VIEW EmployeeManagerView AS

SELECT e.employee\_name, e.company\_name, m.manager\_name

FROM Employee e

JOIN Manages m ON e.employee\_name = m.employee\_name;

**4. Consider the student database:**

**Student (Roll no,name,marks,remark) update remark of student**

**as pass or fail base on marks if marks &gt;=40 pass**

**If marks &lt;=40 fail**

**Update student set remark =’pass’where roll no=2;**

**PL/SQL block to demonstrate use of stored procedure**

A:

-- Create the Student table

CREATE TABLE Student (

Roll\_no INT,

Name VARCHAR(50),

Marks INT,

Remark VARCHAR(10)

);

-- Insert sample data into the Student table

INSERT INTO Student (Roll\_no, Name, Marks, Remark)

VALUES (1, 'John', 75, NULL);

INSERT INTO Student (Roll\_no, Name, Marks, Remark)

VALUES (2, 'Jane', 35, NULL);

-- Create the stored procedure to update the remark based on marks

CREATE OR REPLACE PROCEDURE UpdateStudentRemark (p\_Roll\_no INT)

IS

v\_Marks INT;

BEGIN

-- Retrieve the marks for the given Roll\_no

SELECT Marks INTO v\_Marks

FROM Student

WHERE Roll\_no = p\_Roll\_no;

-- Update the remark based on marks

IF v\_Marks >= 40 THEN

UPDATE Student

SET Remark = 'pass'

WHERE Roll\_no = p\_Roll\_no;

ELSE

UPDATE Student

SET Remark = 'fail'

WHERE Roll\_no = p\_Roll\_no;

END IF;

-- Commit the transaction

COMMIT;

END;

/

**5.Consider the student database:**

**Student (Roll no,name,address,marks,grade) used deleted**

**Trigger.**

**1.Before delete student record cannot be delete if**

**active if active is true**

**2.After delete when student record is deleted then update left**

**student grade**

**1.Before Delete Trigger:**

CREATE OR REPLACE TRIGGER PreventDeleteActiveStudent

BEFORE DELETE ON Student

FOR EACH ROW

BEGIN

IF :OLD.active = 'true' THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Cannot delete an active student');

END IF;

END;

/

**2. After delete Trigger:**

CREATE OR REPLACE TRIGGER UpdateLeftStudentGrade

AFTER DELETE ON Student

FOR EACH ROW

BEGIN

UPDATE Student

SET grade = 'Left'

WHERE Roll\_no = :OLD.Roll\_no;

END;

/

**7.Consider the following database:**

**Employee (emp\_no, name, skill, pay\_rate)**

**Position (posting\_no, skill)**

**Duty-allocation (posting\_no, emp\_no, day, shift)**

**Find SQL queries for the following:**

**1. Get complete details from Duty\_allocation**

**2. Get duty allocation details for Emp\_no 123461 for the month**

**of April 1986**

**3. Find the shift details for employee ‘XYZ’:**

**A:**

2. Get duty allocation details for Emp\_no 123461 for the month of April 1986: To retrieve duty allocation details for a specific employee (Emp\_no 123461) for the month of April 1986, you can use the following SQL query:

SELECT \*

FROM Duty\_allocation

WHERE emp\_no = 123461

AND EXTRACT(MONTH FROM day) = 4

AND EXTRACT(YEAR FROM day) = 1986;

3. Find the shift details for employee 'XYZ': To find the shift details for a specific employee (e.g., 'XYZ'), you can use the following SQL query by joining the Duty\_allocation and Employee tables:

SELECT da.\*

FROM Duty\_allocation da

JOIN Employee e ON da.emp\_no = e.emp\_no

WHERE e.name = 'XYZ';

**8.Consider the given database :**

**Project (project\_id, proj\_name, chief\_arch)**

**Employee (Emp\_id, Emp\_name)**

**Assigned\_To (Project\_id, emp\_id)**

**Find the SQL queries for the following statements :**

**1. Get employee number of employees working on project**

**2. Get details of employees working on project C353**

**3. Obtain details of employees working on both C353 and C354.**

**A:**

Get employee number of employees working on a project:

SELECT emp\_id

FROM Assigned\_To

WHERE project\_id = 'project\_id';

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

//option

SELECT Employee.Emp\_id

FROM Employee

JOIN Assigned\_To ON Employee.Emp\_id = Assigned\_To.emp\_id

WHERE Assigned\_To.Project\_id = <project\_id>;

//

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

Get details of employees working on project C353:

SELECT e.Emp\_id, e.Emp\_name

FROM Employee e

JOIN Assigned\_To at ON e.Emp\_id = at.emp\_id

WHERE at.project\_id = 'C353';

Obtain details of employees working on both C353 & C354:

SELECT Emp\_id, Emp\_name

FROM Employee

WHERE EXISTS (

SELECT \*

FROM Assigned\_To

WHERE Assigned\_To.emp\_id = Employee.Emp\_id

AND project\_id = 'C353'

)

AND EXISTS (

SELECT \*

FROM Assigned\_To

WHERE Assigned\_To.emp\_id = Employee.Emp\_id

AND project\_id = 'C354'

);

//option main

SELECT Employee.\*

FROM Employee

WHERE Employee.Emp\_id IN (

SELECT emp\_id

FROM Assigned\_To

WHERE Project\_id = 'C353'

) AND Employee.Emp\_id IN (

SELECT emp\_id

FROM Assigned\_To

WHERE Project\_id = 'C354'

);

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

**9.Consider the employee database:**

**Employee (employee\_name, street, city,Id,Mobile no,)**

**Works (employee\_name, company\_name, salary)**

**Company (company\_name, city)**

**Manages (employee\_name, manager\_name).**

**1. Find the name of all employees who work for FBC**

**2. Find the name and cities of all employee who works for FBC**

**3. Find the names, street address, and cities of residence of all**

**employees who work for FBC and earn more than $ 10,000.**

**A:**

Find the name of all employees who work for FBC:

SELECT e.employee\_name

FROM Employee e

JOIN Works w ON e.employee\_name = w.employee\_name

WHERE w.company\_name = 'FBC';

Find the name and cities of all employees who work for FBC:

SELECT e.employee\_name, e.city

FROM Employee e

JOIN Works w ON e.employee\_name = w.employee\_name

JOIN Company c ON w.company\_name = c.company\_name

WHERE c.company\_name = 'FBC';

Find the names, street address, and cities of residence of all employees who work for FBC and earn more than $10,000:

SELECT e.employee\_name, e.street, e.city

FROM Employee e

JOIN Works w ON e.employee\_name = w.employee\_name

JOIN Company c ON w.company\_name = c.company\_name

WHERE c.company\_name = 'FBC'

AND w.salary > 10000;

1)SELECT employee\_name

FROM Works

WHERE company\_name = 'FBC';

2)SELECT E.employee\_name, E.city

FROM Employee E

JOIN Works W ON E.employee\_name = W.employee\_name

WHERE W.company\_name = 'FBC';

3) SELECT E.employee\_name, E.street, E.city

FROM Employee E

JOIN Works W ON E.employee\_name = W.employee\_name

WHERE W.company\_name = 'FBC' AND W.salary > 10000;

Write a PL/SQL block to print employee name and salary of all employees above given age.

DECLARE

v\_age\_threshold NUMBER := 30;

BEGIN

FOR emp IN (SELECT employee\_name, salary FROM Employee WHERE age > v\_age\_threshold) LOOP

DBMS\_OUTPUT.PUT\_LINE('Employee Name: ' || emp.employee\_name || ', Salary: ' || emp.salary);

END LOOP;

END;

/

Write a PL/SQL block to print employee name and salary and show salary increment of 10% crossing age of 60. of all employees

DECLARE

v\_salary\_increment\_pct NUMBER := 10;

BEGIN

FOR emp IN (SELECT employee\_name, age, salary FROM Employee) LOOP

DBMS\_OUTPUT.PUT\_LINE('Employee Name: ' || emp.employee\_name || ', Salary: ' || emp.salary);

IF emp.age > 60 THEN

DECLARE

v\_new\_salary NUMBER;

BEGIN

v\_new\_salary := emp.salary + (emp.salary \* (v\_salary\_increment\_pct / 100));

DBMS\_OUTPUT.PUT\_LINE('Salary Increment: ' || v\_salary\_increment\_pct || '%');

DBMS\_OUTPUT.PUT\_LINE('New Salary: ' || v\_new\_salary);

END;

END IF;

END LOOP;

END;

/

Write a implicit cursor to count number of rows updated by update statement.

DECLARE

v\_updated\_rows NUMBER;

BEGIN

UPDATE your\_table

SET your\_column = 'new\_value'

WHERE your\_condition;

v\_updated\_rows := SQL%ROWCOUNT;

DBMS\_OUTPUT.PUT\_LINE('Number of Rows Updated: ' || v\_updated\_rows);

END;

/