Class : TY Comp

Subject: DBMS
Practical Exam Sample Question Bank

SQL

SQL Queries:

Consider the given Database Schema: for problem statement 1

Dept (Deptno , Name , Location, Managerempid)

Employees (name, empid, address, city, dob, date_of_joining,gender, salary, deptno)

Gender must take value ‘M’ or ‘F’.

Project(Projectid, title,city).

Works (empid , Projectid, total_hrs_worked);

Dependant(empid, name_of_dependant, age, relation)

Primary Key is underlined.

1. Develop DDL to implement the above schema enforcing primary key, check constraints (F and M for gender) and foreign key constraints.

```sql
-- Create Dept table
CREATE TABLE Dept (
    Deptno INT PRIMARY KEY,
    Name VARCHAR(100),
    Location VARCHAR(100),
    Managerempid INT
);

-- Create Employees table
CREATE TABLE Employees (
    name VARCHAR(100),
    empid INT PRIMARY KEY,
    address VARCHAR(255),
    city VARCHAR(100),
    dob DATE,
    date_of_joining DATE,
    gender CHAR(1) CHECK (gender IN ('M', 'F')),
    salary DECIMAL(10, 2),
    deptno INT,
    FOREIGN KEY (deptno) REFERENCES Dept(Deptno)
);

-- Create Project table
CREATE TABLE Project (
    Projectid INT PRIMARY KEY,
```

```
    title VARCHAR(100),
    city VARCHAR(100)
);

-- Create Works table
CREATE TABLE Works (
    empid INT,
    Projectid INT,
    total_hrs_worked INT,
    PRIMARY KEY (empid, Projectid),
    FOREIGN KEY (empid) REFERENCES Employees(empid),
    FOREIGN KEY (Projectid) REFERENCES Project(Projectid)
);

-- Create Dependant table
CREATE TABLE Dependant (
    empid INT,
    name_of_dependant VARCHAR(100),
    age INT,
    relation VARCHAR(50),
    PRIMARY KEY (empid, name_of_dependant),
    FOREIGN KEY (empid) REFERENCES Employees(empid)
);
```

2. Insert data in each table .

```
-- Insert data into Dept table
INSERT INTO Dept (Deptno, Name, Location, Managerempid)
VALUES (1, 'HR', 'Building 1', 101),
       (2, 'IT', 'Building 2', 102),
       (3, 'Finance', 'Building 1', 103);

-- Insert data into Employees table
INSERT INTO Employees (name, empid, address, city, dob, date_of_joining, gender,
salary, deptno)
VALUES ('John Doe', 101, '123 Street A', 'City X', '1980-01-15', '2018-02-01', 'M',
50000, 1),
       ('Sarah Lee', 102, '456 Street B', 'City Y', '1990-05-20', '2019-06-15', 'F',
60000, 2),
       ('Sam Smith', 103, '789 Street C', 'City Z', '1985-10-05', '2020-08-10', 'M',
70000, 3);

-- Insert data into Project table
INSERT INTO Project (Projectid, title, city)
VALUES (1, 'Project Alpha', 'City X'),
       (2, 'Project Beta', 'City Y'),
       (3, 'Project Gamma', 'City Z');

-- Insert data into Works table
INSERT INTO Works (empid, Projectid, total_hrs_worked)
VALUES (101, 1, 120),
```

```
       (102, 2, 150),
       (103, 3, 200);

-- Insert data into Dependant table
INSERT INTO Dependant (empid, name_of_dependant, age, relation)
VALUES (101, 'Alice Doe', 15, 'Daughter'),
       (102, 'Bob Lee', 10, 'Son'),
       (103, 'Eve Smith', 5, 'Daughter');
```

3. Add column Mobile number in employee table

```
ALTER TABLE Employees
ADD COLUMN mobile_number VARCHAR(15);
```

4. Update mobile numbers for each employee.

```
UPDATE Employees
SET mobile_number = '1234567890'
WHERE empid = 101;

UPDATE Employees
SET mobile_number = '2345678901'
WHERE empid = 102;

UPDATE Employees
SET mobile_number = '3456789012'
WHERE empid = 103;
```

Develop DML to Populate Database

5. Develop a SQL query to list employees having name starting with ‘S’

```
SELECT * FROM Employees
WHERE name LIKE 'S%';
```

6. Develop a SQL query to list department having location ‘Building 1’

```
SELECT * FROM Dept
WHERE Location = 'Building 1';
```

7. Develop a SQL query to list employee having joining year 2019 to 2020

```
SELECT * FROM Employees
WHERE YEAR(date_of_joining) BETWEEN 2019 AND 2020;
```

SQL Queries:

Consider the given Database Schema: for problem statement 1

Dept (Deptno , Name , Location, Managerempid)

Employees (name, empid, address, city, dob, date_of_joining,gender, salary, deptno)

Gender must take value ‘M’ or ‘F’.

Project(Projectid, title,city).

Works (empid , Projectid, total_hrs_worked);

Dependant(empid, name_of_dependant, age, relation)

Primary Key is underlined.

1. Develop DDL to implement the above schema enforcing primary key, check constraints (F and M for gender) and foreign key constraints.
2. Insert data in each table .
3. Add column Mobile number in employee table
4. Update mobile numbers for each employee.
5. Develop DML to Populate Database
6. Develop a SQL query to list employees having name starting with ‘S’
7. Develop a SQL query to list department having location ‘Building 1’

SQL Queries:

Consider the given Database Schema:

The "Customers" table contains information about the company's customers, including their customer ID, name, email, and country.

The "Orders" table contains information about the orders made by the customers. Each order has an order ID, customer ID (which corresponds to the customer who made the order), order date, and total order amount.

1. Find order ids having total order amount more than 100000

```sql
SELECT order_id
FROM Orders
WHERE total_order_amount > 100000;
```

2. Find the Count number of customers from india

```sql
SELECT COUNT(*) AS count_of_customers
FROM Customers
WHERE country = 'India';
```

3. Find all orders made in Jan 2022

```sql
SELECT order_id, customer_id, order_date, total_order_amount
FROM Orders
WHERE order_date BETWEEN '2022-01-01' AND '2022-01-31';
```

4. Display all customers names starting with ‘S’

```sql
SELECT customer_name
FROM Customers
WHERE customer_name LIKE 'S%';
```

5. How many orders have been placed by each customer? Provide a result that shows the

customer name, email, and the count of orders.

```
SELECT c.customer_name, c.email, COUNT(o.order_id) AS number_of_orders
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_name, c.email;
```

6. What is the total revenue generated by each country? Provide a result that shows the country name and the total revenue.

```
SELECT c.country, SUM(o.total_order_amount) AS total_revenue
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
GROUP BY c.country;
```

Consider the following schema
account(acc-no,branch-name,balance)
depositor(cust-name,acc-no)
borrower (cust-name, loan-no)
loan (loan - no, branch - name, amount)
Write following queries using SQL

1. Create tables using proper primary keys

```
CREATE TABLE account (
    acc_no INT PRIMARY KEY,
    branch_name VARCHAR(50),
    balance DECIMAL(15, 2)
);

CREATE TABLE depositor (
    cust_name VARCHAR(100),
    acc_no INT,
    PRIMARY KEY (cust_name, acc_no),
    FOREIGN KEY (acc_no) REFERENCES account(acc_no)
);

CREATE TABLE borrower (
    cust_name VARCHAR(100),
    loan_no INT,
    PRIMARY KEY (cust_name, loan_no),
    FOREIGN KEY (loan_no) REFERENCES loan(loan_no)
);

CREATE TABLE loan (
    loan_no INT PRIMARY KEY,
    branch_name VARCHAR(50),
    amount DECIMAL(15, 2)
);
```

2. Update information of particular customer

```
UPDATE depositor
```

```
SET acc_no = 123456  -- change to a valid account number
WHERE cust_name = 'Customer Name'; -- change to the customer name you want to update
```

3. Find the customers having loan less than 1 lac

```
SELECT b.cust_name
FROM borrower b
JOIN loan l ON b.loan_no = l.loan_no
WHERE l.amount < 100000;
```

4. Display account number and customer name starting with ‘P’

```
SELECT d.acc_no, d.cust_name
FROM depositor d
WHERE d.cust_name LIKE 'P%';
```

5. Display name of the depositor with balance

```
SELECT d.cust_name, a.balance
FROM depositor d
JOIN account a ON d.acc_no = a.acc_no;
```

6. Find names of all customers who have a loan at the ‘Redwood branch’.

```
SELECT b.cust_name
FROM borrower b
JOIN loan l ON b.loan_no = l.loan_no
WHERE l.branch_name = 'Redwood branch';
```

7. Find all customers who have an account and loan or both

```
SELECT DISTINCT d.cust_name
FROM depositor d
JOIN borrower b ON d.cust_name = b.cust_name
UNION
SELECT d.cust_name
FROM depositor d
UNION
SELECT b.cust_name
FROM borrower b;
```

Physician (reg_no, name, tel_no, city)
Patient (p_name, street, city)
visit(p_name, reg_no, date_of _visit, fees)

1. Create tables and insert values

```
CREATE TABLE Physician (
    reg_no INT PRIMARY KEY,
    name VARCHAR(100),
    tel_no VARCHAR(15),
    city VARCHAR(50)
);
```

```sql
CREATE TABLE Patient (
    p_name VARCHAR(100) PRIMARY KEY,
    street VARCHAR(100),
    city VARCHAR(50)
);

CREATE TABLE visit (
    p_name VARCHAR(100),
    reg_no INT,
    date_of_visit DATE,
    fees DECIMAL(10, 2),
    PRIMARY KEY (p_name, reg_no, date_of_visit),
    FOREIGN KEY (p_name) REFERENCES Patient(p_name),
    FOREIGN KEY (reg_no) REFERENCES Physician(reg_no)
);

-- Insert sample values for demonstration
INSERT INTO Physician (reg_no, name, tel_no, city) VALUES (1, 'Dr. Sharma',
'1234567890', 'Mumbai');
INSERT INTO Physician (reg_no, name, tel_no, city) VALUES (2, 'Dr. Mehta',
'0987654321', 'Nashik');

INSERT INTO Patient (p_name, street, city) VALUES ('Dipak', '123 Main St', 'Nashik');
INSERT INTO Patient (p_name, street, city) VALUES ('Mina', '456 Park Ave', 'Pune');

INSERT INTO visit (p_name, reg_no, date_of_visit, fees) VALUES ('Dipak', 1, '2017-07-
13', 6000);
INSERT INTO visit (p_name, reg_no, date_of_visit, fees) VALUES ('Mina', 2, '2017-07-
13', 4500);
```

2. delete entry of p_name ‘Dipak’

```sql
DELETE FROM Patient
WHERE p_name = 'Dipak';
```

3. Get the patient details of ‘Nashik’ city

```sql
SELECT *
FROM Patient
WHERE city = 'Nashik';
```

4. Count number of physician of ‘Mumbai’

```sql
SELECT COUNT(*) AS physician_count
FROM Physician
WHERE city = 'Mumbai';
```

5. Display a list of p_name in ascending order

```sql
SELECT p_name
FROM Patient
```

```
ORDER BY p_name ASC;
```

6. Get the patient name and fees, who paid fees > 5000

```
SELECT p_name, fees
FROM visit
WHERE fees > 5000;
```

7. Find the name and city of patient who visited physician on 13th july 2017

```
SELECT p.p_name, p.city
FROM Patient p
JOIN visit v ON p.p_name = v.p_name
WHERE v.date_of_visit = '2017-07-13';
```

8. Get the name of the physician and no of patient visited to him

```
SELECT ph.name, COUNT(v.p_name) AS patient_count
FROM Physician ph
JOIN visit v ON ph.reg_no = v.reg_no
GROUP BY ph.name;
```

9. Get date wise fees collected

```
SELECT date_of_visit, SUM(fees) AS total_fees
FROM visit
GROUP BY date_of_visit;
```

10. Display a Physician names who treated ‘Mina’

```
SELECT ph.name
FROM Physician ph
JOIN visit v ON ph.reg_no = v.reg_no
WHERE v.p_name = 'Mina';
```

Consider the given Database Schema: for problem statement

Dept (Deptno , Name , Location, Managerempid)

Employees (name, empid, address, city, dob, date_of_joining,gender, salary, deptno)

Gender must take value ‘M’ or ‘F’.

Project(Projectid, title,city).

Works (empid , Projectid, total_hrs_worked);

Dependant(empid, name_of_dependant, age, relation)

Primary Key is underlined.

1. Develop DDL to implement the above schema enforcing primary key, check constraints (F and M for gender) and foreign key constraints.

```
CREATE TABLE Dept (
    Deptno INT PRIMARY KEY,
    Name VARCHAR(100),
```

```
    Location VARCHAR(50),
    Managerempid INT
);

CREATE TABLE Employees (
    name VARCHAR(100),
    empid INT PRIMARY KEY,
    address VARCHAR(150),
    city VARCHAR(50),
    dob DATE,
    date_of_joining DATE,
    gender CHAR(1) CHECK (gender IN ('M', 'F')),
    salary DECIMAL(10, 2),
    deptno INT,
    FOREIGN KEY (deptno) REFERENCES Dept(Deptno)
);

CREATE TABLE Project (
    Projectid INT PRIMARY KEY,
    title VARCHAR(100),
    city VARCHAR(50)
);

CREATE TABLE Works (
    empid INT,
    Projectid INT,
    total_hrs_worked INT,
    PRIMARY KEY (empid, Projectid),
    FOREIGN KEY (empid) REFERENCES Employees(empid),
    FOREIGN KEY (Projectid) REFERENCES Project(Projectid)
);

CREATE TABLE Dependant (
    empid INT,
    name_of_dependant VARCHAR(100),
    age INT,
    relation VARCHAR(50),
    PRIMARY KEY (empid, name_of_dependant),
    FOREIGN KEY (empid) REFERENCES Employees(empid)
);
```

2. Insert data in each table

```
-- Sample inserts for demonstration

INSERT INTO Dept (Deptno, Name, Location, Managerempid) VALUES (1, 'HR', 'Mumbai',
101);
INSERT INTO Dept (Deptno, Name, Location, Managerempid) VALUES (2, 'Engineering',
'Pune', 102);

INSERT INTO Employees (name, empid, address, city, dob, date_of_joining, gender,
```

```
salary, deptno)
VALUES ('John Doe', 101, '123 Elm St', 'Mumbai', '1980-05-10', '2005-07-15', 'M',
75000, 1);

INSERT INTO Employees (name, empid, address, city, dob, date_of_joining, gender,
salary, deptno)
VALUES ('Jane Smith', 102, '456 Oak St', 'Pune', '1985-03-20', '2010-09-10', 'F',
80000, 2);

INSERT INTO Project (Projectid, title, city) VALUES (1, 'Testing Project', 'Pune');
INSERT INTO Project (Projectid, title, city) VALUES (2, 'Development Project',
'Mumbai');

INSERT INTO Works (empid, Projectid, total_hrs_worked) VALUES (101, 1, 100);
INSERT INTO Works (empid, Projectid, total_hrs_worked) VALUES (102, 2, 200);

INSERT INTO Dependant (empid, name_of_dependant, age, relation) VALUES (101, 'Emily
Doe', 15, 'Daughter');
INSERT INTO Dependant (empid, name_of_dependant, age, relation) VALUES (102, 'Michael
Smith', 12, 'Son');
```

3. “Testing Project”is canceled so delete that entry from project

```
DELETE FROM Project
WHERE title = 'Testing Project';
```

4. Develop a SQL query to display all employees having salary > 50000

```
SELECT name, empid, salary
FROM Employees
WHERE salary > 50000;
```

5. Develop a SQL query to display all projects of “Pune”

```
SELECT *
FROM Project
WHERE city = 'Pune';
```

6. Develop a SQL query to display all dependants of employee id 102

```
SELECT name_of_dependant, age, relation
FROM Dependant
WHERE empid = 102;
```

7. Develop a SQL query to Find the number of employees working on each project.

```
SELECT p.title, COUNT(w.empid) AS number_of_employees
FROM Project p
JOIN Works w ON p.Projectid = w.Projectid
```

```
GROUP BY p.title;
```

8. Develop a SQL query to find average salary of each department

```
SELECT d.Name AS department_name, AVG(e.salary) AS average_salary
FROM Dept d
JOIN Employees e ON d.Deptno = e.deptno
GROUP BY d.Name;
```

Consider the following schema
account(acc-no,branch-name,balance)
depositor(cust-name,acc-no)
borrower (cust-name, loan-no)
loan (loan - no, branch - name, amount)
Write following queries using SQL

1. Create tables using proper primary keys

```
CREATE TABLE account (
    acc_no INT PRIMARY KEY,
    branch_name VARCHAR(50),
    balance DECIMAL(15, 2)
);

CREATE TABLE depositor (
    cust_name VARCHAR(100),
    acc_no INT,
    PRIMARY KEY (cust_name, acc_no),
    FOREIGN KEY (acc_no) REFERENCES account(acc_no)
);

CREATE TABLE borrower (
    cust_name VARCHAR(100),
    loan_no INT,
    PRIMARY KEY (cust_name, loan_no),
    FOREIGN KEY (loan_no) REFERENCES loan(loan_no)
);

CREATE TABLE loan (
    loan_no INT PRIMARY KEY,
    branch_name VARCHAR(50),
    amount DECIMAL(15, 2)
);
```

2. Update information of particular customer

```
UPDATE account
SET branch_name = 'New Branch', balance = 150000
WHERE acc_no = 123456; -- Use the specific account number of the customer you want to
update
```

3. Find the customers having loan less than 1 lac

```
SELECT b.cust_name
FROM borrower b
JOIN loan l ON b.loan_no = l.loan_no
WHERE l.amount < 100000;
```

4. Display account number and customer name starting with ‘P’

```
SELECT d.acc_no, d.cust_name
FROM depositor d
WHERE d.cust_name LIKE 'P%';
```

5. Display name of the depositor with balance

```
SELECT d.cust_name, a.balance
FROM depositor d
JOIN account a ON d.acc_no = a.acc_no;
```

6. Find names of all customers who have a loan at the ‘Redwood branch’.

```
SELECT b.cust_name
FROM borrower b
JOIN loan l ON b.loan_no = l.loan_no
WHERE l.branch_name = 'Redwood branch';
```

7. Find all customers who have an account and loan or both

```
SELECT DISTINCT d.cust_name
FROM depositor d
LEFT JOIN borrower b ON d.cust_name = b.cust_name
UNION
SELECT DISTINCT b.cust_name
FROM borrower b
LEFT JOIN depositor d ON b.cust_name = d.cust_name;
```

8. Find all customers who do not have loan

```
SELECT d.cust_name
FROM depositor d
LEFT JOIN borrower b ON d.cust_name = b.cust_name
WHERE b.loan_no IS NULL;
```

9. Find average account balance at each branch.

```
SELECT branch_name, AVG(balance) AS avg_balance
FROM account
GROUP BY branch_name;
```

10. Find the name of borrower having maximum loan amount

```
SELECT b.cust_name
```

```
FROM borrower b
JOIN loan l ON b.loan_no = l.loan_no
WHERE l.amount = (SELECT MAX(amount) FROM loan);
```

PL/SQL

Unnamed PL/SQLcode block: Use of Control structure and Exception handling is mandatory.

Suggested Problem statement:

1. Consider Tables:
    1. Borrower (Name, Date of Issue, Name of Book, Status)
    2. Fine(Name, Date, Amt)

    Calculate fine for particular student

    A. Check the number of days (from date of issue).
    B. If days are between 10 to 20 then the fine amount will be Rs 2 per day.
    C. If no. of days>20, per day fine will be Rs 20 per day and for days less than 20, Rs. 5 per day.
    D. After submitting the book, status will change from I to R.
    E. if condition of fine is true, then details will be stored into fine table
    F. Also handles the exception by named exception handler or user defined exception handler

```
DECLARE
    -- Declare variables
    v_name Borrower.Name%TYPE;
    v_date_of_issue Borrower.Date_of_Issue%TYPE;
    v_name_of_book Borrower.Name_of_Book%TYPE;
    v_status Borrower.Status%TYPE;
    v_fine_amount NUMBER := 0;
    v_days_borrowed NUMBER;

    -- Exception declaration
    invalid_borrower EXCEPTION;
    v_days_error EXCEPTION;

    -- Declare cursor to fetch borrower details
    CURSOR borrower_cursor IS
        SELECT Name, Date_of_Issue, Name_of_Book, Status
        FROM Borrower
        WHERE Status = 'I'; -- Only borrow records with 'Issued' status

BEGIN
    -- Open the cursor and loop through each borrower record
    FOR borrower_record IN borrower_cursor LOOP
        v_name := borrower_record.Name;
        v_date_of_issue := borrower_record.Date_of_Issue;
        v_name_of_book := borrower_record.Name_of_Book;
        v_status := borrower_record.Status;

        -- Calculate the number of days borrowed
        v_days_borrowed := TRUNC(SYSDATE) - v_date_of_issue;
```

```
        -- Check if borrowed days are less than or equal to zero
        IF v_days_borrowed < 0 THEN
            RAISE v_days_error;
        END IF;

        -- Fine Calculation Logic
        IF v_days_borrowed BETWEEN 10 AND 20 THEN
            v_fine_amount := v_days_borrowed * 2;  -- Rs 2 per day
        ELSIF v_days_borrowed > 20 THEN
            v_fine_amount := v_days_borrowed * 20;  -- Rs 20 per day for > 20 days
        ELSIF v_days_borrowed < 10 THEN
            v_fine_amount := v_days_borrowed * 5;   -- Rs 5 per day for < 10 days
        END IF;

        -- Update Borrower status to 'R' (Returned) if fine is applicable
        IF v_fine_amount > 0 THEN
            -- Store fine details into Fine table
            INSERT INTO Fine (Name, Date, Amt)
            VALUES (v_name, SYSDATE, v_fine_amount);

            -- Update Borrower status to 'Returned' (R)
            UPDATE Borrower
            SET Status = 'R'
            WHERE Name = v_name AND Name_of_Book = v_name_of_book;

                DBMS_OUTPUT.PUT_LINE('Fine for ' || v_name || ' is Rs. ' ||
v_fine_amount);
        ELSE
            DBMS_OUTPUT.PUT_LINE('No fine for ' || v_name);
        END IF;
    END LOOP;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No borrower records found with status "I".');
    WHEN v_days_error THEN
        DBMS_OUTPUT.PUT_LINE('Invalid borrowed days detected for ' || v_name || '.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
```

Write a PL/SQL code block to calculate the area and perimeter of a rectangle for a value of length varying from 10 to 20 and breadth varying from 5 to 10.

Store the length and breadth and the corresponding values of the calculated area and perimeter in an empty table named areas, consisting of three columns, length, breadth and area.

```
CREATE TABLE areas (
    length NUMBER,
    breadth NUMBER,
    area NUMBER,
```

```
    perimeter NUMBER
);

DECLARE
    v_length NUMBER;
    v_breadth NUMBER;
    v_area NUMBER;
    v_perimeter NUMBER;
BEGIN
    -- Loop through length from 10 to 20
    FOR v_length IN 10..20 LOOP
        -- Loop through breadth from 5 to 10
        FOR v_breadth IN 5..10 LOOP
            -- Calculate area and perimeter for each combination
            v_area := v_length * v_breadth;
            v_perimeter := 2 * (v_length + v_breadth);

            -- Insert the calculated values into the 'areas' table
            INSERT INTO areas (length, breadth, area, perimeter)
            VALUES (v_length, v_breadth, v_area, v_perimeter);
        END LOOP;
    END LOOP;

    -- Commit the changes to save the data in the table
    COMMIT;
END;
```

Write a PL/SQL code to calculate Gross and net salary of the employee
Employee (Empid, Name, Basic salary, type) . Type may be Permanent or temporary.
For Permanent employees following is the calculation for salary
DA - 115% of Basic salary
HRA - 12% of basic salary (if HRA is > 20000 then it will be considered as 20000 only)
Calculate Gross salary by adding DA and HRA
Deductions are
Income tax -2000 Rs
Net salary= Gross Salary - Deductions
For Temporary Only basic salary and deductions are there
Prepare a salary table for all

```
CREATE TABLE Employee (
    Empid NUMBER PRIMARY KEY,
    Name VARCHAR2(100),
    Basic_salary NUMBER,
    Type VARCHAR2(20) CHECK (Type IN ('Permanent', 'Temporary'))
);

CREATE TABLE Salary (
    Empid NUMBER,
    Gross_salary NUMBER,
    Net_salary NUMBER,
    CONSTRAINT fk_empid FOREIGN KEY (Empid) REFERENCES Employee(Empid)
);
```

```
DECLARE
    v_empid NUMBER;
    v_name VARCHAR2(100);
    v_basic_salary NUMBER;
    v_type VARCHAR2(20);
    v_DA NUMBER;
    v_HRA NUMBER;
    v_gross_salary NUMBER;
    v_net_salary NUMBER;
    v_deductions NUMBER := 2000;  -- Fixed deduction for income tax

    CURSOR emp_cursor IS
        SELECT Empid, Name, Basic_salary, Type
        FROM Employee;
BEGIN
    -- Open cursor to process each employee
    FOR emp_rec IN emp_cursor LOOP
        v_empid := emp_rec.Empid;
        v_name := emp_rec.Name;
        v_basic_salary := emp_rec.Basic_salary;
        v_type := emp_rec.Type;

        IF v_type = 'Permanent' THEN
            -- Calculate DA (115% of basic salary)
            v_DA := v_basic_salary * 1.15;

            -- Calculate HRA (12% of basic salary, capped at 20000)
            v_HRA := v_basic_salary * 0.12;
            IF v_HRA > 20000 THEN
                v_HRA := 20000;
            END IF;

            -- Calculate Gross salary
            v_gross_salary := v_basic_salary + v_DA + v_HRA;

        ELSIF v_type = 'Temporary' THEN
            -- For temporary employees, only basic salary is gross salary
            v_gross_salary := v_basic_salary;
        END IF;

        -- Calculate Net salary (Gross - Deductions)
        v_net_salary := v_gross_salary - v_deductions;

        -- Insert calculated salaries into the Salary table
        INSERT INTO Salary (Empid, Gross_salary, Net_salary)
        VALUES (v_empid, v_gross_salary, v_net_salary);

    END LOOP;

    -- Commit the transaction
```

```
    COMMIT;
END;
```

Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers).
Write a database trigger on the student table. The System should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added in the Student_Audit table (Action may be delete or update), date of action should be recorded

Student(roll, name)

Student_Audit (roll, name, action, date)

```
CREATE TABLE Student (
    roll INT PRIMARY KEY,
    name VARCHAR2(100)
);

CREATE TABLE Student_Audit (
    roll INT,
    name VARCHAR2(100),
    action VARCHAR2(10),   -- Stores 'UPDATE' or 'DELETE'
    action_date DATE
);

-- Trigger to capture updates on Student records
CREATE OR REPLACE TRIGGER student_update_audit
BEFORE UPDATE ON Student
FOR EACH ROW
BEGIN
    INSERT INTO Student_Audit (roll, name, action, action_date)
    VALUES (:OLD.roll, :OLD.name, 'UPDATE', SYSDATE);
END;
/

-- Trigger to capture deletions from Student records
CREATE OR REPLACE TRIGGER student_delete_audit
BEFORE DELETE ON Student
FOR EACH ROW
BEGIN
    INSERT INTO Student_Audit (roll, name, action, action_date)
    VALUES (:OLD.roll, :OLD.name, 'DELETE', SYSDATE);
END;
/
```

Explain Cursor in detail
Write types of cursors.
Explain with syntax and example

Write a PL/SQL code block to calculate the area and perimeter of a circle
by imputing range for radius as r1 & r2
Store the radius and the corresponding values of the calculated area and perimeter in an

empty table named areas, consisting of three columns, radius and area, perimeter
Use exception handling for checking wrong range values

```
CREATE TABLE areas (
    radius NUMBER,
    area NUMBER,
    perimeter NUMBER
);

DECLARE
    r1 NUMBER := 5;  -- Example starting radius (input)
    r2 NUMBER := 10; -- Example ending radius (input)
    radius NUMBER;
    area NUMBER;
    perimeter NUMBER;
BEGIN
    -- Exception handling for invalid input range
    IF r1 <= 0 OR r2 <= 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Radius values must be positive.');
    ELSIF r1 > r2 THEN
         RAISE_APPLICATION_ERROR(-20002, 'Starting radius (r1) must be less than or
equal to ending radius (r2).');
    END IF;

    -- Loop through the radius range from r1 to r2
    FOR radius IN r1..r2 LOOP
        -- Calculate area and perimeter
        area := 3.14159 * radius * radius;  -- Area = π * r²
        perimeter := 2 * 3.14159 * radius;  -- Perimeter = 2 * π * r

        -- Insert the calculated values into the 'areas' table
        INSERT INTO areas (radius, area, perimeter)
        VALUES (radius, area, perimeter);
    END LOOP;

    -- Commit the transaction to save changes
    COMMIT;

EXCEPTION
    -- Handle any unexpected exceptions
    WHEN OTHERS THEN
        -- Rollback if any exception occurs
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
```

MongoDB

MongoDB Queries: Design and Develop MongoDB Queries using CRUD operations.(Use CRUD operations, SAVE method, logical operators etc.)

.Implement the following MongoDb Query

1. Create a collection named books.

```
db.createCollection("books");
```

2. Insert 5 records with field TITLE,DESCRIPTION,BY,URL,TAGS AND LIKES

```
db.books.insertMany([
  {
    TITLE: "MongoDB for Beginners",
    DESCRIPTION: "An introduction to MongoDB basics.",
    BY: "Alice",
    URL: "http://example.com/mongodb-for-beginners",
    TAGS: ["database", "mongodb", "intro"],
    LIKES: 100
  },
  {
    TITLE: "Advanced MongoDB",
    DESCRIPTION: "A deep dive into MongoDB features.",
    BY: "Bob",
    URL: "http://example.com/advanced-mongodb",
    TAGS: ["database", "mongodb", "advanced"],
    LIKES: 200
  },
  {
    TITLE: "Learning NoSQL",
    DESCRIPTION: "Understanding NoSQL databases.",
    BY: "Charlie",
    URL: "http://example.com/learning-nosql",
    TAGS: ["nosql", "database"],
    LIKES: 150
  },
  {
    TITLE: "MongoDB Performance Tuning",
    DESCRIPTION: "Optimizing MongoDB performance.",
    BY: "David",
    URL: "http://example.com/mongodb-performance-tuning",
    TAGS: ["mongodb", "performance"],
    LIKES: 50
  },
  {
    TITLE: "MongoDB and Node.js",
    DESCRIPTION: "Integrating MongoDB with Node.js.",
    BY: "Eve",
    URL: "http://example.com/mongodb-nodejs",
    TAGS: ["mongodb", "nodejs"],
    LIKES: 120
  }
]);
```

3. Insert 1 more document in the collection with additional field of user name and

comments.

```
db.books.insertOne({
  TITLE: "MongoDB Overview",
  DESCRIPTION: "A comprehensive guide to MongoDB.",
  BY: "John",
  URL: "http://example.com/mongodb-overview",
  TAGS: ["mongodb", "guide"],
  LIKES: 180,
  USERNAME: "john_doe",
  COMMENTS: "Great resource for learning MongoDB!"
});
```

4. Display all the documents whose title is 'mongodb'.

```
db.books.find({ TITLE: "mongodb" });
```

5. Display all the documents written by 'john' or whose title is 'mongodb'.

```
db.books.find({
  $or: [
    { BY: "john" },
    { TITLE: "mongodb" }
  ]
});
```

6. Display all the documents whose title is 'mongodb' and written by 'john'.

```
db.books.find({
  TITLE: "mongodb",
  BY: "john"
});
```

7. Update the title of 'mongodb' document to 'mongodb overview'

```
db.books.updateOne(
  { TITLE: "mongodb" },
  { $set: { TITLE: "mongodb overview" } }
);
```

8. Delete the document titled 'nosql overview'.

```
db.books.deleteOne({ TITLE: "nosql overview" });
```

9. Display exactly two documents written by 'john'.

```
db.books.find({ BY: "john" }).limit(2);
```

MongoDB Queries: Design and Develop MongoDB Queries using CRUD operations.(UseCRUDoperations, SAVE method, logical operators etc.)

.Implement the following MongoDb Query

1. Create a collection named books.

```
db.createCollection("books");
```

2. Insert 5 records with field TITLE,DESCRIPTION,BY,URL,TAGS AND LIKES

```
db.books.insertMany([
  {
    TITLE: "MongoDB for Beginners",
    DESCRIPTION: "An introduction to MongoDB basics.",
    BY: "Alice",
    URL: "http://example.com/mongodb-for-beginners",
    TAGS: ["database", "mongodb", "intro"],
    LIKES: 100
  },
  {
    TITLE: "Advanced MongoDB",
    DESCRIPTION: "A deep dive into MongoDB features.",
    BY: "Bob",
    URL: "http://example.com/advanced-mongodb",
    TAGS: ["database", "mongodb", "advanced"],
    LIKES: 200
  },
  {
    TITLE: "Learning NoSQL",
    DESCRIPTION: "Understanding NoSQL databases.",
    BY: "Charlie",
    URL: "http://example.com/learning-nosql",
    TAGS: ["nosql", "database"],
    LIKES: 150
  },
  {
    TITLE: "MongoDB Performance Tuning",
    DESCRIPTION: "Optimizing MongoDB performance.",
    BY: "David",
    URL: "http://example.com/mongodb-performance-tuning",
    TAGS: ["mongodb", "performance"],
    LIKES: 50
  },
  {
    TITLE: "MongoDB and Node.js",
    DESCRIPTION: "Integrating MongoDB with Node.js.",
    BY: "Eve",
    URL: "http://example.com/mongodb-nodejs",
    TAGS: ["mongodb", "nodejs"],
    LIKES: 120
  }
]);
```

3. Insert 1 more document in the collection with additional field of user name and comments.

```
db.books.insertOne({
  TITLE: "MongoDB Overview",
  DESCRIPTION: "A comprehensive guide to MongoDB.",
  BY: "John",
  URL: "http://example.com/mongodb-overview",
  TAGS: ["mongodb", "guide"],
  LIKES: 180,
  USERNAME: "john_doe",
  COMMENTS: "Great resource for learning MongoDB!"
});
```

4. Display all the documents whose title is 'mongodb'.

```
db.books.find({ TITLE: "mongodb" });
```

5. Display all the documents written by 'john' or whose title is 'mongodb'.

```
db.books.find({
  $or: [
    { BY: "john" },
    { TITLE: "mongodb" }
  ]
});
```

6. Display all the documents whose title is 'mongodb' and written by 'john'.

```
db.books.find({
  TITLE: "mongodb",
  BY: "john"
});
```

7. Update the title of 'mongodb' document to 'mongodb overview'

```
db.books.updateOne(
  { TITLE: "mongodb" },
  { $set: { TITLE: "mongodb overview" } }
);
```

8. Delete the document titled 'nosql overview'.

```
db.books.deleteOne({ TITLE: "nosql overview" });
```

ER Diagram

Conceptual Design using ER features using tools like ERD plus, ER Win etc. (Identifying entities, relationships between entities, attributes, keys, cardinalities, generalization, specialization etc.) Convert the ER diagram into relational tables and normalize the Relational data model.

ER model of a Hospital management using the following description . Each of these entities have their respective attributes which are −

Patients - ID(primary key), name, age,visit_date

Tests- Name(primary key), date, result

Doctor- ID(primary key), name, specialization

Draw an ER Diagram for Hospital management system with Basic and Extended ER Features

Draw an ER Diagram for Library management system with Basic and Extended ER Features

Conceptual Design using ER features using tools like ERD plus, ER Win etc. (Identifying entities, relationships between entities, attributes, keys, cardinalities, generalization, specialization etc.) Convert the ER diagram into relational tables and normalize the Relational data model.

ER diagram of Company has the following description :

a. Company has several departments and Each department may have several locations. Departments schema (name, D_no, Location)
b. Each department has employees Employees schema is (name, empid, address, city, dob, date_of_joining,gender, salary,age)
c. A Manager controls a particular department.
d. Each department is associated with a number of projects(id, title,city).
e. An employee works in only one department but can work on several projects.
f. We also keep track of the number of hours worked by an employee on a single project.
g. Each employee has dependent as Dependent as (D_name, Gender and relationship)

Database Connectivity

Write a program to implement Menu driven MySQL/Oracle database connectivity with any front end language for Python/Java/PHP to implement Database navigation operations (add, delete, edit etc.)

Create a table for employee(empid, name, salary) and perform operations as insert a record, update values, delete particular record, display all record, display employees having salary >50000, display record for particular employee

Write a program using database connectivity to store, edit , delete records