P6_SQL\TechNova.sql

```sql
-- ============================================================
-- Title: Employee Rewards & Performance Management System
-- Description: SQL script to create and manage a database for employee rewards and performance.
-- USER STORY 1: DATABASE SETUP (DDL)
-- ============================================================

-- 1. Create Database
CREATE DATABASE IF NOT EXISTS TechNovaDB;
USE TechNovaDB;

-- 2. Create Tables

-- Department Table
CREATE TABLE Department (
    DeptID INT PRIMARY KEY,
    DeptName VARCHAR(100) NOT NULL UNIQUE,
    Location VARCHAR(100)
);

-- Employee Table
CREATE TABLE Employee (
    EmpID INT PRIMARY KEY,
    EmpName VARCHAR(100) NOT NULL,
    Gender CHAR(1) CHECK (Gender IN ('M','F')),
    DOB DATE,
    HireDate DATE,
    DeptID INT,
    FOREIGN KEY (DeptID) REFERENCES Department(DeptID)
);

-- Project Table
CREATE TABLE Project (
    ProjectID INT PRIMARY KEY,
    ProjectName VARCHAR(100) NOT NULL,
    DeptID INT,
    StartDate DATE,
    EndDate DATE,
    FOREIGN KEY (DeptID) REFERENCES Department(DeptID)
);

-- Performance Table
CREATE TABLE Performance (
    EmpID INT,
    ProjectID INT,
    Rating DECIMAL(3,2) CHECK (Rating BETWEEN 1 AND 5),
    ReviewDate DATE,
    PRIMARY KEY (EmpID, ProjectID),
    FOREIGN KEY (EmpID) REFERENCES Employee(EmpID),
    FOREIGN KEY (ProjectID) REFERENCES Project(ProjectID)
);

```

```sql
-- Reward Table
CREATE TABLE Reward (
    EmpID INT,
    RewardMonth DATE,
    RewardAmount DECIMAL(10,2),
    FOREIGN KEY (EmpID) REFERENCES Employee(EmpID)
);

-- 3. Create Indexes
CREATE INDEX idx_empname ON Employee(EmpName);
CREATE INDEX idx_deptid ON Employee(DeptID);

-- ============================================================
-- USER STORY 2: INSERT AND MANAGE DATA (DML)
-- ============================================================

-- 1. Insert Sample Data

INSERT INTO Department VALUES
(101, 'IT', 'Bangalore'),
(102, 'HR', 'Delhi'),
(103, 'Finance', 'Mumbai'),
(104, 'Sales', 'Hyderabad'),
(105, 'Marketing', 'Pune');

INSERT INTO Employee VALUES
(1, 'Asha', 'F', '1990-07-12', '2018-06-10', 101),
(2, 'Raj', 'M', '1988-04-09', '2020-03-22', 102),
(3, 'Neha', 'F', '1995-01-15', '2021-08-05', 101),
(4, 'Karan', 'M', '1992-02-18', '2019-11-10', 103),
(5, 'Priya', 'F', '1997-12-01', '2022-01-05', 104);

INSERT INTO Project VALUES
(201, 'ERP Upgrade', 101, '2020-02-01', '2020-12-31'),
(202, 'Recruitment Portal', 102, '2021-01-10', '2021-06-30'),
(203, 'Budget Automation', 103, '2021-04-01', '2021-10-15'),
(204, 'Sales Analytics', 104, '2022-02-01', '2022-11-01'),
(205, 'Brand Campaign', 105, '2023-01-05', '2023-06-30');

INSERT INTO Performance VALUES
(1, 201, 4.5, '2020-12-20'),
(2, 202, 4.0, '2021-06-25'),
(3, 201, 4.8, '2021-09-10'),
(4, 203, 3.9, '2021-10-01'),
(5, 204, 4.2, '2022-10-20');

INSERT INTO Reward VALUES
(1, '2023-03-01', 2500),
(2, '2023-03-01', 1800),
(3, '2023-06-01', 3200),
(4, '2023-08-01', 950),
(5, '2023-09-01', 2700);

-- 2. Update one employee’s department
```

```sql
UPDATE Employee
SET DeptID = 105
WHERE EmpID = 5;

-- 3. Delete one reward record where amount < 1000
DELETE FROM Reward WHERE RewardAmount < 1000;

-- ==========================================================
-- USER STORY 3: GENERATE INSIGHTS (DQL + AGGREGATES)
-- ==========================================================

-- 1. Employees who joined after 2019-01-01
SELECT EmpName, HireDate FROM Employee
WHERE HireDate > '2019-01-01';

-- 2. Average performance rating per department
SELECT d.DeptName, ROUND(AVG(p.Rating),2) AS AvgRating
FROM Performance p
JOIN Employee e ON p.EmpID = e.EmpID
JOIN Department d ON e.DeptID = d.DeptID
GROUP BY d.DeptName;

-- 3. List employees with their age
SELECT EmpName,
       TIMESTAMPDIFF(YEAR, DOB, CURDATE()) AS Age
FROM Employee;

-- 4. Total rewards given in the current year
SELECT YEAR(RewardMonth) AS Year, SUM(RewardAmount) AS TotalRewards
FROM Reward
WHERE YEAR(RewardMonth) = YEAR(CURDATE())
GROUP BY YEAR(RewardMonth);

-- 5. Employees with rewards greater than 2000
SELECT e.EmpName, r.RewardAmount
FROM Employee e
JOIN Reward r ON e.EmpID = r.EmpID
WHERE r.RewardAmount > 2000;

-- ==========================================================
-- USER STORY 4: ADVANCED QUERIES (JOINS + SUBQUERIES)
-- ==========================================================

-- 1. Display Employee Name, Department, Project, and Rating
SELECT e.EmpName, d.DeptName, p.ProjectName, perf.Rating
FROM Employee e
JOIN Department d ON e.DeptID = d.DeptID
JOIN Performance perf ON e.EmpID = perf.EmpID
JOIN Project p ON perf.ProjectID = p.ProjectID;

-- 2. Highest-rated employee in each department
SELECT d.DeptName, e.EmpName, perf.Rating
FROM Performance perf
JOIN Employee e ON perf.EmpID = e.EmpID
JOIN Department d ON e.DeptID = d.DeptID
WHERE (e.EmpID, perf.Rating) IN (
    SELECT e2.EmpID, MAX(p2.Rating)
    FROM Performance p2
    JOIN Employee e2 ON p2.EmpID = e2.EmpID
    GROUP BY e2.DeptID
);

-- 3. Employees who have NOT received any rewards
SELECT EmpName
FROM Employee
WHERE EmpID NOT IN (SELECT EmpID FROM Reward);

-- ============================================================
-- USER STORY 5: TRANSACTION CONTROL AND OPTIMIZATION
-- ============================================================

-- 1. Begin a transaction for new employee addition
START TRANSACTION;

INSERT INTO Employee VALUES (6, 'Ravi', 'M', '1998-11-05', '2023-07-01', 101);

INSERT INTO Performance VALUES (6, 201, 4.7, '2023-07-15');

-- If successful, commit, else rollback
COMMIT;
-- ROLLBACK;  -- (Use this if any insert fails)

-- 2. Analyze query performance (example)
EXPLAIN SELECT e.EmpName, d.DeptName, p.ProjectName, perf.Rating
FROM Employee e
JOIN Department d ON e.DeptID = d.DeptID
JOIN Performance perf ON e.EmpID = perf.EmpID
JOIN Project p ON perf.ProjectID = p.ProjectID;

-- Then create indexes (if not already created) and rerun EXPLAIN to see improvement
-- CREATE INDEX idx_projname ON Project(ProjectName);

-- ============================================================
-- BONUS CHALLENGE
-- ============================================================

-- 1. Create a View combining Employee, Department, and Performance
CREATE VIEW EmployeePerformanceView AS
SELECT e.EmpID, e.EmpName, d.DeptName, p.ProjectName, perf.Rating, perf.ReviewDate
FROM Employee e
JOIN Department d ON e.DeptID = d.DeptID
JOIN Performance perf ON e.EmpID = perf.EmpID
JOIN Project p ON perf.ProjectID = p.ProjectID;

-- 2. Stored Procedure: Get top 3 performers by department
DELIMITER $$
CREATE PROCEDURE GetTopPerformers(IN deptName VARCHAR(100))
BEGIN
```

```sql
    SELECT e.EmpName, d.DeptName, perf.Rating
    FROM Performance perf
    JOIN Employee e ON perf.EmpID = e.EmpID
    JOIN Department d ON e.DeptID = d.DeptID
    WHERE d.DeptName = deptName
    ORDER BY perf.Rating DESC
    LIMIT 3;
END$$
DELIMITER ;

-- Example Call:
-- CALL GetTopPerformers('IT');

```