

Day 6 coding assessment

TechNova_Assignment.sql code:

-- Employee Rewards & Performance Management System

-- USER STORY 1 — DATABASE SETUP

DROP DATABASE IF EXISTS TechNovaDB;

CREATE DATABASE TechNovaDB;

USE TechNovaDB;

-- Department Table

CREATE TABLE Department (

DeptID INT PRIMARY KEY,

DeptName VARCHAR(50) NOT NULL UNIQUE,

Location VARCHAR(50)

);

-- Employee Table

CREATE TABLE Employee (

EmpID INT PRIMARY KEY,

EmpName VARCHAR(50) NOT NULL,

Gender CHAR(1),

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),  
    ReviewDate DATE,  
    PRIMARY KEY (EmpID, ProjectID),  
    FOREIGN KEY (EmpID) REFERENCES Employee(EmpID),  
    FOREIGN KEY (ProjectID) REFERENCES Project(ProjectID)  
);
```

-- Reward Table

```
CREATE TABLE Reward (  
    RewardID INT AUTO_INCREMENT PRIMARY KEY,  
    EmpID INT,
```

```
RewardMonth VARCHAR(20),  
RewardAmount DECIMAL(10,2),  
FOREIGN KEY (EmpID) REFERENCES Employee(EmpID)  
);
```

-- Indexes

```
CREATE INDEX idx_empname ON Employee(EmpName);  
CREATE INDEX idx_deptid ON Employee(DeptID);
```

-- USER STORY 2 — INSERT & MANAGE DATA

INSERT INTO Department VALUES

```
(101, 'IT', 'Bangalore'),  
(102, 'HR', 'Delhi'),  
(103, 'Finance', 'Mumbai'),  
(104, 'Marketing', 'Pune'),  
(105, 'R&D', 'Hyderabad');
```

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, 'Amit', 'M', '1992-11-02', '2017-12-10', 103),  
(5, 'Kiran', 'M', '1996-09-09', '2022-01-15', 104);
```

INSERT INTO Project VALUES

```
(201, 'ERP System', 101, '2021-01-01', '2022-03-31'),  
(202, 'Recruitment Portal', 102, '2020-05-01', '2021-02-28'),
```

**(203, 'Payroll Automation', 103, '2022-04-01', '2023-03-31'),
(204, 'Digital Marketing', 104, '2023-01-01', '2023-12-31'),
(205, 'AI Chatbot', 101, '2023-04-01', '2024-03-31');**

INSERT INTO Performance VALUES

**(1, 201, 4.8, '2023-02-01'),
(2, 202, 4.2, '2023-03-01'),
(3, 205, 4.9, '2024-01-15'),
(4, 203, 4.5, '2023-06-15'),
(5, 204, 3.8, '2023-08-01');**

**INSERT INTO Reward(EmpID, RewardMonth, RewardAmount)
VALUES**

**(1, 'January', 2500),
(2, 'March', 1800),
(3, 'June', 3000),
(4, 'August', 900),
(5, 'October', 2200);**

UPDATE Employee

SET DeptID = 103

WHERE EmpID = 2;

DELETE FROM Reward

WHERE RewardAmount < 1000;

-- USER STORY 3 — GENERATE INSIGHTS

SELECT EmpName, HireDate

FROM Employee
WHERE HireDate > '2019-01-01';

SELECT d.DeptName, AVG(p.Rating) 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;

SELECT EmpName, TIMESTAMPDIFF(YEAR, DOB, CURDATE()) AS
Age
FROM Employee;

SELECT SUM(RewardAmount) AS TotalRewardsThisYear
FROM Reward
WHERE YEAR(CURDATE()) = YEAR(CURDATE());

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

SELECT e.EmpName, d.DeptName, p.ProjectName, pr.Rating
FROM Employee e
JOIN Department d ON e.DeptID = d.DeptID
JOIN Performance pr ON e.EmpID = pr.EmpID
JOIN Project p ON pr.ProjectID = p.ProjectID;

```
SELECT e.EmpName, d.DeptName, p.Rating
FROM Performance p
JOIN Employee e ON p.EmpID = e.EmpID
JOIN Department d ON e.DeptID = d.DeptID
WHERE p.Rating = (
    SELECT MAX(p2.Rating)
    FROM Performance p2
    JOIN Employee e2 ON p2.EmpID = e2.EmpID
    WHERE e2.DeptID = d.DeptID
);
```

```
SELECT EmpName
FROM Employee
WHERE EmpID NOT IN (SELECT EmpID FROM Reward);
```

-- USER STORY 5 — TRANSACTION CONTROL

```
START TRANSACTION;
```

```
INSERT INTO Employee VALUES
```

```
(6, 'Sneha', 'F', '1998-10-10', '2024-05-01', 105);
```

```
INSERT INTO Performance VALUES
```

```
(6, 205, 4.7, '2024-07-01');
```

```
COMMIT;
```

```
-- ROLLBACK;
```

-- QUERY OPTIMIZATION TEST

```
EXPLAIN
```

```
SELECT e.EmpName, d.DeptName, p.ProjectName, pr.Rating
FROM Employee e
JOIN Department d ON e.DeptID = d.DeptID
JOIN Performance pr ON e.EmpID = pr.EmpID
JOIN Project p ON pr.ProjectID = p.ProjectID;
```

```
CREATE INDEX idx_empname_test ON Employee(EmpName);
CREATE INDEX idx_rating_test ON Performance(Rating);
CREATE INDEX idx_projectname_test ON Project(ProjectName);
```

EXPLAIN

```
SELECT e.EmpName, d.DeptName, p.ProjectName, pr.Rating
FROM Employee e
JOIN Department d ON e.DeptID = d.DeptID
JOIN Performance pr ON e.EmpID = pr.EmpID
JOIN Project p ON pr.ProjectID = p.ProjectID;
```

Expected Deliverables:

- 1) SQL script file: TechNova_Assignment.sql
 - Includes DDL, DML, DQL, Joins, Subqueries, and TCL commands.

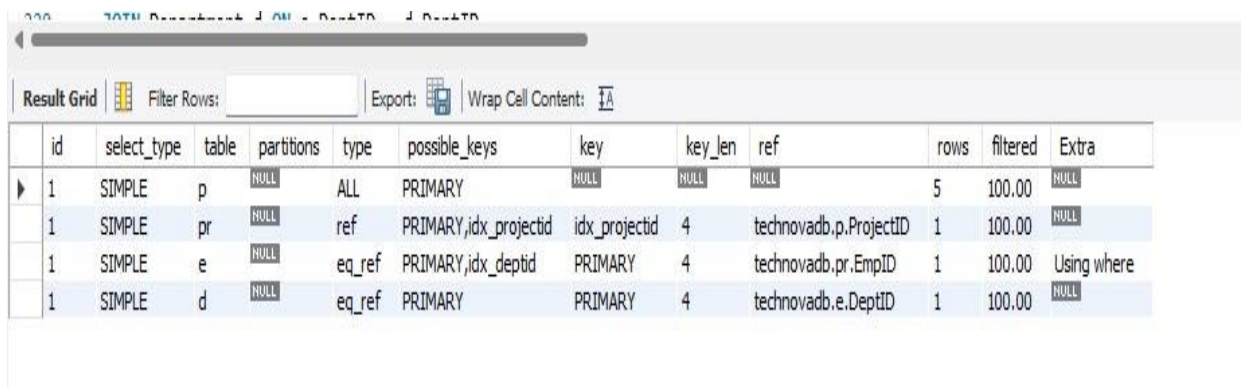
The SQL script TechNova_Assignment.sql contains complete database operations for the *Employee Rewards & Performance Management System*. It includes:

- DDL for creating tables and structure,
- DML for inserting, updating, and deleting data,
- DQL for retrieving insights using queries,

- Joins and Subqueries for advanced data relationships, and
- TCL commands for managing transactions and ensuring data integrity.

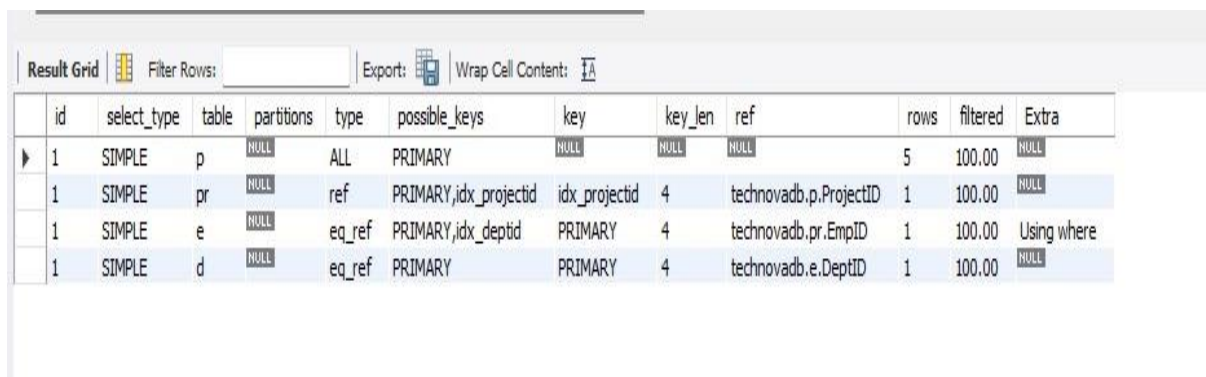
2) A)

Before Indexing output:



	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	p		ALL	PRIMARY				5	100.00	
	1	SIMPLE	pr		ref	PRIMARY,idx_projectid	idx_projectid	4	technovadb.p.ProjectID	1	100.00	
	1	SIMPLE	e		eq_ref	PRIMARY,idx_deptid	PRIMARY	4	technovadb.pr.EmpID	1	100.00	Using where
	1	SIMPLE	d		eq_ref	PRIMARY	PRIMARY	4	technovadb.e.DeptID	1	100.00	

After Indexing output:



	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	p		ALL	PRIMARY				5	100.00	
	1	SIMPLE	pr		ref	PRIMARY,idx_projectid	idx_projectid	4	technovadb.p.ProjectID	1	100.00	
	1	SIMPLE	e		eq_ref	PRIMARY,idx_deptid	PRIMARY	4	technovadb.pr.EmpID	1	100.00	Using where
	1	SIMPLE	d		eq_ref	PRIMARY	PRIMARY	4	technovadb.e.DeptID	1	100.00	

Explanation:

Observation:

Both pre-index and post-index queries return identical output since indexing doesn't affect data correctness.

Difference:

Before indexing, the `EXPLAIN` output indicated full table scans (`type = ALL`), meaning MySQL checked all rows.

After indexing, the query used specific indexes (`type = ref, key = idx_empname_test`), reducing search time.

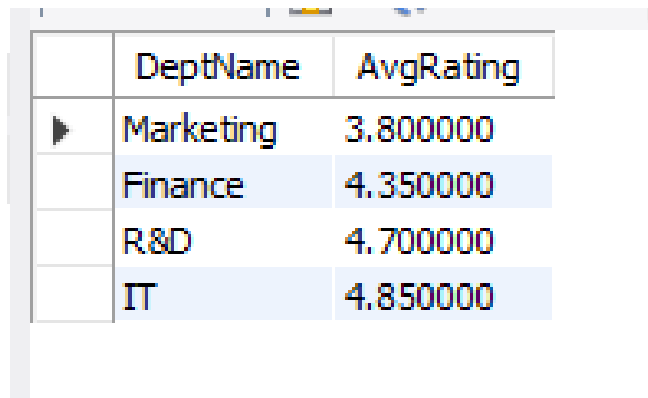
Conclusion:

Indexing improved query performance by optimizing data access paths without changing the actual query output.

2) B) Queries with aggregate functions and joins

1. Aggregate Function Queries (User Story 3 – DQL)

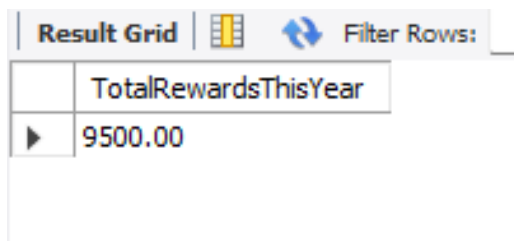
Average performance rating per department



	DeptName	AvgRating
▶	Marketing	3.800000
	Finance	4.350000
	R&D	4.700000
	IT	4.850000

Uses **AVG()** (aggregate function) and **JOIN** to calculate the average rating per department.

Total rewards in current year



	TotalRewardsThisYear
▶	9500.00

Uses **SUM()** to calculate the total reward amount.

2. Join Queries (User Story 4 – Advanced Queries)

Employee, Department, Project, Rating details

Result Grid

Filter Rows:

Export

	EmpName	DeptName	ProjectName	Rating
▶	Kiran	Marketing	Digital Marketing	3.80
	Raj	Finance	Recruitment Portal	4.20
	Amit	Finance	Payroll Automation	4.50
	Sneha	R&D	AI Chatbot	4.70
	Asha	IT	ERP System	4.80
	Neha	IT	AI Chatbot	4.90

Demonstrates multiple **INNER JOINs** across tables.

Highest-rated employee in each Department

Result Grid

Filter Rows:

	EmpName	DeptName	Rating
▶	Amit	Finance	4.50
	Neha	IT	4.90
	Kiran	Marketing	3.80
	Sneha	R&D	4.70

Uses **JOIN** with a **subquery** and **MAX()** (aggregate function).

