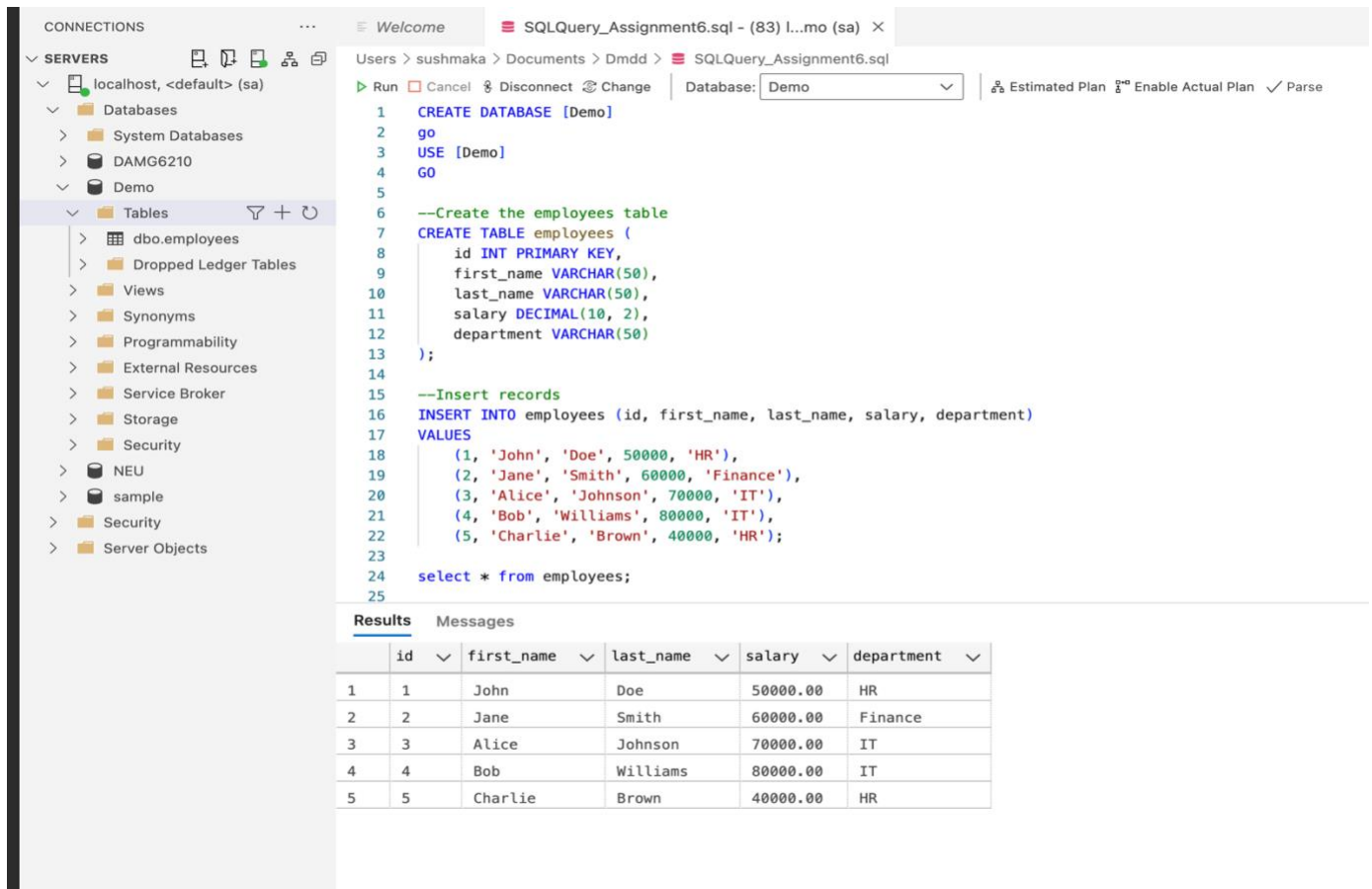


# DAMG6210 - Data Management and Database Design

## Homework 06



The screenshot displays the SQL Server Enterprise Manager interface. On the left, the 'CONNECTIONS' pane shows a tree view of the server hierarchy: 'SERVERS' > 'localhost, <default> (sa)' > 'Databases' > 'Demo' > 'Tables' > 'dbo.employees'. The main window shows a query titled 'SQLQuery\_Assignment6.sql - (83) l...mo (sa)'. The query text is as follows:

```
1 CREATE DATABASE [Demo]
2 GO
3 USE [Demo]
4 GO
5
6 --Create the employees table
7 CREATE TABLE employees (
8     id INT PRIMARY KEY,
9     first_name VARCHAR(50),
10    last_name VARCHAR(50),
11    salary DECIMAL(10, 2),
12    department VARCHAR(50)
13 );
14
15 --Insert records
16 INSERT INTO employees (id, first_name, last_name, salary, department)
17 VALUES
18     (1, 'John', 'Doe', 50000, 'HR'),
19     (2, 'Jane', 'Smith', 60000, 'Finance'),
20     (3, 'Alice', 'Johnson', 70000, 'IT'),
21     (4, 'Bob', 'Williams', 80000, 'IT'),
22     (5, 'Charlie', 'Brown', 40000, 'HR');
23
24 select * from employees;
```

Below the query text, the 'Results' tab is active, showing the output of the query in a table with 6 columns: 'id', 'first\_name', 'last\_name', 'salary', and 'department'. The table contains 5 rows of data.

	id	first_name	last_name	salary	department
1	1	John	Doe	50000.00	HR
2	2	Jane	Smith	60000.00	Finance
3	3	Alice	Johnson	70000.00	IT
4	4	Bob	Williams	80000.00	IT
5	5	Charlie	Brown	40000.00	HR

1.

INSERT INTO employees (id, first\_name, last\_name, salary, department)

SELECT id, first\_name, last\_name, salary, department

FROM contractors

WHERE department != 'HR';

CONNECTIONS

SERVICES

localhost, <default> (sa)

Databases

System Databases

DAMG6210

Demo

Tables

dbo.contractors

dbo.employees

Dropped Ledger Tables

Views

Synonyms

Programmability

External Resources

Service Broker

Storage

Security

NEU

sample

Security

Server Objects

Welcome

SQLQuery\_Assignment6.sql - (83) l...mo (sa) X

Users > sushmaka > Documents > Dmdd > SQLQuery\_Assignment6.sql

Run

Cancel

Disconnect

Change

Database: Demo

Estimated Plan

Enable Actual Plan

Parse

```
30      department VARCHAR(50)
31  );
32
33  --Insert records
34  INSERT INTO contractors (id, first_name, last_name, salary, department)
35  VALUES
36      (101, 'Sarah', 'Connor', 55000, 'IT'),
37      (102, 'Mike', 'Johnson', 48000, 'HR'),
38      (103, 'Emily', 'Davis', 62000, 'Finance'),
39      (104, 'David', 'Brown', 58000, 'IT');
40
41  select * from employees;
42
43  select * from contractors;
44
45  INSERT INTO employees (id, first_name, last_name, salary, department)
46  SELECT id, first_name, last_name, salary, department
47  FROM contractors
48  WHERE department != 'HR';
49
50
```

Results

Messages

	id	first_name	last_name	salary	department
1	101	Sarah	Connor	55000.00	IT
2	102	Mike	Johnson	48000.00	HR
3	103	Emily	Davis	62000.00	Finance
4	104	David	Brown	58000.00	IT

Users > sushmaka > Documents > Dmdd > SQLQuery\_Assignment6.sql

Run Cancel Disconnect Change Database: Demo Estimated Plan Enable Actual Plan Parse

```

30      department VARCHAR(50)
31    );
32
33    --Insert records
34    INSERT INTO contractors (id, first_name, last_name, salary, department)
35    VALUES
36      (101, 'Sarah', 'Connor', 55000, 'IT'),
37      (102, 'Mike', 'Johnson', 48000, 'HR'),
38      (103, 'Emily', 'Davis', 62000, 'Finance'),
39      (104, 'David', 'Brown', 58000, 'IT');
40
41    select * from employees;
42
43    select * from contractors;
44
45    INSERT INTO employees (id, first_name, last_name, salary, department)
46    SELECT id, first_name, last_name, salary, department
47    FROM contractors
48    WHERE department != 'HR';
49
50

```

**Results** Messages

	id	first_name	last_name	salary	department
1	1	John	Doe	50000.00	HR
2	2	Jane	Smith	60000.00	Finance
3	3	Alice	Johnson	70000.00	IT
4	4	Bob	Williams	80000.00	IT
5	5	Charlie	Brown	40000.00	HR
6	101	Sarah	Connor	55000.00	IT
7	103	Emily	Davis	62000.00	Finance
8	104	David	Brown	58000.00	IT

2.  
DELETE FROM employees  
WHERE salary <= 60000;

Users > sushmaka > Documents > Dmdd > SQLQuery\_Assignment6.sql

Run Cancel Disconnect Change Database: Demo Estimated Plan Enable Actual Plan Parse

```

30     department VARCHAR(50)
31 );
32
33 --Insert records
34 INSERT INTO contractors (id, first_name, last_name, salary, department)
35 VALUES
36     (101, 'Sarah', 'Connor', 55000, 'IT'),
37     (102, 'Mike', 'Johnson', 48000, 'HR'),
38     (103, 'Emily', 'Davis', 62000, 'Finance'),
39     (104, 'David', 'Brown', 58000, 'IT');
40
41 select * from employees;
42
43 select * from contractors;
44
45 --Insert data from contractors table, excluding 'HR' department
46 INSERT INTO employees (id, first_name, last_name, salary, department)
47 SELECT id, first_name, last_name, salary, department
48 FROM contractors
49 WHERE department != 'HR';
50
51 --Delete employees with salary less than or equal to $60,000
52 DELETE FROM employees
53 WHERE salary <= 60000;
54

```

**Results** Messages

	id	first_name	last_name	salary	department
1	3	Alice	Johnson	70000.00	IT
2	4	Bob	Williams	80000.00	IT
3	103	Emily	Davis	62000.00	Finance

3.  
 UPDATE employees  
 SET salary = salary \* 1.10  
 WHERE department = 'IT';

Users > sushmaka > Documents > Dmdd > SQLQuery\_Assignment6.sql

Run Cancel Disconnect Change Database: Demo Estimated Plan

```

35 VALUES
36     (101, 'Sarah', 'Connor', 55000, 'IT'),
37     (102, 'Mike', 'Johnson', 48000, 'HR'),
38     (103, 'Emily', 'Davis', 62000, 'Finance'),
39     (104, 'David', 'Brown', 58000, 'IT');
40
41 select * from employees;
42
43 select * from contractors;
44
45 --Insert data from contractors table, excluding 'HR' department
46 INSERT INTO employees (id, first_name, last_name, salary, department)
47 SELECT id, first_name, last_name, salary, department
48 FROM contractors
49 WHERE department != 'HR';
50
51 --Delete employees with salary less than or equal to $60,000
52 DELETE FROM employees
53 WHERE salary <= 60000;
54
55 --Update salary of 'IT' department employees by 10%
56 UPDATE employees
57 SET salary = salary * 1.10
58 WHERE department = 'IT';
59

```

**Results** Messages

	id	first_name	last_name	salary	department
1	3	Alice	Johnson	77000.00	IT
2	4	Bob	Williams	88000.00	IT
3	103	Emily	Davis	62000.00	Finance

4. SELECT TOP 3 \*  
FROM employees  
ORDER BY salary DESC;

SQLQuery\_Assignment6.sql (65) 1 KB (54)

Users > sushmaka > Documents > Dmdd > SQLQuery\_Assignment6.sql

Run Cancel Disconnect Change Database: Demo Estimated Plan Enable Actual Plan Parse

```
43 select * from contractors;
44
45 --Insert data from contractors table, excluding 'HR' department
46 INSERT INTO employees (id, first_name, last_name, salary, department)
47 SELECT id, first_name, last_name, salary, department
48 FROM contractors
49 WHERE department != 'HR';
50
51 --Delete employees with salary less than or equal to $60,000
52 DELETE FROM employees
53 WHERE salary <= 60000;
54
55 --Update salary of 'IT' department employees by 10%
56 UPDATE employees
57 SET salary = salary * 1.10
58 WHERE department = 'IT';
59
60 --Inserting more records
61 INSERT INTO employees (id, first_name, last_name, salary, department)
62 VALUES
63 (6, 'David', 'Miller', 55000, 'Sales'),
64 (7, 'Emily', 'Taylor', 65000, 'Marketing'),
65 (8, 'Frank', 'Garcia', 75000, 'IT'),
66 (9, 'Grace', 'Hall', 45000, 'HR');
67
```

Results Messages

	id	first_name	last_name	salary	department
1	3	Alice	Johnson	77000.00	IT
2	4	Bob	Williams	88000.00	IT
3	6	David	Miller	55000.00	Sales
4	7	Emily	Taylor	65000.00	Marketing
5	8	Frank	Garcia	75000.00	IT
6	9	Grace	Hall	45000.00	HR
7	103	Emily	Davis	62000.00	Finance

Users > sushmaka > Documents > Dmdd > SQLQuery\_Assignment6.sql

Run Cancel Disconnect Change

Database: 

Demo

Estimated Plan Enable Actual Plan Parse

60 --Inserting more records

61 INSERT INTO employees (id, first\_name, last\_name, salary, department)

62 VALUES

63 (6, 'David', 'Miller', 55000, 'Sales'),

64 (7, 'Emily', 'Taylor', 65000, 'Marketing'),

65 (8, 'Frank', 'Garcia', 75000, 'IT'),

66 (9, 'Grace', 'Hall', 45000, 'HR');

67

68 --Select top 3 highest salaries

69 SELECT TOP 3 \*

70 FROM employees

71 ORDER BY salary DESC;

72

73

74

Results Messages

	id	first_name	last_name	salary	department
1	4	Bob	Williams	88000.00	IT
2	3	Alice	Johnson	77000.00	IT
3	8	Frank	Garcia	75000.00	IT

5.  
SELECT \*  
FROM employees  
WHERE first\_name LIKE 'J%n';



Users > sushmaka > Documents > Dmdd > SQLQuery\_Assignment6.sql

Run Cancel Disconnect Change Database: Demo Estimated Plan Enable Ac

```
53 WHERE salary <= 60000;
54
55 --Update salary of 'IT' department employees by 10%
56 UPDATE employees
57 SET salary = salary * 1.10
58 WHERE department = 'IT';
59
60 --Inserting more records
61 INSERT INTO employees (id, first_name, last_name, salary, department)
62 VALUES
63     (6, 'David', 'Miller', 55000, 'Sales'),
64     (7, 'Emily', 'Taylor', 65000, 'Marketing'),
65     (8, 'Frank', 'Garcia', 75000, 'IT'),
66     (9, 'Grace', 'Hall', 45000, 'HR');
67
68 --Select top 3 highest salaries
69 SELECT TOP 3 salary
70 FROM employees
71 ORDER BY salary DESC;
72
73 --Select employees with first name starting with 'J' and ending with 'n'
74 SELECT *
75 FROM employees
76 WHERE first_name LIKE 'J%n';
77
```

Results Messages

id	first_name	last_name	salary	department
----	------------	-----------	--------	------------



```

65      (8, 'Frank', 'Garcia', 75000, 'IT'),
66      (9, 'Grace', 'Hall', 45000, 'HR'));
67
68  --Select top 3 highest salaries
69  SELECT TOP 3 salary
70  FROM employees
71  ORDER BY salary DESC;
72
73  --Select employees with first name starting with 'J' and ending with 'n'
74  SELECT *
75  FROM employees
76  WHERE first_name LIKE 'J%n';
77
78  --Inserting a record
79  INSERT INTO employees (id, first_name, last_name, salary, department)
80  VALUES (10, 'John', 'Smith', 55000, 'Sales');
81
82  select * from employees;

```

Results Messages

	id	first_name	last_name	salary	department
1	3	Alice	Johnson	77000.00	IT
2	4	Bob	Williams	88000.00	IT
3	6	David	Miller	55000.00	Sales
4	7	Emily	Taylor	65000.00	Marketing
5	8	Frank	Garcia	75000.00	IT
6	9	Grace	Hall	45000.00	HR
7	10	John	Smith	55000.00	Sales
8	103	Emily	Davis	62000.00	Finance

Results grid

Users > sushmaka > Documents > Dmdd > SQLQuery\_Assignment6.sql

Run Cancel Disconnect Change Database: Demo Estimated Plan Enable

```

65 (8, 'Frank', 'Garcia', 75000, 'IT'),
66 (9, 'Grace', 'Hall', 45000, 'HR');
67
68 --Select top 3 highest salaries
69 SELECT TOP 3 salary
70 FROM employees
71 ORDER BY salary DESC;
72
73 --Select employees with first name starting with 'J' and ending with 'n'
74 SELECT *
75 FROM employees
76 WHERE first_name LIKE 'J%n';
77
78 --Inserting a record
79 INSERT INTO employees (id, first_name, last_name, salary, department)
80 VALUES (10, 'John', 'Smith', 55000, 'Sales');
81
82 select * from employees;

```

**Results** Messages

	id	first_name	last_name	salary	department
1	10	John	Smith	55000.00	Sales

6.  
 SELECT \*  
 FROM employees  
 WHERE (department = 'HR' AND salary <= 60000)  
 OR (department != 'HR' AND salary > 60000);

SQLQuery\_Assignment6.sql - (83) L...mo (sa)

Users > sushmaka > Documents > Dmdd > SQLQuery\_Assignment6.sql

Run Cancel Disconnect Change Database: Demo Estin

```

65      (8, 'Frank', 'Garcia', 75000, 'IT'),
66      (9, 'Grace', 'Hall', 45000, 'HR');
67
68  --Select top 3 highest salaries
69  SELECT TOP 3 salary
70  FROM employees
71  ORDER BY salary DESC;
72
73  --Select employees with first name starting with 'J' and ending with '
74  SELECT *
75  FROM employees
76  WHERE first_name LIKE 'J%n';
77
78  --Inserting a record
79  INSERT INTO employees (id, first_name, last_name, salary, department)
80  VALUES (10, 'John', 'Smith', 55000, 'Sales');
81
82  select * from employees;
83
84  --Select employees in 'HR' or salary > 60000, but not both
85  SELECT *
86  FROM employees
87  WHERE (department = 'HR' AND salary <= 60000)
88  OR (department != 'HR' AND salary > 60000);

```

**Results** Messages

	id	first_name	last_name	salary	department
1	3	Alice	Johnson	77000.00	IT
2	4	Bob	Williams	88000.00	IT
3	7	Emily	Taylor	65000.00	Marketing
4	8	Frank	Garcia	75000.00	IT
5	9	Grace	Hall	45000.00	HR
6	103	Emily	Davis	62000.00	Finance

7.  
 SELECT department, SUM(salary) as total\_salary  
 FROM employees  
 GROUP BY department;

SQLQuery\_Assignment6.sql - (65) Lines (54) Columns

Users > sushmaka > Documents > Dmdd > SQLQuery\_Assignment6.sql

Run Cancel Disconnect Change Database: Demo Estimated Plan

```

70 FROM employees
71 ORDER BY salary DESC;
72
73 --Select employees with first name starting with 'J' and ending with 'n'
74 SELECT *
75 FROM employees
76 WHERE first_name LIKE 'J%n';
77
78 --Inserting a record
79 INSERT INTO employees (id, first_name, last_name, salary, department)
80 VALUES (10, 'John', 'Smith', 55000, 'Sales');
81
82 select * from employees;
83
84 --Select employees in 'HR' or salary > 60000, but not both
85 SELECT *
86 FROM employees
87 WHERE (department = 'HR' AND salary <= 60000)
88 | OR (department != 'HR' AND salary > 60000);
89
90 --Get total salary for each department
91 SELECT department, SUM(salary) as total_salary
92 FROM employees
93 GROUP BY department;

```

**Results** Messages

	department	total_salary
1	Finance	62000.00
2	HR	45000.00
3	IT	240000.00
4	Marketing	65000.00
5	Sales	110000.00

8.  
 SELECT \*  
 FROM employees  
 ORDER BY department ASC, salary DESC;

SQLQuery\_Assignment6.sql - (83) 1...mo (sa)

Users > sushmaka > Documents > Dmdd > SQLQuery\_Assignment6.sql

Run Cancel Disconnect Change Database: Demo Estimated Plan

```

81
82 select * from employees;
83
84 --Select employees in 'HR' or salary > 60000, but not both
85 SELECT *
86 FROM employees
87 WHERE (department = 'HR' AND salary <= 60000)
88 | OR (department != 'HR' AND salary > 60000);
89
90 --Get total salary for each department
91 SELECT department, SUM(salary) as total_salary
92 FROM employees
93 GROUP BY department;
94
95 --Select employees sorted by department (ASC) and salary (DESC)
96 SELECT *
97 FROM employees
98 ORDER BY department ASC, salary DESC;

```

**Results** Messages

	id	first_name	last_name	salary	department
1	103	Emily	Davis	62000.00	Finance
2	9	Grace	Hall	45000.00	HR
3	4	Bob	Williams	88000.00	IT
4	3	Alice	Johnson	77000.00	IT
5	8	Frank	Garcia	75000.00	IT
6	7	Emily	Taylor	65000.00	Marketing
7	6	David	Miller	55000.00	Sales
8	10	John	Smith	55000.00	Sales

9.

```

CREATE VIEW dept_avg_salary AS
SELECT department, AVG(salary) as average_salary
FROM employees
GROUP BY department;

```

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Servers' tree is expanded to 'localhost, <default> (sa)' > 'Databases' > 'Demo' > 'Tables'. The main window displays a SQL query file named 'SQLQuery\_Assignment6.sql' in the 'Demo' database. The query consists of three parts: a summary query, a sorting query, and a view creation and selection query.

```

90  --Get total salary for each department
91  SELECT department, SUM(salary) as total_salary
92  FROM employees
93  GROUP BY department;
94
95  --Select employees sorted by department (ASC) and salary (DESC)
96  SELECT *
97  FROM employees
98  ORDER BY department ASC, salary DESC;
99
100 --Create a view for average salary by department
101 CREATE VIEW dept_avg_salary AS
102 SELECT department, AVG(salary) as average_salary
103 FROM employees
104 GROUP BY department;
105
106 SELECT * FROM dept_avg_salary;

```

The 'Results' tab shows the output of the first query, displaying a table with 5 rows and 2 columns: 'department' and 'average\_salary'.

	department	average_salary
1	Finance	62000.000000
2	HR	45000.000000
3	IT	80000.000000
4	Marketing	65000.000000
5	Sales	55000.000000

10.

## 1. Index Optimization

Adding an index to the department column can greatly improve query speed. By making it easier to quickly access the rows linked to departments, indexes cut down on the amount of time needed to scan the entire table.

For instance, adding an index to the employee's table's department column speeds up searches that determine the average income for each department. For example, the database can effectively find all HR-related information using the index, which speeds up the process of collecting the average wage for the HR department.

```
-- Create an index on the department column
CREATE INDEX idx_department ON employees(department);
```

## **2. Covering Index**

Performance can be further enhanced by implementing a covering index that considers both department and salary. With this kind of index, the database can obtain all required information straight from the index without having to consult the entire table.

The database can quickly determine the average wage for each department thanks to a covering index on department and salary. The database may retrieve department and salary data from the index alone when a query is run to determine the average wage in the IT department, which expedites the aggregation process.

```
-- Create a covering index on department and salary  
CREATE INDEX idx_department_salary ON employees (department, salary);
```

## **3. Materialized Views**

Repetitive computations can be avoided by developing a materialized view that calculates average pay by department beforehand. The aggregated results are stored in this view for easy retrieval.

For instance, the average salary for every department is kept in a materialized view called `dept_avg_salary_mv`. Faster response times are achieved by the database retrieving the precomputed value from the view rather than recalculating it when the average wage for the Marketing department is queried.

## **4. Query Caching**

The database can save the outcomes of frequently run queries by utilizing query caching. The results of the same query can be retrieved straight from the cache by subsequent executions, negating the requirement for reprocessing.

For instance, the results may be cached if the query to determine the average salary per department is executed more than once. Performance is enhanced since the database provides the cached result immediately when the average wage for the Finance department is requested again.

## **5. Table Partitioning**

To guarantee that queries only target pertinent partitions, the personnel table is partitioned according to the department column. This improves efficiency by lowering the amount of data the database must scan.

For instance, a query to get the average wage for the sales department only scans the sales partition as the workers data has been divided by department. Compared to scanning the full table, this targeted scanning reduces the amount of time needed to process the data.

## **6. Denormalization and Pre-Aggregation**

Storing pre-aggregated data, such as average salaries per department, in a separate table eliminates the need for real-time calculations during queries. This approach provides instant access to aggregated values.

Because the average salary for each department is kept in a distinct table named `department_salary_stats`, queries can extract the average salary for the HR department straight from this table, saving time and effort compared to calculating it on the fly.

## **7. Regular Statistics Update**

Maintaining current database statistics aids in the query optimizer's decision-making process, which results in more effective execution strategies. Faster data aggregation and retrieval are made possible by accurate statistics.

For instance, queries that determine average salaries by department are guaranteed to follow the most effective execution paths when statistics on the `employees` table are updated on a regular basis. When obtaining the average compensation for any department, including IT, this results in optimal performance.

## **References**

Hoffer, J. A., Ramesh, V., & Topi, H. (2016). *Modern database management* (13th ed.). Pearson.