

**DATA MANAGEMENT AND DATABASE DESIGN**  
**HOMEWORK: WEEK- 8**

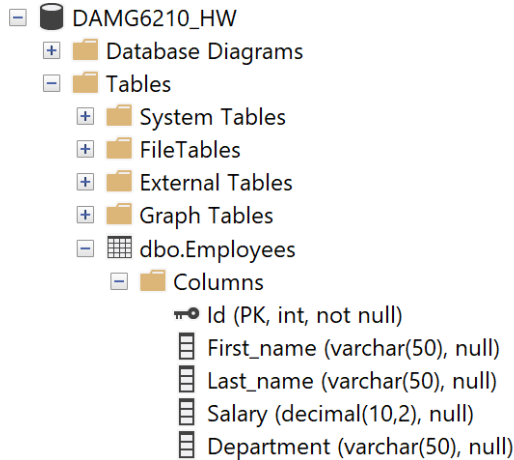
Answer the following questions on the provided schema for the table and sample record

INSERT statements:

Table definition:

```
CREATE TABLE Employees (  
    Id INT PRIMARY KEY,  
    First_name VARCHAR (50),  
    Last_name VARCHAR (50),  
    Salary DECIMAL (10, 2),  
    Department VARCHAR (50)  
);
```

Sol:



SQLQuery1.sql - L...LKUMAR\laksh (71))\*

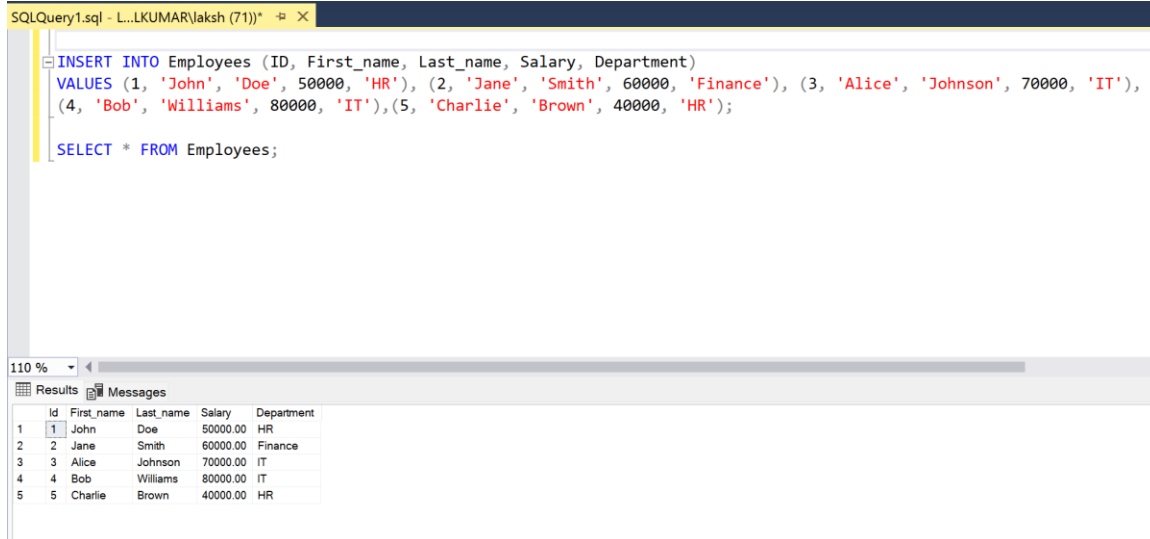
```
CREATE DATABASE DAMG6210_HW;  
GO  
USE DAMG6210_HW  
GO  
CREATE TABLE Employees (  
    Id INT PRIMARY KEY,  
    First_name VARCHAR(50),  
    Last_name VARCHAR(50),  
    Salary DECIMAL(10, 2),  
    Department VARCHAR(50)  
);
```

Sample records:

```
INSERT INTO employees (Id, First_name, Last_name, Salary, Department)
VALUES
```

```
(1, 'John', 'Doe', 50000, 'HR'),
(2, 'Jane', 'Smith', 60000, 'Finance'),
(3, 'Alice', 'Johnson', 70000, 'IT'),
(4, 'Bob', 'Williams', 80000, 'IT'),
(5, 'Charlie', 'Brown', 40000, 'HR');
```

Sol:



The screenshot shows a SQL Developer window with a query editor and a results pane. The query editor contains the following SQL code:

```
INSERT INTO Employees (ID, First_name, Last_name, Salary, Department)
VALUES (1, 'John', 'Doe', 50000, 'HR'), (2, 'Jane', 'Smith', 60000, 'Finance'), (3, 'Alice', 'Johnson', 70000, 'IT'),
(4, 'Bob', 'Williams', 80000, 'IT'), (5, 'Charlie', 'Brown', 40000, 'HR');

SELECT * FROM Employees;
```

The results pane shows the output of the SELECT statement, displaying a table with 5 rows and 5 columns: Id, First\_name, Last\_name, Salary, and Department.

	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

## QUESTIONS:

1.

Sol:

→ Create a new table "Contractors" and add values into it.

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the database structure for 'LKUMAR (SQL Server 16.0.1000.6)'. The 'Tables' folder is expanded, showing 'dbo.Contractors'. The main window displays the SQL script for creating the 'Contractors' table and inserting data. The script is as follows:

```
USE DAMG6210_HW
GO
CREATE TABLE Contractors (
    Id INT PRIMARY KEY,
    First_name VARCHAR(50),
    Last_name VARCHAR(50),
    Salary DECIMAL(10, 2),
    Department VARCHAR(50)
);
INSERT INTO Contractors (Id, First_name, Last_name, Salary, Department)
VALUES (6, 'Lakshmi', 'Kumar', 90000, 'HR'), (7, 'Sanskriti', 'Manoria', 40000, 'Finance'), (8, 'Monika', 'Gundecha', 60000, 'IT'),
(9, 'Nivetha', 'Kumar', 70000, 'IT'), (10, 'Kumar', 'Rajasubramanian', 40000, 'HR'), (11, 'Jayashree', 'Kumar', 90000, 'HR');
SELECT * FROM Contractors;
```

The 'Results' pane shows the data inserted into the 'Contractors' table:

Id	First_name	Last_name	Salary	Department
6	Lakshmi	Kumar	90000.00	HR
7	Sanskriti	Manoria	40000.00	Finance
8	Monika	Gundecha	60000.00	IT
9	Nivetha	Kumar	70000.00	IT
10	Kumar	Rajasubramanian	40000.00	HR
11	Jayashree	Kumar	90000.00	HR

→ Query to insert data into the 'Employees' table from another table 'Contractors' with the same structure, excluding those in the 'HR' department.

INSERT INTO Employees

SELECT \* FROM Contractors WHERE [Department] != 'HR'

SELECT \* FROM Employees;

(Only records with Id 7, 8, 9 have been inserted from Contractors to Employees. Records with Id 6, 10, 11 have not been inserted since those records have the value HR in column Department)

The screenshot shows the SQL Server Enterprise Manager interface. The main window displays the SQL script for inserting data from 'Contractors' into 'Employees'. The script is as follows:

```
INSERT INTO Employees
SELECT * FROM Contractors WHERE [Department] != 'HR'
SELECT * FROM Employees;
```

The 'Results' pane shows the data inserted into the 'Employees' table:

Id	First_name	Last_name	Salary	Department
1	John	Doe	50000.00	HR
2	Jane	Smith	60000.00	Finance
3	Alice	Johnson	70000.00	IT
4	Bob	Williams	80000.00	IT
5	Charlie	Brown	40000.00	HR
6	Sanskriti	Manoria	40000.00	Finance
7	Monika	Gundecha	60000.00	IT
8	Nivetha	Kumar	70000.00	IT

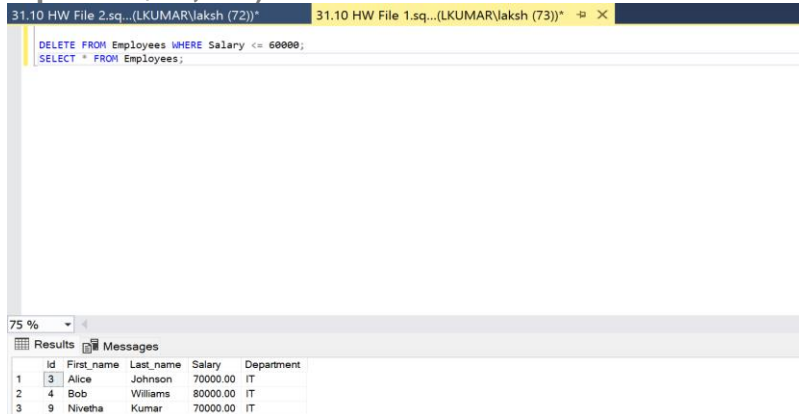
2.

Sol:

→ Query to delete employees from the 'Employees' table which have a salary less or equal to \$ 60000.

```
DELETE FROM Employees WHERE Salary <= 60000;  
SELECT * FROM Employees;
```

(Only records with Id 3, 4, 9 will be displayed from Employees. Records with Id 1,2,5,6,7 have not been displayed since those records have salary less than or equal to \$60,000)



The screenshot shows a SQL Developer window with two tabs. The active tab is '31.10 HW File 2.sql... (LKUMAR\laksh (72))'. It contains the following SQL code:

```
DELETE FROM Employees WHERE Salary <= 60000;  
SELECT * FROM Employees;
```

Below the code, the 'Results' pane shows the output of the SELECT statement. It displays a table with 5 columns: Id, First\_name, Last\_name, Salary, and Department. The table contains 3 rows of data.

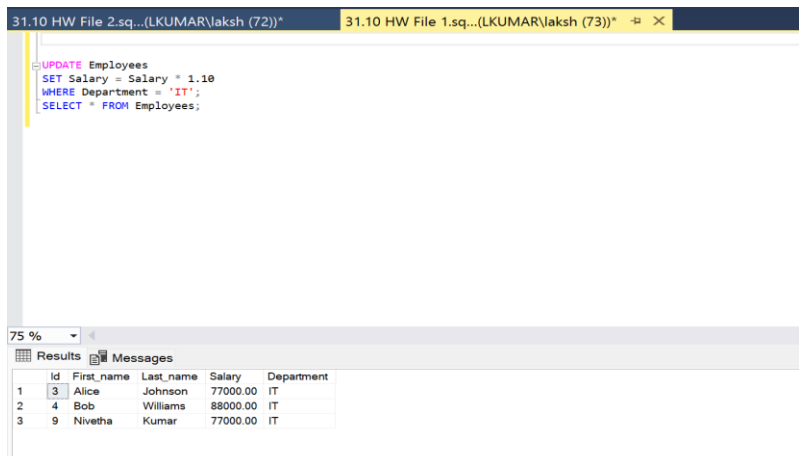
Id	First_name	Last_name	Salary	Department
3	Alice	Johnson	70000.00	IT
4	Bob	Williams	80000.00	IT
9	Nivetha	Kumar	70000.00	IT

3.

Sol:

→ Query to update the salary of employees in the 'IT' department by increasing it by 10%

```
UPDATE Employees  
SET Salary = Salary * 1.10  
WHERE Department = 'IT';  
SELECT * FROM Employees;
```



The screenshot shows a SQL Developer window with two tabs. The active tab is '31.10 HW File 2.sql... (LKUMAR\laksh (72))'. It contains the following SQL code:

```
UPDATE Employees  
SET Salary = Salary * 1.10  
WHERE Department = 'IT';  
SELECT * FROM Employees;
```

Below the code, the 'Results' pane shows the output of the SELECT statement. It displays a table with 5 columns: Id, First\_name, Last\_name, Salary, and Department. The table contains 3 rows of data.

Id	First_name	Last_name	Salary	Department
3	Alice	Johnson	77000.00	IT
4	Bob	Williams	88000.00	IT
9	Nivetha	Kumar	77000.00	IT

Adding more values for Querying purpose:

31.10 HW File 2.sql...(LKUMAR\laksh (72))\* 31.10 HW File 1.sql...(LKUMAR\laksh (73))\*

```
INSERT INTO Employees (ID, First_name, Last_name, Salary, Department)
VALUES (6, 'Lakshmi', 'Kumar', 90000, 'HR'), (7, 'Sanskriti', 'Manoria', 40000, 'Finance'), (8, 'Monika', 'Gundecha', 60000, 'IT'),
(10, 'Kumar', 'Rajasubramanian', 40000, 'HR'), (11, 'Jayashree', 'Kumar', 90000, 'HR'), (5, 'Johnson', 'Mathew', 20000, 'HR');
SELECT * FROM Employees;
```

75 %

Results Messages

	Id	First_name	Last_name	Salary	Department
1	3	Alice	Johnson	77000.00	IT
2	4	Bob	Williams	88000.00	IT
3	5	Johnson	Mathew	20000.00	HR
4	6	Lakshmi	Kumar	90000.00	HR
5	7	Sanskriti	Manoria	40000.00	Finance
6	8	Monika	Gundecha	60000.00	IT
7	9	Nivetha	Kumar	77000.00	IT
8	10	Kumar	Rajasubramanian	40000.00	HR
9	11	Jayashree	Kumar	90000.00	HR

4.

Sol:

➔ Query to display all columns with the top 3 highest salaries from the 'Employees' table

```
SELECT TOP 3 Salary
FROM Employees
ORDER BY Salary DESC;
```

31.10 HW File 2.sql...(LKUMAR\laksh (72))\* 31.10 HW File 1.sql...(LKUMAR\laksh (73))\*

```
SELECT TOP 3 Salary
FROM Employees
ORDER BY Salary DESC;
```

75 %

Results Messages

	Salary
1	90000.00
2	90000.00
3	88000.00

➔ Query to display all columns with the top 3 highest salaries from the 'Employees' table

```
SELECT TOP 3 *  
FROM Employees  
ORDER BY Salary DESC;
```

The screenshot shows a SQL Server Enterprise Manager interface. At the top, there are two tabs: '31.10 HW File 2.sql...(LKUMAR\laksh (72))\*' and '31.10 HW File 1.sql...(LKUMAR\laksh (73))\*'. The active window displays the following SQL query:

```
SELECT TOP 3 *  
FROM Employees  
ORDER BY Salary DESC;
```

Below the query window, there is a 'Results' pane showing the output of the query. The results are displayed in a table with the following columns: Id, First\_name, Last\_name, Salary, and Department. The table contains three rows of data:

	Id	First_name	Last_name	Salary	Department
1	6	Lakshmi	Kumar	90000.00	HR
2	11	Jayashree	Kumar	90000.00	HR
3	4	Bob	Williams	88000.00	IT

5.

Sol:

Query to select employees whose first name starts with 'J' and ends with 'n'

```
SELECT * FROM Employees  
WHERE First_name LIKE 'J%n'
```

The screenshot shows a SQL Server Enterprise Manager interface. At the top, there are two tabs: '31.10 HW File 2.sql...(LKUMAR\laksh (72))\*' and '31.10 HW File 1.sql...(LKUMAR\laksh (73))\*'. The active window displays the following SQL query:

```
SELECT * FROM Employees  
WHERE First_name LIKE 'J%n';
```

Below the query window, there is a 'Results' pane showing the output of the query. The results are displayed in a table with the following columns: Id, First\_name, Last\_name, Salary, and Department. The table contains one row of data:

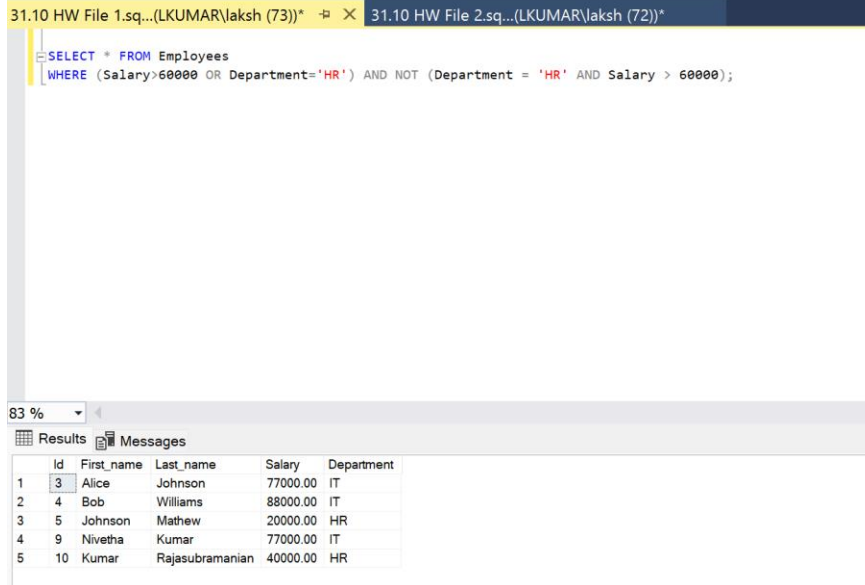
	Id	First_name	Last_name	Salary	Department
1	5	Johnson	Mathew	20000.00	HR

6.

Sol:

Query to select employees who are either in the 'HR' department or have a salary greater than 60000, but not both.

```
SELECT * FROM Employees
WHERE (Salary>60000 OR Department='HR') AND NOT (Department = 'HR'
AND Salary > 60000);
```



The screenshot shows a SQL Developer window with a query editor and a results pane. The query editor contains the following SQL statement:

```
SELECT * FROM Employees
WHERE (Salary>60000 OR Department='HR') AND NOT (Department = 'HR' AND Salary > 60000);
```

The results pane displays a table with 5 rows and 6 columns: Id, First\_name, Last\_name, Salary, Department. The data is as follows:

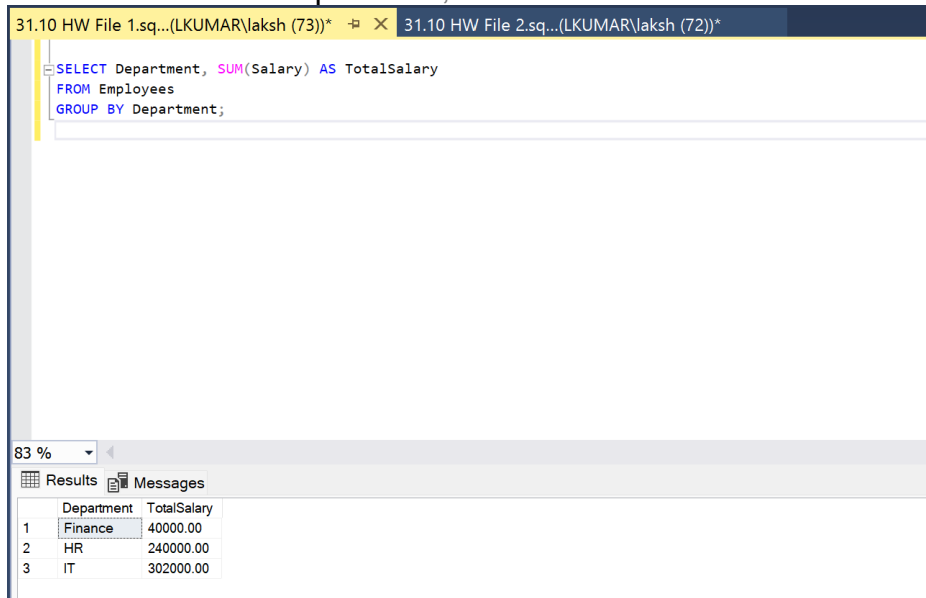
Id	First_name	Last_name	Salary	Department
3	Alice	Johnson	77000.00	IT
4	Bob	Williams	88000.00	IT
5	Johnson	Mathew	20000.00	HR
9	Nivetha	Kumar	77000.00	IT
10	Kumar	Rajasubramanian	40000.00	HR

7.

Sol:

Query to you get the total salary for each department using the `Employees` table?

```
SELECT Department, SUM(Salary) AS TotalSalary
FROM Employees
GROUP BY Department;
```



The screenshot shows a SQL Developer window with a query editor and a results pane. The query editor contains the following SQL statement:

```
SELECT Department, SUM(Salary) AS TotalSalary
FROM Employees
GROUP BY Department;
```

The results pane displays a table with 3 rows and 2 columns: Department, TotalSalary. The data is as follows:

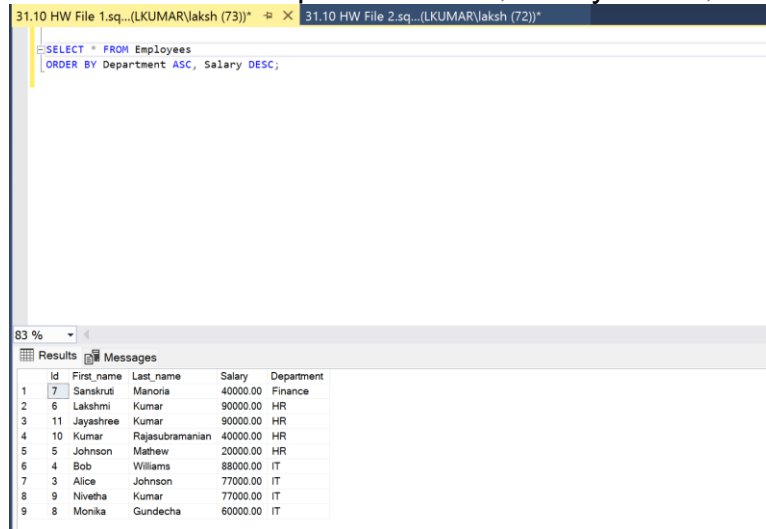
Department	TotalSalary
Finance	40000.00
HR	240000.00
IT	302000.00

8.

Sol:

Query to select employees from the `Employees` table, sorted by department in ascending order, and then by salary in descending order within each department.

```
SELECT * FROM Employees  
ORDER BY Department ASC, Salary DESC;
```



The screenshot shows a SQL query window with the query: `SELECT * FROM Employees ORDER BY Department ASC, Salary DESC;`. Below the query, the results are displayed in a table with columns: Id, First\_name, Last\_name, Salary, and Department. The results are sorted by department (Finance, HR, IT) and then by salary in descending order within each department.

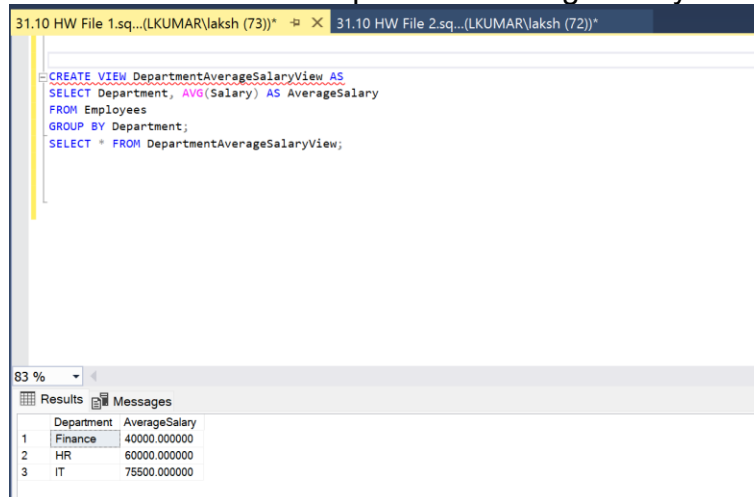
Id	First_name	Last_name	Salary	Department
7	Sanskriti	Manoria	40000.00	Finance
6	Lakshmi	Kumar	90000.00	HR
11	Jayashree	Kumar	90000.00	HR
10	Kumar	Rajsubramanian	40000.00	HR
5	Johnson	Mathew	20000.00	HR
4	Bob	Williams	88000.00	IT
3	Alice	Johnson	77000.00	IT
9	Nivetha	Kumar	77000.00	IT
8	Monika	Gundecha	60000.00	IT

9.

Sol:

Query to create a view that shows the average salary for each department from the employees` table?

```
CREATE VIEW DepartmentAverageSalaryView AS  
SELECT Department, AVG(Salary) AS AverageSalary  
FROM Employees  
GROUP BY Department;  
SELECT * FROM DepartmentAverageSalaryView;
```



The screenshot shows a SQL query window with the query: `CREATE VIEW DepartmentAverageSalaryView AS SELECT Department, AVG(Salary) AS AverageSalary FROM Employees GROUP BY Department; SELECT * FROM DepartmentAverageSalaryView;`. Below the query, the results are displayed in a table with columns: Department and AverageSalary. The results show the average salary for each department: Finance (40000.000000), HR (60000.000000), and IT (75500.000000).

Department	AverageSalary
Finance	40000.000000
HR	60000.000000
IT	75500.000000



10. What is a strategy to optimize a query that frequently retrieves department and average salary columns data from the `employees` table?

Sol:

Strategies to optimize a query that frequently retrieves department and average salary columns data from the Employees table.

- ➔ Maintenance: Ensure indexes are maintained regularly, indexes are not fragmented and also that queries run in a systematic manner.
- ➔ Caching: Implement a cache to store the department and average salary columns data in order to reduce load on the database.
- ➔ Usage of appropriate hardware: Make sure database server has the right type of hardware (i.e.) CPU, Memory, Storage etc.
- ➔ Denormalizing the data: In case the data wouldn't have to be changed frequently, denormalize the data and store average salary in a separate table directly. In this way, frequent calculations can be eliminated.
- ➔ Indexing: Look out if the table has been indexed properly. Make sure to put an index on the column "Department" in order to speed up grouping and retrieval of data.
- ➔ Partitioning: If the table is extremely large, partition the table. This helps deal with poor performance in case of frequent querying or retrieval.