📘 **SQL Assignment**

**Section A – Leet Code Questions**

In this section, you will practice solving **query-based challenges** inspired by real interview-style

problems. These focus on applying SQL concepts like joins, filtering, and conditional logic to

small but practical datasets.

1️⃣**Replace Employee ID With The Unique Identifier   
ALTER TABLE NAME**

**RENAME EMPLOYEE ID TO EMPLOYEE TAG**

2️⃣**Product Sales Analysis I**

3️⃣**Customer Who Visited but Did Not Make Any Transactions**

**SELECT CUSTOMERID, NAME FROM TABLE**

**WHERE STATUS = ‘VISTED’**

**AND TRANSACTION IS NULL;**

4️⃣**Employee Bonus**

**SELECT EMPLOYEE ID, NAME, SALARY, SALARY\*0.10 AS BONUS FROM TABLE**

**Section B – Adventure Works Table**

This section uses the **Adventure Works sample database**, which is widely used for SQL

learning. You will analyze business-related data such as **customer orders and product**

**pricing**

5️⃣**SalesLT. Sales Order Header**

● Find the top 5 customers by total number of orders placed.   
select top (5) customerID, sum(TotalDue) as TotalNumber from [AdventureWorksLT2017].[SalesLT].[SalesOrderHeader]

GROUP by CustomerID order by sum(TotalDue) DESC ;

● Find customers who placed at least 3 orders and have spent more than 2000 in total.   
select CustomerID from [AdventureWorksLT2017].[SalesLT].[SalesOrderHeader]

group by CustomerID

Having count(PurchaseOrderNumber) >= 3

AND sum(TotalDue) > 2000;

6️⃣**SalesLT. Product**● List all products that have a List Price greater than the average List Price.   
SELECT PRODUCTID, Name, AVG(ListPrice) AS AVGPRICE

FROM [AdventureWorksLT2017].[SalesLT].[PRODUCT]

WHERE ListPrice > (select AVG(ListPrice) from [AdventureWorksLT2017].[SalesLT].[PRODUCT]);

● Find the product categories (Product Category ID) that have more than 5 products

with a ListPrice greater than 1000.   
SELECT ProductCategoryID FROM [AdventureWorksLT2017].[SalesLT].[PRODUCT]

WHERE ListPrice > 1000

GROUP BY ProductCategoryID

HAVING COUNT(\*) > 5 ;

**Section C SQL Hands on**

In this part, you will **design and populate your own table** called Employees. You will:

● Create a table with multiple columns.

● Insert at least 15 records of your choice (ensuring diversity across departments).

● Write queries using aggregation (AVG, SUM, MAX), GROUP BY, and HAVING to analyze

the dataset.

**Q1.** Create a table named Employees with the following columns:   
CREATE TABLE Empolyees(

EmpID INT PRIMARY KEY,

EmpName VARCHAR(50),

Department VARCHAR(50),

Salary DECIMAL(10,2),

YearsOfService int

);

● Emp ID (INT, Primary Key)

● Emp Name (VARCHAR(50))

● Department (VARCHAR(50))

● Salary (DECIMAL(10,2))

● Years Of Service (INT)

**Q2.** Insert at least **15 records of your own choice** into the Employees table.   
insert into Empolyees(EmpID, EmpName, Department, Salary, YearsOfService)

VALUES

(1, 'Ismita kC', 'Marketing', 75000.00, 5),

(2, 'John Doe', 'IT', 85000.00, 7),

(3, 'Jane Smith', 'Finance', 95000.00, 10),

(4, 'Michael Brown', 'HR', 65000.00, 3),

(5, 'Emily Davis', 'IT', 78000.00, 6),

(6, 'Robert Wilson', 'Marketing', 72000.00, 4),

(7, 'Linda Johnson', 'Finance', 88000.00, 8),

(8, 'James Taylor', 'HR', 60000.00, 2),

(9, 'Patricia Moore', 'IT', 82000.00, 5),

(10, 'David Anderson', 'Marketing', 77000.00, 6),

(11, 'Barbara Thomas', 'Finance', 91000.00, 9),

(12, 'William Jackson', 'HR', 63000.00, 3),

(13, 'Susan White', 'IT', 87000.00, 7),

(14, 'Charles Harris', 'Marketing', 76000.00, 4),

(15, 'Jessica Martin', 'Finance', 94000.00, 10);

👉 Make sure you include employees from at least **3 different departments** (e.g., IT, HR,

Finance, Sales, etc.).

**Q3.** From the Employees table, find the **average salary in each department**.   
SELECT Department, avg(Salary) as AverageSalary from Empolyees

group by Department;

**Q4.** From the Employees table, find the **total salary expenditure by each department**.   
SELECT Department, sum(Salary) as TotalSalary from Empolyees

group by Department;

**Q5.** List the **departments that have more than 3 employees**.   
SELECT Department, EmpName, count(\*) from Empolyees

group by Department

HAVING COUNT(\*) > 3;

**Q6.** Find the **employee(s) with the maximum years of service in each department**.

SELECT Department, MAX(YearsOfService) as MaximumYearOfService from Empolyees

group by Department