**Exact Numeric**

**1. Scenario: Employee Salary Management**

* **Problem:** You have an **Employees** table with columns **EmployeeID** (int), **Name** (varchar), **BaseSalary** (decimal(10, 2)), and **Bonus** (decimal(10, 2)). Write a query to calculate the **TotalSalary** for each employee, which is the sum of **BaseSalary** and **Bonus**. Return **EmployeeID**, **Name**, and **TotalSalary**.
* **Task:** Write the SQL query to achieve this.

**Solution - select EmployeeID, Name, (BaseSalary + Bonus) as TotalSalary**

**from Employees;**

**--------------------------------------------------------------------------------------------------------------------------**

**2. Scenario: Product Pricing Analysis**

* **Problem: In the Products table, there are columns ProductID (int), ProductName (varchar), Price (decimal(8, 2)), and QuantityInStock (int). Write a query to find all products where the total value in stock (calculated as Price \* QuantityInStock) is greater than $1,000. Return ProductID, ProductName, and the calculated TotalValue.**
* **Task: Write the SQL query to retrieve these products.**

**Solution - select ProductID, ProductName, (Price \* QuantityInStock) as TotalValue**

**from Products**

**where (Price \* QuantityInStock) > 1000;**

**--------------------------------------------------------------------------------------------------------------------------**

**3. Scenario: Student Grades Calculation**

* **Problem: The Grades table contains columns StudentID (int), Subject (varchar), and Score (numeric(5, 2)). Write a query to find the average score of each student across all subjects. Return StudentID and the calculated AverageScore rounded to two decimal places.**
* **Task: Write the SQL query to calculate the average score.**

**Solution - select StudentID, round(avg(Score), 2) as AverageScore**

**from Grades**

**group by StudentID;**

**--------------------------------------------------------------------------------------------------------------------------**

**4. Scenario: Financial Transactions Overview**

* **Problem: In the Transactions table, you have columns TransactionID (int), Amount (decimal(12, 2)), and TransactionDate (datetime). Write a query to find all transactions where the amount is negative (indicating expenses) and occurred in the last 30 days. Return TransactionID, Amount, and TransactionDate.**
* **Task: Write the SQL query to retrieve the transaction details.**

**Solution - select TransactionID, Amount, TransactionDate**

**from Transactions**

**where Amount < 0**

**and TransactionDate >= dateadd(day, -30, getdate());**

**--------------------------------------------------------------------------------------------------------------------------**

**5. Scenario: Inventory Cost Calculation**

* **Problem: The Inventory table includes columns ItemID (int), ItemName (varchar), CostPerItem (decimal(10, 2)), and StockQuantity (int). Write a query to calculate the total inventory cost for each item (calculated as CostPerItem \* StockQuantity). Return ItemID, ItemName, and the calculated TotalCost.**

**Task: Write the SQL query to determine the total inventory cost.**

**Solution - select ItemID, ItemName, (CostPerItem \* StockQuantity) as TotalCost**

**from Inventory;**

**--------------------------------------------------------------------------------------------------------------------------**

**Scenario: Employee Compensation Analysis**

* **Problem: You have an Employees table with columns EmployeeID (int), Name (varchar), BaseSalary (decimal(10, 2)), Bonus (decimal(10, 2)), and Deductions (decimal(10, 2)). Write a query to calculate the NetSalary for each employee, which is defined as BaseSalary + Bonus - Deductions. Return EmployeeID, Name, and the calculated NetSalary for employees with a NetSalary greater than $50,000.**
* **Task: Write the SQL query to achieve this.**

**Solution - select EmployeeID, Name, (BaseSalary + Bonus - Deductions) as NetSalary**

**from Employees**

**where (BaseSalary + Bonus - Deductions) > 50000;**

**--------------------------------------------------------------------------------------------------------------------------**

**Scenario: Product Profitability Analysis**

* **Problem: In the Products table, there are columns ProductID (int), ProductName (varchar), CostPrice (decimal(10, 2)), SellingPrice (decimal(10, 2)), and QuantitySold (int). Write a query to determine the total profit for each product, calculated as (SellingPrice - CostPrice) \* QuantitySold. Return ProductID, ProductName, and the calculated TotalProfit for products where the TotalProfit is greater than $1,000.**

**Task: Write the SQL query to retrieve these products.**

**Solution - select ProductID, ProductName, (SellingPrice - CostPrice) \* QuantitySold as**

**TotalProfit**

**from Products**

**where (SellingPrice - CostPrice) \* QuantitySold > 1000;**

**--------------------------------------------------------------------------------------------------------------------------**

**Scenario: Student Performance Evaluation**

* **Problem: The Students table contains columns StudentID (int), Name (varchar), and GPA (decimal(3, 2)). Write a query to find students who have a GPA greater than or equal to 3.5. Additionally, calculate the number of credits they completed based on the following conditions: if GPA is 3.5 or higher, they have completed at least 30 credits; if GPA is less than 3.5, they have completed 15 credits. Return StudentID, Name, and calculated CreditsCompleted.**
* **Task: Write the SQL query to analyze student performance.**

**Solution - select StudentID, Name,**

**case**

**when GPA >= 3.5 then 30**

**else 15**

**end as CreditsCompleted**

**from Students**

**where GPA >= 3.5;**

**--------------------------------------------------------------------------------------------------------------------------**

**Scenario: Transaction Fee Calculation**

* **Problem: In the Transactions table, you have columns TransactionID (int), TransactionAmount (decimal(12, 2)), and TransactionDate (datetime). Write a query to calculate a transaction fee of 2% on all transactions that occurred in the last 60 days and had a transaction amount greater than $500. Return TransactionID, TransactionAmount, and the calculated TransactionFee.**
* **Task: Write the SQL query to retrieve the transaction fee details.**

**Solution - select TransactionID, TransactionAmount, TransactionAmount \* 0.02 as**

**TransactionFee**

**From Transactions**

**where TransactionDate >= dateadd(day, -60, getdate())**

**and TransactionAmount > 500;**

**--------------------------------------------------------------------------------------------------------------------------**

**Scenario: Inventory Turnover Ratio Calculation**

* **Problem: The Inventory table includes columns ItemID (int), ItemName (varchar), CostPerItem (decimal(10, 2)), StockQuantity (int), and TotalSales (decimal(12, 2)). Write a query to calculate the InventoryTurnoverRatio, which is defined as TotalSales / (CostPerItem \* StockQuantity). Return ItemID, ItemName, and the calculated InventoryTurnoverRatio for items with a ratio greater than 2.**
* **Task: Write the SQL query to determine the inventory turnover ratio.**

**Solution – select ItemID, ItemName, TotalSales / (CostPerItem \* StockQuantity) as**

**InventoryTurnoverRatio**

**from Inventory**

**where (CostPerItem \* StockQuantity) > 0**

**and (TotalSales / (CostPerItem \* StockQuantity)) > 2;**