# Database Basics MS SQL Retake Exam – 09 April 2025

Exam problems for the ["Database Basics" course @ SoftUni](https://softuni.bg/trainings/4841/ms-sql-january-2025).  
Submit your solutions in the SoftUni Judge system at [Judge](https://judge.softuni.org/Contests/5360/MS-SQL-Retake-Exam-8-April-2025).

# Tech Store

# Section 1. DDL (30 pts)

You have been given the E/R Diagram of the **TechStore** database.



Create a database called **TechStore**. You need to create **7 tables**:

* **Manufacturers** – Contains information about the manufacturers that produce tech products;
* **Categories** – Contains information about product categories. Each product belongs to a category;
* **Products** – Contains information about individual tech products, including their type, price, and specifications;
* **Stores**– Contains information about physical or online stores where products are sold. Each store operates within a specific product category;
* **Customers** – Contains information about customers who purchase products from the stores;
* **Orders** – Contains information about customer orders, including order date, product, and store details;
* **StoresProducts** – A mapping table between stores and the products they offer;

**NOTE: Keep in mind that Judge doesn't accept "ALTER" statement and square brackets naming (when the names are not keywords).**

**NOTE: Please keep in mind that in case you have to work with a date, you have to use the exact same data type, described in the models tables. If you don't use the correct type, the Judge system won't accept your submission as correct.**

You have been tasked to **create the tables in the database by the following models**:

### ****Manufacturers****

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| **Id** | **Integer** from **0** to **2,147,483,647** | **PK**, **Unique** table identification, **Identity** |
| **Name** | **String** up to **50** symbols, **Unicode** | **Null** is **not** allowed |
| **Country** | **String** up to **50** symbols, **Unicode** | **Null** is **not** allowed |

### Categories

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| **Id** | **Integer** from **0** to **2,147,483,647** | **PK**, **Unique** table identification, **Identity** |
| **Name** | **String** up to 3**0** symbols, **Unicode** | **Null** is **not** allowed, **Unique** |

### Products

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| **Id** | **Integer** from **0** to **2,147,483,647** | **PK**, **Unique** table identification, **Identity** |
| **Name** | **String** up to **100** symbols, **Unicode** | **Null** is **not** allowed |
| **Price** | **Decimal**, up to **18 digits**,  **2 after decimal point** | **Null** is **not** allowed |
| **Specifications** | **String** up to **1000** symbols, **Unicode** | **Null** is **allowed** |
| **ManufacturerId** | **Integer** from **0** to **2,147,483,647** | **Relationship** with table **Manufacturers**,  **Null** is **not** allowed |
| **CategoryId** | **Integer** from **0** to **2,147,483,647** | **Relationship** with table **Categories**,  **Null** is **not** allowed |

### ****Stores****

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| **Id** | **Integer** from **0** to **2,147,483,647** | **PK**, **Unique** table identification, **Identity** |
| **Name** | **String** up to **100** symbols, **Unicode** | **Null** is **not** allowed |
| **IsOnline** | **Boolean (bit)** | **Null** is **not** allowed |

### ****StoresProducts****

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| **StoreId** | **Integer** from **0** to **2,147,483,647** | **PK**, **Unique** table identification, **Relationship** with table **Stores**, **Null** is **not** allowed |
| **ProductId** | **Integer** from **0** to **2,147,483,647** | **PK**, **Unique** table identification, **Relationship** with table **Products**, **Null** is **not** allowed |

### ****Customers****

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| **Id** | **Integer** from **0** to **2,147,483,647** | **PK**, **Unique** table identification, **Identity** |
| **Name** | **String** up to **80** symbols, **Unicode** | **Null** is **not** allowed |
| **PhoneNumber** | **String** up to **20** symbols, **Unicode** | **Null** is **not** allowed |
| **Email** | **String** up to **80** symbols, **Unicode** | **Null** is **allowed** |

### ****Orders****

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| **Id** | **Integer** from **0** to **2,147,483,647** | **PK**, **Unique** table identification, **Identity** |
| **OrderDate** | **DateTime2** | **Null** is **not** allowed |
| **CustomerId** | **Integer** from **0** to **2,147,483,647** | **Relationship** with table **Customers**,  **Null** is **not** allowed |
| **ProductId** | **Integer** from **0** to **2,147,483,647** | **Relationship** with table **Products**,  **Null** is **not** allowed |
| **StoreId** | **Integer** from **0** to **2,147,483,647** | **Relationship** with table **Stores**,  **Null** is **not** allowed |

## Database design

Submit all of yours **CREATE** **statements** to Judge (only the creation of tables).

# Section 2. DML (10 pts)

**Before you start, you have to import "DataSet.sql". If you have created the structure correctly, the data should be successfully inserted.**

In this section, you have to do some data manipulations:

## Insert

Let's **insert** some sample data into the database. Write a **query to add the following records** into the corresponding tables. All IDs (**Primary Keys**) should be **auto-generated**.

### ****Customers****

|  |  |  |
| --- | --- | --- |
| **Name** | **PhoneNumber** | **Email** |
| Marko Petrovic | 0888-123456 | marko.petrovic@email.rs |

### ****Products****

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Price** | **ManufacturerId** | **CategoryID** |
| Asus ZenBook 14 | 1199.99 | 6 | 2 |
| Sony WF-1000XM5 | 299.99 | 10 | 10 |

### ****StoresProducts****

|  |  |
| --- | --- |
| **StoreId** | **ProductId** |
| 2 | 61 |
| 12 | 62 |

### ****Orders****

|  |  |  |  |
| --- | --- | --- | --- |
| **OrderDate** | **CustomerId** | **ProductId** | **StoreId** |
| 2025-03-04 | 21 | 61 | 2 |
| 2024-12-20 | 21 | 62 | 12 |
| 2025-01-25 | 18 | 14 | 1 |
| 2025-02-26 | 7 | 31 | 20 |

## Update

**Increase Prices of Affordable Products**

In this task, you will update the Products table by **increasing the prices of all products** whose price is **below $500.00**.  
The **increase** should be **15% of the current price**.

## Delete

**Delete Customers Without Email (and Their Orders)**

In this task, you will remove all customers from the Customers table who **hadn’t provided an email address**.  
If these customers have placed any orders, those orders must be deleted first to preserve referential integrity.

# Section 3. Querying (40 pts)

**You need to start with a fresh dataset, so recreate your DB and import the sample data again ("DataSet.sql**"**).**

## Find Recent Orders

Write a SQL query to retrieve all orders that were placed **after January 1st, 2025**.

Required columns:

* **OrderId**
* **OrderDate** (formatted as **'MM-dd'**)
* **CustomerId**
* **StoreId**
* **ProductId**

Additional requirements:

* Only include orders with a date **after** **'2025-01-01'**
* Sort the results by:
  + OrderDate in **descending** order
  + Then by CustomerId in **ascending** order
  + Then by StoreId in **ascending** order
  + Then by ProductId in **ascending** order

### Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **OrderId** | **OrderDate** | **CustomerId** | **StoreId** | **ProductId** |
| 60 | 04-10 | 20 | 3 | 59 |
| 59 | 04-09 | 19 | 4 | 10 |
| 58 | 04-09 | 20 | 20 | 13 |
| 57 | 04-08 | 19 | 1 | 1 |
| 55 | 04-08 | 19 | 7 | 18 |
| 56 | 04-08 | 19 | 15 | 42 |
| 52 | 04-07 | 12 | 18 | 37 |
| ... | ... | ... | ... | ... |

## Manufacturers From Specific Countries

Write a SQL query to retrieve the **name** and **country** of all manufacturers whose country name **starts with the letter "S"**. The results must be sorted by the **country’s** **name in ascending order, then by Name in ascending order**.

Required columns:

* **Manufacturer**
* **Country**

### Example

|  |  |
| --- | --- |
| **Manufacturer** | **Country** |
| Razer | Singapore |
| LG | South Korea |
| Samsung | South Korea |

## Customers Ordered Products Produced in China

Write a SQL query to retrieve the **names and emails** of all customers who have ordered   
**products manufactured in China**.

* **Return each customer only once**, even if they ordered multiple such products
* **Only** include **customers** who have a **non-null email**
* Sort the results by the customer’s **name in ascending order**

Required columns:

* **CustomerName**
* **PhoneNumber**
* **Email**

### Example

|  |  |  |
| --- | --- | --- |
| **CustomerName** | **PhoneNumber** | **Email** |
| Daan van Dijk | 020-1234567 | daan.vd@email.nl |
| Javier Gómez | 91-4567890 | j.gomez@email.es |
| Luca Bianchi | 089-1234567 | l.bianchi@email.it |
| Thomas Meier | 040-5566778 | th.meier@email.de |

## Customers with Multiple Orders

Write a SQL query to retrieve a list of **customers who have made more than one order**.

* Only include customers who have made **more than one order**
* Sort the result by the number of orders **in descending order**, then by **name** in ascending

Required columns:

* **CustomerName**
* **OrdersCount**

### Example

|  |  |
| --- | --- |
| **CustomerName** | **OrdersCount** |
| Carlos Fernández | 6 |
| Thomas Meier | 6 |
| Daan van Dijk | 5 |
| Isabelle Laurent | 5 |
| ... | ... |

## Average Price of Products Produced by Country

Write a SQL query to calculate the **average price of all products** grouped by the **manufacturer’s country**, and also display the **number of manufacturers** from each country.

* **Format** the **average price** to **2 decimal places – the 'N2' format should be applied**
* **Sort** the result by:
  + **ProducersCount** (descending)
  + Then by **AvgPrice** (ascending)

Required columns:

* **CountryName**
* **ProducersCount**
* **AvgProductPrice**

### Example

|  |  |  |
| --- | --- | --- |
| **CountryName** | **ProducersCount** | **AvgProductPrice** |
| USA | 5 | 1,075.94 |
| Taiwan | 3 | 487.90 |
| China | 3 | 813.99 |
| South Korea | 2 | 742.69 |
| Singapore | 1 | 127.49 |
| Japan | 1 | 686.66 |

## High-Value Store Analysis

The TechStore management team is interested in identifying **which stores are** performing well by **offering high-priced products**. As part of your analysis, write a SQL query that lists only those stores that sell **high-value products** and meet specific criteria.

Write a SQL query that:

* Retrieves **stores** that sell **products priced above $800**
* For each store, calculates:
  + The **number of high-value products** it sells
  + The **average price** of these products
* Only includes stores that offer **at least 4 products above $800**
  + **Order** the final result by **AveragePrice** (descending),
* **Format** the **average** **price** to **2 decimal places – the 'N2' format should be applied**

Required columns:

* **StoreName**
* **ProductCount**
* **AveragePrice**

### Example

|  |  |  |
| --- | --- | --- |
| **StoreName** | **ProductCount** | **AveragePrice** |
| ByteBazaar | 4 | 2,074.74 |
| ElectroMart | 4 | 1,974.50 |
| Digital Galaxy | 10 | 1,619.30 |
| Tech World | 4 | 1,024.74 |

# Section 4. Programmability (20 pts)

## Count Products by Manufacturer

Write a **User-Defined Function** called: udf\_GetProductCountByManufacturer

It should receive a **manufacturer name** as a parameter and returns the **number of products** produced by that manufacturer.

The function should:

* Take a single parameter: the name of the manufacturer (as **NVARCHAR(50)**)
* Return the **count of products** (as **INT**) that belong to the given manufacturer
* The function should search for an **exact match** on the manufacturer’s name
* If **no manufacturer** is found, **return 0**

### Examples

|  |
| --- |
| **Query** |
| **SELECT dbo.udf\_** GetProductCountByManufacturer **('Apple')** |
| **Example Output** |
| **ProductsCount** |
| **6** |

## Get Orders by Customer

Create a stored procedure named usp\_GetOrdersByCustomer

It should accept a **customer's name** and **returns all orders made by that customer**.

* ProductName
* StoreName
* OrderDate (formatted as 'MM-dd-yyyy')
* Price (price formatted to 2 decimal places – the 'N2' format should be applied)

Results must be **ordered by OrderDate descending**, then by **ProductName ascending**

### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **Query** | | | |
| **EXEC usp\_GetOrdersByCustomer 'Carlos Fernández'** | | | |
| **Output** | | | |
| **ProductName** | **StoreName** | **Orderdate** | **Price** |
| Nintendo Switch OLED | TechExpress | 04-07-2025 | 349.99 |
| Sony SRS-XB43 | SmartTech Pro | 04-06-2025 | 199.99 |
| Xiaomi Mi Smart Band 7 | Xiaomi Central | 04-03-2025 | 59.99 |
| Razer Huntsman Mini | NovaTech Online | 04-02-2025 | 119.99 |
| Galaxy S23 Ultra | Samsung ElectroZone | 04-01-2025 | 1,199.99 |
| iPhone 14 Pro | Tech World | 04-01-2025 | 1,099.99 |