# Database Basics MS SQL Exam – 11 Aug 2020

**Database Fundamentals MSSQL – Bakery**

*Your friend is opening his “bio” bakery. Since he is a lucky one to have you as a friend/programmer you decide to take part of his new journey – selling healthy food to people. Based on his requirements you should create the initial database frame. Then you have to do some data manipulations. Finally, you have to do some work on the programmability part.*

# Section 1. DDL

**For this section put your queries in judge and use: “*SQL Server run queries and check DB*”.**

You have been given the E/R Diagram of the bakery:



Crate a database called **Bakery**. You need to create **7 tables**:

* **Products** – contains information about the products that are being sold in our bakery.
* **Ingredients** – contains information about fruits, vegetables, spices and so on. Has relation to both products and distributors.
* **ProductsIngredients** – mapping table between products and ingredients.
* **Distributors** – contains information about distributors – organizations that deliver ingredients.
* **Customers** – contains information about the customers that use our products.
* **Countries** – contains info for origin place (ingredients), general office(distributors) or homeland (customers).
* **Feedbacks** – contains information about the feedback that the customers give while evaluating some of the products

**Countries**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Id | Integer from 0 to 2,147,483,647 | Unique table identificator, Identity |
| Name | String up to 50 characters, Unicode | Unique |

**Customers**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Id | Integer from 0 to 2,147,483,647 | Unique table identificator, Identity |
| FirstName | String up to 25 symbols, Unicode |  |
| LastName | String up to 25 symbols, Unicode |  |
| Gender | **Character** with **exactly** 1 symbol | Could be: '*M*' or '*F*' |
| Age | Integer from 0 to 2,147,483,647 |  |
| PhoneNumber | String 10 **characters** long. | String length is **exactly 10** **chars** long. |
| CountryId | Integer from 0 to 2,147,483,647 | Relationship with table Countries |

**Products**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Id | Integer from 0 to 2,147,483,647 | Unique table identificator, Identity |
| Name | String up to 25 symbols, Unicode | Unique |
| Description | String up to 250 symbols, Unicode |  |
| Recipe | String with unlimited number of symbols (**max**),  Unicode |  |
| Price | Decimal number used for **money** calculations | **Non-positive** numbers are not allowed |

**Feedbacks**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Id | Integer from 0 to 2,147,483,647 | Unique table identificator, Identity |
| Description | String up to 255 symbols, Unicode |  |
| Rate | **Decimal** number with two-digit precision | Rate is between 0 and 10 |
| ProductId | Integer from 0 to 2,147,483,647 | Relationship with table Products |
| CustomerId | Integer from 0 to 2,147,483,647 | Relationship with table Customers |

**Distributors**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Id | Integer from 0 to 2,147,483,647 | Unique table identificator, Identity |
| Name | String up to 25 symbols, Unicode | Unique |
| AddressText | String up to 30 symbols, Unicode |  |
| Summary | String up to 200 symbols, Unicode |  |
| CountryId | Integer from 0 to 2,147,483,647 | Relationship with table Countries |

**Ingredients**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Id | Integer from 0 to 2,147,483,647 | Unique table identificator, Identity |
| Name | String up to 30 symbols, Unicode |  |
| Description | String up to 200 symbols, Unicode |  |
| OriginCountryId | Integer from 0 to 2,147,483,647 | Relationship with table Countries |
| DistributorId | Integer from 0 to 2,147,483,647 | Relationship with table Distributors |

**ProductsIngredients**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| ProductId | Integer from 0 to 2,147,483,647 | Unique table identificator, Relationship with table Products, Required |
| IngredientId | Integer from 0 to 2,147,483,647 | Unique table identificator, Relationship with table Ingredients, Required |

## Database design

Submit all of your create statements to Judge.

# Section 2. DML

**For this section put your queries in judge and use: “*SQL Server run skeleton, run queries and check DB*”.**

**Before you start you have to import “*Скелет*”. 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 Id’s should be auto-generated.

**Distributors**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **CountryId** | **AddressText** | **Summary** |
| Deloitte & Touche | 2 | 6 Arch St #9757 | Customizable neutral traveling |
| Congress Title | 13 | 58 Hancock St | Customer loyalty |
| Kitchen People | 1 | 3 E 31st St #77 | Triple-buffered stable delivery |
| General Color Co Inc | 21 | 6185 Bohn St #72 | Focus group |
| Beck Corporation | 23 | 21 E 64th Ave | Quality-focused 4th generation hardware |

**Customers**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FirstName** | **LastName** | **Age** | **Gender** | **PhoneNumber** | **CountryId** |
| Francoise | Rautenstrauch | 15 | M | 0195698399 | 5 |
| Kendra | Loud | 22 | F | 0063631526 | 11 |
| Lourdes | Bauswell | 50 | M | 0139037043 | 8 |
| Hannah | Edmison | 18 | F | 0043343686 | 1 |
| Tom | Loeza | 31 | M | 0144876096 | 23 |
| Queenie | Kramarczyk | 30 | F | 0064215793 | 29 |
| Hiu | Portaro | 25 | M | 0068277755 | 16 |
| Josefa | Opitz | 43 | F | 0197887645 | 17 |

## Update

We’ve decided to switch some of our ingredients to a local distributor. Update the table **Ingredients** and change the **DistributorId** of "Bay Leaf", "Paprika" and "Poppy" to 35. Change the **OriginCountryId** to 14 of all ingredients with **OriginCountryId equal to** 8.

## Delete

Delete all **Feedbacks** which relate to **Customer** with **Id** 14 or to **Product** with **Id** 5.

# Section 3. Querying

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

**For this section put your queries in judge and use: “*SQL Server prepare DB and run queries*”.**

## Products by Price

Select all **products** ordered by **price** (descending) then by name (ascending).

Required columns:

* Name
* Price
* Description

Example:

|  |  |  |
| --- | --- | --- |
| **Name** | **Price** | **Description** |
| Oxygen bread | 27.39 | Morbi ut odio. |
| Pizza(small) | 27.27 | In sagittis dui vel nisl. Duis ac nibh. |

## Negative Feedback

Select all **feedbacks** alongside with the customers which gave them. Filter only feedbacks which have **rate** below **5.0**. Order results by ProductId (descending) then by Rate (ascending).

Required columns:

* ProductId
* Rate
* Description
* CustomerId
* Age
* Gender

Example:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ProductId** | **Rate** | **Description** | **CustomerId** | **Age** | **Gender** |
| 30 | 2.04 | I did not like the product | 23 | 27 | M |
| 27 | 4.16 | Meh.. | 20 | 57 | F |

## Customers without Feedback

Select all customers **without** feedbacks. Order them by customer id (ascending).

Required columns:

* CustomerName – customer’s first and last name, concatenated with space
* PhoneNumber
* Gender

Example:

|  |  |  |
| --- | --- | --- |
| **CustomerName** | **PhoneNumber** | **Gender** |
| Rachel Bishop | 0779574407 | F |
| Irene Peters | 0995086966 | F |

## Customers by Criteria

Select customers that are **either** at least 21 old **and** contain “**an**” in their first name **or** their phone number ends with “38” **and** are **not** from Greece. Order by first name (ascending), then by age(descending).

Required columns:

* FirstName
* Age
* PhoneNumber

Example:

|  |  |  |
| --- | --- | --- |
| **FirstName** | **Age** | **PhoneNumber** |
| Amanda | 30 | 0886609909 |
| Antonio | 49 | 0781375797 |
| Edward | 55 | 0988359338 |

## Middle Range Distributors

Select all distributors which distribute **ingredients** used in the making process of all products having average rate between **5** and **8** (inclusive). Order by distributor name, ingredient name and product name all ascending.

Required columns:

* DistributorName
* IngredientName
* ProductName
* AverageRate

Example:

|  |  |  |  |
| --- | --- | --- | --- |
| **DistributorName** | **IngredientName** | **ProductName** | **AverageRate** |
| Alprazolam | Cinnamon | Nicotine Pleasure | 5.260000 |
| Arinase | Peppercorn | Panetone | 5.400000 |
| … | … | … | … |

## Country Representative

Select all countries with their most active distributor (the one with the greatest number of ingredients). If there are several distributors with most ingredients delivered, list them all. Order by country name then by distributor name.

Required columns:

* CountryName
* DistributorName

Example:

|  |  |
| --- | --- |
| **CountryName** | **DisributorName** |
| Albania | Arinase |
| Andorra | Allopurinol |
| Andorra | SPF 17 |
| … | … |

# Section 4. Programmability

**For this section put your queries in judge and use: “*SQL Server run skeleton, run queries and check DB*”.**

## Customers with Countries

Create a view named **v\_UserWithCountries** which selects all **customers** with **their** **countries**.

Required columns:

* CustomerName – first name plus last name, with space between them
* Age
* Gender
* CountryName

Example usage:

|  |  |  |  |
| --- | --- | --- | --- |
| **Query** | | | |
| SELECT TOP 5 \*  FROM v\_UserWithCountries  ORDER BY Age | | | |
| CustomerName | Age | Gender | CountryName |
| Paul Wells | 6 | M | Philippines |
| Jeremy Banks | 7 | M | Brazil |
| Marie Hudson | 7 | F | Bulgaria |
| … | … | … | … |

## Delete Products

Create a trigger that deletes all of the relations of a product upon its deletion.

Example usage:

|  |
| --- |
| **Query** |
| DELETE FROM Products WHERE Id = 7 |
| Response |
| (1 row(s) affected)  (3 row(s) affected)  (1 row(s) affected)  (1 row(s) affected) |