# Exercises: Data Definition and Data Types

## Create Database

Now it’s time to create SQL queries. In that task (and the several following it) you will be required to create the database **using only SQL queries**. Firstly, just **create new database named Minions.**

## Create Tables

In the newly created database Minions add table **Minions (Id, Name, Age)**. Then add new table **Towns (Id, Name).** Set **Id** columns of both tables to be **primary key** as **constraint**.

## Alter Minions Table

Change the structure of the Minions table to have **new column TownId** that would be of the same type as the **Id** column of **Towns table**. Add **new constraint** that makes **TownId** **foreign key** and references to **Id** column of **Towns** table.

## Insert Records in Both Tables

**Populate both tables** with sample records given in the table below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Minions** | | | |  | **Towns** | |
| **Id** | **Name** | **Age** | **TownId** |  | **Id** | **Name** |
| 1 | Kevin | 22 | 1 |  | 1 | Sofia |
| 2 | Bob | 15 | 3 |  | 2 | Peter |
| 3 | Steward | NULL | 2 |  | 3 | Victoria |

Use only SQL queries.

## Truncate Table Minions

**Delete all the data** from the Minions table using **SQL query.**

## Drop All Tables

**Delete all tables** from the Minions database using **SQL query**.

## Create Table People

Using **SQL query** create table **People** with columns:

* **Id** – unique number for every person there will be **no more than 231-1** **people.** (Auto incremented)
* **Name** – full name of the person will be **no more than 200 Unicode characters**. (Not null)
* **Picture** – image with **size up to** **2 MB.** (Allow nulls)
* **Height** – In meters. Real number precise up to **2 digits** after floating point. (Allow nulls)
* **Weight** – In kilograms. Real number precise up to **2 digits** after floating point. (Allow nulls)
* **Gender** – Possible states are **m** or **f.** (Not null)
* **Birthdate –** (Not null)
* **Biography** – detailed biography of the person it can contain **max allowed Unicode characters.** (Allow nulls)

Make **Id** primary key. Populate the table with only **5 records**.

## Create Table Users

Using **SQL query** create table **Users** with columns:

* **Id** – unique number for every user. There will be **no more than 263-1 users.** (Auto incremented)
* **Username** – unique identifier of the user will be **no more than 30 characters (non Unicode).** (Required)
* **Password** – password will be **no longer than 26 characters (non Unicode).** (Required)
* **ProfilePicture** – image with **size up to 900 KB.**
* **LastLoginTime**
* **IsDeleted** – shows if the user deleted his/her profile. Possible states are **true** or **false**.

Make **Id** primary key. Populate the table with exactly **5 records**.

## Change Primary Key

Using **SQL queries** modify table **Users** from the previous task. First **remove current primary key** then create **new primary key** that would be the **combination** of fields **Id** and **Username**.

## Add Check Constraint

Using **SQL queries** modify table **Users**. Add **check constraint** to ensure that the values in the Password field are **at least 5 symbols** long.

## Set Default Value of a Field

Using **SQL queries** modify table **Users**. Make the **default value** of **LastLoginTime** field to be the **current time.**

## Set Unique Field (Grade = Pass!)

Using **SQL queries** modify table **Users**. Remove **Username** field from the primary key so only the field **Id** would be primary key. Now **add unique constraint** to the **Username** field to ensure that the values there are **at least 3 symbols** long.

## Movies Database

Using **SQL queries** create **Movies** database with the following entities:

* **Directors** (Id, DirectorName, Notes)
* **Genres** (Id, GenreName, Notes)
* **Categories** (Id, CategoryName, Notes)
* **Movies** (Id, Title, DirectorId, CopyrightYear, Length, GenreId, CategoryId, Rating, Notes)

Set most **appropriate data types** for each column. **Set primary key** to each table. Populate each table with exactly **5 records**. Make sure the columns that are present in 2 tables would be of the **same data type**. Consider which fields are always required and which are optional.

## Car Rental Database

Using **SQL queries** create **CarRental** database with the following entities:

* **Categories** (Id, CategoryName, DailyRate, WeeklyRate, MonthlyRate, WeekendRate)
* **Cars** (Id, PlateNumber, Manufacturer, Model, CarYear, CategoryId, Doors, Picture, Condition, Available)
* **Employees** (Id, FirstName, LastName, Title, Notes)
* **Customers** (Id, DriverLicenceNumber, FullName, Address, City, ZIPCode, Notes)
* **RentalOrders** (Id, EmployeeId, CustomerId, CarId, TankLevel, KilometrageStart, KilometrageEnd, TotalKilometrage, StartDate, EndDate, TotalDays, RateApplied, TaxRate, OrderStatus, Notes)

Set most **appropriate data types** for each column. **Set primary key** to each table. Populate each table with only **3 records**. Make sure the columns that are present in 2 tables would be of the **same data type**. Consider which fields are always required and which are optional.

## Hotel Database (Grade = Great!)

Using **SQL queries** create **Hotel** database with the following entities:

* **Employees** (Id, FirstName, LastName, Title, Notes)
* **Customers** (AccountNumber, FirstName, LastName, PhoneNumber, EmergencyName, EmergencyNumber, Notes)
* **RoomStatus** (RoomStatus, Notes)
* **RoomTypes** (RoomType, Notes)
* **BedTypes** (BedType, Notes)
* **Rooms** (RoomNumber, RoomType, BedType, Rate, RoomStatus, Notes)
* **Payments** (Id, EmployeeId, PaymentDate, AccountNumber, FirstDateOccupied, LastDateOccupied, TotalDays, AmountCharged, TaxRate, TaxAmount, PaymentTotal, Notes)
* **Occupancies** (Id, EmployeeId, DateOccupied, AccountNumber, RoomNumber, RateApplied, PhoneCharge, Notes)

Set most **appropriate data types** for each column. **Set primary key** to each table. Populate each table with only **3 records**. Make sure the columns that are present in 2 tables would be of the **same data type**. Consider which fields are always required and which are optional.

## Create Lexicon Database

Now create bigger database called **Lexicon.** You will use same database in the future tasks. It should hold information about

* **Towns** (Id, Name)
* **Addresses** (Id, AddressText, TownId)
* **Departments** (Id, Name)
* **Employees** (Id, FirstName, MiddleName, LastName, JobTitle, DepartmentId, HireDate, Salary, AddressId)

**Id** columns are **auto incremented** starting from 1 and increased by 1 (1, 2, 3, 4…). Make sure you **use appropriate data types** for each column. Add **primary** and **foreign keys** **as constraints** for each table. Use **only SQL queries**. Consider which fields are always required and which are optional.

## Backup Database

Backup the database **Lexicon** from the previous tasks into a file named “**lexicon-backup.bak**”. Delete your database from SQL Server Management Studio. Then restore the database from the created backup.

## Basic Insert

Use the **Lexicon** database and insert some data **using SQL queries**.

* **Towns:** Sofia, Plovdiv, Varna, Burgas
* **Departments:** Engineering, Sales, Marketing, Software Development, Quality Assurance
* **Employees:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Job Title** | **Department** | **Hire Date** | **Salary** |
| Mel Gibson Gibson | .NET Developer | Software Development | 01/02/2013 | 3500.00 |
| Chuck Norris Norris | Senior Engineer | Engineering | 02/03/2004 | 4000.00 |
| Greta Garbo Garbo | Intern | Quality Assurance | 28/08/2016 | 525.25 |
| Meryl Strep Strep | CEO | Sales | 09/12/2007 | 3000.00 |
| Peter Pan Pan | Intern | Marketing | 28/08/2016 | 599.88 |

## Basic Select All Fields (Grade = Excellent!)

Use the **Lexicon** database and first select all records from the **Towns**, then from **Departments** and finally from **Employees** table.

## Basic Select All Fields and Order Them

Modify queries from previous problem by sorting:

* **Towns** - alphabetically by name
* **Departments** - alphabetically by name
* **Employees** - descending by salary

## Basic Select Some Fields

Modify queries from previous problem to show only **some of the columns**. For table:

* **Towns** – Name
* **Departments** – Name
* **Employees** – FirstName, LastName, JobTitle, Salary

## Increase Employees Salary (Grade = Expert!)

Use **Lexicon** database and **increase the salary** of all employees by **10%.** Then show **only Salary** column for all in the **Employees** table.