**SQL Manual**

You can have a scenario where you have multiple databases each having their own tables. You may have a database to manage customer tables, and another for product, etc.

You can create a table using two ways. One is a query and one is a table.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Click save, and then give the table a name. Click refresh, and then you will see that dbo.Employee has been created as a table. Right click on the table and you can add values to the table by going to edit top 200 rows.

A screenshot of a computer program

Description automatically generated

After adding the values:

A screenshot of a computer

Description automatically generated

Right click the table and select top 1000 rows. You will now see the rows:

A screenshot of a computer

Description automatically generated

In the same way, you can right click and create a database as well.

**Query to create database**



**Query to create table**



Make sure that all queries or scripts are run in the database that you want to execute them in. You can select the database in the dropdown.

**Making comments**

You can add two dashes to make anything into a comment.

**Inserting data into a table**

A screenshot of a computer

Description automatically generated

You can insert rows or multiple rows of data as above.

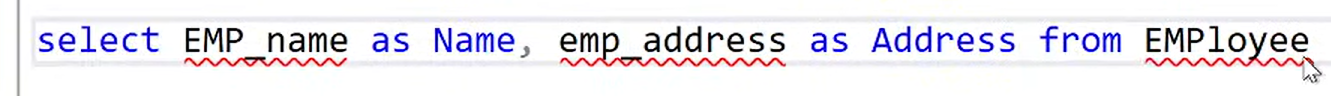
**Selecting columns**

A close-up of words

Description automatically generated

You can select all columns, or you can choose to select only some columns.

**Using an alias when selecting columns**



This only changes the name for the selection purpose, but not the name of the column in the DB.

**Deleting rows from a table**

A blue text on a white background

Description automatically generated

Above we are saying to delete all rows from the Employe table.

Deleting tables

A close up of a sign

Description automatically generated

You can delete tables with the query above.

**CRUD Operations (Create, Read, Update, and Delete).**

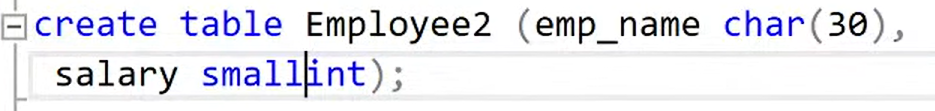
**Data Types**

You must design your database with the correct data types

**A screenshot of a computer

Description automatically generated**

So based on the type of data, you can can state the data type.

****

**Precision**

When defining the data type, you can state the precision. So if you want to store 123, you want to store it as five digits and 2 decimal places.

**A screenshot of a computer

Description automatically generated**

So a practical example of this is:

**A screenshot of a computer

Description automatically generated**

**Storing the data considering UNICODE**

A close up of a computer screen

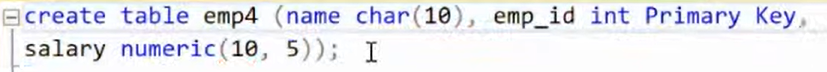
Description automatically generated

Sometimes people have letters which don’t fall in ASCII, and may be foreign names. You can use nchar data type and you must use the N keyword when entering the value into the table.

**Identity**

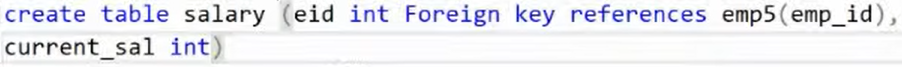
We all have some sort of a unique identity, for example employee id’s or passport numbers. There could be many davids, or johns in this world. When you store information in a table, you have to be able to uniquely identify each and every row. A unique ID is essentially a primary key constraint. You cant have two rows with the same ID in a table. Also, a null ID is not allowed.

Using the primary key constraint.



Referential Identity concept

When creating a table which holds a foreign key, it is used to reference a different table.



So we create a new table named salary and give it a column named eid which is a foreign key to the emp5 table which column emp\_id. You can only insert values into the eid column in the foreign key table if the emp\_id in the emp5 table exists, otherwise it will not be allowed.

**Unique, not null, and Default Constraints**

Lets create a new table in a database.

CREATE TABLE Citizens (

Citizen\_ID INT PRIMARY KEY,

Citizen\_Name VARCHAR(100) NOT NULL,

NI\_Number VARCHAR(20) UNIQUE,

Country VARCHAR(50) DEFAULT 'United Kingdom');

Notice how the national insurance number has a UNIQUE constraint and how the country has a default constraint. That is because we want the NI number to be unique (without being the primary key) and we want the country to have a default value of united kingdom. We have stated that the name cannot be a NULL value, so the user is forced to insert a value for it. We then insert values into the table:

INSERT INTO Citizens (Citizen\_ID, Citizen\_Name, NI\_Number, Country)

VALUES

(1, 'John Doe', 'AB123456C', 'United Kingdom'),

(2, 'Jane Smith', 'CD789012D', 'United Kingdom'),

(3, 'Alice Johnson', 'EF345678E', 'United Kingdom'),

(4, 'Bob Wilson', 'GH901234F', 'United Kingdom'),

(5, 'Eva Davis', 'IJ567890G', 'United Kingdom');

**Identity constraint**

Notice how you have been putting in the values 1,2,3,4,5 for the primary key. However, you do not have to do that. You can apply the identity constraint to the primary key to force it to auto increment.

CREATE TABLE Citizens (

Citizen\_ID INT PRIMARY KEY IDENTITY(1,1),

Citizen\_Name VARCHAR(100),

NI\_Number VARCHAR(20) UNIQUE,

Country VARCHAR(50) DEFAULT 'United Kingdom'

);

You can now insert into without putting the primary key in there:

INSERT INTO Citizens (Citizen\_Name, NI\_Number, Country)

VALUES

('Mary Johnson', 'KL123456H', 'United Kingdom'),

('Tom Williams', 'MN789012I', 'United Kingdom'),

('Grace Brown', 'OP345678J', 'United Kingdom');

Lets say that you wanted to alter the table that was created before and you wanted to add the identity constraint to it. You can do this with the use of the ALTER key word.

ALTER TABLE Citizens

ALTER COLUMN Citizen\_ID INT PRIMARY KEY IDENTITY(1,1);

**Difference between primary key and unique**

The unique constraint allows a NULL value, whereas the primary key does not.

**Foreign Key Constraint**

Lets assume that we are creating a voters table that needs to be linked to the citizen table through the citizen\_id. We state the foreign key references keywords and connect the citizen\_id.

A screenshot of a computer code

Description automatically generated

**Check Constraint**

We can check if certain columns have a condition. For the example below we have added a check if the age is greater than or equal to 18. We have also added a check to state that the height must be greater or equal to 120. Also, the mobile number must have a length which is equal to 11. If these conditions are not met, then a value cannot be added into the respective columns. Finally, we have declared the Citizen\_ID column so that it exists in both tables. We then add a foreign key constraint linking the two columns which have the same name.

CREATE TABLE Citizen\_info (

Info\_ID smallint PRIMARY KEY,

Age smallint CHECK (Age >= 18),

Height smallint CHECK (Height >= 120),

Mobile\_no INT UNIQUE CHECK (LEN(Mobile\_no) = 11),

Citizen\_ID INT,

FOREIGN KEY (Citizen\_ID) REFERENCES Citizens(Citizen\_ID)

);

**Foreign Key Constraint with the correct constraints**

CREATE TABLE Citizen\_info (

Info\_ID smallint PRIMARY KEY IDENTITY (1,1),

Age smallint CHECK (Age >= 18),

Height smallint CHECK (Height >= 120),

Mobile\_no bigint UNIQUE CHECK (LEN(Mobile\_no) = 11),

Citizen\_ID int,

FOREIGN KEY (Citizen\_ID) REFERENCES Citizens(Citizen\_ID)

);

INSERT INTO Citizen\_info (Age, Height, Mobile\_no, Citizen\_ID)

VALUES

(25, 160, 12345678901, 1),

(30, 175, 23456789012, 2),

(22, 150, 34567890123, 3);

select \* from Citizens

select \* from Citizen\_info

In the code above, I have created a table named Citizen info which will be connected to the citizens table. The info id column is the primary key and the age, height, and mobile number are all checked for specific inputs. The citizen ID has a foreign key constraint. The values are inserted into the table, and notice how I do not need to add the info id as it is auto incremented. I add the citizen id which exists in the citizens table. If I try to add a citizen Id in the citizens info table that does not exist in the citizens info table, an error will be received. The results look like this:

A screenshot of a computer

Description automatically generated

**Creating a Schema**

You can create tables and put them inside schemas. Schemas are just like folders or containers. Inside a database, you can have different schemas, and inside that schema, you will have tables. Lets now create a schema named bus drivers and then put two tables inside the bus\_drivers schema named employees and salary.

CREATE SCHEMA BusDrivers

CREATE TABLE BusDrivers.Employees (

Employee\_id SMALLINT PRIMARY KEY IDENTITY (1,1),

Age SMALLINT CHECK (Age >= 18),

Height SMALLINT CHECK (Height >= 100)

);

CREATE TABLE BusDrivers.Salary (

Salary INT,

CURRENCY NVARCHAR(10) DEFAULT N'£',

Employee\_id SMALLINT,

FOREIGN KEY (Employee\_id) REFERENCES BusDrivers.Employees(Employee\_id)

);

-- Inserting three rows into the BusDrivers.Employees table

INSERT INTO BusDrivers.Employees (Age, Height)

VALUES

(25, 175),

(30, 160),

(22, 180);

-- Inserting corresponding rows into the BusDrivers.Salary table

INSERT INTO BusDrivers.Salary (Salary, Employee\_id)

VALUES

(50000, 1),

(60000, 2),

(45000, 3);

SELECT \* FROM BusDrivers.Employees

SELECT \* FROM BusDrivers.Salary

A screenshot of a computer

Description automatically generated

A schema is a great way of putting your tables into categories.