INTRO TO SQL
WEEK 2

APRIL 18, 2023 – MAY 4TH 2023

WEEK 2: APRIL 25 & APRIL 28TH

INSTRUCTOR: SYLVIA VARGAS SQLSYLVIA@GMAIL.COM

LINKEDIN: HTTPS://WWW.LINKEDIN.COM/IN/SYLVIAVARGAS/

BLOG: HTTPS://SYLVIAVARGAS.COM/

BLOG (IN PROGRESS): HTTP://SHESATECHIE.ORG/

ALL SLIDES ON GITHUB: HTTPS://GITHUB.COM/SQLSYLVIA/GDI-SQL

WEEK 2 AGENDA – MORE DATA MANIPULATION

We will continue using the **W3Schools** Database for class.

https://www.w3schools.com/sql/trysql.asp?filename=trysql_editor

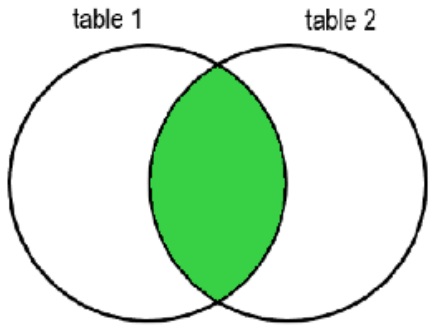
This week's class topics include:

- JOIN REVIEW
- OUTER JOINS
- SUBQUERIES and Common Table Queries
- INSERT
- UPDATE
- ALTER TABLE
- CREATE TABLE

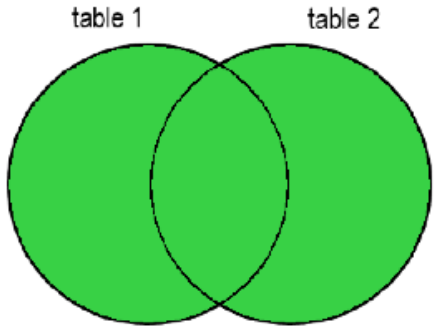
JOIN REVIEW

- These VENN DIAGRAMS illustrate the output of various JOIN commands.

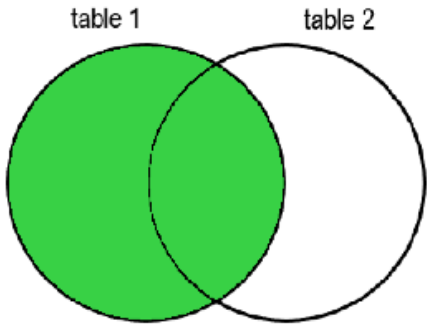
INNER JOIN



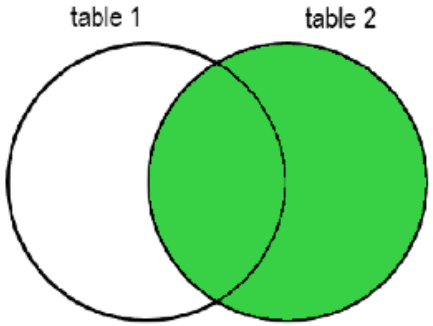
FULL OUTER JOIN
CARTESIAN PRODUCT



LEFT JOIN

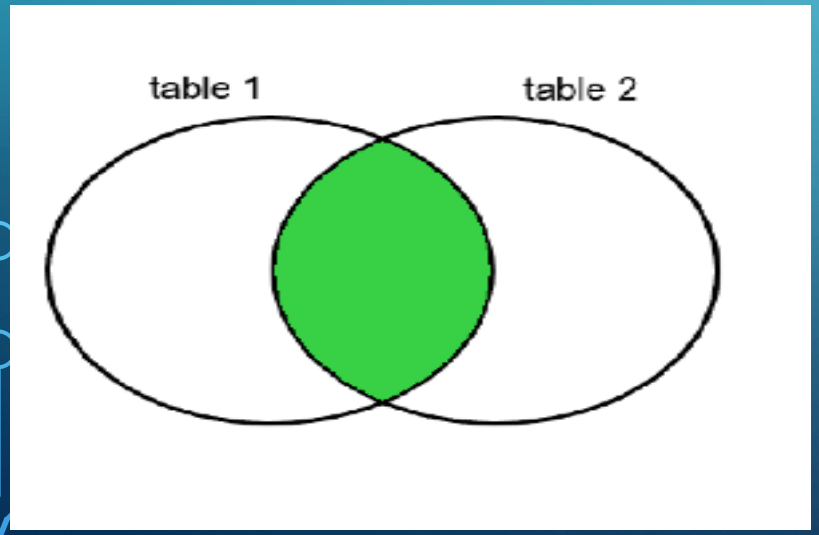


RIGHT JOIN



JOIN

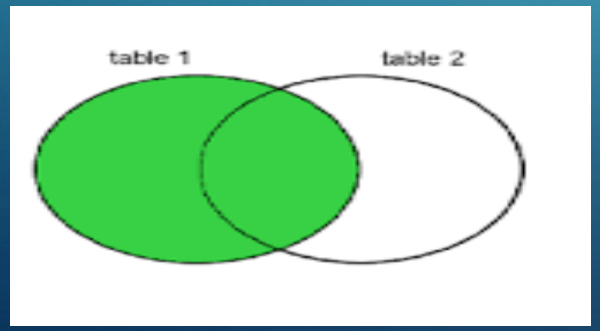
The inner join will keep only the information from the two joined tables that is related




```
SELECT Employees.LastName
      , Orders.orderID
FROM Orders
JOIN Employees
ON Employees.EmployeeID =
Orders.EmployeeID
```

LEFT JOIN = LEFT OUTER JOIN

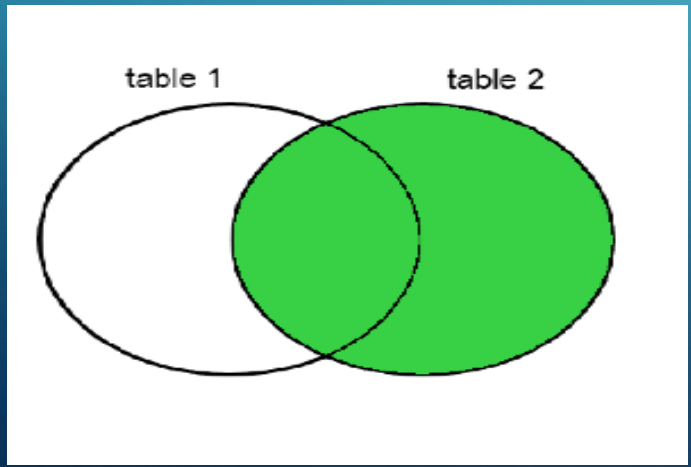
A LEFT JOIN returns all the rows from the left table and the matching rows from the right table. If there is no match in the right table, the result will contain NULL values.




```
SELECT Employees.LastName
, Orders.OrderID
FROM Employees
LEFT JOIN Orders
  ON orders.EmployeeID =
      Employees.EmployeeID ;
```

RIGHT JOIN
= RIGHT OUTER JOIN

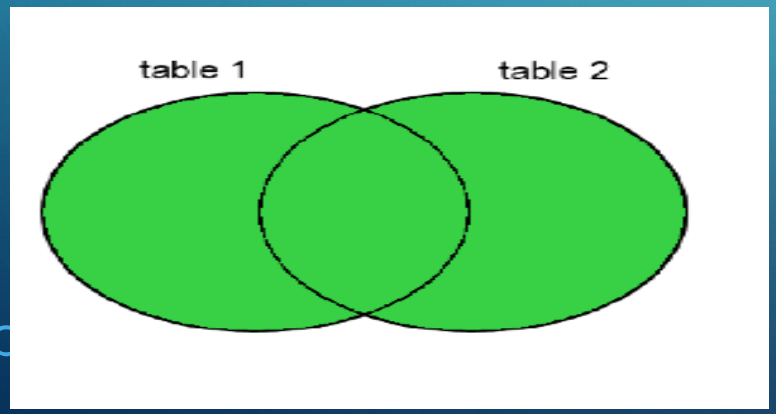
A RIGHT JOIN returns all the rows from the right table and the matching rows from the left table. If there is no match in the left table, the result will contain NULL values.




```
SELECT Employees.LastName
      , Orders.OrderID
FROM Orders
RIGHT JOIN Employees
  ON orders.EmployeeID =
       Employees.EmployeeID ;
```

FULL OUTER JOIN = CARTESIAN PRODUCT

OUTER....



SQL Statement:

```
SELECT Orders.customerID, OrderDetails.ProductID
FROM Orders
JOIN OrderDetails
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 101528

CustomerID	ProductID
90	11
90	42
90	72

REFERENCES FOR SQL JOINS

- https://www.freecodecamp.org/news/sql-join-types-inner-join-vs-outer-join-example/#:~:text=There%20are%20three%20types%20of,RIGHT%20JOIN%20%2C%20and%20FULL%20JOIN%20

HAVING CLAUSE – USE WITH GROUP BY

The **HAVING** statement is used to filter data after it has been grouped by the **GROUP BY** statement.

```
SELECT Orders.customerID,
SUM(OrderDetails.quantity) AS total_quantity
FROM Orders
JOIN OrderDetails
    ON OrderDetails.OrderId =
        Orders.OrderID
GROUP BY Orders.customerID
HAVING SUM(OrderDetails.quantity) > 100
```

UNION

The **UNION** operator is used to combine the results of two or more

SELECT statements into a single result set. The **SELECT** statements must have the same number of columns, and the columns must have compatible data types.

Duplicate rows are automatically removed from the result set.

```
SELECT 'Customer' as PersonType, CustomerName, City, Country
 FROM Customers
 WHERE  Country ='Germany'
UNION
SELECT 'Supplier' as PersonType, SupplierName, City, Country
 FROM Suppliers
WHERE  Country ='Germany'
```

SUBQUERIES

A SUBQUERY IS A QUERY NESTED INSIDE ANOTHER STATEMENT SUCH AS SELECT, INSERT, UPDATE, OR DELETE.

```
SELECT
    AVG(Total) average_product_quantity
FROM
    (SELECT SUM(Quantity) as Total
    FROM OrderDetails
    GROUP BY ProductID)
```

COMMON TABLE EXPRESSIONS (CTE)

- The **WITH** clause is considered “temporary” because the result is not permanently stored anywhere in the database schema. It acts as a temporary view that only exists for the duration of the query.

SUBSELECT VERSUS COMMON TABLE EXPRESS

INSERTING DATA - EXAMPLE 1

```
INSERT INTO Suppliers (supplierID
        , supplierName
        , ContactName
        , Address, City, PostalCode
       , Country, Phone)
VALUES (99
    ,'Sylvia The Supplier’
    , 'Sylvia'
    , 'One Microsoft Way’
    , 'Redmond', '10481’
    , 'USA', '(425) 555-1212’);
```

INSERTING DATA WITH A COMMON TABLE EXPRESSION

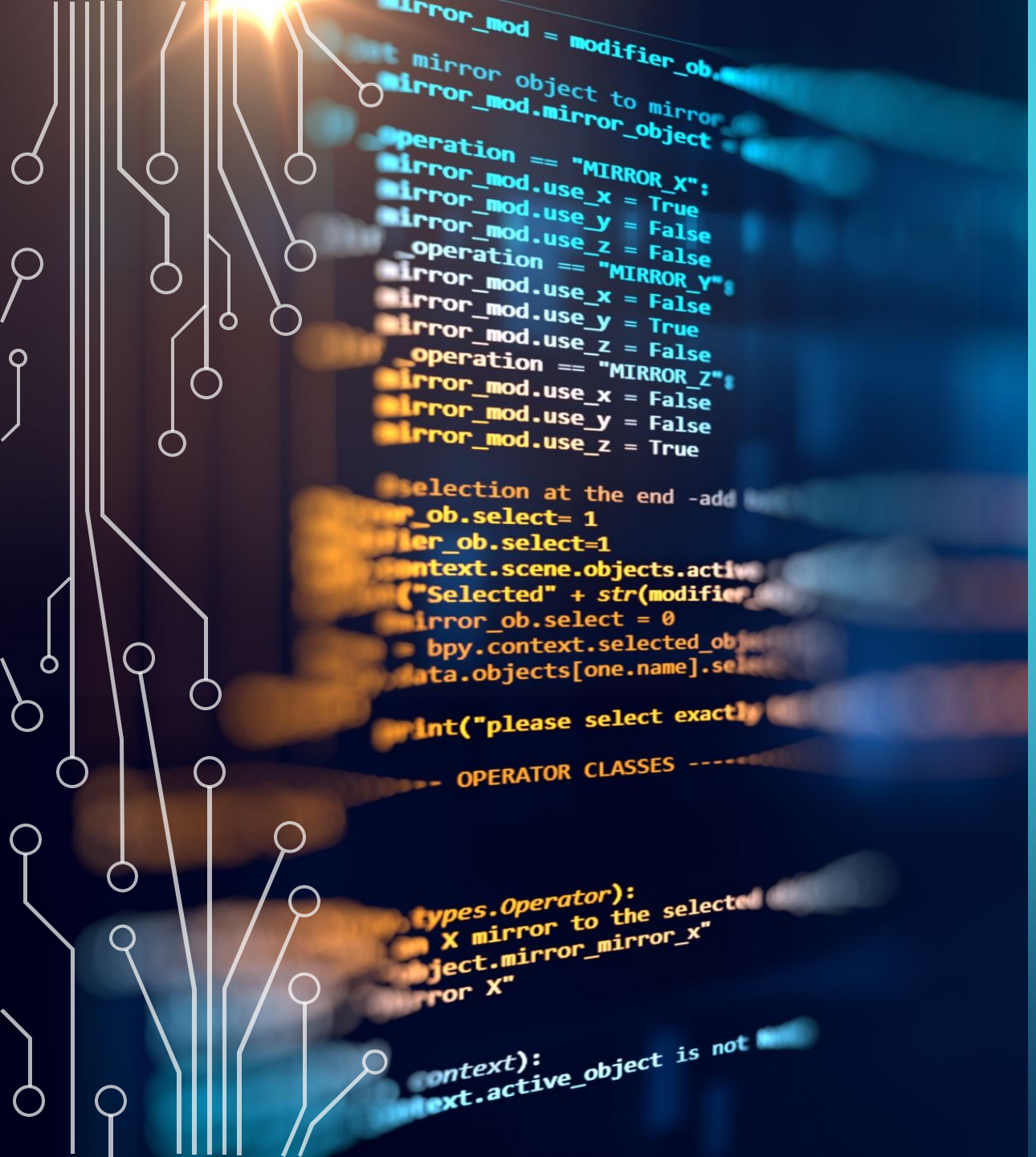
```
WITH Data as
( SELECT 99 as SupplierId
      , 'John The Supplier' as SupplierName
      , 'John' as ContactName
      , 'One Microsoft Way' as Address
      , 'Redmond' as City
      , '10481' as PostalCode
      , 'USA' as Country
       , '(425) 555-1212' as Phone )
INSERT INTO Suppliers (supplierID, supplierName,
ContactName, Address, City, PostalCode, Country,
Phone)
SELECT supplierID, supplierName, ContactName
            , Address, City, PostalCode, Country, Phone
FROM Data;
```

UPDATING DATA

```
UPDATE Suppliers
SET Phone='(800) 123-4567'
WHERE SupplierID = 99;
```

DELETING DATA

```
DELETE Suppliers
WHERE SupplierID = 99;
```



DAY 4

CREATING TABLES, VIEWS

SQL DATA TYPES

SQL INDEXES AND CONSTRAINTS

CREATE TABLE FROM A QUERY

You can create one table from another by adding a SELECT statement at the end of the CREATE TABLE statement:

```
CREATE TABLE CountryDataTable as

  SELECT 'Customer' as PersonType
      , CustomerName
      , City, Country
    FROM Customers

  UNION

   SELECT 'Supplier' as PersonType
    ,  SupplierName
    , City, Country
   FROM Suppliers

;
```

CREATE VIEW

A SQL VIEW is a virtual table based on the result-set of an SQL statement.

Why use a VIEW

A view will always show up-to-date data when the data from the tables in the view change!!

```
CREATE VIEW CountryData as
  SELECT 'Customer' as PersonType
      , CustomerName
      , City, Country
    FROM Customers
  UNION
   SELECT 'Supplier' as PersonType
    ,  SupplierName
    , City, Country
   FROM Suppliers
;
```

DATABASE STORAGE AND DATA TYPES

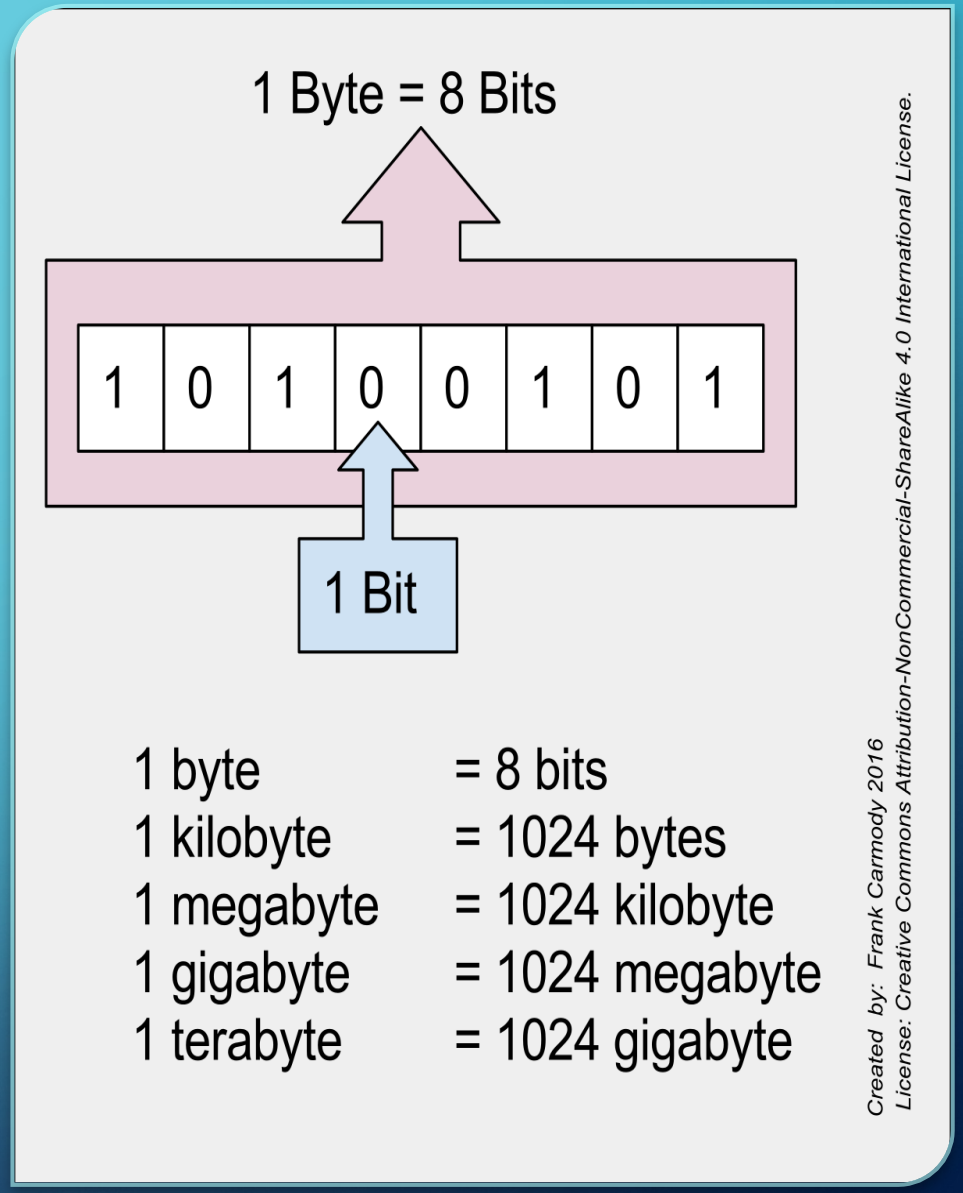
When creating a database table, the data type not only indicates the how type of data will be stored but how much space will be used.

A database administrator, architect or software developer needs to be aware of how much space is used by the data types because it affects the performance and cost of the database.

In MySQL table has a maximum row size limit of 65,535 bytes, even if the storage engine is capable of supporting larger rows. BLOB and TEXT columns only contribute 9 to 12 bytes toward the row size limit because their contents are stored separately from the rest of the row.

CHARACTER STRINGS WHAT IS ASCII AND UNICODE?

- ASCII and UNICODE are used in storing characters into computer code.
- ASCII - map the numeric values 0-127 to various Western characters and control codes (newline, tab, etc.). Note that values 0-127 fit in the lower 7 bits in an 8-bit byte.
- UNICODE is the solution for languages with characters exceeding more than 128 characters. Languages like Chinese, Russian, Hebrew.



STRING DATA TYPES

Data Type	Description	Example
char(n)	Fixed width character string Max size 8,000 characters	CHAR(50) – will use 50 bytes of storage for each record.
varchar(n)	Variable width character string Max size 8,000 characters	VARCHAR(50) will use up to 50 bytes for storage. But will use less if only 2 characters are stored.
text	Variable width character string Max size: 2GB	
nchar(n)	Fixed width Unicode string Max size: 4,000 characters	NCHAR(50) – will use 100 bytes of storage because it will assume additional storage for special characters. Only use if your system supports non-Western languages.
nvarchar(n)	Fixed width Unicode string Max size: 4,000 characters	VARCHAR(50) will use up to 100 bytes for storage. But will use less if only 2 characters are stored.

NUMERIC DATA TYPES (MYSQL)

Data Type	Description
BOOL	Boolean data type: - Zero is considered as false - 1 is considered as true.
INT(*size*) or INTEGER(size)	A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The *size* parameter specifies the maximum display width (which is 255)
BIGINT(*size*)	A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The *size* parameter specifies the maximum display width (which is 255)
DEC(*size*, *decimal points*) or DECIMAL(*size*, *decimal points*)	An exact fixed-point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter.
FLOAT(size)	A floating point number. Used in scientific calcuations.
DOUBLE(size)	A floating point number that is more precise than float and can store 64 bits

DATE AND TIME DATA TYPES

Data Type	Description	When to Use
date	Store a date only. From January 1, 0001 to December 31, 9999.	Use if time information is not required.
datetime	From January 1, 1753 to December 31, 9999 with an accuracy of 3.33 milliseconds	Use if date and time accuracy include to a millisecond
datetime2	From January 1, 0001 to December 31, 9999 with an accuracy of 100 nanoseconds	Use if date and time accuracy to the nanosecond. Mostly used in businesses that have high transactions from web, financial, manufacturing business.
datetimeoffset	The same as datetime2 with the addition of a time zone offset	Use if the timezone is required in capturing data. Best practice is to store dates in GMT and then when querying show in the timezone needed.
timestamp	Stores a unique number that gets updated every time a row gets created or modified.	The timestamp value is based upon an internal clock and does not correspond to real time. Each table can only have one timestamp variable

TABLE CONSTRAINTS AND INDEXES

SQL CONSTRAINTS are used to ensure the quality of the data in a table

The following constraints are commonly used in SQL:

- NOT NULL - Ensures that a column cannot have a NULL value (ie. Must a have a value)
- UNIQUE - Ensures that all values in a column are different
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- CREATE INDEX - Used to create and retrieve data from the database very quickly
- FOREIGN KEY - Prevents actions that would destroy links between tables
- CHECK - Ensures that the values in a column satisfy a specific condition
- DEFAULT - Sets a default value for a column if no value is specified

CREATE TABLE PRIMARY KEY

- PRIMARY KEY is one more columns to uniquely identify each row in a table.
- PRIMARY KEY will not allow any duplicate values in table for that the PRIMARY KEY column/s
- PRIMARY KEY is used as an index to quickly retrieve a specific row of a table.

MySQL:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (ID)
);
```

SQL Server / Oracle / MS Access:

```
CREATE TABLE Persons (
    ID int NOT NULL PRIMARY KEY,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```

CREATE TABLE WITH UNIQUE

- The UNIQUE constraint ensures that all values in a column are different.
- Both the **UNIQUE** and **PRIMARY** KEY constraints provide a guarantee for uniqueness for a column or set of columns.
- A PRIMARY KEY constraint automatically has a UNIQUE constraint.
- However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255) NOT NULL,
    Age int,
PRIMARY KEY(ID),
UNIQUE(LastName,FirstName)
);
```

CREATE TABLE WITH UNIQUE

- The UNIQUE constraint ensures that all values in a column are different.
- Both the **UNIQUE** and **PRIMARY** KEY constraints provide a guarantee for uniqueness for a column or set of columns.
- A PRIMARY KEY constraint automatically has a UNIQUE constraint.
- However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255) NOT NULL,
    Age int,
PRIMARY KEY(ID),
UNIQUE(LastName,FirstName)
);
```

CREATE TABLE WITH DEFAULT DEFINITION

- The DEFAULT constraint is used to set a default value for a column.
- The default value will be added to all new records, if no other value is specified.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255) DEFAULT  'New York'
);
```

CREATE TABLE WITH CHECK CONSTRAINT

- The DEFAULT constraint is used to set a default value for a column.
- The default value will be added to all new records, if no other value is specified.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CHECK (Age>=18)
);
```

ALTER TABLE

- ADD COLUMN
- MODIFY COLUMN

```
ALTER TABLE [Employees]
ADD COLUMN Country VARCHAR(50);
```

```
ALTER TABLE [Employees]
MODIFY COLUMN Country VARCHAR(100);
```

OPTIONAL HOMEWORK

Using W3Schools database:
https://www.w3schools.com/sql/trysql.asp?filename=trysql_editor

- Add 2 new customers to the Customers' table.
- Add 1 new supplier to the Suppliers' table.
- Add a new column for Country in the Employees table and Update 2 or more records in the Employee table with values for Country.