# Introduction to SQL

* Structured Query Language
* It is **NOT** a programming language. It is a **declarative query** language.
  + You tell the program what needs to be done instead of delivering the implementation details to do what needs to be done.
* It performs operations against a relational database.

## Database

* It is just an organize collection of data stored in some organized format.
* They allow us to input, manage, organize, and retrieve data quickly.
* Traditionally, it is organized into “tables” and each table will have a row and a column.
  + Rows will be the same as “records”
  + Columns will be the same as “fields”
* Data is the intersection between a row and a column

## RDBMS

* It stands for Relational Database Management System.
  + SQL is a specific style of RDBMS
* It upholds specified relationships between tables or our data.
* It includes functions that maintain the security, accuracy, integrity, and consistency of the data.

# SQL Sublanguages

## DDL

* Data Definition Language
* It is for the creation/alteration of table structures.
* Create – Most commonly used to create tables and their columns.
  + Can also be used to create other things.
* Alter – Will change the column of the table.
* Truncate – Removes all the data in a table.
  + You cannot roll back the changes.
* Drop – drops the table.

## DML

* Data Manipulation Language
* It is for changing/manipulating/modifying the data within a table.
* Insert – Add row(s) to your table
* Select – Retrieves the data from a table for us to read
* Update – Modify the rows in the table
* Delete – Delete the rows in the table

## DQL

* Data Query Language
* It is for reading the data from a table.
* It is bit controversial in that some people do not believe this is part of SQL sublanguage but for our case it will be.
* All it has is a bunch of select.

## TCL

* Transaction Control Language
* They are used to manage transactions.
  + Think of as methods in C#.
  + They are a logical work unit that will perform single or multiple statements in a database.
  + They help prevent data inconsistency because they will either execute all the statements inside the transaction or it won’t persists any of the statements at all.
  + Ex: You have a checking account, and you want to transfer some money to your savings account.
    - During the transaction of updating your checking account to have less money your database failed.
    - If you didn’t use a transaction, then it seems like the transfer took your money from your checking account and your savings account didn’t have that $100.
* Begin – It will start our transaction.
* Commit – To permanently save any transaction into the database.
* Savepoint – It is used to temporarily save a transaction.
* Rollback – Restores the database to the last committed state or save point.

# Aggregate functions

* They take a set of data and returns a single calculate valued from the data set
* Sum – Summation of the set of data
* Count – Gets the total number of rows that is returned
* Avg – Average of the set of data
* Min – Finds the least data
* Max – Finds the highest data

# Scalar functions

* It will return a new data (some sort of transformation for every data in the set)
* Upper – return the uppercase version of the string
* Lower – opposite of upper
* Length – gives the number of characters in a string
* GetDate

# Constraints

* They are a way for you to limit the data that will come into your table (hence the name “constraints)
* It will specify either one or more rules that the data you are inputting in that column

## Some commonly used constraints

* Type
  + Restricts what data type to input in a column.
* Unique
  + Data in this column cannot have repeating values.
  + No duplicates
  + It can still have multiple null data.
* Not null
  + Will make sure that the data in this column will always have a value.
* Default
  + Will provide a default value in this column if you do not provide a value.
* Check
  + Adds an extra condition on the data.
* Primary key
  + Data in this column is unique and not null implicitly.
  + Acts as the unique identifier for the records in this table.
  + A composite key is just a primary key that consists of multiple columns together.
* Foreign key
  + Data in this column reference the primary key of another table.
  + Establishes the relationships between 2 columns in the same table or different tables.

# Multiplicity

* It is a way to describe the numerical relationships between 2 tables.
* We will be using the primary and foreign keys constraints to establish these relationships
* There are 3 main relationships

## One to One

* One row in Table A is directly related to one row in table B and it goes both ways (bi-directional)
* Establish a foreign key with a unique constraint.

## One to Many

* One row in Table A is related to many rows in Table B
* Established by making the foreign key column to be non-unique

## Many to Many

* Many rows in Table A is related to many rows in Table B
* You must use join table to create many to many relationships
  + Join tables are separate, independent tables.
  + Consists of 2 columns that are both foreign keys.
  + One column will reference Table A and the other column will reference Table B
  + The primary key of the join table is commonly structured as composite key.

# Normalization

* It is a design pattern that reduces/eliminated data redundancy and data duplication.
* Must always ensure Referential Integrity Atomic data
  + Meaning data has to be broken down into smallest possible meaningful unit
  + Use atomic as a way for you to remember, an atom is the smallest possible way to break something down (for now at least)

## 0NF – Zero Normal Form

* No normalization
* Data redundancy and data duplication is everywhere.
* Trying to find information is incredible hard to do (for humans)
* Great to find information for robots (no complex joins needed to find the correct info)

## 1NF – 1st Normal Form

* Table must have a primary key.
* All data must be atomic.
  + Data must be broken down into the smallest possible unit.
  + No composite column.

## 2NF – 2nd Normal Form

* Table needs to be in 1NF
* Remove any partial dependencies.
  + A partial dependency is when some of your columns does not correlate with all the columns that creates a composite key.
    - Ex: [Partial Dependency in DBMS (tutorialspoint.com)](https://www.tutorialspoint.com/Partial-Dependency-in-DBMS)
  + Having composite keys might lead to only having partial dependency.
  + By good practice, just have a single column that is a primary key will prevent partial dependency and make your table in 2NF.

## 3NF – 3rd Normal Form

* Table needs to be in 2NF.
* Remove transitive dependencies.
  + It is when a column in the table does not depend on the primary key.
* Everything must describe the key, the whole key, and nothing but the key.

## Referential Integrity

* It is a concept used to maintain our relationships between our tables without having missing information.
* Essentially that means we cannot delete or update data in a table that have existing relationships with another table.
* You must also **always point your foreign key constraint to reference a valid unique key** to identify a row in another table (although you can point a foreign key to a null value it is not recommended to avoid violating this concept).