Structured Query Language

- Structured Query Language (SQL) was developed by the IBM Corporation in the late 1970's.

- SQL was endorsed as a United States national standard by the American National Standards Institute (ANSI) in 1992 [SQL-92].

- Each DBMS manufacturer follows the ANSI standard, but also adds extended features unique to their SQL

SQL DDL and DML

- **SQL statements can be divided into two categories:**
  - **Data definition language (DDL)** statements
    - Used for creating tables, relationships and other structures.
  - **Data manipulation language (DML)** statements.
    - Used for queries and data modification.

"Structured Query Language"

- It is NOT a programming language
- It is NOT PROCEDURAL
- It does not process one record at a time, rather, it is a SET processing language
- All inputs to SQL are tables
- Output from a query is a table referred to as the "**Answer Set**"
- Some queries may produce "interim" temporary answer sets

"Structured Query Language"

- It is a relatively simple language – brief syntax, few commands
- It is a relatively powerful language – a few lines of code can accomplish a LOT of work

USE statement

```
USE <database>;
```

- Tells the Query Engine which database you want to use for your query

SELECT statement

```
SELECT <column1>, <column2>, <column3>,
   <literal>, <math expression>
   FROM  <table A> ;
```

Examples

```
select *
     from  nwEmployees;
use Northwinds
     select *
     from  nwEmployees;
select EmployeeID, LastName, FirstName
     from nwEmployees
```

SELECT statement

- Literals may be either **`'Character'`** (in quotes) or Numeric
- Math expressions

  Only use with columns defined as numeric data types

+	Add
-	Subtract
*	Multiply
/	Divide
**	Exponent

SELECT statement

- Rename a column in the answer set with “AS”

- Concatenate character columns with **`CONCAT(column1, 'literal', column2)`** (multiple input columns combined into a single output column)

- Comment out a line of code by prefixing it with “- -”

Examples

```
select 'Roster', LastName, FirstName
     from nwEmployees
select 'Roster' as 'Type', LastName, FirstName
     from nwEmployees
select 22, LastName, FirstName
     from nwEmployees
select 2*2, LastName, FirstName
     from nwEmployees
SELECT CONCAT(FirstName,' ',LastName)
     FROM nwEmployees
```

SELECT statement with WHERE clause

```
SELECT <column1>, <column2>, <column3>,
   <literal>, <math expression> AS <label>
   FROM  <tableA>
   WHERE  <condition> ;
```

SQL Basics

- The WHERE clause selects a subset of ROWs to appear in the answer set
- The condition in the WHERE clause takes this format:

  < operand > < operator > < operand >
- Operands may be columns or literals or expressions
- Operator may be

=	Equals	Like
<>	Not equals	Between
>	Greater than	In
<	Less than	

- Operator may be In or Like
  - In (literal, literal, literal)
  - Like `'string'` with % or _ as a wildcard
    - % = zero or more of any character
    - _ = exactly one of any character
- Multiple conditions may be joined with Boolean operators
  - AND, OR
- Conditions may be negated with Boolean operator
  - NOT

- **`Between`** is INCLUSIVE

- Answer Set rows may be sorted with “Order By”
  - Options are ASC or DESC
  - Defaults to ASC

- **`Boolean`** expressions are not English !
  - **`NOT`** negates the whole condition

```
SELECT Customerid, ContactName, Region, Country
      FROM nwCustomers
      where Country = 'Brazil‘
SELECT Customerid, ContactName, Region, Country
      FROM nwCustomers
      where Country NOT = 'Brazil'
SELECT Customerid, ContactName, Region, Country
      FROM nwCustomers
      where NOT Country = 'Brazil'
```

- When combining **WHERE** conditions using Boolean operators, please make a habit of using parentheses

```
SELECT Productname, SupplierID, CategoryID,
       UnitPrice, UnitsInStock
   FROM nwProducts
   WHERE SupplierID = 1 AND CategoryID = 2 OR
       CategoryID = 3 AND UnitPrice > 20 OR
       UnitsInStock < 12;

SELECT Productname, SupplierID, CategoryID,
       UnitPrice, UnitsInStock
   FROM nwProducts
   WHERE SupplierID = 1 AND (CategoryID = 2 OR
       CategoryID = 3 AND UnitPrice > 20) OR
       UnitsInStock < 12;
```

Examples

```
select Customerid, ContactName, Region, Country
      from nwCustomers
select Customerid, ContactName, Region, Country
      from nwCustomers
      where Country = 'Brazil'
select Customerid, ContactName, Region, Country
      from nwCustomers
      where Country <> 'Brazil'
select ProductID, ProductName, UnitPrice
      from nwProducts
      where UnitPrice > 60
```

Examples

```
select ProductID, ProductName, UnitPrice
      from nwProducts
      where UnitPrice between 20 and 30
select ProductID, ProductName, categoryid, UnitPrice
      from nwProducts
      where UnitPrice between 20 and 30
      and categoryid in (2, 4, 6)
select ProductID, ProductName, QuantityPerUnit
      from nwProducts
      where QuantityPerUnit like '%jars%'
```

Distinct:

- SQL cannot easily determine whether or not a row is a duplicate of another row
- The answer set may contain duplicate rows
- The “distinct” keyword before a column removes duplicates form the answer set
  - 87 Customers, each one has a country
  - How many distinct countries are they from?

Examples

Using distinct

```
Select CompanyName, ContactName, Country
       from nwCustomers
Select Country
       from nwCustomers
Select Distinct Country
       from nwCustomers
```

Handling Dates in MySQL

- MySQL supports DATE, DATETIME, and TIMESTAMP data types
- Columns with a data type of “TIMESTAMP” are stored as a 4-byte binary integer representing the number of seconds since 1970-01-01 00-00-00 UTC. TIMESTAMP has a range of '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC.
- If no value is provided for the TIME portion of a DATETIME column, it defaults to 00:00.00.0000
- To make it easier for humans to deal with date/time, MySQL allows us to reference dates/times in this format:

  YYYY-MM-DD and HH:MM.SS.nnn
- If you pass the date to MySQL as text in YYYY-MM-DD format, it will automatically convert it to the proper binary number
- If you pass the time to MySQL as text in HH:MM.SS.nnn format, it will automatically convert it to the proper binary number

Handling Dates in MySQL

YYYY-MM-DD and hh:mm.ss.nnn

- YYYY is four digits from 1000 through 9999 that represent a year.
- MM is two digits, ranging from 01 to 12, that represent a month in the specified year.
- DD is two digits, ranging from 01 to 31 depending on the month, that represent a day of the specified month.
- hh is two digits, ranging from 00 to 23, that represent the hour.
- mm is two digits, ranging from 00 to 59, that represent the minute.
- ss is two digits, ranging from 00 to 59, that represent the second.
- nnn is zero to three digits, ranging from 0 to 999, that represent the fractional seconds.

Examples: Using DATES

```
SELECT Now();
SELECT Curdate();
SELECT Curtime();
SELECT Lastname, Firstname, Extract(Year From HireDate) AS HireYear
    FROM NWEmployees;
SELECT EmployeeID, Lastname, Firstname,
        ROUND(DATEDIFF(HireDate, BirthDate)/365,0) AS HIRE_AGE
    FROM NWEmployees;
SELECT Lastname, Firstname, Date_Format(BirthDate,%M)
    FROM NWEmployees;
```

Date_Format(Date,Format)

Specifier	Description
%D	Day of the month with English suffix (0th, 1st, 2nd, 3rd, ...)
%d	Day of the month, numeric (00..31)
%e	Day of the month, numeric (0..31)
%f	Microseconds (000000..999999)
%H	Hour (00..23)
%h	Hour (01..12)
%I	Hour (01..12)
%i	Minutes, numeric (00..59)
%j	Day of year (001..366)
%k	Hour (0..23)
%l	Hour (1..12)
%M	Month name (January..December)
%m	Month, numeric (00..12)
%p	AM or PM
%r	Time, 12-hour (hh:mm:ss followed by AM or PM)
%S	Seconds (00..59)
%s	Seconds (00..59)
%T	Time, 24-hour (hh:mm:ss)
%U	Week (00..53), where Sunday is the first day of the week; WEEK() mode 0

MySQL Date Functions

https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html