SQL Queries for Beginners

Understanding the basics of SQL Queries and their uses

Disclaimer

The purpose of this tutorial is simply to provide an introduction to a variety of topics in Microsoft SQL. It is in no way a substitute for a full for-credit course on SQL, and should students want to pursue SQL on a more serious level they should look to the university courses on the subject

A note on the background of Microsoft SQL:

Back in the 1970's Dr. E. F. Codd constructed the theory behind what is now accepted as the definitive model for **relational database management systems** (RDBMS). In the Beginner Database Management Tutorial we spend a great deal of time going through the intricacies of the RDBMS. In 1979 a language was developed by IBM to use Codd's model. The language was called Structured English Query Language (SEQUEL), later shortened to become SQL (still pronounced "sequel"). Today, SQL is accepted as the standard RDBMS language. SQL is easier to learn than most programing languages, as it is much closer to standard english, and the scope of it's abilities is much smaller. Regardless, SQL is one of the most important languages to have at least some ability with, and it is still the most powerful way to access structured data.

Let's get some data:

- Download the CSV files for this tutorial
- Create a new database in SQLite
- Run this code:

```
Drop table Sales;
Drop table Sales_Rep;
Drop table Fiscal_Calendar;
Drop table Product;


Create table Sales(
Sale_ID INT NOT NULL,
SALES_Rep_ID INT,
Product_ID INT,
Date DATE,
PRIMARY KEY(Sale_ID)
);

Create table Sales_Rep(
```

```
Sales_Rep_ID INT NOT NULL,
First_Name VARCHAR,
Last_Name VARCHAR,
PRIMARY KEY(Sales_Rep_ID)
);

Create table Fiscal_Calendar(
Day DATE NOT NULL,
Day_Of_Week VARCHAR,
Week_Number INT
);

Create table Product(
Product_ID INT NOT NULL,
Product_Name VARCHAR,
Product_Class VARCHAR,
Price DECIMAL,
COGS DECIMAL,
PRIMARY KEY (Product_ID)
)
```

Importing the data

- Select the tables one at a time from the navigator bar
- Go to the "Data" tab in each table, and select Import Data
- Pick the CSV file associated with that table

The SELECT, FROM, WHERE format

The core idea of a SQL query comes from these three functions. They essentially ask the question "Show me (SELECT) the data from (FROM) these tables with some conditions applied (WHERE)"

Query Basics:

- SELECT/SELECT DISTINCT: *What columns do you want?*
- FROM: *What tables do you want us to pull the data from?*
- WHERE: *What rows do you want?*
    - Conditions: =/AND/OR
    - The LIKE operator
    - Wildcards: "%" (many characters) -or- "_" (one character)
- ORDER BY: *How do you want the rows to show?*
    - ASC/DESC
    - Limit

Aggregate Functions

- Types of aggregates: COUNT, SUM, MAX/MIN, AVG, ROUND
- GROUP BY: groups the aggregated function into buckets

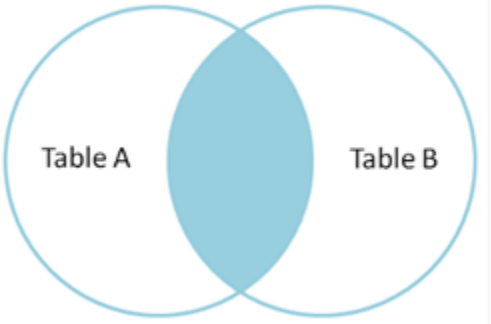
Joining Multiple Tables

- INNER JOIN, RIGHT/LEFT JOIN, OUTER JOIN, CROSS JOIN

A visual representation of SQL join statements:

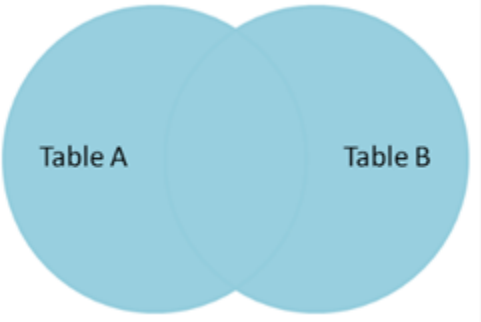
INNER JOIN:

```
SELECT * FROM TableA
INNER JOIN TableB
ON TableA.name = TableB.name
```



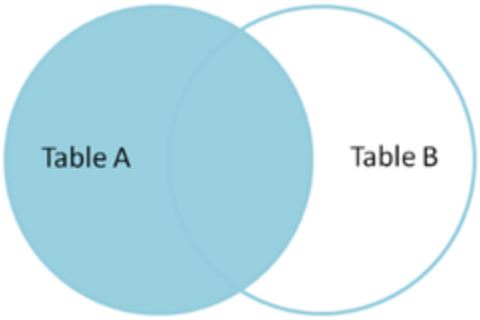
FULL OUTER JOIN:

```
SELECT * FROM TableA
FULL OUTER JOIN TableB
ON TableA.name = TableB.name
```



LEFT OUTER JOIN:

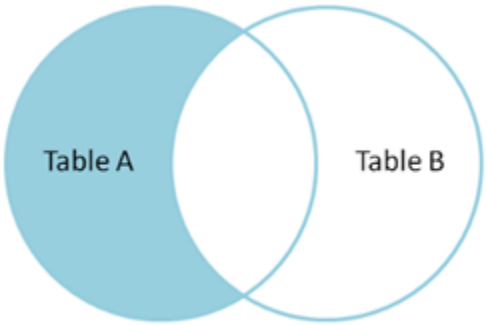
```
SELECT * FROM TableA
LEFT OUTER JOIN TableB
ON TableA.name = TableB.name
```



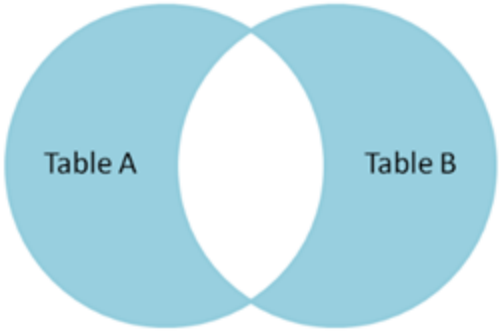
CROSS JOIN: Cannot be expressed in Venn Diagrams, but essentially gives a Cartesian Product. A Cartesian Product is the product of two sets: the product of set X and set Y is the set that contains all ordered pairs (x, y) for which x belongs to X and y belongs to Y. Cross joins should generally be avoided at all costs, as they often join data in a way that is haphazard and not useful.

Other useful ways to join tables using NULL statements:

```
SELECT * FROM TableA
LEFT OUTER JOIN TableB
ON TableA.name = TableB.name
WHERE TableB.id IS null
```




```
SELECT * FROM TableA
FULL OUTER JOIN TableB
ON TableA.name = TableB.name
WHERE TableA.id IS null
OR TableB.id IS null
```



Source for charts:
https://blog.codinghorror.com/a-visual-explanation-of-sql-joins/