Lesson 2 - Retrieving Data

The Select Statement

- This will probably be your most frequently used statement - retrieves data from one or more tables
- Requires (at a minimum) *what you want to select* and *from where you want to select it*

Retrieving Individual Columns

- Simple select statement has the following format:

```
SELECT prod_name
FROM Products;
```

- Column name is specified after **SELECT** keyword, and the database from which to retrieve the data is specified after **FROM**

Retrieving Multiple Columns

- Multiple columns can be selected with a similar **SELECT** statement:

```
SELECT prod_id, prod_name, prod_price
FROM Products;
```

Retrieving All Columns

- All columns can also be retrieved using a wildcard **SELECT** statement:

```
SELECT *
FROM Products;
```

Retrieving Distinct Rows

- To retrieve distinct values (or unique values) within a table, you can use the **SELECT DISTINCT** statement:

```
SELECT DISTINCT vend_id
FROM Products;
```

Limiting Results

- Different implementations of SQL have different keywords to limit results
- MSSQL - **TOP n**
    - **Should be after SELECT keyword**
- DB2 - **FETCH FIRST n ROWS ONLY**
- Oracle - **WHERE ROWNUM <= n**
- MySQL, MariaDB, PostgreSQL, SQLite - **LIMIT n**

Comments

- Inline comments can be created by using **--** or **#** (this is less supported)
- Multiple Line comments can be created using **/* THIS IS A COMMENT */**

Notes:

- Unless data is sorted specifically, data will be returned in no order of significance.
- Multiple statements must be separated by semicolons
- SQL language is case insensitive; i.e., SELECT == select (however tables, columns and values may not be case insensitive, depending on DBMS config)

- Whitespace is ignored in SQL statements

Lesson 3 - Sorting Data

Sorting Data

- Data can be sorted using the **ORDER BY** keyword
    - This should be the last clause in your statement

Sorting by multiple columns

- You can sort data by more than one column (for example if there are multiple employees with the same last name)
- This is done using **ORDER BY** and then separating the two columns with a comma.
    - i.e., **ORDER BY prod_price, prod_name**

Sorting by Column Position

- **ORDER BY** supports ordering by relative column position
    - i.e., **ORDER BY 2, 3**
    - This can save time because you don't have to retype column names
    - The downside is that you can mistakenly specify the incorrect column, rather than explicitly typing it out

Specifying Sort Direction

- You can sort data in descending order, rather than ascending order (this is the default)
- This is done using **ORDER BY [column name] DESC**
    - The **DESC** applies only to the column name directly preceding it in the case of multiple columns being sorted