Data Query Language (DQL)

DSC 301: Lecture 4

February 4, 2021

Lecture Objectives

- Access Database Server
- Examine Flights data (in Excel)
- Log in to db server
- Examine Flights data on Server
- SELECT statements (single table queries)

Flights data (in Excel)

- Download `Flights.csv` from Blackboard
- Determine how many flights departing Seattle on Christmas day that were delayed more than 30 minutes.
- Determine the carriers that service `CLT`.
- Make your own query and find the answer.
- Observe the difficulty in querying data in this format

Gain Access to Server

- `https://www.dbsoln.com`
- Enter UNE email address
- Copy login credentials

Connect to Database Server

- Open MySQL Workbench - enter credentials and connect
  - Use commands at the terminal or command line interface
- View features of software after connecting

SELECT Statements

- `SELECT` (without `FROM`)
  - Technically works, but trivial, impractical, and nonsensical
  - `SELECT 2+3;`
  - `SELECT 'Hello World!';`
- `SELECT FROM`
  - Besides trivial case (see above), ALL `SELECT` statements have `FROM` clause
  - Basic SELECT FROM syntax

```
SELECT <column_X> FROM <Table_Y>;
```

Example: `SELECT fid FROM Flights;`
where `fid` is a column in the `Flights` table.

  - **Note**: Use **Semicolon** to terminate `SQL` statements
- `SELECT` multiple columns by separating column names by comma(s):

```
SELECT <column1, column2> FROM <Table_Y>;
```

Example: `SELECT flight, origin, dest FROM Flights;`

- Wildcard - used to select ALL columns in a table (not advisable to use!)

```
SELECT * FROM Flights;
```

- `LIMIT` - Use the `LIMIT` clause to limit the number of records returned by the `SELECT` query. The syntax of the `LIMIT` clause is: `LIMIT [skip] number_of_records`. If a single argument is used, it represents the maximum number of records returned by the query, starting with the first. An optional integer N is used to skip the first N records. The following query string returns the first 100 records (if 100 records exist).

```
SELECT <column1> FROM <table_Y> LIMIT 100;
```

Example: Return the departure time of the first 100 flights in the database.

```
SELECT dept_time FROM Flights LIMIT 100;
```

- `ORDER BY` clause

  The `ORDER BY` clause sorts the resulting record set in ascending (`ASC`) or descending (`DESC`) order using one or more columns. Ascending order is the default. `ORDER BY` can be used with or without the `WHERE` clause. Null values will always be displayed first. The general query statement using an `ORDER BY` clause is given as follows.

```
SELECT <column1> FROM <table_Y> ORDER BY <column2>;
```

- Results sorted by `column1`, then by `column2`.

```
SELECT * FROM <table_Y> ORDER BY <column1, column2>;
```

- Results sorted by column two (i.e., the second column). Specify the column number in the `ORDER BY` clause.

```
SELECT * FROM <table_Y> ORDER BY 2;
```

  It is important to note that SQL follows ASCII character sort order. In particular, numeric digits 0 - 9 come before capital letters A - Z, which come before lower case letters a - z. Additionally, because all values are considered characters, SQL would sort the numeric values 1, 2, 3, 5, 8, 13, 21, 34, and 55 as 1, 13, 2, 21, 3, 5, 55, and then 8.

- Lastly, query results can be ordered even by columns not requested in the `SELECT` statement (same as above). Note in code below, `column2` is not displayed, but the results, `column1` is nevertheless sorted by `column2`.

```
SELECT <column1> FROM <table_Y> ORDER BY <column2>;
```

Odds and ends

- Broom icon - Cleans SQL code
- SQL syntax is not case sensitive (i.e., `SELECT` same as `select`)

- **Comments** - are brief explanations or annotations that make code more user friendly. Comments are not executed but useful to user to indicate the process or understand the results of a query, etc.
  - Single line comment syntax: MySQL `#` . Standard SQL `--`.
    - That is, `--` works with all flavors of SQL, while `#` is a MySQL thing.
  - Multiple lines or Block comments: `/*` starts the block and `*/` ends the block.
    - Note: This can be used within the code as well.

  Example:

```
SELECT fid, flight, /* tailnum,*/ origin FROM Flights;
```

Examples

1. Display the entire data set (all columns and all rows).

```
SELECT * FROM Flights;
```

2. List tail numbers from the first 150 flights.

```
SELECT tailnum FROM Flights LIMIT 150;
```

3. List the flight ID and tail numbers of 10 flights starting with the 6th. Note: First number after `LIMIT` in code below is an offset (i.e., excludes the first 5 records).

```
SELECT fid, tailnum FROM Flights LIMIT 5,10;
```

4. List the flight ID and tail numbers of 10 flights starting with the 1st. This example is for comparison with the example above. Note: `LIMIT 0,10` is the same as `LIMIT 10`.

```
SELECT fid,tailnum FROM Flights LIMIT 0,10;
```

5. List flight number, origin and destination sorted by flight number in ascending order.

```
SELECT flight, origin, dest FROM Flights ORDER BY flight;
```

6. List flight number, origin and destination sorted by carrier in descending order.

```
SELECT flight,origin,dest FROM Flights ORDER BY carrier desc;
```

Next Time

`WHERE` clause - used to filter rows (i.e., selection operator) based on criteria.