My SQL Select statement

Objectives

1. Retrieving data
2. Selecting specific columns
3. Selecting specific rows with the WHERE clause
4. Limiting data output
5. Ordering data
6. Grouping data
7. Removing duplicate items
8. Some function

1. Retrieving data

The following SQL statement is one of the most common ones. It is also one of the most expensive ones:

' * ' is use to get all the column. If there are too many column with unnecessary information, it will waste time and memory.

```
mysql> SELECT * FROM Cars;
+----+------------+--------+
| Id | Name       | Cost   |
+----+------------+--------+
|  1 | Audi       |  52642 |
|  2 | Mercedes   |  57127 |
|  3 | Skoda      |   9000 |
|  4 | Volvo      |  29000 |
|  5 | Bentley    | 350000 |
|  6 | Citroen    |  21000 |
|  7 | Hummer     |  41400 |
|  8 | Volkswagen |  21600 |
+----+------------+--------+
8 rows in set (0.00 sec)
```

2. Selecting specific columns

We can use the SELECT statement to retrieve specific columns that we need. The column names follow the SELECT word. If you use more than 1 table with same column's name, use [tablename].[columnname] to point exact column you want to get.

```
        mysql> SELECT Name, Cost FROM Cars;
        +------------+--------+
        | Name       | Cost   |
        +------------+--------+
        | Audi       |  52642 |
        | Mercedes   |  57127 |
        | Skoda      |   9000 |
        | Volvo      |  29000 |
        | Bentley    | 350000 |
        | Citroen    |  21000 |
        | Hummer     |  41400 |
        | Volkswagen |  21600 |
        +------------+--------+
8 rows in set (0.00 sec)
```

2. Selecting specific columns

We can rename the column names of the returned result set. For this, we use the `AS` clause.

```
mysql> SELECT Name, Cost AS Price FROM Cars;
+------------+--------+
| Name       | Price  |
+------------+--------+
| Audi       |  52642 |
| Mercedes   |  57127 |
| Skoda      |   9000 |
| Volvo      |  29000 |
| Bentley    | 350000 |
| Citroen    |  21000 |
| Hummer     |  41400 |
| Volkswagen |  21600 |
+------------+--------+
8 rows in set (0.00 sec)
```

3. Selecting specific rows with the WHERE Clause

In the following examples, we are going to use the Orders table.

```
mysql> SELECT * FROM Orders;
+----+------------+------------+
| Id | OrderPrice | Customer   |
+----+------------+------------+
|  1 |       1200 | Williamson |
|  2 |        200 | Robertson  |
|  3 |         40 | Robertson  |
|  4 |       1640 | Smith      |
|  5 |        100 | Robertson  |
|  6 |         50 | Williamson |
|  7 |        150 | Smith      |
|  8 |        250 | Smith      |
|  9 |        840 | Brown      |
| 10 |        440 | Black      |
| 11 |         20 | Brown      |
+----+------------+------------+
11 rows in set (0.00 sec)
```

Here we see all the data from the Orders table.

3. Selecting specific rows with the WHERE Clause

Next, we want to select get data with condition:

```
mysql> SELECT * FROM Orders WHERE Id=6;
+----+------------+------------+
| Id | OrderPrice | Customer   |
+----+------------+------------+
|  6 |         50 | Williamson |
+----+------------+------------+
1 row in set (0.00 sec)3 rows in set (0.00 sec)
```

The above SQL statement selects a row which has Id 6.

```
mysql> SELECT * FROM Orders WHERE Customer="Smith";
+----+------------+----------+
| Id | OrderPrice | Customer |
+----+------------+----------+
|  4 |       1640 | Smith    |
|  7 |        150 | Smith    |
|  8 |        250 | Smith    |
+----+------------+----------+
```

The above SQL statement selects all orders created by Smith customer.

3. Selecting specific rows with the WHERE Clause

We can also use the `LIKE` keyword to look for a specific pattern in the data.

```
mysql> SELECT * FROM Orders WHERE Customer LIKE "B%";
+----+------------+----------+
| Id | OrderPrice | Customer |
+----+------------+----------+
|  9 |        840 | Brown    |
| 10 |        440 | Black    |
| 11 |         20 | Brown    |
+----+------------+----------+
3 rows in set (0.00 sec)
```

This SQL statement selects all orders from customers whose names begin with 'B' character.

4. Limiting data output

As we mentioned above, retrieving all data is expensive when dealing with large amounts of data. We can use the `LIMIT` clause to limit the data amount returned by the statement.

This `LIMIT` clause limits the number of rows returned to 4:

```
mysql> SELECT * FROM Cars LIMIT 4;
+----+----------+-------+
| Id | Name     | Cost  |
+----+----------+-------+
|  1 | Audi     | 52642 |
|  2 | Mercedes | 57127 |
|  3 | Skoda    |  9000 |
|  4 | Volvo    | 29000 |
+----+----------+-------+
4 rows in set (0.00 sec)
```

With two arguments, the `LIMIT` returns rows beginning from an offset value.
`LIMIT` [start offset], [number limit]

```
mysql> SELECT * FROM Cars LIMIT 2, 4;
+----+---------+--------+
| Id | Name    | Cost   |
+----+---------+--------+
|  3 | Skoda   |   9000 |
|  4 | Volvo   |  29000 |
|  5 | Bentley | 350000 |
|  6 | Citroen |  21000 |
+----+---------+--------+
4 rows in set (0.00 sec)
```

5. Ordering data

To make output data is sorted, use the `ORDER BY` clause.

The `ORDER BY` clause is followed by the column on which we do the sorting.

There are 2 type of sorting data:

- The `ASC` keyword sorts the data in ascending order - this is default type.
- The `DESC` in descending order.

Using:

```
ORDER BY [column's name] [sort type]
```

```
mysql> SELECT Name, Cost
       FROM Cars
       ORDER BY Cost DESC;
+------------+--------+
| Name       | Cost   |
+------------+--------+
| Bentley    | 350000 |
| Mercedes   |  57127 |
| Audi       |  52642 |
| Hummer     |  41400 |
| Volvo      |  29000 |
| Volkswagen |  21600 |
| Citroen    |  21000 |
| Skoda      |   9000 |
+------------+--------+
8 rows in set (0.00 sec)
```

6. Grouping data

The `GROUP BY` clause is used to combine database records with identical values into a single record. It is often used with the aggregation functions.

Say we wanted to find out, the sum of each customers' orders.

```
mysql> SELECT SUM(OrderPrice) AS Total, Customer FROM Orders GROUP BY Customer;
+-------+------------+
| Total | Customer   |
+-------+------------+
|   440 | Black      |
|   860 | Brown      |
|   340 | Robertson  |
|  2040 | Smith      |
|  1250 | Williamson |
+-------+------------+
5 rows in set (0,11 sec)
```

NOTE: `GROUP BY` clause must after condition in `WHERE` clause and must before `ORDER BY` (if use `ORDER BY`).

7. Removing duplicate items

The `DISTINCT` keyword is used to select only unique items from the result set.

First, we have this query:

```
mysql> SELECT Customer FROM Orders WHERE Customer LIKE 'B%';
+----------+
| customer |
+----------+
| Brown    |
| Black    |
| Brown    |
+----------+
```

This time we have selected customers whose names begin with B character. We can see that Brown is mentioned twice.

To remove duplicates, we use the `DISTINCT` keyword:

```
mysql> SELECT DISTINCT Customer FROM Orders WHERE Customer LIKE 'B%';
+----------+
| customer |
+----------+
| Brown    |
| Black    |
+----------+
```

8. Some function

There are some useful function you can use: avg(), count(), max(), min(), sum(),...

Say we wanted to figure out, how many orders were placed by Brown customer. We would utilize the COUNT() function.

```
mysql> SELECT COUNT(Customer) AS "Orders by Brown" FROM Orders WHERE Customer="Brown";
+-----------------+
| Orders by Brown |
+-----------------+
|               2 |
+-----------------+
1 row in set (0.00 sec)
```

The customer has placed two orders.

THE
END