30 Examples to Master SQL

A comprehensive practical tutorial

Soner Yıldırım

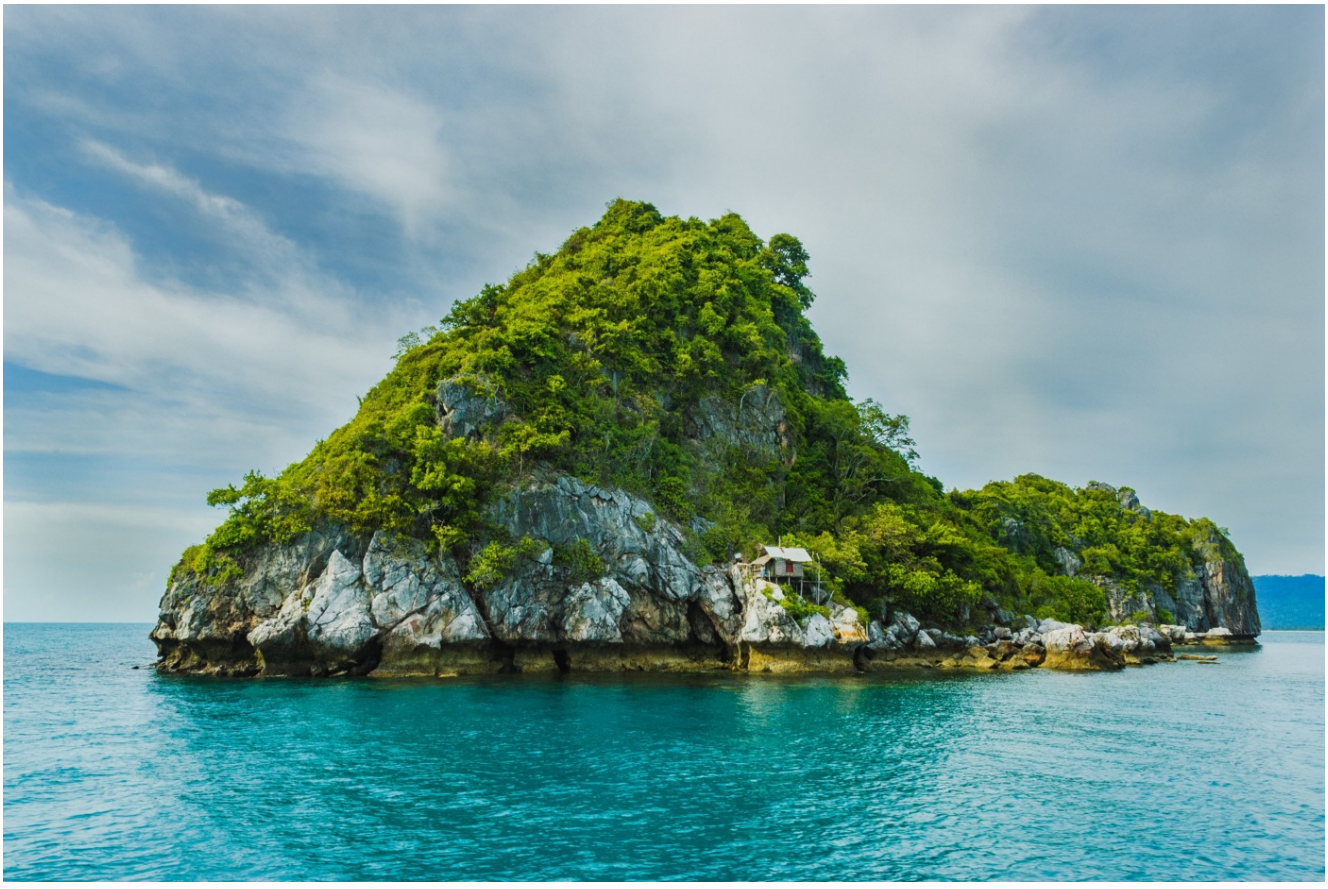

Photo by Tom Winckels on Unsplash

SQL is a programming language that is used to manage data stored in tabular form (i.e. tables) in relational databases.

A relational database consists of multiple tables that relate to each other. The relation between tables is formed in the sense of shared columns.

There are many different relational database management systems (e.g. MySQL, PostgreSQL, SQL Server). The SQL

syntax they adapt might differ slightly. However, the difference is very small so if you learn how to use one, you can easily switch to another one.

In this article, we will go over 30 examples that cover the following operations with SQL:

- Create a database and tables
- Insert data into tables
- Delete data from tables
- Update tables
- Query tables using a wide variety of select statements

There are many alternatives to use SQL in your machine or on the cloud. I'm currently using MySQL on a linux machine through the terminal. Another commonly used alternative is to install MySQL Workbench.

Example 1

We first connect to the MySQL server from the terminal and create a database.

```
~$ sudo mysql -u root
```

We will be prompted to enter the password. We are now connected to the MySQL server in our machine.

The following command creates a database called "retail".

```
mysql> create database retail;
mysql> use retail;
```

We are not in the retail database which does not contain any tables yet.

Example 2

We will first create a table called “customer” using the create table command.

```
mysql> create table customer (
    -> cust_id int primary key,
    -> age int,
    -> location varchar(20),
    -> gender varchar(20)
    -> );
```

We define the name of the columns and associated data types inside the parenthesis. The cust_id column is specified as the primary key.

> *Primary key is the column that uniquely identifies each row. It is like the index of a pandas dataframe.*

Example 3

We will create the second table which is called “orders”.

```
mysql> create table orders (
```

```
-> order_id int primary key,
-> date date,
-> amount decimal(5,2),
-> cust_id int,
-> foreign key (cust_id) references customer(
-> on delete cascade
-> );
```

In the beginning, we mentioned that relational tables are related to each other by means of shared columns. A column that relates two tables is a foreign key.

> *Foreign key is what relates a table to another one.*
> *Foreign key contains the primary key of another table.*

The cust_id column in the orders table is a foreign key and related the orders table to the customer table. We specify this condition while creating the table.

In the last line, we specify another condition with “on delete cascade” phrase. It tells MySQL what to do when a row in the customer table is deleted. Each row in the orders table belongs to a customer. Each row in the customer table contains a unique customer id and represents a customer. If a row in the customer table is removed, it means we do not have that customer any more. As a result, the orders that belonged to that customer do not have an associated customer id anymore. “On delete cascade” indicates that orders that do not have an associated customer id will also be deleted.

Example 4

The retail database contains two tables now. We can view the tables exist in a database using the show tables command.

```
mysql> show tables;+-----------------+
| Tables_in_retail |
+-----------------+
| customer         |
| orders           |
+-----------------+
```

Note: The commands in SQL ends with a semi-colon (“;”).

Example 5

The desc or describe commands provide an overview of the table in terms of column names, data types, and some additional information.

```
mysql> desc orders;+----------+--------------+--
| Field    | Type         | Null | Key | Default
+----------+--------------+------+-----+---------
| order_id | int(11)      | NO   | PRI | NULL
| date     | date         | YES  |     | NULL
| amount   | decimal(5,2) | YES  |     | NULL
| cust_id  | int(11)      | YES  | MUL | NULL
+----------+--------------+------+-----+---------
```

Example 6

We can modify existing tables. For instance, the alter table command can be used to add a new column or delete an existing column.

Let's add a column to the orders table called "is_sale".

```
mysql> alter table orders add is_sale varchar(20)
```

We write the column name and data type along with the add keyword.

```
mysql> desc orders;+----------+--------------+---
| Field    | Type         | Null | Key | Default
+----------+--------------+------+-----+--------
| order_id | int(11)      | NO   | PRI | NULL
| date     | date         | YES  |     | NULL
| amount   | decimal(5,2) | YES  |     | NULL
| cust_id  | int(11)      | YES  | MUL | NULL
| is_sale  | varchar(20)  | YES  |     | NULL
+----------+--------------+------+-----+--------
```

The is_sale column has been added to the orders table.

Example 7

The alter table can also be used to delete a column with a minor change in the syntax.

```
mysql> alter table orders drop is_sale;
```

The drop keyword is used instead of the add. We also do not have to write the data type to drop a column.

Example 8

We have tables but they do not contain any data. One way to populate tables is the insert statement.

```
mysql> insert into customer values (
    -> 1000, 42, 'Austin', 'female'
    -> );
```

The specified values are inserted into the columns in the same order. Thus, we need to keep the order consistent.

Example 9

We can insert multiple rows at the same time by separating each row.

```
mysql> insert into customer values
    -> (1001, 34, 'Austin', 'male'),
    -> (1002, 37, 'Houston', 'male'),
    -> (1003, 25, 'Austin', 'female'),
    -> (1004, 28, 'Houston', 'female'),
    -> (1005, 22, 'Dallas', 'male'),
    -> ;
```

I have added some more rows and also populated the orders table in the same way.

There are other ways to populate the tables with data. For instance, we can load a csv file using the load data infile or load data local infile statements.

Example 10

The delete from statement can be used to delete existing rows in a table. We need to identify the rows to be deleted by providing a condition. For instance, the statement below will delete the row with an order id of 17.

```
mysql> delete from orders
    -> where order_id = 17;
```

If we do not specify a condition, all rows in the given table are deleted.

Example 11

We can also update an existing row. Let's update a row in the orders table.

```
+----------+------------+--------+----------+
| order_id | date       | amount | cust_id |
+----------+------------+--------+----------+
|        1 | 2020-10-01 |  24.40 |    1001 |
+----------+------------+--------+----------+
```

This is the first row in the orders table. We want to change the order amount to 27.40.

```
mysql> update orders
    -> set amount = 27.40
    -> where order_id = 1;mysql> select * from o
+----------+------------+--------+---------+
| order_id | date       | amount | cust_id |
+----------+------------+--------+---------+
|        1 | 2020-10-01 |  27.40 |    1001 |
+----------+------------+--------+---------+
```

We write the updated values after the set keyword. The rows to be updates are identified by providing the conditions after the where keyword.

Example 12

If we want to create a table by copying the structure of an existing table, we can use the create table statement with the like keyword.

```
mysql> create table orders_copy like orders;mysq
+-----------------+
| Tables_in_retail |
+-----------------+
| customer        |
| orders          |
| orders_copy     |
+-----------------+
```

The orders_copy table has the same structure as the orders table but does not contain any data.

Example 13

We can also create a copy of an existing table with the data by using the create table and select statements together.

```
mysql> create table new_orders
    -> select * from orders;
```

It seems like a combination of two separate statements. The first line creates the table and the second line populates it with the data in the orders table.

Example 14

The drop table statement can be used to delete tables in a database.

```
mysql> drop table orders_copy, new_orders;mysql>
+------------------+
| Tables_in_retail |
+------------------+
| customer         |
| orders           |
+------------------+
```

We have successfully dropped the tables created in the

previous example.

We have two relational tables in a database. The following examples will demonstrate how we can retrieve data from these tables using select queries.

Example 15

The simplest query is to view all the columns in a table.

```
mysql> select * from orders
    -> limit 3;+----------+------------+--------
| order_id | date       | amount | cust_id |
+----------+------------+--------+---------+
|        1 | 2020-10-01 |  27.40 |    1001 |
|        2 | 2020-10-01 |  36.20 |    1000 |
|        3 | 2020-10-01 |  65.45 |    1002 |
+----------+------------+--------+---------+
```

The “*” selects all the columns and limit keyword puts a constraint on the number of rows to be displayed.

Example 16

We can select only some of the columns by writing the name of the columns instead of “*”.

```
mysql> select order_id, amount
    -> from orders
    -> limit 3;+----------+--------+
| order_id | amount |
```

```
+----------+--------+
|        1 |  27.40 |
|        2 |  36.20 |
|        3 |  65.45 |
+----------+--------+
```

Example 17

We can specify a condition for the rows to be selected using the where clause. The following query will return all the orders made on 2020–10–01.

```
mysql> select * from orders
    -> where date = '2020-10-01';+----------+---
| order_id | date       | amount | cust_id |
+----------+------------+--------+---------+
|        1 | 2020-10-01 |  27.40 |    1001 |
|        2 | 2020-10-01 |  36.20 |    1000 |
|        3 | 2020-10-01 |  65.45 |    1002 |
+----------+------------+--------+---------+
```

Example 18

The where clause accepts multiple conditions. Let's add another condition on the query in the previous example.

```
mysql> select * from orders
    -> where date = '2020-10-01' and amount > 50;
| order_id | date       | amount | cust_id |
+----------+------------+--------+---------+
|        3 | 2020-10-01 |  65.45 |    1002 |
```

```
+----------+------------+--------+----------+
```

Example 19

We may want to sort the query results which can be done by using the order by clause.

The following query will return the orders on 2020–10–02 and sort them based on the amount.

```
mysql> select * from orders
    -> where date = '2020-10-02'
    -> order by amount;+----------+------------+-
| order_id | date       | amount | cust_id |
+----------+------------+--------+----------+
|        5 | 2020-10-02 |  18.80 |    1005 |
|        6 | 2020-10-02 |  21.15 |    1009 |
|        4 | 2020-10-02 |  34.40 |    1001 |
|        7 | 2020-10-02 |  34.40 |    1008 |
|        8 | 2020-10-02 |  41.10 |    1002 |
+----------+------------+--------+----------+
```

Example 20

The order by clause sorts the rows in ascending order by default. We can change it to descending with the desc keyword.

```
mysql> select * from orders
    -> where date = '2020-10-02'
```

```
    -> order by amount desc;+----------+------------+--------+---------+
| order_id | date       | amount | cust_id |
+----------+------------+--------+---------+
|        8 | 2020-10-02 |  41.10 |    1002 |
|        4 | 2020-10-02 |  34.40 |    1001 |
|        7 | 2020-10-02 |  34.40 |    1008 |
|        6 | 2020-10-02 |  21.15 |    1009 |
|        5 | 2020-10-02 |  18.80 |    1005 |
+----------+------------+--------+---------+
```

Example 21

SQL is a versatile language which can also be used as a data analysis tool. It provides many functions to analyze and transform data while querying from a database.

For instance, we can count the number of unique days in the orders table.

```
mysql> select count(distinct(date)) as day_count
    -> from orders;+-----------+
| day_count |
+-----------+
|         4 |
+-----------+
```

The orders table contain orders in 4 different days. The “as” keyword is used to rename the column in the query result. Otherwise the name of the column would be “count(distinct(date))”.

Example 22

There are 4 different days in the orders table. We can also find out how many orders exist for each day. The group by clause will help us accomplish this task.

```
mysql> select date, count(order_id) as order_cour
    -> from orders
    -> group by date;+------------+-------------
| date       | order_count |
+------------+-------------+
| 2020-10-01 |           3 |
| 2020-10-02 |           5 |
| 2020-10-03 |           6 |
| 2020-10-04 |           2 |
+------------+-------------+
```

We count the orders and group them by the date column.

Example 23

We will calculate the average order amount in each day and order the results based on average amount in descending order.

```
mysql> select date, avg(amount)
    -> from orders
    -> group by date
    -> order by avg(amount) desc;+------------+--
| date       | avg(amount) |
+------------+-------------+
```

```
| 2020-10-01 |   43.016667 |
| 2020-10-04 |   42.150000 |
| 2020-10-03 |   37.025000 |
| 2020-10-02 |   29.970000 |
+------------+-------------+
```

Example 24

We want to modify the query in the previous example and only include days with an average amount higher than 30.

```
mysql> select date, avg(amount)
    -> from orders
    -> group by date
    -> having avg(amount) > 30
    -> order by avg(amount) desc;+------------+-
| date       | avg(amount) |
+------------+-------------+
| 2020-10-01 |   43.016667 |
| 2020-10-04 |   42.150000 |
| 2020-10-03 |   37.025000 |
+------------+-------------+
```

It is important to note that the order of the statements in the query matters. For instance, it gives an error if we put the order by clause before the having clause.

Example 25

We want to find out the maximum order amount for each day.

```
mysql> select date, max(amount)
    -> from orders
    -> group by date;+------------+-------------+
| date       | max(amount) |
+------------+-------------+
| 2020-10-01 |       65.45 |
| 2020-10-02 |       41.10 |
| 2020-10-03 |       80.20 |
| 2020-10-04 |       50.10 |
+------------+-------------+
```

Example 26

We want to combine multiple aggregate functions in a select statement. To demonstrate it, let's elaborate on the previous example. We want to see the difference between the maximum order and minimum order of each customer. We also want to sort the results based on the difference in ascending order and display the first three.

```
mysql> select cust_id, max(amount) - min(amount)
    -> from orders
    -> group by cust_id
    -> order by dif desc
    -> limit 3;+---------+-------+
| cust_id | dif   |
+---------+-------+
|    1007 | 46.00 |
|    1009 | 28.95 |
|    1002 | 24.35 |
+---------+-------+
```

The dif column is obtained by substracting the minimum amount from the maximum amount.

Example 27

We are switching to the customer table now. Let's find out how many female and male customers we have in each city. We need to write both location and gender columns in the group by clause.

```
mysql> select location, gender, count(cust_id)
    -> from customer
    -> group by location, gender;+----------+---
| location | gender | count(cust_id) |
+----------+--------+----------------+
| Austin   | female |              2 |
| Austin   | male   |              1 |
| Dallas   | female |              2 |
| Dallas   | male   |              2 |
| Houston  | female |              2 |
| Houston  | male   |              1 |
+----------+--------+----------------+
```

Example 28

The customer and orders tables are related to each other based on the cust_id column. We can query data from both tables using SQL joins.

We want to see average order amount for each city in the

customer table.

```
mysql> select customer.location, avg(orders.amou
    -> from customer
    -> join orders
    -> on customer.cust_id = orders.cust_id
    -> group by customer.location;+----------+--
| location | avg       |
+----------+-----------+
| Austin   | 33.333333 |
| Dallas   | 34.591667 |
| Houston  | 44.450000 |
+----------+-----------+
```

Since we select columns from two different tables, the column names are specified with associated table name. The second, third, and fourth line of the query above joins the customer and orders table based on the cust_id column in each table.

Please note that the column names do not have to be the same. Whatever column name we provide with the “on” keyword, the comparison or matching is done based on these columns.

Example 29

We want to see the average age of customers who have an order on 2020–10–03.

```
mysql> select avg(c.age) as avg_age
```

```
    -> from customer c
    -> join orders o
    -> on c.cust_id = o.cust_id
    -> where o.date = '2020-10-03';+---------+
| avg_age |
+---------+
| 30.0000 |
+---------+
```

We can use aliases for the table names as well. It comes in handy when we need to type the table names many times.

Example 30

We want to see the location of customer who has the highest amount of order.

```
mysql> select c.location, o.amount
    -> from customer c
    -> join orders o
    -> on c.cust_id = o.cust_id
    -> where o.amount = (select max(amount) from
    -> ;+----------+--------+
| location | amount |
+----------+--------+
| Dallas   |  80.20 |
+----------+--------+
```

We have a nested select statement in this query. The condition on the amount is calculated using a separate select statement from the orders table.

This task can be completed in a different way. I have chosen this method to introduce the idea of nested queries.

Conclusion

I believe the 30 examples in this article provide a comprehensive introduction to SQL. We have covered the following topics:

- Creating a database with relational tables
- Modifying tables
- Inserting data into tables
- Deleting data from tables
- Writing queries to retrieve data from tables

There are, of course, more advanced queries and operation that can be done with SQL. It is better to move on to the more advanced operations once you are comfortable working with the basics.

Thank you for reading. Please let me know if you have any feedback.