6 Query Examples to Boost Your SQL Skills

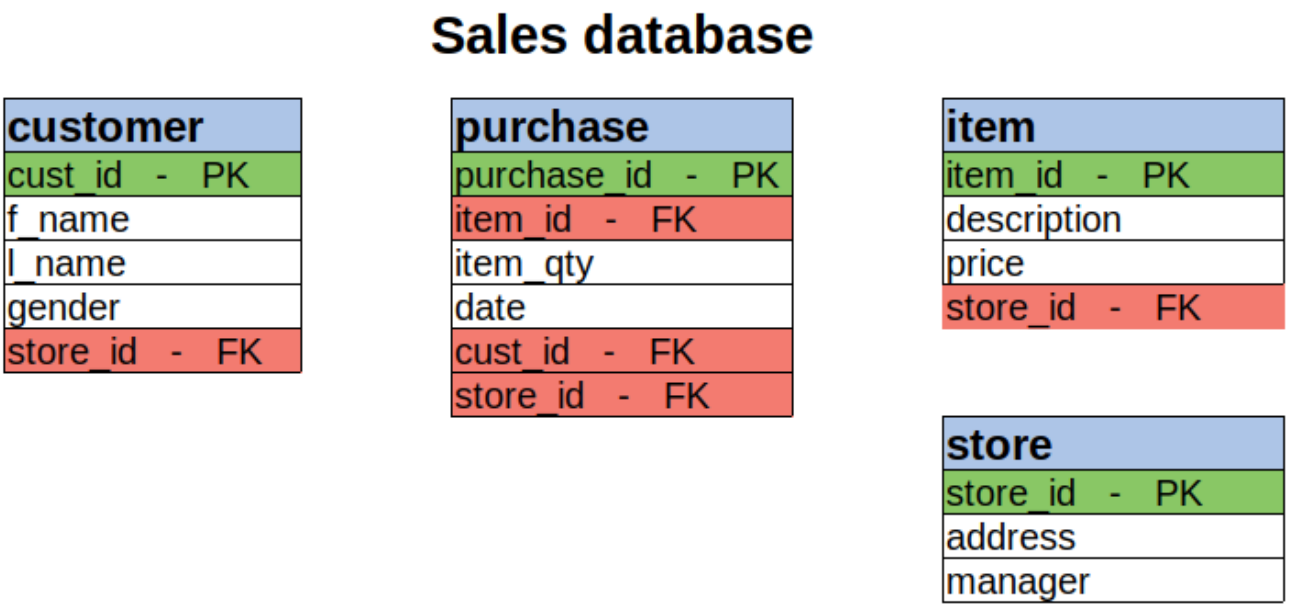
Aggregating, joining, filtering, and more

Photo by SpaceX on Unsplash

SQL is a programming language used to manage data in a relational database. Data is in tabular form with labelled rows and columns. A relational database typically consists of many tables that are related by means of shared columns.

In this article, we will go over 6 examples to query a relational database. The examples aim to solve data analysis and manipulation tasks including filtering, aggregation, sorting, and joining tables.

I previously created a sales database with 4 tables. The following figure illustrates the structure of the database and tables.



(image by author)

The columns marked with green are the primary keys and the ones in pink represent the foreign keys.

- Primary key is the column that uniquely identifies each row. It is like the index of a pandas dataframe.
- Foreign key is what relates a table to another one. Foreign key contains the primary key of another table. For instance, the "item_id" in the purchase table is a foreign key. It stores the rows from the primary key in the item table.

After the brief introduction, let's start with the examples.

Note: There are many relational database management systems (e.g. MySQL, PostgreSQL, SQL Server). Although the SQL syntax is mostly the same for all, there might be

small differences. We will be using MySQL in this article.

Example 1

The item table contains the price, description, and store information for each item. We may want to find the average item price for each store.

```
mysql> select * from item limit 3;
+---------+-------------+-------+----------+
| item_id | description | price | store_id |
+---------+-------------+-------+----------+
|       1 | apple       |  2.45 |        1 |
|       2 | banana      |  3.45 |        1 |
|       3 | cereal      |  4.20 |        2 |
+---------+-------------+-------+----------+
```

We may want to find the average item price for each store.

```
mysql> select store_id, avg(price)
    -> from item
    -> group by store_id;+----------+------------
| store_id | avg(price) |
+----------+------------+
|        1 |   1.833333 |
|        2 |   3.820000 |
|        3 |   3.650000 |
+----------+------------+
```

The store_id and price columns are selected and then grouped by the store_id. The aggregation function which

is “avg” in this case is applied while selecting the column.

Example 2

Let’s find out the name of the customer who has made the most number of purchases. This task requires to retrieve data from two tables.

The name is selected from the customer table and the number of purchases is calculated using the purchase table. Thus, we need to join these two tables.

```
mysql> select concat(customer.f_name," ", custome
    -> count(purchase.cust_id) as number_of_purch
    -> from customer
    -> join purchase
    -> on customer.cust_id = purchase.cust_id
    -> group by name;+--------------+-----------
| name         | number_of_purchases |
+--------------+---------------------+
| Adam Gelvin  |                   2 |
| Alisha T.    |                   1 |
| Elaine Smith |                   2 |
| Jane Doe     |                   2 |
| John Doe     |                   2 |
| Robert Sam   |                   1 |
+--------------+---------------------+
```

The shared column is the cust_id so we use it as the condition for joining the tables.

Example 3

We want to sort the dates based on the total spent amount. This task also requires to retrieve data from two tables.

We select the date, item_id, and item_qty from the purchase table. In order to calculate the amount, we need the price of an item which is accessed through the item table.

```
mysql> select p.date, sum(p.item_qty * i.price) a
    -> from purchase p
    -> join item i
    -> on p.item_id = i.item_id
    -> group by p.date;+------------+-----------
| date       | total_amount |
+------------+--------------+
| 2020-05-10 |        52.95 |
| 2020-05-11 |         8.70 |
+------------+--------------+
```

The total amount column is calculated by multiplying the price and quantity. One of the nice things about SQL is that it allows for doing such calculations and aggregations when selecting the column.

You may notice that we can also use aliases for table names to shorten the code and ease the typing.

The purchase table contains purchases from only two

days. We can confirm by applying the distinct function on the date column.

```
mysql> select distinct(date) from purchase;+----
| date       |
+------------+
| 2020-05-10 |
| 2020-05-11 |
+------------+
```

Example 4

Let's do a slightly more complicated examples. Consider a case where we need to calculate the total purchase amount for females and males.

This tasks involves joining three tables.

- Gender from customer table
- Item quantity from purchase table
- Item price from item table

```
mysql> select c.gender, sum(p.item_qty * i.price)
    -> from customer c
    -> join purchase p on c.cust_id = p.cust_id
    -> join item i on p.item_id = i.item_id
    -> group by c.gender;+--------+--------------
| gender | total_amount |
+--------+--------------+
| F      |        29.15 |
| M      |        32.50 |
```

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

The multiple join operations can be combined in a query in a chain-like style.

Example 5

Consider we want to find the ids of customers who purchased ice cream. After joining the purchase and item tables, we need to apply a filter using the where clause.

```
mysql> select p.cust_id, i.description
    -> from purchase p
    -> join item i
    -> on p.item_id = i.item_id
    -> where i.description = "icecream";+--------
| cust_id | description |
+---------+-------------+
|       4 | icecream    |
|       3 | icecream    |
|       5 | icecream    |
+---------+-------------+
```

Example 6

In the item table, there is a store associated with each item. We want to find the manager of the store which is associated with “cereal”.

One way to solve this task is to join tables. Another option is to implement nested select statements which can be

done as follows.

```
mysql> select manager from store
    -> where store_id = (
    -> select store_id from item
    -> where description = "cereal"
    -> );+---------+
| manager |
+---------+
| Max     |
+---------+
```

We select the manager from the store table based on a condition which is specified with another select statement. The desired store_id is found by filtering the description column in the item table.

Conclusion

We have done 6 examples to query a relational database. One of the key characteristics of a relational database is to have many tables to store data.

The tables are related by means of shared columns. The data we need is typically spread out in multiple tables. Thus, we often need to join tables or use nested select statements.

In order to retrieve the required data efficiently, we need to be able to write complex queries. The best way to get comfortable with writing such queries is to practice.

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