Review of SQL JOINS

Overview on SQL Interview Questions and Topics

Jose Marcial Portilla

6 min read

May 16, 2016

--

About Me: I teach Python,Data Science, Interviewing, and SQL on Udemy.

In this article I've written up a quick guide to remind people how JOINS work in case they are going to find themselves interviewing soon.

JOINS

Pretty much for any SQL interview you're going to need to know how the different types of JOINS work. I'll present the most common joins here along with a few venn diagram representations of what is going on. You should note however, that venn diagram analogies can only go so far with SQL.

We'll start with two tables, Registrations and Logins:

```
Registrations       Logins
id name              id  name
-- ----              --  ----
1  Bobby             1   Yolanda
2  Xavier            2   Bobby
3  Albert            3   William
4  Zack              4   Albert
```

These tables represent a social media website with two tables. Registrations has an event id and then the name of who registered, and Logins has an event id and the name of who logged in. This is a very basic example (and may not actually make sense since Yolanda and William are shown to have never registered) but we will use it to show the effects of different JOINS. A quick note, Albert and Bobby are the only names in both tables. Other names that start with letters at the end of the alphabet are not in both tables.

INNER JOIN

An *INNER JOIN* will result with the set of records that match in both tables.

```
SELECT * FROM Registrations
INNER JOIN Logins
ON Registrations.name = Logins.name

id  name       id   name
--  ----       --   ----
1   Albert     2    Albert
```

```
3   Bobby       4     Bobby
```


An INNER JOIN, (also written LEFT INNER JOIN)

. . .

FULL OUTER JOIN

A FULL OUTER JOIN will result in the set of all records in both tables.

```
SELECT * FROM Registrations
FULL OUTER JOIN Logins
ON Registrations.name = Logins.name

id    name        id    name
--    ----        --    ----
1     Albert      2     Albert
2     Xavier      null  null
3     Bobby       4     Bobby
4     Zack        null  null
null  null        1     Yolanda
null  null        3     William
```

Note how if there is no match, the missing side will produce a null.

FULL OUTER JOIN

. . .

LEFT OUTER JOIN

A LEFT OUTER JOIN results in the set of records that are in the left table, if there is no match with the right table, the results are null.

```
SELECT * FROM Registrations
LEFT OUTER JOIN Logins
ON Registrations.name = Logins.name

id  name       id    name
--  ----       --    ----
1   Albert     2     Albert
2   Xavier     null  null
3   Bobby      4     Bobby
4   Zack       null  null
```

LEFT OUTER JOIN

. . .

SELF JOIN

The last join to talk about doesn't really have a venn diagram that can properly visualize it. A Self Join is used to join a table to itself as if the table were two tables, by temporarily renaming at least one table in the SQL statement.

Quick example of the syntax:

```
SELECT a.column_name, b.column_name
FROM table1 a, table1 b
WHERE a.common_field = b.common_field;
```

Using WHERE Clauses for Exclusions

We can use the OUTER JOINS we previously saw along with some WHERE clauses to exclude records we don't want. Since we know some JOIN statements will result in null values, we can use this to our advantage.

For example, if we wanted to produce a set of records that only appear in the Registration table, we can specify to:

```
SELECT * FROM Registrations
LEFT OUTER JOIN Logins
ON Registrations.name = Logins.name
WHERE Logins.id IS null

id  name       id     name
--  ----       --     ----
2   Xavier     -      -
4   Zack       -      -
```

A LEFT OUTER JOIN with a WHERE clause excluding nulls in the logins table

. . .

Another example of this is trying to produce a set of records that is unique to the Left Table and the Right Table. In other words, for our case: Produce a table that has the records that are unique to either Registrations or Logins.

```
SELECT * FROM Registrations
FULL OUTER JOIN Logins
ON Registrations.name = Logins.name
WHERE Registrations.id IS null
OR Logins.id IS null

id    name       id    name
```

```
--    ----       --    ----
2     Xavier     --    ---
4     Zack       --    ---
--    ---        1     Yolanda
--    ---        3     William
```

Example Interviews

Now for some example interview questions. Most basic SQL interview questions consist of telling you the schema of two tables and then asking you to produce various queries from them. For roles where you will be doing basic querying of a database (for example a Data Scientist who will have to query Amazon Redshift, or maybe you job requires you to use HIVE) you can expect to be more concerned with querying a database versus creating tables. This article will focus only on query questions.

Interview Question 1

Let's continue with our theme of registrations and logins. You're interviewing at a large social media company and are given two tables: Registrations and Logins. Each table consists of a user id and a date timestamp of the event. The first 4 rows of each table are shown below:

```
Registrations       Logins
id date              id  date
-- ----              --  ----
1  2015-01-10        1   2015-02-12
3  2015-01-12        1   2015-03-04
5  2015-02-02        3   2015-03-15
9  2015-02-09        4   2015-04-22
```

Each user has only one registration entry, but could have multiple login entries (or

none). The question:

Write a query that will give the number of times each user logged in within their first week of registration (i.e. within 7 days after registering for the site).

Interview Question 2

You are interviewing for a position at a CRM company. Your given two tables, Employees and Departments. The Employees table consists of the employee id number, the department id, the supervisor id, the employee name, and their salary. The Departments table consists of the department id and the department name. Below are the an example two rows of each table.

```
Employees                                Departments
empid deptid supid name  salary          deptid   name
--    --     --    ---   ----            ---      ----
1     7      12    Bob   90              7        Marketing
2     8      11    Amy   110             1        Sales
```

We'll tackle several questions for this pair of tables. Note: There will be more than one way to answer these questions.

Can you list employees by name who have a larger salary than their supervisor?

Interview Question 3

You are interviewing for an online marketplace company and are presented with three tables:

```
Table 1: Salespeople
ID     Name     Age    Salary
--     --       --     ---
```

```
1       Albert    40      90000
2       Bob       38      88000

Table 2: Customers

ID      Name          City         Industry
--      -----         -----        ---
5       Hooli         Sunnyvale    V
7       PiedPiper     Palo Alto    C

Table 3: Orders

Number     Order_Date      Cust_id     Salesperson_id    Amount
--         -----           -----       ------            ------
10         3/2/12          5           1                 11000
20         4/8/14          7           2                 50000
```

Now for a few questions!

Write a query to list the names of all the salespeople that have an order with Hooli.

Write a query the returns the names of all salespeople that do not have an order with Hooli.

MISCELLANEOUS QUESTIONS

Here are a couple of questions you may get asked that don't have a direct query involved, along with some answers.

What is the difference between UNION and JOIN?

A JOIN allows a user to find records on another table based on a given condition between the two tables.

A UNION operations allows a user to add 2 similar dat sets to create a resulting data set that contains all the data from the source data sets. The syntax for a UNION looks like:

```
SELECT * FROM Table_A
UNION
SELECT * FROM Table_B;
```

What is the difference between UNION and UNION ALL?

UNION ALL is almost the same at UNION except it will return all rows of both datasets, even if there are duplicates.

What is the difference between WHERE and HAVING?

Both WHERE and HAVING serve as filters for querying records in a database. The difference is that the WHERE clause can only be applied on static non-aggregated columns, unlike HAVING which can be applied on aggregated columns. For instance if you need to continue filtering *after* having done a GROUP BY command, you can use HAVING.

Hope this helps anyone who needed the refresher!

Sql

Programming

Data Science

--

Written by Jose Marcial Portilla

5.3K Followers

Data Scientist and Online Instructor

Follow