/*SQL HACKS*/

Table of Contents

Introduction

This document contains hacks for SQL. It will guide you through important Window functions along with those that are not supported by MySQL, common errors that are made while writing queries and the best tips to be followed while writing them. Best resources links are given which includes advance topics as well.

Windows Functions

It's important to know operation of windows functions like Lag, Lead, and NTILE.

- The LEAD() and LAG() function in MySQL are used to get preceding and succeeding value of any row within its partition. These functions are termed as non-aggregate Window functions. Refer the link below for these:

  https://www.geeksforgeeks.org/mysql-lead-and-lag-function/

- There is no Windows function for calculating Cumulative Sum or Running Total in MySQL. Hence we can employ the following method:

  https://popsql.com/learn-sql/mysql/how-to-calculate-cumulative-sum-running-total-in-mysql

- Similarly for finding median we can use the following method:

```
set @rownum:= -1;
select avg(mtest) from
(select mtest,
@rownum := @rownum + 1 as rng
from student order by mtest) as tt
where rng in (ceil(@rownum/2), floor(@rownum/2));
```

Self Join(Application)

Self joins can be used for comparing row within same table. For example people with same names, blood groups. It is extensively used in industries for modelling any kind of hierarchy, finding duplicates, etc. Go through the given example very carefully.

E.g.: From a table of employees, create a list of employees and their manager:

Table name: employee_table

Relevant columns: employee_name, employee_id, manager_id

Query:

```
SELECT e1.employee_name AS employee,
E2.employee_name AS manager
FROM employee_table AS e1
LEFT JOIN employee_table AS e2
ON e1.manager_id = e2.employee_id;
```

/*Notice the common column on which tables are joined*/

At Your Fingertips

- The sequence of execution of a query in SQL follows this hierarchy :
  From -> where -> select
  Hence despite of having “Where” clause we needed “Having” clause to filter groups based on specific conditions.

- **SUM CASE WHEN** is a powerful way to aggregate and count based on conditions. To calculate total number of orders shipped and On Hold:-

```
SELECT
        SUM (CASE
                WHEN status = 'Shipped' THEN 1
         ELSE 0
                END) AS 'Shipped',
        SUM (CASE
                WHEN status = 'On Hold' THEN 1
         ELSE 0
                END) AS 'On Hold' from Orders;
```

- Most popular concepts asked in Interviews:

1. CASE WHEN
2. Self joins. Common in product
3. DISTINCT and GROUP BY
4. Left vs Outer joins
5. UNION
6. SUM and COUNT
7. Date-time manipulation
8. String formatting
9. Window functions like rank and row
10. Subqueries

Best Practices

- Best way to write column names: Write "from Table_Name" and then "select col", in this way col names appear in drop down and no typing mistake!

- Debug your query along the way. Write it in pieces to check that your logic and syntax work.

- Using **Common Table Expressions** (CTEs), we can write and maintain complex queries via increased readability and unlike temp table we can refer to derived table multiple times in a single query. Reference for CTEs:

  https://www.mysqltutorial.org/mysql-cte/

- Use Index to improve the speed of data retrieval on a table when dealing with huge datasets. Though speed comes along with additional writes and storage to maintain it. Reference for Index:

  https://www.mysqltutorial.org/mysql-index/mysql-create-index/

Common Mistakes

- While using **NULL**, always use "IS NULL" instead of "= NULL".

  Correct: Select * from table where H_Date **is null**;

  Incorrect: Select * from table where H_Date = null;

- Trying to use non-aggregate columns in the SELECT statement with a GROUP BY. If you aren't grouping on the variable, it shouldn't be in the SELECT statement. Fullname is non aggregate column in the example:

  Correct: Select **class**, Count (*) from student **Group by Class**;

  Incorrect: Select class, **Fullname**, Count (*) from student **Group by Class**;

- **Commas and parentheses**. Be very careful that you have the right commas in the right place. E.g.: while writing multiple col names, using functions.

- Be careful about when to use count (col_name) and count (Distinct col_name). Consider given student table for the difference:

S.ID	S.Name	Marks
1	Ron	55
2	MARY	89
3	PETER	100
4	WENDY	99
5	Tobeee	88
6	Prateik	95
7	Rupa	98
8	Y43T	75
9	Ramish	87
10	Rupa	81

- To find number of distinct name we will use:
  Select count (distinct S.name) from student;

- To find number of rows we will use:
  Select count (S.name) from student; OR

  Select count (*) from student;

  /*count (*) counts all the rows of Primary Key col, hence all the rows of a table.*/

- Avoid using “&”, “AND”, “OR” inside **IN statements**. Use ",". E.g.:

  Correct: Select * from student where **Marks In (98,100)**
  Incorrect: Select * from student where Marks In (98 & 100)

SQL Resources

Some of the best SQL resources are listed here. These include beginner, intermediate and in some cases advanced topics. Check them out:

1. Zachary Thomas' SQL Questions https://lnkd.in/g-JJzuD
2. Select * SQL: https://selectstarsql.com/
3. Leetcode: https://lnkd.in/g3c5JGC
4. DataCamp course: https://lnkd.in/gii9n9m
5. LinkedIn Learning: https://lnkd.in/gQXFc4n
6. Window Functions: https://lnkd.in/g3RtPCJ
7. HackerRank: https://lnkd.in/grv_9sB
8. W3 Schools: https://lnkd.in/gJPfrrv
9. CodeAcademy: https://lnkd.in/gT5xmpN
10. SQLZoo: https://sqlzoo.net/