**SQL Interview Questions (Discover)**

Q: What are the different types of JOINS used in SQL? (The differences and when to use each one)

* INNER JOIN: It will only include rows from the tables where the join conditions are satisfied, so in this case if table A or table B have the same employee\_id:

Diagram

Description automatically generated

* LEFT JOIN: returns the matches rows from both tables, BUT it also returns all the UNMATCHED rows from the left side as well-> it will show NULL for cells on the right that didn’t match on the left

Diagram

Description automatically generated

* RIGHT JOIN: similar to Left join, it will return all matched rows between table A and table B but it will also return all UNMATCHED rows from the right side as well -> it will show NULL for cells on the left that didn’t match on the right

Diagram

Description automatically generated’

* FULL OUTER JOIN: it returns every row between the two tables, with no duplicates. Basically it will contain all the matched rows between the two tables, all the unmatched rows from a left join, and all the unmatched rows from a right join with NULL values present where there wasn’t a matched row between A and B

Diagram

Description automatically generated

* CROSS JOIN: cartesian product for all combinations of A and B: you get every combination of every row in table A matched with every row in table B.
  + If the first table has x number of rows and the second table has n number of rows, cross join gives x\*n number of rows in the output. You should avoid cross join on larger tables because it might return a vast number of records and SQL Server requires a lot of computing power (CPU, memory and IO) for handling such extensive data.

Diagram

Description automatically generated

Diagram

Description automatically generated

* SELF JOIN: You join a table with itself so it appears twice in the query. You would do this for **hierarchical data.** We would learn which employee reports to which manager

A picture containing graphical user interface

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Link: <https://blog.quest.com/an-overview-of-sql-join-types-with-examples/#:~:text=The%20cross%20join%20joins%20each,ID%201%2C2%20and%203>.

**Q: Difference between the HAVING clause vs the WHERE clause and when to use each?**

* The WHERE clause filters or is applied to individual rows BEFORE grouping (First)

Graphical user interface, application

Description automatically generated

* The HAVING clause filters data AFTER GROUPING

A picture containing diagram

Description automatically generated

* A query can contain BOTH a WHERE clause and a HAVING clause

Graphical user interface, text, application

Description automatically generated

For example, imagine that you're joining the titles and publishers tables to create a query showing the average book price for a set of publishers. You want to see the average price for only a specific set of publishers - perhaps only the publishers in the state of California. And even then, you want to see the average price only if it is over $10.00.

1. You can establish the first condition by including a WHERE clause, which discards any publishers that are not in California, before calculating average prices.
2. The second condition requires a HAVING clause, because the condition is based on the results of grouping and summarizing the data.

Graphical user interface, text, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**Q: Say you have a dataset with a payment history profile of 24 characters where each month of the payment cycle is characterized by a character of the past 24 months**

***Example*** *if the month is April we would put a character representing each month a customer had used their card in that month*

Apr Mar Feb Jan Dec Nov Oct Sep Aug

1st 2nd 3rd 4th 5th 6th 7th 8th 9th ….

**Q: How would you pull all the rows of every person that made at a payment in February (the 3rd slot)?**

A: Since the 24-character variable is stored as a string I could use the **LIKE operator** to select the character representing the 3rd month and use # and \* to specify that I want the 3rd entry and don’t care about anything after the 3rd month.

* The percent sign (%): represents zero, one, or multiple numbers or characters
* The underscore (\_): represents a single number or character

<https://www.tutorialspoint.com/postgresql/postgresql_like_clause.htm>

<https://www.sqlshack.com/overview-of-the-sql-like-operator/>

<https://www.w3schools.com/sql/sql_wildcards.asp>

|  |  |  |
| --- | --- | --- |
| Question: | Answer: | Explanation |
| Find a text that begins with 200 | Where SALARY::text LIKE ‘200**%**’ | Looks for any text that starts with 200, but doesn’t care about any text after |
| Find a text that has the sequence of 200 anywhere in the string | Where Salary::text LIKE ‘**%**200**%**’ | The **%** sandwiched between the 200 will search through the whole character string |
| Find a text where the 3rd position has a ‘0’ in it | Where Salary::text LIKE  ‘\_ \_ 0’ | The \_ designated a single character of any type, so we want 2 of any character and then a 0 on the 3rd position |
| Find any values that start with 2 and are at least 3 characters in length? | WHERE Salary::text LIKE  ‘2\_ \_%’ | Here the \_ \_ signifines any 2 characters and we don’t care if it has more or not |
| Find any customer name that begins with a b, s, or p | WHERE customerName LIKE  ‘[bsp]%’ | The bracket is an ‘OR’ statement standing any letter in there |
| Find any customer name that does NOT start with an b, d, or f | Where CustomerName LIKE  ‘[!bdf]%’ | The ! signifies NOT or anything but these values |
| Find a customer name that starts with a, b,c, or d | Where CustomerName Like  ‘[a-d]’ | The ‘-‘ represents a sequence so any and all values between a and d: a,b,c,d |
| Find all rows that have phone numbers formatted correctly: 999-999-9999 | Where Phone\_Number LIKE  ‘[0-9][0-9][0-9] – [0-9][0-9][0-9] – [0-9][0-9][0-9][0-9]’ | This shows that we accept any numeric value between 0-9 in first position and then the ‘-‘ split the phone number up |
| Find all rows that have the character string of ‘100% Free’ | Where StickerName LIKE  ‘%100**!%** Free%’ **ESCAPE ‘!’** | Here, we need to ensure that SQL doesn’t get confused with the % sign so we put the ! before the ‘%’ sign to show we want to specifically find this character. We want to keep the sql usage of % in the other cases normal |

**Q: Rank vs DenseRank vs RowNumber:**

* The main difference between the three is how they handle ties between two rows.
  + Rank(): If there is a tie between two records they will both have the same ranking, but the next ranking will jump depending on how many ties:  
    **(1,1,3,4,5,6,6,6,9):** here two people tied for 1st, the next entry gets 3rd place
  + DenseRank():if there is a tie between two ties, there is no jumping over any numbers this means there is a sequential ranking of 1,2,3, etc. when ties:  
    **(1,1,2,3,4,5,5,5,6):** here two people tied for 1st, the next entry gets 2nd place
  + Row\_Number(): it doesn’t handle ties well and will just default to picking one row over the other randomly. It will NOT return any duplicate rankings.  
    (1,2,3,4,5,6,7,8,9): even if two rows are tied, they will get assigned a sequential #

Table

Description automatically generated

Q: How would I pull a random number of 100 rows in a dataset? (What function?)

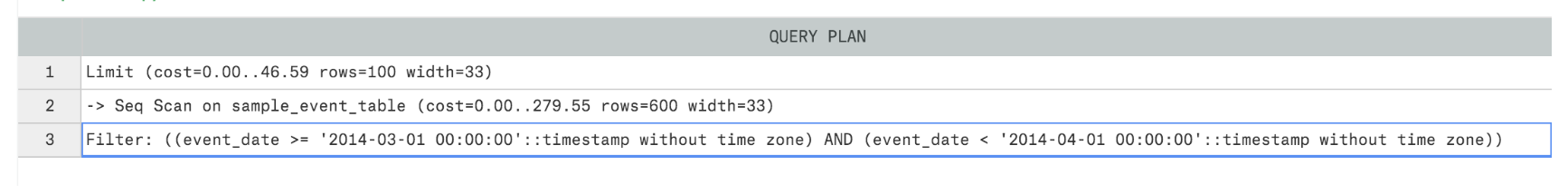
* You could use the RAND() function to order the dataset a certain way for id\_number and then select the LIMIT 100 to pull those values:   
  **select \*   
  from tablea  
  ORDER by RAND()  
  LIMIT 100**

Q: How would I improve a long running query for RUNTIME?

* Reducing the table size initially by filtering the data: using where clauses on certain date ranges
* Use the LIMIT clause to only pull the first 100 rows if our analysis is only looking at the top 100 customers
* Making Joins less complicated -> join certain tables first in which the row number will be less when joining with other tables in the future
* Select only specific columns you want to look at anything instead of select \*
* Add the EXPLAIN operator at the beginning of any working query to get a sense of how long it will take:

Text

Description automatically generated

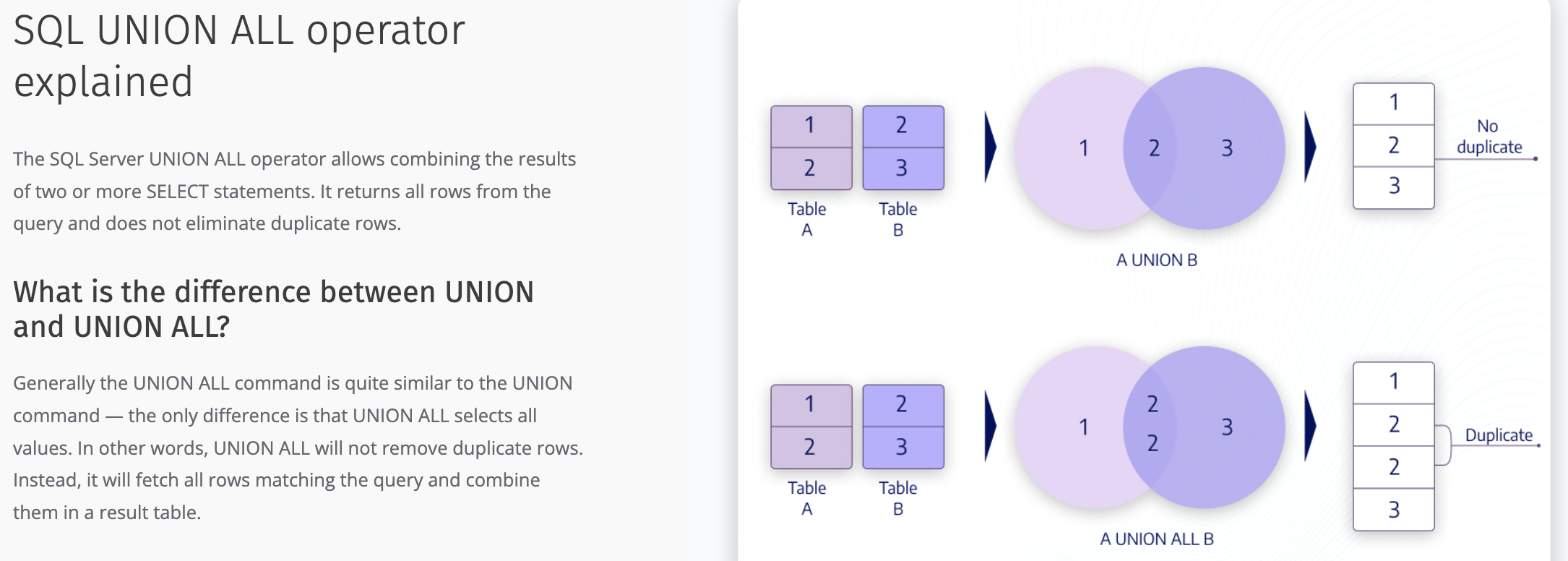


* The entry at the bottom of the list is executed first. So this shows that the WHERE clause, which limits the date range, will be executed first.
* Then, the database will scan 600 rows (this is an approximate number). You can see the cost listed next to the number of rows—higher numbers mean longer run time. You should use this more as a reference than as an absolute measure.
* To clarify, this is most useful if you run EXPLAIN on a query, modify the steps that are expensive, then run EXPLAIN again to see if the cost is reduced. Finally, the LIMIT clause is executed last and is really cheap to run (24.65 vs 147.87 for the WHERE clause)

Q: UNION vs UNION ALL?

<https://www.devart.com/dbforge/sql/sqlcomplete/union-vs-union-all.html>

* You use UNION or UNION all when you want to stack 2 tables vertically on one another. This means that table A and table B have the SAME number of columns with the SAME column names
* We use UNION, if we want to look for any duplicate rows that may exist when stacking table A and table B on top of one another. Using UNION will **eliminate** any duplicate rows between table A and table B and only leave DISTINCT records.



Q: Say I have a field called customer id that is 10 characters long (0000012345) but is stored as a string. However, say that dataset was converted into numeric (12345) and you want to bring it back to a 10-character string field (0000012345)?

How would you add the leading 0s back into the field and make sure that every entry was 10 characters long?

A: Use LPAD and RPAD to add the leading zero’s back.

*Select (customerID, 10, ‘0’) as customer\_ID\_padded*

*From tableA*

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

The long way would be using case when statements for the length and adding the leading zeroes that way for each length type less than 10: case when length(customer\_id) = 7 then CONCAT( ‘000’, customer\_id)

Q: How would I extract just the DATE from a DATE-TIMESTAMP field?

A: You can use the DATE() function to get ONLY the date

SELECT CAST(DateTime\_column\_name/ Expression as date)

Table

Description automatically generated

<https://www.golinuxcloud.com/sql-getdate-date-only/#Overview_of_SQL_DateTime_datatype>

Q: What function would you use to convert a string into an integer or numeric field?

<https://learnsql.com/cookbook/how-to-convert-a-string-to-a-numeric-value-in-postgresql/#:~:text=Use%20the%20%3A%3A%20operator%20to,convert%20between%20different%20data%20types>.

* You can use the “**::”** operator to convert a column to a different data type

Graphical user interface, text, application, email

Description automatically generated

A: Use the CAST function to convert a string to numeric

Graphical user interface, text, application, email, Teams

Description automatically generated

**Q: How would you remove duplicate rows?**

A: You could use the DISTINCT function

Graphical user interface, text, application, email

Description automatically generated

<https://learnsql.com/cookbook/how-to-eliminate-duplicate-rows-in-sql/#:~:text=Simply%20use%20the%20DISTINCT%20keyword,on%20the%20columns%20you%20listed>.