NYU

Data Science Bootcamp Fall'25

Week 4: SQL for DS Interviews

Intro

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- The data in RDBMS is stored in database objects called **tables**.
- A table is a collection of related data entries and it consists of columns and rows.
- A record, also called a row, is each individual entry that exists in a table.
- A column is a vertical entity in a table that contains all information associated with a specific field in a table.

Tables

SALES
Date
Order_id
Item_id
Customer_id
Quantity
Revenue

ITEMS
Item_id
Item_name
Price
department

CUSTOMERS
Customer_id
First_name
Last_name
Address

NYU

SELECT

Syntax

SELECT Columns

FROM Table_name;

We can use wildcard character ***** to select all columns.

We can also use Limit at the end of the query to limit the number of records fetched.

Example:

SELECT *

FROM Sales

LIMIT 10;

WHERE -filtering

Syntax

SELECT Columns

FROM Table_name

WHERE

Condition

We can use Where clause to filter the rows that are fetched by our query. Some of the operators that we can use are – LIKE, =, >,<, IS NULL, IN, BETWEEN, etc.

Eg: Pull sample of 20 sales from 05 January 2023

```
SELECT *

FROM Sales

WHERE Date =”01-05-2023”

LIMIT 20;
```

NYU **Sales –** Date, Order_id, Item_id, Customer_id, Quantity, Revenue

AND, OR, NOT operator

The `WHERE` clause can contain one or many `AND` operators.

The `AND` operator is used to filter records based on more than one condition.

Eg.Pull up customers from Germany and Berlin

```
SELECT * FROM Customers
WHERE Country='Germany' AND
City='Berlin';
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

NYU

Eg. Select only the customers that are NOT from Spain:

```
SELECT * FROM Customers
WHERE NOT Country = 'Spain';
```

Eg. Select all customers that either: are from Spain and starts with either "G", or starts with the letter "R":

```
SELECT * FROM Customers
WHERE Country = 'Spain' AND
CustomerName LIKE 'G%' OR
CustomerName LIKE 'R%';
```

NYU

Aggregate Functions

- COUNT(column)
  - count all non null values in the column count(*) will count all rows in the table
- COUNT(DISTINCT column)
  - count all distinct values in the column
- SUM(column) and AVG(column)
  - calculates the sum and average of a column
- MIN(column) and MAX(column)
  - computes the max and min value in a column

SELECT columns, **aggregate_fun** (column) **FROM** table **WHERE** condition **GROUP BY** columns

Note*-We **must** group by all non aggregate columns

Eg: For each day in January 2023 how much revenue did we generate and how many sales did we have?

```
SELECT Date, SUM(Revenue) as Rev
COUNT( DISTINCT Order_id) as Cnt
FROM Sales
WHERE Date BETWEEN “01-01-2023”
AND “01-31-2023”
GROUP BY Date;
```

NYU

SALES – Date, Order_id, Item_id, Customer_id, Quantity, Revenue

ORDER BY

Syntax:

```
SELECT columns
                aggregate_fun (column)
FROM table WHERE
condition GROUP BY
columns
ORDER BY columns ASC/DESC
```

Eg: How many items do we have in each department. Sort the departments in descending order

```
SELECT department, COUNT(item_id) AS
num_items

FROM Items

GROUP BY department

ORDER BY num_items DESC,
department DESC;
```

Items – item_id, item_name, price, department

HAVING - WHERE condition for aggregates

Syntax

```
SELECT columns
        aggregate_fun (column)
FROM table
WHERE condition
GROUP BY
columns
HAVING condition
```

Eg: Pull any order that cost at least $1000 sorted by order revenue descending.

```
SELECT Order_id, SUM(Revenue) as Rev
FROM Sales
GROUP BY order_id
HAVING Rev >=1000
ORDER BY Rev DESC;
```

NYU

SALES – Date, Order_id, Item_id, Customer_id, Quantity, Revenue

SQL Column Functions

- **CAST(column AS dtype)**

Changes a column's datatype (int64, string, float64 are the most common dtypes

- **UPPER() and LOWER()**

Adjusts the case of a string field for easier string matching

- **LIKE '%string%'**

To match on 'string' with % acting as a wildcard (this is actually a conditional, not a function)

Eg: What was the average order value in 2022

SELECT SUM(Revenue)/**SUM(DISTINCT** order_id) AS Avg_order_val

FROM Sales

WHERE CAST (Date **AS** string) **LIKE** '%-%-2022' ;

SALES – Date, Order_id, Item_id, Customer_id, Quantity, Revenue

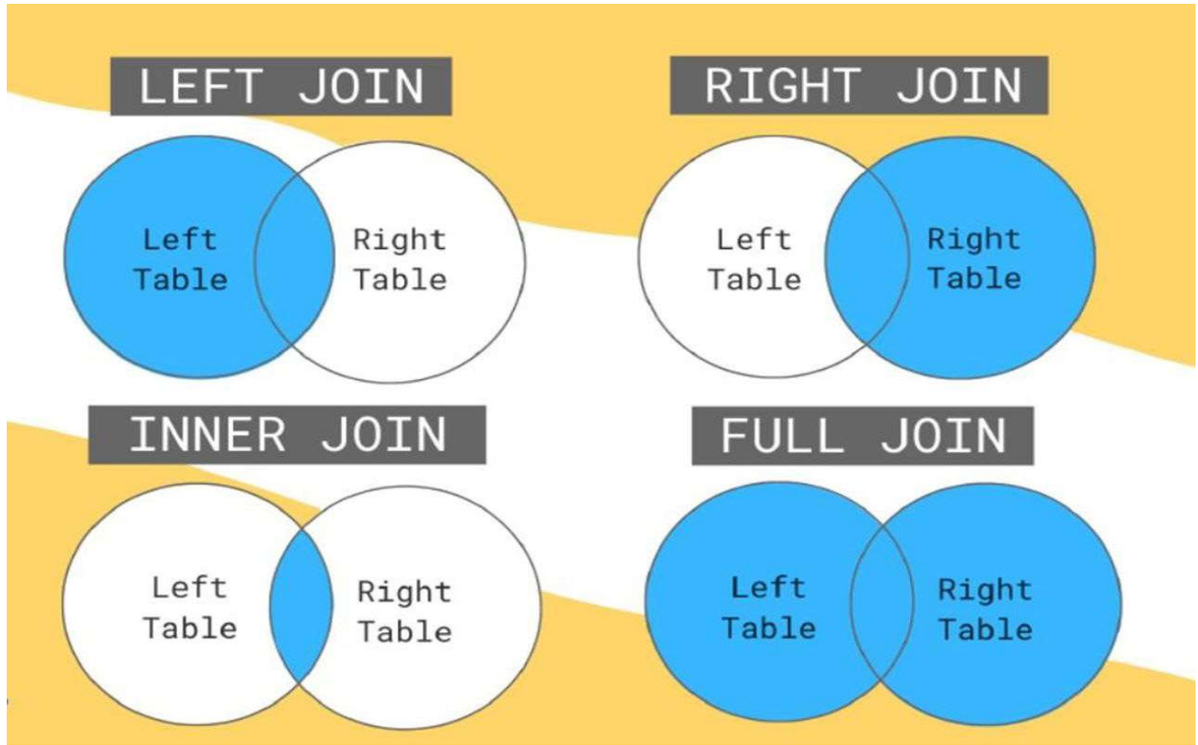
SQL Joins

Joins in SQL are used to combine rows from two or more tables based on related columns between them.

LEFT JOIN (or LEFT OUTER JOIN): Returns all rows from the left table and matching rows from the right table. If there are no matching rows in the right table, NULL values are included for the columns from the right table.

INNER JOIN: Returns rows from both tables where there is a match between the columns in both tables. If there is no match, the rows are not included in the result set.

FULL JOIN (or FULL OUTER JOIN): Returns all rows from both tables, matching rows from both tables where available. If there is no match, NULL values are included for the columns from the table where no match was found.



Syntax:

```
SELECT columns FROM
table1 as A JOIN table2
as B ON A.column =
B.column;
```

Eg: How much revenue has every item we sell generated?

```
SELECT i.item_id,
SUM(s.revenue) as rev,
FROM Items as i LEFT
JOIN Sales AS s ON
i.item_id=s.item_id
GROUP BY item_id;
```

SALES – Date, Order_id, Item_id, Customer_id, Quantity, Revenue

ITEMS – Item_id, Item_name, price, department

NYU

Subquery

- Subqueries, also known as nested queries or inner queries, are queries that are nested within another query.
- They allow you to use the result of one query as a part of another query.
- Subqueries can be used in various parts of a SQL statement, such as SELECT, FROM, WHERE, HAVING, and so on.

Syntax:

```
SELECT columns
FROM table
        WHERE column_val [<,>,IN,
        etc.] (SELECT ...)
```

NYU

Eg: Pull the sales that generated more revenue than order ‘2567’.

```
SELECT order_id,
        SUM(revenue) as rev
FROM sales
GROUP BY order_id
HAVING rev > (
            SELECT SUM(revenue)
            FROM sales WHERE
            order_id = ‘2567’);
```


SALES – Date, Order_id, Item_id, Customer_id, Quantity, Revenue

Tips:

- Pay attention to the **order of tables** when you are joining them [17]
- Remember to **group by** every column you aren't aggregating.
- When using a conditional for null values, you cannot use '=' and MUST use '**IS NULL**'.

- Use **DISTINCT** when values might be duplicated across multiple rows.

- Don't use distinct on a table's key- it isn't necessary to dedupe a key that is unique.

- Make sure you are talking about your code and thought process while you write it during coding interviews.