30 Must Know Data Analyst SQL Interview Questions and Answers

SQL is a powerful programming language that is widely used for managing and analyzing data. If you're looking to become a data analyst or an existing one who is looking for a new job, it's important to be prepared to answer SQL interview questions.

Here are some of the most common SQL interview questions that you may be asked during our data analyst SQL interview, along with advice on how to answer them.

1. What is SQL and what are its main features?

SQL, or Structured Query Language, is a standard programming language for accessing and manipulating databases. SQL is renowned for its simple yet powerful syntax, which makes it easy to query data in a variety of ways. SQL also supports a wide range of data types, including numeric, text, and date/time values.

2. What are some of the most common SQL commands?

Some of the most common SQL commands are CREATE TABLE, INSERT INTO, UPDATE, DELETE, and SELECT.

CREATE TABLE is used to create a new table in a database.

INSERT INTO is used to insert data into a table.

UPDATE is used to update data in a table.

DELETE is used to delete data from a table.

SELECT is used to select data from a table.

3. How can SQL be used to analyze data?

SQL provides a number of built-in functions that can be used to perform various types of data analysis. For example, the COUNT function can be used to counting the number of records in a table, while the SUM function can be used to calculate the sum of numeric values in a column. By using these and other SQL functions, data analysts can quickly and easily perform complex data analysis tasks.

For example, a data analyst might use SQL to count the number of orders placed on a website each day. The following SQL query would return the total number of orders for each day in the dataset:

SELECT COUNT (*) AS “Total Orders”

FROM orders

GROUP BY order_date

4. What are some common errors that occur when writing SQL queries?

One common error that occurs when writing SQL queries is forgetting to include a WHERE clause. Without a WHERE clause, your query will return all rows from the table you’re querying, which can make it difficult to find the specific information you’re looking for. Another common error is using incorrect syntax, which can lead to unexpected results or errors when your query is executed. Finally, it’s important to make sure that your SQL queries are properly formatted and easy to read; otherwise, they may be difficult for others to understand or debug if something goes wrong.

For example, the following SQL query would return all rows from the orders table, regardless of the order_date:

```
SELECT *

FROM orders
```

This would return a very large dataset that would be difficult to work with. To fix this, we can add a WHERE clause to filter the data by order_date:

```
SELECT *

FROM orders

WHERE order_date = '2018–01–01'
```

5. What's the difference between a primary and foreign key?

A primary key is a column (or set of columns) in a database table that uniquely identifies each row in the table. A foreign key is a column (or set of columns) in one table that contains values that match the primary key values in another table. Foreign keys are used to create relationships between tables; for example, a foreign key in a "customer" table could reference the primary key in an "orders" table, linking each customer with their respective orders.

6. What's a SQL join and how is it used?

A SQL join is used to combine data from two or more tables into a single result set. Joins are performed using the JOIN keyword, followed by the name of the table to join with. There are a number of different types of joins, including inner joins, outer joins, and self-joins. Inner joins return rows from both tables that have matching values in the specified columns, while outer joins return all rows from both tables, including rows with no matching values. Self-joins are used to join a table to itself; for example, you could use a self-join to find all customers who live in the same city as another customer.

7. What's a subquery and how is it used?

A subquery is a SQL query that is embedded within another SQL query. Subqueries are often used to find data that satisfies certain conditions; for example, you could use a subquery to find all customers who live in the same city as a particular customer. Subqueries can be used with various SQL commands, including SELECT, FROM, WHERE, and ORDER BY.

For example, consider the following customer table:

table: customers

id	name	city
1	John Smith	New York
2	Jane Doe	Boston
3	Joe Shmo	Chicago

Suppose we want to find all customers who live in the same city as customer with id=1. We could use the following SQL query:

```
SELECT * FROM customers WHERE city IN (SELECT city FROM customers WHERE id = 1)
```

```
SELECT * FROM customers WHERE city IN (SELECT city FROM customers WHERE id = 1 AND id != 1)
```

8. What are some of the most important SQL data types?

SQL supports a number of different data types, including numeric, text, date/time, and Boolean values. Numeric values include integers and floating-point numbers, while text values include character strings and date/time values

include date, time, and timestamp values. Boolean values can either be TRUE or FALSE.

9. What's an index and how is it used?

An index is a database structure that is used to improve the performance of SQL queries. Indexes can be created on columns in a table, and they are typically used to speed up searches for specific values in those columns. When a query is executed, the database engine will first check to see if an index exists for the columns that are being searched; if an index exists, the engine will use the index to quickly locate the desired data, which can improve query performance.

10. What's the difference between a view and a table?

A view is a virtual table that is based on the results of an SQL query. Views are often used to provide security or simplify complex queries. For example, you could create a view that only includes customer information that is relevant to your current project. Tables, on the other hand, are database structures that actually store data.

11. What's the difference between a WHERE and a HAVING clause?

The WHERE clause is used to filter rows from a table based on specified conditions; for example, you could use a WHERE clause to find all customers who live in a particular city. The HAVING clause is used to filter rows from a table based on aggregated values; for example, you could use a HAVING clause to find all customers who have placed more than 10 orders.

12. What does the ORDER BY keyword do?

The ORDER BY keyword is used to sort the results of an SQL query in ascending or descending order. By default, ORDER BY will sort the results in ascending order; to sort the results in descending order, you can use the DESC keyword.

13. What's a primary key?

A primary key is a column or set of columns that uniquely identify a row in a table. Primary keys must contain unique values, and they cannot be NULL.

14. What's a foreign key?

A foreign key is a column or set of columns that contains values that match the primary key values in another table. Foreign keys are used to create relationships between tables; for example, a foreign key in a "customer" table could reference the primary key in an "orders" table, linking each customer with their respective orders

15. How do window functions work?

Window functions are a type of SQL function that operates on a set of rows and returns a single value. Window functions are typically used to calculate aggregated values, such as sums or averages, over a specified window of rows. For example, you could use a window function to calculate the average order total for each customer.

16. What's the difference between an inner join and an outer join?

Inner joins return only rows that have matching values in both tables; for example, if your inner join a "customer" table with an "orders" table, only

customers who have placed orders will be returned. Outer joins, on the other hand, return all rows from both tables, including rows with no matching

17. What are some of the most common SQL functions?

Some of the most common SQL functions are SUM (), AVG (), COUNT (), MIN (), and MAX (). These functions are used to calculate aggregated values, such as sums, averages, or counts.

18. How have you used SQL to solve a problem?

This is a common SQL interview question that is designed to assess your real-world experience with the language. When answering this question, be sure to describe a specific problem that you were able to solve using SQL. This will help to show the interviewer that you have a good understanding of how SQL can be used in practice.

19. What's the difference between a lag and lead function in SQL?

Lag and lead functions are used to access data from a previous or future row in a table. Lag functions return data from a row that is preceding the current row, while lead functions return data from a row that is following the current row.

For example,

customer table:

customer_id	name
1	John Smith
2	Jane Doe
3	Joe Bloggs
4	Sarah Patel
5	Arnold Smith

If the current row is customer_id 3 (Joe Bloggs), a lag function would return customer_id 2 (Jane Doe), while a lead function would return customer_id 4 (Sarah Connor).

20. Write a SQL query to select the second-highest salary in the engineering department.

```
SELECT DISTINCT salary

FROM employee

WHERE department = 'engineering'

ORDER BY salary DESC LIMIT 1 OFFSET 1;
```

This SQL query will select the second-highest salary from the engineering department by first selecting all distinct salaries from employees in the engineering department, then ordering them in descending order, and finally selecting the top 2 salaries.

21. What is a correlated subquery?

A correlated subquery is a type of SQL query that contains a reference to a value from outer query. Correlated subqueries are typically used when you want to find rows from a table that match certain conditions, but you can only know those conditions after examining other rows in the same table.

For example, you could use a correlated subquery to find all employees who make more than the average salary in their department. In this case, you would need to calculate the average salary for each department before you could compare each employee's salary to it.

22. What's a SQL aggregate function and how is it used?

A SQL aggregate function is a function that performs a calculation on a set of values and returns a single value. Common aggregate functions include COUNT, SUM, MAX, and MIN. Aggregate functions are often used with the GROUP BY clause to return one result per group; for example, you could use the COUNT() function to find the number of customers in each city.

For example, consider the following customer table:

table: customers

id	name	city
1	John Smith	New York
2	Jane Doe	Boston
3	Joe Shmo	Chicago
4	John Smith	New York

Suppose we want to find the number of customers in each city. We could use the following SQL query:

SELECT city, COUNT (*) AS "Number of Customers"

FROM customers

GROUP BY city

This would return the following result:

city	Number of Customers
Boston	1
Chicago	1
New York	2

23. When would you not want to use a window function in a SQL?

Window functions are a type of SQL function that return a value for each row in the query result, based on values from other rows in the same result. For example, you could use a window function to calculate the running total of all order totals in your customer orders table.

Window functions are not typically used with aggregate functions, because the results would not make sense. For example, if you tried to find the average salary for each department using a window function, you would end up with the same average salary for every department, because the window function would calculate the average salary for each row in the result set (which would be all employees in all departments).

24. How can you find duplicate rows in a SQL table?

There are a few ways to find duplicate rows in a SQL table. One way is to use the GROUP BY clause to group together rows with the same values in the columns you're interested in. For example, suppose we have a customer orders table with the following data:

table: customer_orders

id	name	city
1	John Smith	New York
2	Jane Doe	Boston
3	Joe Shmo	Chicago
4	John Smith	New York

If we wanted to find all the duplicate rows, we could use the following SQL query:

```
SELECT name, city
FROM customer_orders
GROUP BY name, city
HAVING COUNT (*) > 1
```

This would return the following result:

name	city
John Smith	New York

25. How do you optimize a SQL query?

There are a few different ways to optimize a SQL query. One way is to make sure that the columns you're interested in are indexed, so the database can more quickly find the data you're looking for. Another way is to use the EXPLAIN command to see how the database will execute your query, and then make changes to your query based on that information. Finally, you can use query hints to give database-specific instructions on how to execute your query.

Another way you can optimize a SQL query is to use a tool like SQL Profiler to see where the bottlenecks are in your query and then make changes accordingly.

26. How do you find the top 5 customers by sales?

There are a few different ways to find the top 5% of customers by sales. One way is to use the GROUP BY clause to group together rows with the same values in the columns you're interested in. For example, suppose we have a customer orders table with the following data:

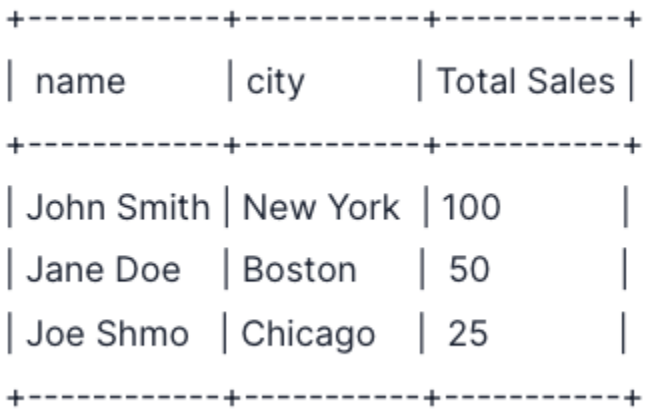
table: customer_orders

id	name	city
1	John Smith	New York
2	Jane Doe	Boston
3	Joe Shmo	Chicago
4	John Smith	New York

If we wanted to find the top 5 customers by sales, we could use the following SQL query:

```
SELECT name, city, SUM(sales) AS “Total Sales”

FROM customer_orders

GROUP BY name, city

ORDER BY “Total Sales” DESC

LIMIT 5
```

This would return the following result:



name	city	Total Sales
John Smith	New York	100
Jane Doe	Boston	50
Joe Shmo	Chicago	25

27. What is a relational database?

A relational database is a database that stores data in tables. Tables are similar to folders in a file system, where each table stores data about a particular subject. For example, a customer orders table might store data about customer orders, and a product table might store data about products.

Relational databases are the most common type of database, and they are used by most businesses because they are easy to use and easy to scale.

28. What types of relationships are there with a database?

There are three types of relationships in a database: one-to-one, one-to-many, and many-to-many.

A one-to-one relationship is when each row in one table is related to only one row in another table. For example, a customer table might have a one-to-one relationship with an orders table, where each customer is related to only one order.

A one-to-many relationship is when each row in one table is related to multiple rows in another table. For example, a customer's table might have a one-to-many relationship with an orders table, where each customer is related to multiple orders.

A many-to-many relationship is when each row in one table is related to multiple rows in another table, and each row in the other table is related to multiple rows in the first table. For example, a customer's table might have a many-to-many relationship with a products table, where each customer is related to multiple products and each product is related to multiple customers.

29. What DDL and DML?

DDL (Data Definition Language) is a language used to create and modify database structures like tables, views, and indexes.

30. Write a query to get the total two-day rolling average for sales by day.

For example, suppose we have a sales table with the following data:

table: sales

id	date	sales
1	2017-01-01	100
2	2017-01-02	200
3	2017-01-03	300
4	2017-01-04	400
5	2017-01-05	500

If we wanted to get the total two-day rolling average for sales by day, we could use the following SQL query:

SELECT date, sales, AVG(sales) OVER (ORDER BY date ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS “Two-Day Rolling Average”

FROM sales

GROUP BY date

ORDER BY date

This would return the following result:

date	sales	Two-Day Rolling Average
2019-01-01	100	62.
2019-01-02	50	37
2019-01-03	25	56
2019-01-04	75	50