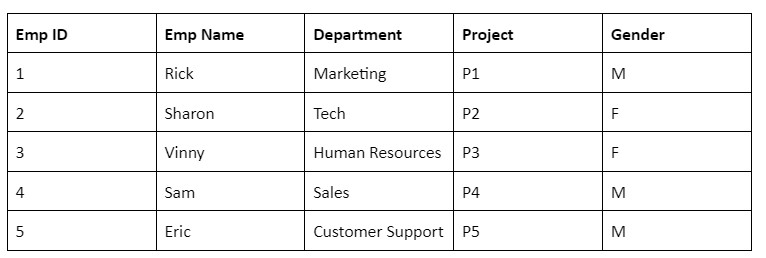
**SQL**

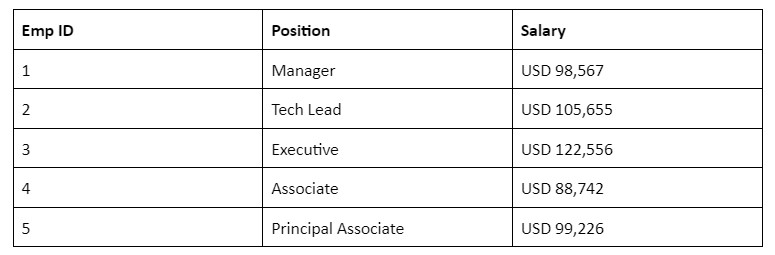
**Questions:**

1. Explain primary keys, unique keys, and foreign keys in SQL
2. What are the subsets of SQL?
3. What do you understand by DDL?
4. Explain the main features of a relational database
5. What are network databases?
6. What do you understand by normalization in a database?
7. What are consistent and inconsistent dependencies? Elaborate with an example.
8. What are some SQL operators and their functions?
9. What are views and indexes in SQL?
10. What do you understand about the unique index in SQL?
11. Explain primary keys, unique keys, and foreign keys in SQL
12. What are the subsets of SQL?
13. What do you understand by DDL?
14. Explain the main features of a relational database
15. What are network databases?
16. What do you understand by normalization in a database?
17. What are consistent and inconsistent dependencies? Elaborate with an example.
18. What are some SQL operators and their functions?
19. What are views and indexes in SQL?
20. What do you understand about the unique index in SQL?

**Table 1:**



‍**Table 2:**



Following are the SQL interview questions to expect at data analyst interviews based on queries:

1. Write a query to fetch salary records from a table in descending order.
2. Write a query to fetch the record with the highest salary from the tables.
3. What query will you use to calculate odd and even records in a table?
4. What query would you use to identify which position draws the maximum salary from the tables?
5. Write a query to find which project associates with the position that pays the highest salary.
6. Write a query to determine which employee draws the highest salary from the table.
7. Write a query to identify whether male or female employees make more on average.

**1. When would you use the GROUP BY statement?**

The GROUP BY clause enables us to compute aggregate values alongside discrete data by collapsing the distinct rows of our database into summary rows that share a particular characteristic.

If you were to attempt to formulate a query selecting a number of rows with a WHERE clause, and then display an aggregate value alongside, you would find that the query would return an error. This is because SQL is not able to display, in a single table, the results of our WHERE query as a list of values conjoined with the aggregate value you are looking for.

**2. What are the most common aggregate functions in SQL? What do they do?**

An [aggregate function](https://www.interviewquery.com/learning-paths/sql/beginner-sql-questions/aggregate-functions) performs a calculation on a set of values and returns a single value summarizing the set. The three most common aggregate functions in SQL are: COUNT, SUM, and AVG.

* **COUNT** - Returns the number of items of a group.
* **SUM** - Returns the sum of ALL or DISTINCT values in an expression.
* **AVG** - Returns the average of values in a group (and ignores NULL values).

**3. What is a unique key in SQL?**

In SQL, a unique key is one or more columns or fields that identify a record in a database. Tables can have multiple unique keys, which is a difference between unique keys and primary keys. With unique keys, only one NULL value is accepted for the column and it cannot have duplicate values.

**4. What is the difference between UNION and UNION ALL?**

UNION and UNION ALL are SQL operators used to concatenate two or more result sets. This allows us to write multiple SELECT statements, retrieve the desired results, then combine them together into a final, unified set.

The main difference between UNION and UNION ALL is that:

* **UNION:** only keeps unique records.
* **UNION ALL:** keeps all records, including duplicates.

**5. What is the difference between a RIGHT JOIN and a LEFT JOIN?**

The two most common types of joins in SQL are LEFT and RIGHT. The main difference is that these JOIN operators deal with **matched and unmatched rows**.

A LEFT JOIN includes all records from the left table and **matched rows from the right**. A RIGHT JOIN returns all rows from the right table and **unmatched rows from the left**.

**6. What is the difference between a table and a view?**

Tables contain data and they are made up of columns and rows. A view is a virtual table, which generally is dependent on data from the table for its display.

One use case for a view can be found if you wanted to look at a subset of data from a table. You could create a view using the SELECT command to query the data.

**7. What SQL operator is used for pattern matching?**

The LIKE operator is used to search for a specific pattern within a column. It is used with a WHERE clause to query specific columns.

LIKE is used with two wildcard characters:

* % represents a 0, 1 or multiple characters
* \_ (underscore) represents a single character

Here are a few examples:

WHERE Employee Name LIKE '%r' - Finds matches that end with "r"

WHERE Employee NAME LIKE '%gh%' - Finds matches that include "gh" in any position `

WHERE Employee NAME LIKE '\_ch%' - Finds matches with "ch" in the second and third positions

WHERE Employee NAME LIKE 'g%r' - Finds matches that start with "g" and end with "r"

**8. What command would you use to update data in a table?**

You update an existing table with the UPDATE command in SQL. It is used with SET (which includes the updated information) and WHERE to select the specific instance.

**Example:** In the table ‘Employees’, you want to change the emergency contact, ContactName, for an employee with EmployeeID 3.

UPDATE Employees

SET ContactName = "Bob Smith"

WHERE EmployeeID = 3;

**9. Which operator is used to select values within a range? What types of values can be selected?**

The BETWEEN operator is used to select values within a range. You can use numbers, texts or dates with BETWEEN.

One important thing to note: the BETWEEN operator includes both the start and end dates.

SELECT EmployeeID

FROM Employees

WHERE EmployeeID BETWEEN 378 AND 492

[**10. Given the tables users and rides, write a query to report the distance traveled by each user in descending order.**](https://www.interviewquery.com/questions/distance-traveled)

For this question, you need to accomplish two things: the first is to figure out the total distance traveled for each user\_id, and the second is to order from greatest to least eachuser\_id by a calculated distance traveled.

This question has been asked in [Uber data analyst interviews](https://www.interviewquery.com/interview-guides/uber-data-analyst).

[**11. Write a query to find all the users that are currently “Excited” and have never been “Bored” within a campaign.**](https://www.interviewquery.com/questions/always-excited-users)

For this medium SQL problem, assume you work at an advertising firm. You have a table of users’ impressions of ad campaigns over time. Each user\_id from these campaigns has an attached impression\_id, categorized as either “Excited” or “Bored”. You will need to assess which users are “Excited” by their most recent campaign, and have never been “Bored” in any past campaign.

**Note:** This is the type of question that might get asked for a [marketing analyst job](https://www.interviewquery.com/p/marketing-analyst-role).

[**12. Write a SQL query to select the second-highest salary in the engineering department.**](https://www.interviewquery.com/questions/2nd-highest-salary)

To answer this question, you need the name of the department to be associated with each employee in the employees table to understand which department each employee is a part of.

The “department\_id” field in the employees table is associated with the “id” field in the departments table. You can call the “department\_id” a foreign key because it is a column that references the primary key of another table, which in this case is the “id” field in the departments table.

Based on this shared field, you can join both tables using INNER JOIN to associate the department name with their employees.

SELECT salary

FROM employees

INNER JOIN departments

ON employees.department\_id = departments.id

With the department name in place you can now look at the employees of the Engineering team and sort by salary to find the second highest paid.

[**13. Given a table of bank transactions, write a query to get the last transaction for each day.**](https://www.interviewquery.com/questions/last-transaction)

**More Context:** The table includes the columns: id, transaction\_value and created\_at (representing the time for each transaction).

Since our goal in this problem is to pull the last transaction from each day, you want to group the transactions by the day they occurred and create a chronological order within each day from which you can retrieve the latest transaction.

To accomplish the task of grouping and order, create a modified version of the bank\_transactions table with an added column denoting the chronological ordering of transactions within each day.

To partition by date, you can use an OVER() statement. After partitioning, you should use a descending order so that the first entry in each partition is the last transaction chronologically. Here is how that query can be written:

SELECT (\*), ROW\_NUMBER() OVER(PARTITION BY DATE(created\_at)

ORDER BY created\_at DESC)

AS ordered\_time

[**14. Write a query to debug an error and select the top five most expensive projects by budget-to-employee ratio.**](https://www.interviewquery.com/questions/project-budget-error)

**More context:** You are given two tables. A projects table and another that maps employees to their projects, called employee\_projects. In this question, however, a bug exists that is causing duplicate rows in the employee\_projects table.

**Example:**

**Input:**

projects table

| **column** | **type** |
| --- | --- |
| id | INTEGER |
| title | VARCHAR |
| state\_date | DATETIME |
| end\_date | DATETIME |
| budget | INTEGER |

employee\_projects table

| **Column** | **Type** |
| --- | --- |
| project\_id | INTEGER |
| employee\_id | INTEGER |

**Output:**

| **Column** | **Type** |
| --- | --- |
| title | VARCHAR |
| budget\_per\_employee | FLOAT |

This is a good example of a logic-based SQL problem. Although there are a few steps to the solution, the actual SQL queries are fairly simple. *HINT: One way to do the debugging is to group by columns project\_id simply and employee\_id. By grouping by both columns, you are creating a table that provides distinct values on project\_id and employee\_id, thereby excluding any duplicates.*

[**15. You have a table that represents the total number of messages sent between two users by date on Facebook Messenger. Answer these questions:**](https://www.interviewquery.com/questions/conversations-distribution)

* What are some insights that could be derived from this table?
* What do you think the distribution of the number of conversations created by each user per day looks like?
* Write a query to get the distribution of the number of conversations created by each user by day in 2020.

This question tests your data sense, as well as your SQL writing skills. It has also appeared in [Facebook data analyst interviews](https://www.interviewquery.com/interview-guides/facebook-data-analyst).

To answer the first part of the question regarding insights, there are a number of metrics you could evaluate. You can find the total number of messages sent per day, the number of conversations being started, or the average number of messages per conversation. All of these metrics seek to find users’ level of engagement and connectivity.

[**16. Write a SQL query to create a histogram of the number of comments per user in the month of January 2020.**](https://www.interviewquery.com/questions/comments-histogram)

This intermediate SQL question has been asked in [Amazon data analyst interviews](https://www.interviewquery.com/interview-guides/amazon-data-analyst). Here is a partial answer from Interview Query:

What does a histogram represent, and what kind of story does it tell? In this case, you are interested in using a histogram to represent the distribution of comments each user has made in January 2020. A histogram with bin buckets of size one means that you can avoid the logical overhead of grouping frequencies into specific intervals.

For example, if you want a histogram of size five, you would have to run a SELECT statement like so:

SELECT

CASE WHEN frequency BETWEEN 0 AND 5 THEN 5

WHEN frequency BETWEEN 5 AND 10 THEN 10 etc..

[**18. Given a table of students and their SAT test scores, write a query to return the two students with the closest test scores by score difference.**](https://www.interviewquery.com/questions/closest-sat-scores)

Given that this problem is referencing one table with only two columns, you have to [self-reference](https://www.interviewquery.com/learning-paths/sql/medium-sql-questions/self-joins-and-cross-joins) different creations of the same table. It is helpful to think about this problem in the form of two different tables with the same values.

There are two parts to this question:

* The first part compares each combination of students and their SAT scores.
* The second part is figuring out which two students’ scores are then the closest.

[**19. Write a query to support or disprove the hypothesis: Clickthrough Rate (CTR) is dependent on search rating.**](https://www.interviewquery.com/questions/search-ctr)

This question provides a table that represents search results on Facebook, including a query, a position, and a human rating.

[**20. Write a query to get the number of customers that were upsold by purchasing additional products.**](https://www.interviewquery.com/questions/upsell-transactions)

For this problem, you are given a table of product purchases. Each row in the table represents an individual product purchase.

**Note:** If the customer purchased two things on the same day, that does not count as an upsell, as they were purchased within a similar timeframe. We are looking for a customer returning on a different date to purchase a product.

This question is a little tricky because you have to note the dates that each user purchased products. You can’t just group by the user\_id and look where the number of products purchased is greater than one because of the upsell condition.

You have to group by both the date field and the user\_id to get each transaction broken out by day and user:

SELECT

user\_id

, DATE(created\_at) AS date

FROM transactions

GROUP BY 1,2

The query above will now give us a user\_id and date field for each row. If there exists a duplicate user\_id then you know that the user purchased on multiple days, which satisfies the upsell condition. What comes next?

[**21. Given the transactions table below, write a query that finds the third purchase of every user.**](https://www.interviewquery.com/questions/third-purchase)

**Note:** Sort the results by the user\_id in ascending order. If a user purchases two products at the same time, the lower id field is used to determine which is the first purchase.

**Example:**

**Input:**

transactions table

| **Columns** | **Type** |
| --- | --- |
| id | INTEGER |
| user\_id | INTEGER |
| created\_at | DATETIME |
| product\_id | INTEGER |
| quantity | INTEGER |

**Output:**

| **Columns** | **Type** |
| --- | --- |
| user\_id | INTEGER |
| created\_at | DATETIME |
| product\_id | INTEGER |
| quantity | INTEGER |

Here is a helpful hint for this question: You need an indicator of which purchase was the third by a specific user. Whenever you are thinking of ranking a dataset, it is helpful to then immediately think of a specific window function you can use. You need to apply the RANK function to the transactions table. The RANK function is a window function that assigns a rank to each row in the partition of the result set.

SELECT \*, RANK() OVER (

PARTITION BY user\_id ORDER BY created\_at ASC

) AS rank\_value

FROM transactions

LIMIT 100

**[23. Write a query to get the top three highest employee salaries for each department.](https://www.interviewquery.com/questions/top-three-salaries)**

For this problem, you are given an employees and a departments table.

**Note:** If the department contains less than three employees, the top two or top one highest salaries should be listed.

Here’s a hint: You need to order the salaries by department. A window function is useful here. Window functions enable calculations within a certain partition of rows. In this case, the RANK() function would be useful. What would you put in the PARTITION BY and ORDER BY clauses?

Your window function can look something like this:

RANK() OVER (PARTITION BY id ORDER BY metric DESC) AS ranks

**Note:** When you substitute for the actual id and metric fields, make sure the substitutes are relevant to the question asked and aligned with the data provided to you.