🎯 Question: Explain the [hashtag#PIVOT](https://www.linkedin.com/feed/hashtag/?keywords=pivot&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A7235595571728642048) and [hashtag#UNPIVOT](https://www.linkedin.com/feed/hashtag/?keywords=unpivot&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A7235595571728642048) operations.   
  
📚 Explanation:  
  
PIVOT and UNPIVOT are powerful SQL operations that allow you to transform data, making it easier to analyze and report.  
  
### 🌟 PIVOT:  
The PIVOT operation rotates rows into columns, effectively transforming your data into a more readable and analyzable format.  
  
🔧 Example:  
Consider a sales table:  
  
| Year | Product | Sales |  
|------|---------|-------|  
| 2023 | A | 100 |  
| 2023 | B | 150 |  
| 2024 | A | 200 |  
| 2024 | B | 250 |  
  
PIVOT the data to compare sales across products:  
  
| Year | A | B |  
|------|-----|-----|  
| 2023 | 100 | 150 |  
| 2024 | 200 | 250 |  
  
SQL Syntax:  
  
SELECT Year, [A], [B]  
FROM  
 (SELECT Year, Product, Sales  
 FROM Sales) AS SourceTable  
PIVOT  
(  
 SUM(Sales)  
 FOR Product IN ([A], [B])  
) AS PivotTable;  
  
  
### 🌟 UNPIVOT:  
The UNPIVOT operation does the opposite—it converts columns back into rows, which can be useful when you need to normalize your data.  
  
🔧 Example:  
Using the pivoted data above:  
  
| Year | A | B |  
|------|-----|-----|  
| 2023 | 100 | 150 |  
| 2024 | 200 | 250 |  
  
UNPIVOT to get back the original structure:  
  
| Year | Product | Sales |  
|------|---------|-------|  
| 2023 | A | 100 |  
| 2023 | B | 150 |  
| 2024 | A | 200 |  
| 2024 | B | 250 |  
  
SQL Syntax:  
  
SELECT Year, Product, Sales  
FROM  
 (SELECT Year, [A], [B]  
 FROM PivotTable) AS PivotedTable  
UNPIVOT  
(  
 Sales FOR Product IN ([A], [B])  
) AS UnpivotTable;  
  
  
### 📈 Why It Matters:  
- PIVOT is ideal for generating cross-tab reports, making it easier to compare metrics across different categories.  
- UNPIVOT is essential for data normalization, which is critical for maintaining data integrity and preparing data for further analysis.  
  
💼 Pro Tip: Mastering PIVOT and UNPIVOT can significantly enhance your data transformation and reporting capabilities in SQL, making you more efficient in handling complex data scenarios.

**40 MOST ASKED SQL Interview Questions**

**INTERVIEW QUESTIONS**

1. What is the difference between **DELETE** and **TRUNCATE** command?
2. What is the difference between **DROP** and **TRUNCATE** command?
3. What are the different **SUBSETS OF SQL**?
4. What do you mean by **DBMS**? What are the different types?
5. What do you mean by **Table and Field**?
6. What are **JOINTS** in SQL?
7. What is the difference between **CHAR** and **VARCHAR**?
8. What is **Primary Key**?
9. What are **Constraints**?
10. What is the Difference Between **SQL** and **MySQL**?
11. What is a **UNIQUE KEY**?
12. What is **FOREIGN KEY?**
13. What is **DATA integrity**?
14. What are **CLUSTERED INDEX** and **NON-CLUSTERED INDEX**?
15. Write an **SQL Query** to display the Current Date.
16. Explain the Different Types of **JOINTS**.
17. What is **De-Normalization**?
18. What are **ENTITIES** and **RELATIONSHIPS**?
19. What is an **INDEX**?
20. Explain **types of INDEX**
21. What is **Normalization**? What are its **Advantages**?
22. Explain different types of **NORMALIZATION**
23. What is **ACID** property in a database?
24. What do you mean by **"Trigger" in SQL**?
25. What are the different types of **operators in SQL**?
26. Are **NULL** values the same as **ZERO**?
27. What is the difference between **CROSS JOIN** and **NATURAL JOIN**?
28. What is a **SUBQUERY in SQL**?
29. What are the different types of **SUBQUERY**?
30. Write an SQL Query to **count the records of a table.**
31. Write an SQL query to find the names of **employees that begin with 'A'.**
32. Write a Query to get the **third highest salary** of an employee from the employee table.
33. What is the need for **Group Function in SQL**?
34. What are **relationships and types of relationships**?
35. How can you insert **NULL VALUES** in columns while inserting Data?
36. What is the main difference between **BETWEEN and IN** operators?
37. Why are **SQL functions used**?
38. What is the need for **MERGED STATEMENT**?
39. What do you mean by **recursive stored procedures**?
40. What is **CLAUSE** in SQL?

SQL Interview Questions asked in Sapient and other MNC's!  
  
Interviewer: Why do we need SQL and NoSQL?  
  
Me: SQL is used for structured data with fixed schemas, ensuring ACID properties, making it ideal for transactions. NoSQL handles unstructured or semi-structured data, providing scalability and flexibility, suitable for big data and real-time applications.  
  
Interviewer: Define CTE and subquery and how they affect time complexity.  
  
Me: A Common Table Expression (CTE) is a temporary result set for complex queries, improving readability. A subquery is a query within another query, used for filtering. CTEs can be more efficient as they avoid repeated calculations, while subqueries might increase time complexity due to nested executions.  
  
  
Interviewer: Exchange seats of odd students with even students, and if the last student is odd, don't change his seat.  
Me:  
WITH Students AS (  
 SELECT id, ROW\_NUMBER() OVER (ORDER BY id) AS row\_num  
 FROM StudentTable  
)  
UPDATE Students  
SET id = CASE  
 WHEN row\_num % 2 = 1 THEN LEAD(id) OVER (ORDER BY row\_num)  
 WHEN row\_num % 2 = 0 THEN LAG(id) OVER (ORDER BY row\_num)  
END  
WHERE row\_num < (SELECT MAX(row\_num) FROM Students) OR row\_num % 2 = 0;  
  
Interviewer: Find the last person to fit in a bus based on weight. Bus capacity is 1000.  
Me:   
SELECT person\_id  
FROM (SELECT person\_id, weight, SUM(weight) OVER (ORDER BY person\_id) AS running\_total  
 FROM Passengers) AS sub  
WHERE running\_total <= 1000  
ORDER BY person\_id DESC  
LIMIT 1;  
  
Interviewer: Find the second highest salary using a CTE.  
Me:   
WITH SalaryRank AS (  
 SELECT salary, DENSE\_RANK() OVER (ORDER BY salary DESC) AS rank  
 FROM Employees  
)  
SELECT salary  
FROM SalaryRank WHERE rank = 2;

Data Engineering Interview Questions !!  
  
  
1) Explain about the types of Integration Runtime in ADF and use case of each?  
2) Explain the use of different triggers in ADF  
3) What are the optimization techniques you have used in your project. Implementation with use-cases.  
4) How do you unit test your code in databricks   
5) Why can't we always use broadcast in Spark   
6) Where to use Coalesce and where to use Repartition   
7) Explain how you handle data skew ? Salting in detail.  
8) How do you handle Merge-conflicts in Git   
9) How to schedule you pipeline in data bricks   
10) Difference b/w Parquet table, Delta table and delta live tables  
11) Explain spark architecture in detail  
12) What are the benefits of delta table ? Cross-questions on this and data formats.  
13) Difference b/w CDC and SCDs, when to use which  
14) Two sum problem   
15) SQL Questions - Basics questions on window functions(Lead/Lag and rolling

[Fractal](https://www.linkedin.com/company/fractal-analytics/) SQL Interview Questions for Data Scientist 2024.  
  
1. Explain the difference between INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN in SQL. When would you use each?  
2. What are window functions in SQL? Provide examples of their usage and discuss scenarios where they are particularly useful.  
3. How does SQL handle NULL values in different operations? Discuss the implications of NULLs in JOINs and aggregations.  
4. Describe the process of indexing in SQL databases. How do indexes improve query performance, and what are the trade-offs?  
5. What are common techniques to optimize SQL queries? Discuss how you would improve the performance of a slow-running query.  
6. Explain the concept of normalization in SQL databases. What are the different normal forms, and how do they impact database design?  
7. What is a Common Table Expression (CTE) in SQL, and how does it differ from subqueries? Provide examples of when to use CTEs.  
8. Discuss the differences between SQL and NoSQL databases. When would you choose SQL over NoSQL and vice versa?  
9. Explain the ACID properties in SQL databases. Why are they important for transactional consistency?  
10. What are stored procedures in SQL, and how can they be used to encapsulate business logic? Discuss the advantages and potential drawbacks of using stored procedures.

Scenario Based SQL interview questions:  
  
1. Finding the Second Highest Salary:  
Scenario: You have an employees table with columns "employee\_id", "name", and "salary".   
Q. Write a query to find the second highest salary in the company.  
A. SELECT MAX(salary)   
FROM employees   
WHERE salary < (SELECT MAX(salary) FROM employees);  
  
2. Employee Hierarchy:  
Scenario: You have an employees table with columns "employee\_id", "name", and "manager\_id" (which is a self-reference to the employee\_id of the employee's manager).   
Q. Write a query to find the names of all employees and their respective managers.  
A. SELECT [e.name](http://e.name/) as employee\_name, [m.name](http://m.name/) as manager\_name  
FROM employees e  
LEFT JOIN employees m ON e.manager\_id = m.employee\_id;  
  
3. Finding Duplicate Records:  
Scenario: You have a users table with columns "user\_id", "email", and "created\_at". Some users have registered multiple times with the same email. Q. Write a query to find all duplicate email addresses.  
A. SELECT email, COUNT(\*) as num\_occurrences  
FROM users  
GROUP BY email  
HAVING COUNT(\*) > 1;  
  
4. Inactive Users:  
Scenario: You have a users table with columns "user\_id", "last\_login", and "status".   
Q. Write a query to find all users who have not logged in for more than 6 months and are still marked as "active".  
A. SELECT user\_id, last\_login  
FROM users  
WHERE status = 'active'   
AND last\_login < DATE\_SUB(CURDATE(), INTERVAL 6 MONTH);  
  
5. Customer Purchase Behavior:  
Scenario: You have two tables, customers (with columns "customer\_id", "name", and "registration\_date") and orders (with columns "order\_id", "customer\_id", "order\_date", "order\_amount").   
Q. Write a query to find customers who made their first purchase within 30 days of registration.  
A. SELECT c.customer\_id, [c.name](http://c.name/), MIN(o.order\_date) as first\_order\_date  
FROM customers c  
JOIN orders o ON c.customer\_id = o.customer\_id  
GROUP BY c.customer\_id, [c.name](http://c.name/)  
HAVING DATEDIFF(MIN(o.order\_date), c.registration\_date) <= 30;  
  
6. Product Sales Ranking:  
Scenario: You have a products table with columns "product\_id", "product\_name", and "price", and a sales table with columns "sale\_id", "product\_id", and "quantity".   
Q. Write a query to rank products by total sales revenue.  
A. SELECT p.product\_name, SUM(s.quantity \* p.price) as total\_revenue,  
RANK() OVER (ORDER BY SUM(s.quantity \* p.price) DESC) as sales\_rank  
FROM products p  
JOIN sales s ON p.product\_id = s.product\_id  
GROUP BY p.product\_name;  
  
7. Customer Retention Analysis:  
Scenario: You have a customers table and an orders table.   
Q: Write a query to find customers who placed an order in their first month of registration but have not placed any orders in the last 6 months.  
A: SELECT c.customer\_id, [c.name](http://c.name/)  
FROM customers c  
JOIN orders o ON c.customer\_id = o.customer\_id  
GROUP BY c.customer\_id, [c.name](http://c.name/)  
HAVING MIN(o.order\_date) <= DATE\_ADD(c.registration\_date, INTERVAL 1 MONTH)  
AND MAX(o.order\_date) < DATE\_SUB(CURDATE(), INTERVAL 6 MONTH);

Spark submit config

Can you explain --num-executors, --executor-cores, and --executor-memory configurations in spark-submit?  
  
1. --num-executors :  
Specifies the number of executor instances to launch. The number of executors determines the level of parallelism in your Spark job. More executors generally mean more tasks can run in parallel, potentially reducing the overall runtime.  
  
If you have a cluster with 10 nodes and each node can handle 4 executors, you might set --num-executors to 40. This would allow you to maximize the usage of the cluster resources.  
  
2. --executor-cores :   
Determines the number of CPU cores allocated to each executor. Each executor can run multiple tasks in parallel, depending on the number of cores assigned to it. For instance, if --executor-cores is set to 4, each executor can run 4 tasks concurrently.  
  
If you have 8 cores per node and decide to allocate 2 cores per executor (--executor-cores 2), each node could run 4 executors concurrently. This setting balances the task parallelism and the number of executors.  
  
3. --executor-memory :  
This configuration determines how much memory is available for each executor to store data and perform computations. More memory per executor can reduce the need for disk I/O (spillover), which slows down performance.  
  
Together, these three configurations define how your application utilizes the cluster’s resources. For instance, if you have a total of 80 cores and 640GB of memory across your cluster, and you set --num-executors to 20, --executor-cores to 4, and --executor-memory to 16GB, then:  
You’ll use all 80 cores (20 executors \* 4 cores).  
You’ll utilize 320GB of memory (20 executors \* 16GB).  
  
This leaves you with unused memory, which might prompt you to adjust your configuration, perhaps increasing the number of executors or the memory per executor to better leverage available resources.  
  
Sample Spark Submit:  
  
spark-submit \  
 --master yarn \  
 --deploy-mode cluster \  
 --num-executors 20 \  
 --executor-cores 4 \  
 --executor-memory 16G \  
 --driver-memory 4G \  
 --conf spark.yarn.submit.waitAppCompletion=false \  
 [my\_pyspark\_script.py](http://my_pyspark_script.py/)

https://codeinspark.com/courses/spark-client-mode-cluster-mode/

Daily issues faced by Data Engineers:  
  
1. How do you handle job failures in an ETL pipeline?  
2. What steps do you take when a data pipeline is running slower than expected?  
3. How do you address data quality issues in a large dataset?  
4. What would you do if a scheduled job didn't trigger as expected?  
5. How do you troubleshoot memory-related issues in Spark jobs?  
6. What is your approach to handling schema changes in source systems?  
7. How do you manage data partitioning in large-scale data processing?  
8. What do you do if data ingestion from a third-party API fails?  
9. How do you resolve issues with data consistency between different data stores?  
10. How do you handle out-of-memory errors in a Hadoop job?  
11. What steps do you take when a data job exceeds its allocated time window?  
12. How do you manage and monitor data pipeline dependencies?  
13. What do you do if the output of a data transformation step is incorrect?  
14. How do you address issues with data duplication in a pipeline?  
15. How do you handle and log errors in a distributed data processing job?

Here are the top 10 most-asked PySpark coding questions in 2024:  
  
1. How do you remove duplicates from a PySpark DataFrame?  
2. Write a PySpark code to filter rows based on a condition using `filter()` or `where()`.  
3. How do you perform an inner join between two DataFrames in PySpark?  
4. Explain how to group data in a DataFrame using `groupBy()` and aggregate it using `agg()` or `sum()`.  
5. How do you add a new column to a PySpark DataFrame based on an existing column?  
6. Write PySpark code to calculate the average, min, and max values of a column.  
7. How do you repartition a PySpark DataFrame and explain when you would use `repartition()` vs `coalesce()`?  
8. Write PySpark code to apply a custom UDF to a DataFrame column.  
9. How do you cache or persist a DataFrame, and what’s the difference between them?  
10. Write PySpark code to handle null values in a DataFrame, either by dropping or filling them.

ere are some recent Data Engineering interview questions you might face in 2024:  
  
1. Difference Between ETL and ELT: Explain how these processes differ in terms of where and when data transformation occurs. Be prepared to discuss scenarios where one might be more beneficial than the other.  
  
2. Designing Scalable Data Pipelines: How would you design a data pipeline to ensure scalability? This question assesses your ability to think ahead and build systems that can handle future growth.  
  
3. SQL Query Optimization: What strategies would you use to optimize SQL queries for performance? This question tests your understanding of indexing, query plans, and other optimization techniques.  
  
4. Data Pipeline Reliability: What steps would you take to ensure the reliability of a data pipeline? Discuss monitoring, alerting, error handling, and recovery strategies.  
  
5. Handling Inconsistent Data from APIs: How would you manage a situation where your team receives inconsistent data from a third-party API? This question evaluates your problem-solving skills and ability to manage external dependencies.  
  
6. Data Security and Privacy: Describe the measures you would take to handle data security and privacy, including compliance with regulations like GDPR.  
  
[7.Star](http://7.star/) vs. Snowflake Schema: Explain the differences between star and snowflake schemas, including their use cases and advantages.  
  
8. Big Data Deployment: Describe the steps involved in deploying a big data solution, including data integration, storage, and processing frameworks like Spark and Hadoop.

[Amazon](https://www.linkedin.com/company/amazon/) Pyspark interview questions for 𝗗𝗮𝘁𝗮 𝗘𝗻𝗴𝗶𝗻𝗲𝗲𝗿 2024.  
  
1. How do you deploy PySpark applications in a production environment?  
2. What are some best practices for monitoring and logging PySpark jobs?  
3. How do you manage resources and scheduling in a PySpark application?  
4. Write a PySpark job to perform a specific data processing task (e.g., filtering data, aggregating results).  
5. You have a dataset containing user activity logs with missing values and inconsistent data types. Describe how you would clean and standardize this dataset using PySpark.  
6. Given a dataset with nested JSON structures, how would you flatten it into a tabular format using PySpark?  
8. Your PySpark job is running slower than expected due to data skew. Explain how you would identify and address this issue.  
9. You need to join two large datasets, but the join operation is causing out-of-memory errors. What strategies would you use to optimize this join?  
10. Describe how you would set up a real-time data pipeline using PySpark and Kafka to process streaming data.  
11. You are tasked with processing real-time sensor data to detect anomalies. Explain the steps you would take to implement this using PySpark.  
12. Describe how you would design and implement an ETL pipeline in PySpark to extract data from an RDBMS, transform it, and load it into a data warehouse.  
13. Given a requirement to process and transform data from multiple sources (e.g., CSV, JSON, and Parquet files), how would you handle this in a PySpark job?  
14. You need to integrate data from an external API into your PySpark pipeline. Explain how you would achieve this.  
15. Describe how you would use PySpark to join data from a Hive table and a Kafka stream.  
16. You need to integrate data from an external API into your PySpark pipeline. Explain how you would achieve this.