DATA ENGINEERING CONCEPTUAL INTERVIEW QUESTIONS

(Azure & ETL Based Questions)

**Scenario 1:**

You're tasked with designing a data warehouse for a large e-commerce company that sells a variety of products online. The company wants to analyze customer behavior, track sales performance, and optimize inventory management. **How would you approach this project, considering your three years of experience in data engineering?**

(How would you structure the data warehouse to meet the company's needs? What technologies and methodologies would you employ, and how would you ensure scalability and performance?)

**Answer:**

To design a data warehouse for a large e-commerce company using Azure Data Factory (ADF), I would start by gathering requirements to understand the needs for customer behavior analysis, sales performance tracking, and inventory management optimization. I would structure the data warehouse using a star schema with fact tables for sales and inventory, and dimension tables for customers, products, dates, and suppliers. Technologies like Azure Synapse Analytics for scalable data storage, Azure Data Lake Storage for raw data, and Azure Databricks for complex transformations would be employed. I would use ADF for orchestrating ETL workflows, ensuring data is ingested, transformed, and loaded efficiently. To ensure scalability and performance, I would implement partitioning and indexing in Synapse, use data archival strategies, and leverage auto-scaling features. Data governance and security would be maintained through Azure RBAC, encryption, and data lineage tracking. Monitoring and maintenance would involve Azure Monitor and ADF's built-in features to track pipeline performance and set up alerts for any anomalies. This approach ensures the data warehouse meets the company's analytical needs while being scalable, performant, and secure.

* 1. What kind of data validation checks can we perform?

**Answer:**

1. **Integrity Checks**: Ensuring data relationships are maintained (e.g., referential integrity).
2. **Format Checks**: Verifying data adheres to predefined formats (e.g., email addresses, phone numbers).
3. **Range Checks:** Validating data falls within acceptable ranges (e.g., dates, numerical values).
4. **Completeness Checks**: Ensuring all required fields are populated.
5. **Consistency Checks**: Verifying data consistency across different sources or tables.
6. **Uniqueness Checks**: Ensuring unique constraints are maintained (e.g., primary keys).

These checks help maintain data quality and reliability within the data warehouse.

* 1. Now give me the answer in terms of what technologies and methodologies would you employ?

**Answer:**

1. **SQL-based databases** (e.g., PostgreSQL, MySQL) for storing structured data efficiently.
2. **ETL (Extract, Transform, Load)** tools like Apache Spark or Talend for data integration and transformation.
3. **Data modeling tools** such as ERwin or PowerDesigner to design and manage the data warehouse schema.
4. Cloud platforms like Azure, AWS or GCP for scalability and flexibility.

As for methodologies, I'd follow Agile principles for iterative development and collaboration, ensuring alignment with business requirements and continuous improvement.

* 1. Tell me what is Data Modeling in simplest way and use a real-life example as well.

**Answer:**

Data modeling is like creating a blueprint for organizing information in a way that makes sense. It's about designing how data will be structured, stored, and accessed in a database.

*Think of it this way: imagine you're planning a road trip. Before you hit the road, you'd probably use a map to figure out your route, where to stop for gas, food, and rest. That map is like a data model—it helps you visualize and plan how to organize and navigate the trip efficiently.*

Similarly, in a data model for a business, you'd plan how different pieces of information (like customer details, product information, and sales data) are connected and stored in a database. This helps ensure that when you need to retrieve or analyze data, everything is structured in a logical and efficient way.

* 1. Now tell me what is a star schema in data modeling? Keep it simple, okay?

**Answer:**

Imagine a star in the sky with different points radiating out from its center. In a star schema, the center represents a central table called the fact table, which contains the core information you want to analyze, like sales transactions. The points around the center are the dimension tables, which provide additional details about the data in the fact table, like customers, products, and time. Just like how the points of a star connect back to its center, the dimension tables in a star schema connect to the fact table, allowing for easy analysis and retrieval of data.

* 1. So now tell me what is a Spark session relating to a real life scenario and why this is required?

**Answer:**

Imagine you're organizing a big party with lots of friends coming over. You have to set up games, prepare food, decorate the house, and more. But doing all this by yourself is too much work. So, you call your friends and ask them to help you.

Think of a Spark session like the party's main organizer. This organizer (the Spark session) is the one who:

1. **Invites Helpers:** Just like you invite friends to help you with the party, the Spark session gathers resources (like computers) to help process data.
2. **Gives Instructions:** The organizer gives instructions to everyone on what they should do, like setting up games or making sandwiches. Similarly, the Spark session gives tasks to the resources, like reading data or analyzing it.
3. **Keeps Everything Running Smoothly:** The organizer makes sure that everything is going smoothly and that everyone knows what they're doing. The Spark session manages all the jobs and makes sure the data processing is done efficiently.
   1. Can you give me a real world example for star schema?

**Answer:**

Imagine you're managing a retail store that sells products online and in physical stores. You want to analyze sales data to understand customer behavior and product performance.

**In a star schema:**

The fact table at the center would contain sales transactions, with details like sales amount, quantity sold, and dates of purchases.

The dimension tables around the fact table would include:

* Customer dimension: Details about customers such as their names, addresses, and demographics.
* Product dimension: Information about products like product names, categories, and prices.
* Time dimension: Data related to time, such as dates, months, and years.

Each dimension table is connected to the fact table, forming a star-like structure. This setup allows you to analyze sales data from different perspectives, such as by customer, product, or time, making it easier to gain insights into sales performance and customer behavior.

* 1. What do you mean by inferred schema in a spark? Tell me in a simplest way.

**Answer:**

**In Spark, an inferred schema is like a detective figuring out the structure of your data without you explicitly telling it.** Imagine you have a bunch of files with data, but you haven't described what each piece of data means. Spark will take a peek at the data and guess how it's structured—like deducing that the first column might be a name, the second might be an age, and so on. This guesswork is what we call an inferred schema. It's handy when you have lots of data and you don't want to manually specify every detail about its structure.

**SCENARIO 2**

You're working for a financial services company that handles large volumes of transactions daily. They want to build a data pipeline to analyze transaction patterns, detect fraud, and generate reports. How woul­­­d you design this data pipeline, considering your three years of experience in data engineering?

**Answer:**

To design this data pipeline, I would follow these steps:

1. **Data Ingestion:**

* Sources: Transaction data from multiple sources like databases, APIs, and log files.
* Tools: Use Apache Kafka for real-time data ingestion and batch processing tools like Apache Nifi for periodic data ingestion.

1. **Data Storage:**

* Data Lake: Store raw data in a data lake (e.g., AWS S3, Azure Data Lake) to handle large volumes of unstructured data.
* Data Warehouse: Use a relational database (e.g., Amazon Redshift, Google BigQuery) for structured, processed data to facilitate fast queries and analytics.

1. **Data Processing:**

* ETL/ELT Tools: Use Apache Spark or Apache Flink for large-scale data processing and transformations.
* Data Validation: Implement data validation checks like format, range, completeness, and consistency checks to ensure data quality.

1. **Data Modeling:**

* Schema Design: Create a star schema for the data warehouse with fact tables for transactions and dimension tables for customer details, account information, and time.

1. **Real-time Analytics and Fraud Detection:**

* Stream Processing: Use Spark Streaming or Apache Flink to process and analyze transaction data in real-time to detect anomalies and potential fraud.

1. **Reporting and Visualization:**

* BI Tools: Utilize business intelligence tools like Tableau or Power BI to create dashboards and reports for stakeholders to monitor transaction patterns and performance.

1. **Monitoring and Logging:**

* Tools: Implement monitoring tools like Prometheus and Grafana to track the health of the data pipeline and ensure smooth operation.

**QUESTION**:Can you define what is streaming data in the simplest form?

**Answer:**

Streaming data is like a continuous flow of information that keeps coming in real-time, similar to a river that keeps flowing without stopping. Instead of getting a snapshot of data all at once, like taking a picture, streaming data is more like watching a live video that keeps updating as new events happen. It's data that arrives continuously and needs to be processed and analyzed on the fly, without waiting for it to stop.

**SCENARIO 3:** You're tasked with building a scalable data pipeline for a rapidly growing streaming service. The company wants to collect and process real-time user engagement data to provide personalized content recommendations, monitor system performance, and analyze viewing trends. With your three years of experience in data engineering, how would you design and implement this data pipeline?

**Answer:**

To design and implement a scalable data pipeline for a streaming service using Azure Data Factory and Databricks, I would start by leveraging Azure Data Factory (ADF) to orchestrate the data flow. First, I would set up ADF to ingest real-time user engagement data from various sources, such as web logs, mobile app interactions, and streaming events. This data would be collected using Azure Event Hubs or Azure IoT Hub, which can handle high-throughput data streams efficiently.

Next, I would create ADF pipelines to move the ingested data into a staging area in Azure Data Lake Storage (ADLS). Once the data is securely stored, I would use Azure Databricks for data manipulation and transformation. In Databricks, I would develop ETL (Extract, Transform, Load) processes using Apache Spark to clean, enrich, and aggregate the data. This includes filtering out any irrelevant information, transforming raw data into a structured format, and performing aggregations to extract meaningful insights.

After processing the data in Databricks, the transformed data would be loaded back into ADLS, and then ADF would move it to an Azure SQL Data Warehouse or Azure Synapse Analytics for further analysis and reporting. This final step ensures that the data is readily available for generating personalized content recommendations, monitoring system performance, and analyzing viewing trends.

To ensure the pipeline's scalability and reliability, I would implement automated monitoring and alerting using Azure Monitor and configure ADF for error handling and retry policies. This approach, combining the powerful orchestration capabilities of Azure Data Factory with the robust data processing abilities of Databricks, would result in a highly efficient and scalable data pipeline capable of meeting the streaming service's needs.

**QUESTION**: What is partitioning in SQL?

**Answer:**

Partitioning in SQL is a technique used to divide a large database table into smaller, more manageable pieces called partitions. Each partition can be stored and accessed separately, but together they still represent the whole table.

Here's a simple explanation:

*Imagine you have a big book with thousands of pages. Instead of trying to carry the entire book around, you split it into smaller sections—one section for each chapter. This way, you can easily carry, read, or search through just one chapter at a time without having to handle the whole book.*

**QUESTION**: What is Data Skewness in Spark?

**Answer:**

Data skewness in Spark occurs when some partitions have significantly more data than others, leading to imbalanced workloads. This imbalance can cause performance issues, as some tasks take much longer to complete than others, slowing down the entire job.

*Let's consider a basket containing either apples. If there are a few very large apples that take up a lot of space in the basket, while the rest are small, the basket becomes unbalanced. This situation mirrors data skewness in Spark. The partitions representing these large fruits will require more processing time than the partitions with smaller fruits, causing performance issues similar to data skewness in Spark.*

**QUESTION:** What is Repartition in Spark?

**Answer:**

Repartitioning is a process of redistributing data across a specified number of partitions. It helps balance the workload and improve performance by ensuring that each partition has a more even amount of data to process. Repartitioning can either increase or decrease the number of partitions based on the desired distribution of data for efficient processing.

*Think of repartitioning in Spark like organizing books on a bookshelf. Initially, books might be scattered across different shelves, with some shelves having more books than others. Repartitioning is like rearranging the books so that each shelf has a similar number of books, making it easier to find and access them. Similarly, in Spark, repartitioning redistributes data across partitions so that each partition has a more balanced workload, improving processing efficiency, just like organizing books on a shelf improves accessibility.*

**QUESTION:** What is Check Pointing in Spark?

**Answer:**

Check pointing in Spark is like taking a snapshot of your work during a long journey. It's a way to save the intermediate state of your Spark job to disk so that if something goes wrong, you don't have to start from the beginning. Instead, you can restart from the checkpoint, saving time and resources. It's especially useful for long-running jobs or jobs with complex transformations, ensuring fault tolerance and easier recovery in case of failures.

**QUESTION:** What is Adaptive Query Execution?

**Answer:**

AQE in Spark dynamically optimizes the execution plan during runtime based on factors like data distribution and resource availability, leading to faster and more efficient query processing.

*It is like having a GPS on a bike trip that changes your route based on real-time road conditions. Similarly, AQE adjusts Spark's execution plan during query processing based on the actual data and resource availability, making the journey smoother and more efficient.*

SQL BASED QUESTIONS

**1**. Difference between SQL and PL/SQL ?

**Answer:**

SQL is a language used to query and manipulate data in relational databases, while PL/SQL is a procedural extension of SQL used to write stored procedures, functions, and other programmatic constructs within Oracle databases. SQL is primarily used for data retrieval and manipulation, while PL/SQL allows for more complex logic, such as loops, conditional statements, and exception handling, making it suitable for building applications and business logic within the database.

**2**. What is a stored procedure?

**Answer:**

A stored procedure is a named set of SQL statements that perform a specific task. It's like a reusable script stored in the database that can be executed whenever needed. Stored procedures are often used for tasks like data manipulation, transaction management, or implementing business logic within the database.

**3**. How do you create a small stored procedure?

**Answer:**

CREATE PROCEDURE ProcedureName

AS

BEGIN

-- SQL statements for the procedure go here

END;

**4**. Have you worked on SQL functions?

**Answer:**

Yes, I've worked extensively with functions in SQL. Functions are incredibly useful for performing calculations, data transformations, and encapsulating reusable logic within the database. I've created scalar functions to calculate values based on input parameters, table-valued functions to return sets of data, and even aggregate functions for summarizing data. Functions have allowed me to streamline queries, improve code readability, and maintain consistency across database operations.

**5**. Difference between stored procedures and functions

**Answer:**

Stored procedures can return multiple values, execute DDL statements, and manage transactions, while functions return a single value and are primarily used for calculations or data transformation tasks. Stored procedures are invoked using the EXECUTE keyword, whereas functions are called directly in SQL queries or expressions.

**6**. What are views with examples

**Answer:**

Views in SQL are virtual tables that display data from one or more base tables. They provide a way to simplify complex queries, hide sensitive data, and present a customized view of the data to users.

CREATE VIEW EmployeeDetails AS

SELECT EmployeeID, FirstName, LastName, Department

FROM Employees;

**7**. If Any Changes Happen in the Main Tables Used in the Views, Will the View Update Automatically?

**Answer:**

In most cases, yes, if any changes occur in the main tables used in views, the view will reflect those changes automatically. Views are essentially virtual tables that display data from the underlying base tables. Therefore, any modifications, such as insertion, updating, or deletion, made to the base tables will be immediately reflected in the view when queried. This ensures consistency and real-time access to the most up-to-date data through the view.

**8**. Is Insertion, Updating, or Deletion Possible Directly on the View?

**Answer:**

Yes, insertion, updating, and deletion operations can be performed directly on a view in SQL under certain conditions. If the view is based on a single table or meets specific criteria defined as updatable, such as having a simple join or being based on a single table, then these operations are typically allowed. However, for more complex views involving multiple tables, aggregations, or joins, direct modification operations may not be possible or may require additional considerations, in which case, modifications would need to be performed on the underlying base tables instead of the view itself.

**9**. What Are Indexes, and Why Are They Required?

**Answer:**

Indexes in SQL are like the index at the back of a book, which helps you quickly find specific information. In databases, indexes are data structures that help speed up the retrieval of rows from a table based on the values in certain columns. They work by organizing the data in a way that makes it faster to search, sort, and retrieve information, similar to how an index in a book helps you locate relevant pages quickly. Indexes are required to improve the performance of database queries, especially for tables with large amounts of data, by reducing the amount of data that needs to be scanned to find the desired information.

**10**. Tell me the Types of Indexes.

**Answer:**

* **Primary Index:** Uniquely identifies each row in a table based on the primary key column(s).
* **Unique Index:** Enforces uniqueness on one or more columns, allowing NULL values (except in NOT NULL columns).
* **Clustered Index:** Dictates the physical order of data in a table, typically only one per table.
* **Non-Clustered Index:** Creates a separate structure for index entries, enabling multiple indexes per table.
* **Composite Index:** Involves multiple columns, optimizing queries with multiple filtering or sorting conditions.
* **Covering Index:** Includes all columns needed for a query, eliminating the need to access the actual table.
* **Bitmap Index:** Used for columns with a limited number of distinct values, represented by bitmaps for faster query performance.

**11**. Difference Between Clustered and Non-Clustered Index.

**Answer:**

A Clustered Index determines the physical order of data in a table, with the data itself organized based on the index key. Only one clustered index is allowed per table. In contrast, a non-clustered index creates a separate structure for index entries, pointing to the corresponding data rows, and multiple non-clustered indexes can exist per table.

SAMPLE SQL QUERIES

1). Write an SQL query to find the count of the total occurrences of a particular character – ‘n’ in the FullName field.

SELECT FullName,

LENGTH(FullName) - LENGTH(REPLACE(FullName, 'n', ''))

FROM EmployeeDetails;

2). Write an SQL query to update the employee names by removing leading and trailing spaces.

UPDATE EmployeeDetails

SET FullName = LTRIM(RTRIM(FullName));

3). Fetch all the employees who are not working on any project.

SELECT EmpId FROM EmployeeSalary

WHERE Project IS NULL;

4) . Write an SQL query to fetch employee names having a salary greater than or equal to 5000 and less than or equal to 10000.

SELECT FullName FROM EmployeeDetails

WHERE EmpId IN (SELECT EmpId FROM EmployeeSalary WHERE Salary BETWEEN 5000 AND 10000);

5). Write an SQL query to find the current date-time.

SELECT SYSDATE FROM DUAL;

6). Write an SQL query to fetch all the Employee details from the EmployeeDetails table who joined in the Year 2020.

SELECT \* FROM EmployeeDetails

WHERE DateOfJoining

BETWEEN '2020/01/01' AND '2020/12/31';

7). Write an SQL query to fetch all employee records from the EmployeeDetails table who have a salary record in the EmployeeSalary table.

SELECT \* FROM EmployeeDetails E

WHERE EXISTS (SELECT \* FROM EmployeeSalary S WHERE E.EmpId = S.EmpId);

8). Write an SQL query to fetch the project-wise count of employees sorted by project’s count in descending order.

SELECT Project, count(EmpId) EmpProjectCount

FROM EmployeeSalary

GROUP BY Project ORDER BY EmpProjectCount DESC;

9). Write a query to fetch employee names and salary records. Display the employee details even if the salary is not present for the employee.

SELECT E.FullName, S.Salary

FROM EmployeeDetails E

LEFT JOIN EmployeeSalary S ON E.EmpId = S.EmpId;

10). Write an SQL query to join 3 tables.

SELECT column1, column2

FROM TableA

JOIN TableB ON TableA.Column3 = TableB.Column3

JOIN TableC ON TableA.Column4 = TableC.Column4;

HANDLING SPARK JOB FAILURES QUESTIONS

**Q1) Diagnosing a Spark Job Failure:**

Your Spark job fails consistently at 40% progress. How would you go about diagnosing the cause of this failure? What specific logs or metrics would you check?

**Answer:**

To diagnose a Spark job that fails consistently at 40% progress, I would start by checking the Spark application logs for any error messages or stack traces around the time of the failure. These logs can provide insights into what might be going wrong. I would also look at the executor logs to see if any specific tasks or stages are failing.

Next, I would use the Spark UI to identify which stage is failing. The UI can help pinpoint if there's an issue like data skew, where some tasks have much more data than others, or if there's a problem with shuffling data between stages.

Additionally, I would monitor metrics such as memory usage, CPU utilization, and disk I/O to see if there are resource bottlenecks causing the job to fail. If the failure is due to running out of memory, I might need to increase executor memory or optimize the job to use less memory.

By checking these logs and metrics, I can identify the root cause of the failure and take the necessary steps to fix it.

**Q2) Resuming a Failed Job:**

A long-running Spark job fails after processing a significant amount of data. How would you approach resuming or recovering this job to avoid reprocessing all the data?

**Answer:**

To resume or recover a long-running Spark job that fails after processing a significant amount of data, I would enable checkpointing to save intermediate results periodically. This way, if the job fails, it can restart from the last checkpoint instead of starting over. Additionally, I would look into using Spark's external shuffle service and fault-tolerant storage systems like HDFS or S3 to persist data at critical points in the pipeline.

**Q3) Handling Out of Memory Errors:**

Your Spark job fails due to out of memory errors. What strategies would you use to resolve this issue and prevent it from happening in the future?

**Answer:**

If my Spark job fails due to out of memory errors, I would:

1. **Increase Memory**: Allocate more memory to executors and the driver.
2. **Optimize Partitions**: Increase the number of partitions to spread the data load.
3. **Use Disk Storage**: Store intermediate data on disk instead of in memory.
4. **Optimize Code**: Simplify transformations to use less memory.
5. **Tune Garbage Collection**: Adjust garbage collection settings for better memory management.

These steps help prevent out of memory errors and ensure smoother job execution.

**Q4) Dealing with Skewed Data**:

A Spark job fails because of data skew, where a few partitions have much more data than others. How would you handle and mitigate data skew in your Spark jobs?

**Answer:**

To handle and mitigate data skew in Spark jobs, I would start by increasing the number of partitions to ensure a more balanced data distribution. I would also use salting techniques, adding random keys to heavily skewed keys, which helps distribute data more evenly across partitions. Additionally, for joins, I would leverage techniques like broadcasting smaller tables or applying skew join hints to optimize the process. These strategies effectively address data skew and enhance job performance.

IMPROVING SPARK JOB PERFORMANCE

1. **Optimizing Spark Job Performance:**

A Spark job is taking longer than expected to complete. What steps would you take to diagnose and optimize its performance? Consider aspects like data partitioning, shuffling, and resource allocation.

1. **Tuning Spark Configurations:**

Your Spark job performance is suboptimal despite sufficient resources. What Spark configuration parameters would you adjust to improve performance, and why?

1. **Improving Shuffle Performance:**

A Spark job involves heavy shuffling of data, leading to performance bottlenecks. What techniques would you use to improve shuffle performance?

1. **Caching and Persistence:**

You have iterative operations in your Spark job that recompute the same data multiple times. How would you use caching or persistence to speed up your job?

APACHE SPARK BASED QUESTIONS

1. Apache Spark is a lighting real-time processing Framework.
2. What are the main features of Apache Spark?
3. What is a Resilient Distribution Dataset in Apache Spark?
4. What is a Transformation in Apache Spark?
5. What are security options in Apache Spark?
6. How will you monitor Apache Spark?

**Answer:**

To monitor an Apache Spark job, I would use the Spark UI, which provides detailed information about the job's progress, including stages, tasks, and executor metrics like task duration, shuffle read/write, and memory usage. I would regularly check the Spark UI to track job performance and identify any issues. Additionally, I would review the application logs provided by the Spark cluster for any error messages or warnings to help troubleshoot problems. This approach allows me to effectively monitor and manage the performance of Spark jobs.

1. What are the main libraries of Apache Spark?
2. What are the main functions of Spark Core in Apache Spark?
3. How will you do memory tuning in Spark?
4. What are the two ways to create RDD in Spark?
5. What are the main operations that can be done on a RDD in Apache Spark?
6. What are the common Transformations in Apache Spark?
7. What are the common Actions in Apache Spark?
8. What is a Shuffle operation in Spark?
9. What are the operations that can cause a shuffle in Spark?
10. What is purpose of Spark SQL?

**Answer:**

The purpose of Spark SQL is to allow querying and processing of structured data using SQL queries within the Apache Spark framework. It provides a way to seamlessly combine SQL queries with Spark's powerful data processing capabilities, making it easier to work with structured data stored in various formats like JSON, Parquet, or databases. Spark SQL also integrates with data visualization tools and supports querying big data efficiently.

1. What is a DataFrame in Spark SQL?
2. What is a Parquet file in Spark?

**Answer:**

1. What is the difference between Apache Spark and Apache Hadoop MapReduce?
2. What are the main languages supported by Apache Spark?
3. What are the file systems supported by Spark?
4. What is a Spark Driver?

**Answer:**

In Spark, the Driver is like the brain of the operation. It receives instructions (code) from the user, decides how to break down the tasks, and coordinates with the worker computers in the cluster to get the job done. It manages the overall workflow, keeps track of progress, and gathers results to deliver the final output. Essentially, it's the control center that guides the entire process of data processing in Spark.

1. What is an RDD Lineage?

**Answer:**

RDD Lineage in Apache Spark is a record of all the transformations that were applied to an RDD (Resilient Distributed Dataset) to create it. It provides a detailed history of the operations performed on the dataset, allowing Spark to recompute lost data by tracing back through the transformations. This lineage information is essential for fault tolerance, as it enables Spark to rebuild data by reapplying transformations to the original data sources in case of node failures.

1. What are the two main types of Vector in Spark?

**Answer:**

In Apache Spark, the two main types of vectors used for representing data in machine learning algorithms are **DenseVector** and **SparseVector**.

* **Dense Vector**

A Dense Vector is used to represent data where most of the elements are non-zero. It stores all elements explicitly.

* **Sparse Vector**

A SparseVector is used for data where most of the elements are zero. It efficiently stores only the non-zero elements and their indices.

1. What are the different deployment modes of Apache Spark?
2. What is lazy evaluation in Apache Spark?
3. What are the core components of a distributed application in Apache Spark?
4. What is the difference in cache() and persist() methods in Apache Spark?

**Answer:**

In Apache Spark, both the cache and persist methods are used to store RDDs (Resilient Distributed Datasets) or DataFrames in memory for faster access during subsequent operations. The main difference between them lies in the flexibility of storage levels.

**Cache():**

**Default Storage Level:** The cache method stores the data in memory only, using the default storage level which is MEMORY\_ONLY.

**Use Case:** Use cache when you are sure that the data can fit into memory and you want to optimize for speed by avoiding recomputation of the dataset.

**Syntax:** df.cache()

**Persist():**

**Custom Storage Levels:** The persist method allows you to specify different storage levels, providing more flexibility in how and where the data is stored. You can store the data in memory, on disk, or both, and even replicate it across nodes.

**Storage Levels Include:**

* MEMORY\_ONLY: Store RDD as deserialized objects in memory.
* MEMORY\_AND\_DISK: Store RDD as deserialized objects in memory and spill to disk when memory is insufficient.
* MEMORY\_ONLY\_SER: Store RDD as serialized objects in memory.
* MEMORY\_AND\_DISK\_SER: Store RDD as serialized objects in memory and spill to disk.
* DISK\_ONLY: Store RDD only on disk.
* MEMORY\_ONLY\_2, MEMORY\_AND\_DISK\_2, etc.: Same as the above but with replication across nodes.

**Use Case:** Use persist when you need more control over the storage strategy, particularly when dealing with large datasets that might not fit entirely in memory or when you need fault tolerance through replication.

**cache()**: It is a shorthand for persist() with the default storage level MEMORY\_ONLY.

**persist():** Itallows specifying various storage levels, offering more flexibility in handling data that might not fit in memory.

1. How will you remove data from cache in Apache Spark?
2. What is the use of SparkContext in Apache Spark?
3. Do we need HDFS for running Spark application?
4. What is Spark Streaming?
5. How does Spark Streaming work internally?
6. What is a Pipeline in Apache Spark?
7. How does Pipeline work in Apache Spark?

**Answer:**

In Apache Spark, a pipeline is a sequence of stages that are executed to process data. Each stage typically performs a specific task or transformation on the data. Here's how the pipeline works:

1. **Data Source:** The pipeline starts with reading data from a source, such as a file system (like HDFS, S3, etc.), a database, or a streaming source.

2. **Transformations:** Once the data is read, Spark applies various transformations to manipulate the data. Transformations include operations like filtering, mapping, aggregating, joining, and more. These transformations are defined as a Directed Acyclic Graph (DAG) of stages.

3. **Actions:** Actions are operations that trigger the execution of the pipeline. Examples include collecting results to the driver program, writing data to an output source, or triggering computations for machine learning models.

4. **Execution:** Spark optimizes the pipeline's execution plan based on the DAG of transformations and dependencies. It then distributes the tasks across the cluster's executors, which are responsible for processing the data in parallel.

5. **Fault Tolerance:** Throughout the execution, Spark ensures fault tolerance by maintaining lineage information (i.e., how each RDD or DataFrame was derived from the original data). If a task or executor fails, Spark can recompute the affected data partitions using the lineage information.

6. **Result:** Once all stages are executed successfully, the final result or output is obtained from the actions performed in the pipeline.

Overall, the pipeline in Apache Spark enables efficient and scalable data processing through distributed computation, leveraging its in-memory computing capabilities to handle large-scale data sets effectively.

1. What is the difference between Transformer and Estimator in Apache Spark?
2. What are Managed and External Table in Spark?

**Managed Table:** In a managed table, Spark takes full control of both the metadata (schema) and the data itself. All information about the table, including its structure and location, is stored within Spark’s internal catalog (e.g., the Hive Metastore). Managed tables are the default type of table in Spark and are typically used when you want Spark to handle the entire lifecycle of the table, from creation to data storage and cleanup.

**External Table:** On the other hand, external tables are designed to access data stored outside of Spark’s control. In an external table, Spark manages only the metadata (schema) in its internal catalog, while the actual data resides in an external storage system like HDFS, S3, or a relational database. External tables are ideal for scenarios where data is shared across different Spark applications or when you want to leverage existing data stored in external locations without moving or copying it.

*In Apache Spark, there are two main types of tables: managed and external. Managed tables are simple and managed by Spark, while external tables allow exploring data beyond Spark’s internal storage. Understanding these differences helps you make better data management decisions for your Spark applications. Now, you’re ready to make the most of Spark’s capabilities for big data processing.*

INTRODUCTION BASED QUESTIONS

**1. Can you tell us about yourself?**

"I am a data engineer with over three years of experience in big data, machine learning, data science, and cloud technologies. I currently work as an Associate Consultant at Capgemini India, where I specialize in developing and optimizing data pipelines. My role involves handling large volumes of data, ensuring high performance and efficiency, and collaborating with teams to solve complex problems. I have a strong background in debugging production notebooks, streamlining workflow processes, and authoring comprehensive documentation for ETL processes. My goal is to leverage my technical skills and experience to deliver high-quality solutions that drive business success."

**2. What are some of the data projects you’ve worked on?**

At Everest Insurance, I built and maintained data pipelines that handle over 1 TB of data daily from various sources like TPA, SoR, and ADLS Data Lake. I enhanced efficiency by debugging production notebooks, which reduced runtime by 35%, and optimized SQL queries to cut data retrieval times by 50%.

At Hindustan Unilever, I engineered Intelligent Pipelines for parallel data loading and re-constructed over 250 data pipelines with comprehensive functionality. I also merged multiple Azure Data Pipelines, improving efficiency by 40%.

In my previous role at Burberry London, I managed multiple change requests for live reports, achieving a 99% accuracy rate. I conducted rigorous Unit and Regression Testing to ensure robustness and reliability and collaborated closely with business users for seamless production deployment.

During my tenure at Millennial Garage, I led the development of an EDA Dashboard with interactive graphs and ML algorithms, which generated detailed reports within 5 seconds. I also integrated algorithms for enhanced analytics accuracy and implemented data cleaning and validation techniques.

**3. How do you handle tight deadlines and pressure?**

"I handle tight deadlines and pressure by prioritizing tasks, maintaining clear communication with my team, and breaking down projects into manageable parts. For example, at Burberry London, I managed multiple change requests simultaneously by collaborating closely with various development teams, ensuring timely and accurate modifications. I also focus on continuous improvement, such as debugging and optimizing workflows, which significantly reduced runtime and increased efficiency in my projects at Everest Insurance and Hindustan Unilever."

**4. How do you ensure the quality and accuracy of your data solutions?**

"I ensure the quality and accuracy of my data solutions through rigorous testing, documentation, and continuous monitoring. At Burberry London, I conducted thorough Unit and Regression Testing to ensure the robustness and reliability of reports. I also authored comprehensive documentation for data pipelines at Everest Insurance, which improved onboarding efficiency and provided a clear reference for ongoing maintenance. Additionally, I debugged and optimized production notebooks, which helped in identifying and resolving issues promptly, ensuring high-quality data processing."

**5. Can you describe a challenging project and how you overcame the challenges?**

"One of the most challenging projects I worked on was at Hindustan Unilever, where I had to re-construct over 250 data pipelines with comprehensive functionality. The main challenge was to ensure seamless integration and functionality within a tight timeframe. I tackled this by breaking down the project into smaller tasks, prioritizing critical pipelines, and setting up a periodic scheduling system for parallel data loading. This approach allowed me to manage the workload efficiently and complete the project on time. Additionally, I collaborated with team members to debug and resolve issues, ensuring the pipelines were robust and reliable."

**6. What technologies and tools are you most proficient in?**

I am highly proficient in various data engineering technologies and tools. I have extensive experience with Python, SQL, and cloud platforms like Azure. I have built and maintained data pipelines from various data sources like APIs, TPA, SoR, and ADLS Data Lake. I am also skilled in using Databricks for data processing and debugging notebooks. For data visualization and machine learning tasks, I use libraries such as Plotly, Streamlit, and Seaborn. Additionally, I have experience with web development technologies from my time as a Full Stack Web Developer at SEO Design Pvt. Ltd.

UNITY CATALOG QUESTIONS

1. **What is Unity Catalog in Azure? Explain me in the simplest way.**

**Answer:**

Unity Catalog in Azure is like a super organized library for all your data.

Imagine a giant library where each book is a piece of data, and the library has many sections (departments, categories, etc.). Unity Catalog helps you:

* **Organize the Books (Data):** It categorizes and labels all the books properly, so you can easily find what you need.
* **Control Access:** It decides who can read, borrow, or even add new books to the library. Some sections might be restricted to certain people.
* **Track Usage:** It keeps a log of who has borrowed or returned books, ensuring everyone is accountable.

So, if your company has a lot of data from different departments, Unity Catalog helps keep everything in order, ensures only the right people access the right data, and keeps track of how the data is used.