**Intodution:**

Hello, I am Souvik, currently working as a Data Engineer with five years of experience at Cognizant. I joined Cognizant back in 2019, right after my Graduation. Throughout my tenure, I have specialized in the Banking and Financial Services (BFS) domain, where I have had the privilege of working closely with one of the largest financial institutions in the United States.

My technical expertise spans a wide range of technologies, with a strong focus on cloud platform and big data solutions. I have hands-on experience with Google Cloud Platform (GCP), leveraging tools such as PySpark, Dataproc, Google Cloud Storage, Google Cloud Function, Cloud SQL and BigQuery to develop and optimize data pipelines.

**Detailed Project Overview:**

Now, coming to my current project, we are building a robust and scalable data pipeline on Google Cloud Platform (GCP) to meet the complex analytical needs of our end users, including data analysts and business stakeholders. The data originates from an SFTP server, and as per the business requirements, it is first ingested into our Data Landing Zone (DLZ) or RAW layer on GCP.

From the DLZ or RAW layer, the data is initially transferred 'as is' to the BRONZE layer in Google Cloud Storage (GCS). This step is essential to preserve the original data and allows us to revert to the unmodified version if required, ensuring compliance with our data governance policies. Next, we conduct thorough Data Quality (DQ) checks using Apache Deequ on the BRONZE layer data. These checks are crucial to ensure the integrity and quality of data before it is processed further, as any discrepancies at this stage could lead to significant issues for our downstream users.

After the DQ checks, the data is ingested into the SILEVR layer in GCS, where it is stored in Delta format. Here, we perform schema restructuring, including standardization and partitioning to optimize the data for queries. This layer is where we create and manage our entity tables based on the requirements and specifications provided by our business analysts. These tables serve as the foundation for the downstream processes.

In the next phase, we execute complex joins and aggregations across multiple entity tables to produce derived datasets that are crucial for business intelligence and decision-making. This transformed data is then loaded into the GOLD layer in BigQuery, which is optimized for high-performance analytics. This layer is critical for our downstream users, including data analysts and business analysts, who rely on timely and accurate data to drive insights and reports.

To ensure that our pipeline meets the defined SLAs and OLAs, we have automated the entire process using Google Cloud Functions. These functions are triggered whenever a new file arrives in the DLZ, ensuring that the pipeline runs seamlessly and that data is processed and made available in a timely manner. The automation also includes audit logging and monitoring to ensure that any issues are promptly identified and resolved, minimizing the impact on downstream users.

Throughout the project, I've been deeply involved in every phase, from planning to execution and validation. In the planning phase, we conducted a detailed analysis of the existing schema, access patterns, and data sensitivity (such as PII, PCI, and demographics). We also evaluated the costs and restructured the schema to fit best practices in cloud environments, such as denormalization and partitioning. During execution, we focused on creating robust ETL jobs, ensuring data quality through validation jobs, and maintaining comprehensive audit logs. Post-load, we have also validated data integrity, ensured that access policies were correctly applied, and obtained sign-offs from the business.

**DPEV = Design, Plan, Execute, Validate**

**1. Gathering Requirements and Designing the Pipeline**

In the initial phase, gathering requirements is essential for creating a pipeline that meets the needs of the end users (e.g., data analysts, business stakeholders) while adhering to business requirements. Here's how we approach this step:

* **Stakeholder Engagement**: Engage with key stakeholders, including business analysts, data analysts, and data governance teams. Identify the type of data being processed, the key metrics they need for analysis, and any regulatory requirements such as data privacy (PII, PCI).
* **Understanding Data Sources**: Understand the structure and format of the data originating from the SFTP source, and how frequently it is updated. This will guide how the pipeline handles data ingestion frequency, incremental loads, or full reloads.
* **Define Data Architecture**: Based on the business needs, define the architecture that includes various layers (DLZ, BRONZE, SILVER, GOLD). The requirements should help decide how raw data is ingested into the BRONZE layer, transformed in the SILVER layer, and then aggregated in the GOLD layer.
* **Data Quality & Governance**: Gather requirements related to data quality (DQ) standards. Define what checks (e.g., completeness, uniqueness, and integrity) need to be applied and how the pipeline will handle bad data (i.e., segregate failed records in a separate history layer). Understanding governance policies will help in designing a solution that aligns with data retention and audit requirements.
* **Scalability & Performance**: Capture requirements around the expected data volume and query performance, which will guide partitioning, indexing, and schema design in the SILVER and GOLD layers. Ensure that the design can scale to accommodate future data growth.

**2. Planning and Estimation**

Once requirements are clear, careful planning and estimation are necessary to ensure the pipeline is built within the timeline and budget while maintaining quality and reliability.

* **Resource and Time Estimation**: Based on the size of the data and the complexity of transformations (e.g., complex joins, aggregations), estimate the processing time for each stage (e.g., ingestion, DQ checks, transformations). Identify potential bottlenecks, such as data shuffling during joins, and plan resource allocation accordingly.
* **Cost Evaluation**: Use GCP’s cost calculators to evaluate the costs associated with data storage in GCS, compute costs for Dataproc or Databricks clusters, BigQuery queries, and Cloud Functions. Plan the project to remain within budget constraints while ensuring the architecture meets business requirements.
* **Schema Restructuring and Partitioning**: Evaluate existing schemas and identify areas where denormalization, partitioning, or clustering will improve performance. Estimate the effort required to transform the schema to an optimized cloud-native format.
* **Data Sensitivity and Compliance**: Plan how sensitive data will be handled (e.g., anonymization or encryption of PII, PCI data), and estimate the effort required to implement security and compliance controls like IAM policies, VPCs, and service accounts.
* **Timeline Planning**: Break down the project into phases, such as ingestion, quality checks, transformation, and final reporting. Set milestones, deadlines, and buffer periods to accommodate unforeseen challenges.

**3. Execution of the Pipeline**

The execution phase involves building, testing, and deploying the pipeline while ensuring minimal disruption to the business.

* **Ingestion Layer (DLZ to BRONZE)**: Build ETL jobs that extract data from the SFTP server and load it into the BRONZE layer. This involves handling different file formats, incremental loads, or full reloads based on the business logic. For automation, configure Cloud Functions to trigger the pipeline when a new file arrives.
* **Data Quality Checks**: Implement data quality checks in the BRONZE layer using Apache Deequ. This involves writing Spark or PySpark jobs to run checks on data completeness, uniqueness, and consistency. Failed records can be moved to a "Silver\_HISTORY" layer for debugging, while valid records are passed to the next layer.
* **Transformation and Entity Creation in SILVER Layer**: Perform data cleansing and transformations based on business rules. Create entity tables that align with the business’s logical data model. Ensure that the data is optimized for querying by applying techniques like partitioning and clustering, which enhances performance.
* **Aggregation and Joins in GOLD Layer**: Perform complex transformations, such as multi-table joins and aggregations, to derive meaningful datasets for reporting. Load these derived datasets into BigQuery for use by downstream users, ensuring the data is ready for high-performance analytical queries.
* **Automation and Monitoring**: Automate the entire workflow using Google Cloud Functions to ensure that every stage runs smoothly. Integrate Google Cloud Logging and Monitoring to capture logs, errors, and metrics across the pipeline. This helps in tracking performance, ensuring SLAs and OLAs are met, and troubleshooting issues.

**4. Validations After Loading from One Layer to Another**

Validation is critical after data is transferred from one layer to another to ensure data quality, integrity, and compliance with the business requirements. Key validations include:

* **Schema Validation**: Ensure that the schema in the SILVER layer matches the expected structure. For instance, if a column is supposed to be non-null, ensure that no null values have crept in during the transformations.
* **Data Integrity Checks**: Compare record counts between layers to ensure no data has been dropped or duplicated during transitions between layers (BRONZE to SILVER or SILVER to GOLD). This can be done using row count checks and hash-based validations.
* **Data Quality Rechecks**: After loading the data into SILVER and GOLD layers, recheck critical data quality constraints like uniqueness, range checks, and completeness to ensure that the transformations didn’t introduce any anomalies.
* **Audit Logs**: Ensure that all actions, such as file arrival, transformation jobs, and data loads, are logged and auditable. Capture critical metadata (e.g., file names, load times, row counts) to help in tracking and troubleshooting issues in case of discrepancies.
* **Performance and Query Optimization Validation**: Ensure that the transformations, partitioning, and clustering in the SILVER layer are optimized for performance. Run performance tests to ensure that downstream queries in BigQuery meet the expected SLAs, especially if the data analysts rely on real-time or near-real-time reports.
* **Access and Security Validation**: Verify that IAM roles and permissions are correctly applied to ensure that only authorized users can access data in different layers (BRONZE, SILVER, GOLD). Validate encryption and data masking for sensitive information in compliance with PII/PCI regulations.

By following these steps, the pipeline will not only meet functional requirements but also ensure high performance, scalability, and data quality throughout the process.

**Scenario Based Questions from Current Project:**

**1. Data Volume and Scalability:**

* *Scenario*: "You’re working with high-volume data in your BRONZE layer, and the ingestion load is increasing every day. However, downstream processes (e.g., DQ checks, transformations) are now slowing down. What changes would you suggest to scale the pipeline to handle the growing data volume?"
* **Expected Response**: You might discuss options like partitioning strategies, using Databricks auto-scaling clusters, improving data partitioning in GCS, optimizing join strategies, or re-architecting the pipeline to run tasks in parallel across different layers.

**2. Data Quality and Governance:**

* *Scenario*: "Suppose some data quality checks are failing intermittently in the BRONZE layer due to corrupt or incomplete data. How would you ensure that the downstream users only consume good quality data while investigating the root cause?"
* **Expected Response**: You could explain how you would move bad records to a "Silver\_HISTORY" table, as you do now, but also suggest alerting mechanisms (using Google Cloud Monitoring) for immediate notifications. You might also implement re-ingestion strategies from DLZ or set up retries on failures while improving the DQ framework by adding more robust checks.

**3. Schema Evolution:**

* *Scenario*: "How would you handle a situation where the schema of incoming data changes unexpectedly (e.g., new columns are added) during ingestion? How can you ensure that this doesn’t break the pipeline?"
* **Expected Response**: You could propose leveraging Delta Lake's schema evolution capabilities and discuss how Databricks handles schema merging. You could also mention monitoring new schemas using Databricks’ audit logs and versioning mechanisms or setting up checks to validate schema integrity before allowing new columns to be propagated to downstream layers.

**4. Performance Optimization in BigQuery:**

* *Scenario*: "After loading data into the GOLD layer in BigQuery, analysts report slow performance when running complex queries. How would you optimize the dataset to improve query performance?"
* **Expected Response**: Discuss strategies like partitioning and clustering in BigQuery, denormalizing tables, or creating materialized views for frequently queried datasets. You might also suggest optimizing join conditions and aggregation logic within Databricks before data is loaded into BigQuery.

**5. Data Consistency Across Layers:**

* *Scenario*: "Data analysts are noticing inconsistent data between the BRONZE and SILVER layers, particularly during periods of high ingestion. What strategies can you implement to maintain consistency between layers?"
* **Expected Response**: You could suggest using ACID transactions in Delta Lake to ensure data consistency, even under concurrent loads. You might also discuss the use of versioning in Delta format to track changes and the implementation of robust error-handling mechanisms to prevent data loss or duplication during transitions between layers.

**6. Handling Duplicate Records:**

* *Scenario*: "The pipeline sometimes ingests duplicate records from the BRONZE layer into the SILVER layer. What mechanisms would you use to ensure the SILVER layer only contains unique, clean data?"
* **Expected Response**: Explain how you would handle deduplication by leveraging the MERGE or UPSET operation in Delta Lake. You could also propose adding a deduplication step during data ingestion based on a unique key or identifier and explain how Databricks allows for this kind of incremental data load.

**7. Automation and Failures:**

* *Scenario*: "Your pipeline is triggered via Cloud Functions when new files arrive in the DLZ. What would you do if a file is corrupted or causes the pipeline to fail in one of the intermediate layers (BRONZE to SILVER or SILVER to GOLD)?"
* **Expected Response**: Discuss setting up error-handling mechanisms that log and retry failures. You could mention Cloud Monitoring for alerting on failure and using Databricks' fault-tolerant capabilities to retry failed steps. Implementing a checkpointing mechanism within the pipeline could also be an approach to recover from specific points of failure without re-running the entire pipeline.

**8. Managing Historical Data:**

* *Scenario*: "The data volumes are growing rapidly, and older data in the SILVER and GOLD layers needs to be archived. How would you implement a cost-effective solution for archiving historical data?"
* **Expected Response**: Explain strategies such as moving historical data to a low-cost, cold storage option in GCS, setting up a data retention policy, and leveraging Delta Lake's time travel feature to ensure that historical data can be retrieved when necessary. You could also automate this process using Cloud Composer to trigger archiving jobs periodically.

**9. Multi-Region Data Management:**

* *Scenario*: "Your pipeline is now required to serve users in multiple geographical regions, but data access latency is a concern. How would you architect the pipeline to support multi-region data access while keeping costs manageable?"
* **Expected Response**: You might propose creating regional replicas of the data in both SILVER and GOLD layers in GCS and BigQuery, leveraging GCP’s multi-region storage options. You could also explain the importance of caching frequently accessed data in different regions to minimize latency and how tools like Cloud CDN can help optimize access times for users in different locations.

**10. Security and Compliance Challenges:**

* *Scenario*: "Some of the data you're handling contains PII, and new regulations require additional levels of data security and access control. How would you implement these controls in your current GCP-based architecture?"
* **Expected Response**: Describe implementing GCP IAM policies to restrict access to sensitive data, encrypting data at rest using GCS or BigQuery encryption features, and using field-level encryption or masking techniques for PII data. You could also discuss setting up detailed audit logs and ensuring compliance with regulations such as GDPR and HIPAA by using tools like Google Cloud’s Data Loss Prevention (DLP) API.

**11. Pipeline Latency and SLA Violations:**

* *Scenario*: "There are reports of pipeline delays, and you're breaching SLA agreements for some critical jobs. How would you troubleshoot and reduce the pipeline latency?"
* **Expected Response**: Propose options such as optimizing cluster configurations in Databricks (e.g., auto-scaling, using spot instances), fine-tuning the Spark jobs to reduce shuffling, and using caching mechanisms to store intermediate results. You could also mention leveraging monitoring tools like Cloud Monitoring to identify bottlenecks and optimize the frequency of file ingestion triggers via Cloud Functions.

**12. Data Lineage and Auditability:**

* *Scenario*: "The compliance team has requested detailed data lineage reports to track how data flows through your pipeline. How would you implement this, ensuring that every transformation and data movement is auditable?"
* **Expected Response**: Discuss integrating Delta Lake’s built-in audit logs and using Databricks’ notebooks to document transformations and data movements. You might also suggest using tools like Google Data Catalog to automate metadata management and track data lineage across GCP services.

**13. Disaster Recovery Plan:**

* *Scenario*: "If GCS or BigQuery were to experience downtime, what would your disaster recovery plan look like? How can you ensure minimal downtime for the data pipeline?"
* **Expected Response**: You could mention setting up a multi-region disaster recovery plan, using GCP’s automated backups, and leveraging Delta Lake’s time travel features to recover lost data. Additionally, outline how you would use Cloud Functions or Composer to handle job failures and trigger recovery actions such as re-running jobs or loading from checkpoints.

**STAR:** Situation, Task, Action, Result

**Ensuring Data Quality Automation Using Apache Deequ**

**S - Situation:**

In the current project, we faced a recurring issue where low-quality or incomplete data was entering our pipeline at the BRONZE layer from upstream systems (SFTP). This poor data quality led to downstream users, including data analysts and business analysts, reporting incorrect insights and inconsistencies in their reports, affecting their decision-making and timelines. Since the pipeline handles large volumes of data, manual checks were time-consuming and prone to human error. Furthermore, there were no automated mechanisms to flag these issues in real-time, leading to delays in meeting SLAs with business stakeholders.

**T - Task:**

We needed to automate data quality checks to identify issues at the earliest stage, in the BRONZE layer, and prevent bad data from flowing downstream to the SILVER and GOLD layers. This automation needed to be efficient, scalable, and seamlessly integrated into the pipeline. Additionally, we had to ensure that the pipeline runs without manual intervention while providing logging and monitoring for quick troubleshooting.

**A - Action:**

1. **Implemented Apache Deequ**: We integrated the Apache Deequ framework to automate data quality checks in the pipeline. This included:
   * Checking for completeness (e.g., ensuring no missing values in critical columns).
   * Verifying uniqueness and integrity constraints.
   * Performing business-specific checks (e.g., ensuring non-negative values for certain financial metrics).
2. **Automated DQ Checks**: We embedded these data quality checks at the BRONZE layer as part of the ETL pipeline, running them each time a new file was ingested from the Data Landing Zone (DLZ).
3. **Categorized Data Based on Quality**: We divided the incoming data into:
   * **dq\_passed\_df**: Data that passed all quality checks.
   * **bad\_records\_df**: Data that failed the quality checks, stored separately for further investigation.
4. **Automated Error Logging and Monitoring**: We integrated Google Cloud Logging to capture any errors or quality failures in real-time, triggering alerts through Google Cloud Monitoring. This ensured immediate visibility into data quality issues and their impact on downstream systems.
5. **Fail-Safe Mechanism**: Data that failed quality checks was not allowed to flow to the SILVER layer, preventing the propagation of bad data to the entity tables or GOLD layer in BigQuery, thus protecting downstream users.

**R - Result:**

* **Improved Data Quality**: By automating data quality checks, we reduced the number of errors that reached downstream systems by over 90%. This greatly improved the confidence of data analysts and business stakeholders in the reports they generated.
* **Time Savings**: Automation eliminated the need for manual data checks, saving our team countless hours and ensuring that the pipeline met SLAs without delays.
* **Real-Time Alerting**: With Google Cloud Monitoring and Logging in place, the team could quickly identify and respond to any data quality issues in real-time, minimizing the impact on downstream users.
* **Increased Stakeholder Satisfaction**: The business was able to trust the data insights generated from the pipeline, leading to faster decision-making and improved performance in data-driven projects.