1. What is your Data platform stack, ETL / ELT and Visualisation?
Answer:
My data platform stack includes:

- **ETL/ELT:** Google BigQuery, Azure Databricks (PySpark, SQL), AWS Lambda, AWS S3, PostgreSQL, Fivetran, Stitch, Syncari, REST APIs, dbt, Airflow
- **Cloud & Infra:** AWS (CloudWatch, EventBridge, DynamoDB, RDS), Azure, Google Cloud Platform
- **Orchestration & DevOps:** Airflow, GitHub CI/CD, Step Functions
- **Visualization:** Tableau, Looker Studio, Google Sheets, Excel
  I've implemented both batch and real-time pipelines with data flowing from APIs to cloud warehouses, applying transformation logic in dbt, Databricks, and SQL.

2. What is the entire lifecycle to operate a data lake end-to-end?
Answer:
My usual work:

1. **Ingestion**: Pull data from 3rd-party platforms and internal systems (Fivetran, APIs, Lambda).
2. **Storage**: Store raw data in AWS S3/GCS or BigQuery.
3. **Processing & Transformation**: Clean and transform data in Databricks (PySpark) and dbt.
4. **Data Modeling**: Create scalable models in BigQuery/PostgreSQL/Databricks.
5. **Orchestration**: Schedule and monitor workflows (Airflow, EventBridge, Step Functions).
6. **Monitoring & QA**: Check pipeline health using CloudWatch and Databricks logs.
7. **Serving & Visualization**: BI tools - Tableau, Looker Studio

3. What did you work on across the lifecycle of the data and what are the tools that you used?
Answer:
I've handled end-to-end pipeline development:

- **Ingestion**: Built custom connectors with Lambda, APIs, Fivetran, and Syncari.
- **Transformation**: Used dbt for SQL-based models, and PySpark on Databricks for advanced processing.
- **Storage/Warehousing**: Worked with BigQuery and PostgreSQL for structured storage.
- **Monitoring**: Used CloudWatch, EventBridge, and Databricks workflow logs.
- **Serving**: Enabled reporting through Tableau

4. What is a rather complex workflow you've worked on and its targeted use case?
Answer:
A complex example was the **GoTo pipeline**:

- **Use case**: Transitioning from fragmented systems to a unified, cloud-native data pipeline that fed marketing automation platforms.
- **Workflow**: Data flowed from Web APIs → AWS Lambda and Step Functions → Azure Databricks (transformation and batching) → Marketo via REST API → Syncari for final operational sync.
- **Challenge**: Zero downtime, high data fidelity, and scalable structure.
- **Impact**: Improved marketing automation, reporting accuracy, and reduced manual errors.

5. Data Processing knowledge (data warehousing)
Answer:
I have deep hands-on experience with:

- **Data Warehousing**: Google BigQuery, PostgreSQL
- **Transformation Tools**: dbt, Databricks (SQL, PySpark)
- **Optimizations**: Refactored 170+ SQL scripts for scalability and cost reduction; implemented partition trimming and job retries in Databricks.

6. Data orchestration tool (Airflow is what the team uses)
Answer:
Yes, I've worked with **Apache Airflow** to:

- Schedule ETL pipelines
- Manage dependencies and retries
- Integrate with AWS and BigQuery Additionally, I've used **AWS EventBridge**, **Step Functions**, and **CloudWatch** for monitoring on cloud-native stacks.

7. Data Platform knowledge (DevOps experience)
Answer:
I've implemented DevOps practices such as:

- **CI/CD Pipelines** using GitHub Actions
- **Code review culture in 2X** via PRs and Git workflows
- **Environment management** across staging and prod in AWS and BigQuery
- **Monitoring and automated failure handling** using CloudWatch, Step Functions, and custom retry logic in Databricks

8. Large Data and Dynamic Data Handling
I've worked with large and dynamic datasets from platforms like Salesforce, Marketo, and various ad platforms. To optimize performance and manage data effectively:

- **TrimPartition**: Implemented TrimPartition logic in Databricks for efficient processing, prevent unnecessary data retention.
- **Incremental Models**: Used incremental models in dbt to process only new or changed data.
- **BigQuery Optimizations**:
    - Refined heavy queries to reduce costs and improve runtime.
    - Used **partitioning** to divide tables into segments based on specific columns, making queries more efficient.
    - Implemented **clustering** for frequently queried columns to optimize scan times.

  DynamoDB: Designed retry and queuing logic to handle high-load Marketo to Syncari payloads effectively.