**Self intro:**

My name is Komal Nikhade, and I am from Nagpur.

I hold a Bachelor's degree in Computer Technology from RTMNU. For the past 3+ years,

I have been working as an Azure Data Engineer at brizkrieg solutions pune, and I am globally certified as microsoft azure data engineer.

""My technical skills include data storage solutions like Azure Data Lake Storage and Azure SQL Database,

and data integration tools such as Azure Data Factory and Synapse Analytics.

My proficiency extends to big data technologies like Apache Hadoop, data processing frameworks such as PySpark,

and Azure Databricks, along with SQL databases. I am skilled in programming languages such as Python.

I have successfully completed two projects, one in the retail domain and another in the e-commerce domain.

--------------------------------------------------

**1. Project Context:**

"Our Retail Analysis project focuses on developing an efficient and scalable data pipeline using Azure Data Factory,

Databricks, and Azure Data Lake Storage (ADLS) to manage and process large volumes of retail data.

The goal is to centralize data from various sources, transform it according to business logic, and make it available for

analytics and reporting."

**end to end flow:**

We maintained a control table in Azure SQL to track essential information about our source tables, such as their names,

loading criteria, incremental column, and the last ingestion date.

We set up an ADF pipeline to orchestrate the entire data workflow,

including activities to extract data from various source systems, transform it according to business logic, and load it into Azure Data

Lake Storage Gen2 (ADLS Gen2).

The pipeline started with a Lookup Activity to retrieve details from the control table.

Then, we used a ForEach Activity to iterate over each source table for processing.

Within the ForEach Activity, we used Copy Activity to transfer data from on-premises SQL Server to Azure Data Lake Storage (ADLS) Gen2.

We ensured a secure connection using Self-hosted Integration Runtime.

after copy activity using the notebooks, activity call the databricks notebooks and process the data and loaded into delta table

in the notebook using mount point we connact to adls cretae a dataframe a rawdata and loade into bronze zone

We moved data from the Bronze to Silver Zone, where we performed various transformations like adding new columns,

date transformations, applying conditions, and deriving new columns to clean and enhance the data.

As per business requirements, we aggregated data from multiple Silver zone tables and loaded it into the Gold zone.

This aggregated data was then used for reporting. After completing the data processing tasks, we updated the control

table with the date for the next run using a Stored Procedure Activity. We scheduled pipeline execution using triggers

for daily updates.

**Roles and Responsibilities Summary**

"I managed Azure Data Factory (ADF) pipelines to extract, transform, and load data from on-premises SQL Server to Azure Data Lake

Storage Gen2. My responsibilities included ensuring secure data transfer using Self-hosted Integration Runtime,

performing data transformations with Databricks notebooks, and structuring data into Bronze, Silver, and Gold

zones as per the Medallion architecture.

I updated control tables in Azure SQL to track source table details and scheduled daily pipeline executions for timely data

updates and reporting."

**Possible Challenges and Solutions in the Project**

1**. Out of Memory (OOM) Issues:**

During the transformation of data from the Bronze to Silver zone, we encountered Out of Memory (OOM) errors in our Databricks environment. This happened because the large amount of data and complex transformations needed more memory than was available.

To fix this, we:

1. **Partitioned Data Efficiently:** We broke the data into smaller, more manageable parts to spread the workload evenly.
2. **Optimized Spark Settings:** We adjusted the memory and CPU settings to better handle the data.
3. **Processed in Batches:** We processed smaller chunks of data at a time to reduce memory usage.
4. **Cached Intermediate Results:** We stored intermediate results to avoid recalculating them, which saved memory.

**2. Data Skew**:

Data skew occurred when some parts of our data had many more records than others, causing an uneven workload. This imbalance made some tasks take much longer, slowing down the entire pipeline.

We tackled data skew by:

1. **Salting Keys:** Adding extra values to the data keys to spread the data more evenly.
2. **Custom Partitioning:** Using custom methods to split the data more uniformly.
3. **Monitoring and Adjusting:** Regularly checking the data distribution and making changes to keep it balanced.

**Data Consistency and Quality Issues**

1. **Issue:** Keeping data consistent and high-quality throughout the pipeline was a big challenge.
2. **Explanation to Interviewer:** In the early stages of data ingestion and processing, we faced problems with data consistency and quality. We had to deal with different data formats, missing values, and duplicates, which made it hard to ensure accurate and reliable data transformations.