- **What are the most significant experiences you had in building data platforms? What roles have you played during this time?**

  Most Significant Experiences:

  1- Data Migration using Azure Synapse that includes fetching data from on premise servers to data lake, applying transformations in delta lake using python, sparksql and pyspark and building data warehousing and storing that into the dedicated pool (data warehouse) for further reporting and analysis.

  2- Reporting Universe Design where I built end to end ETL pipelines and KPI dashboards using Azure Data Factory and Power BI.

  3- A Fintech KYC application for avoiding chargebacks that uses several checks such as document segmentation, face detection and text processing using OCR.

  Roles:

  I played the roles of Data Engineer and Data Scientist and the roles were purely developer in nature.

- **How does the ETL processes differ when ingesting data from a NoSQL database & a SQL database? Do you have any experience around that, and if yes, please explain briefly based on your experience the differences and the difficulties of each type of technology.**

  Differences:

  NoSQL databases mostly store data in structure-less or semi-structured format that allows a lot of flexibility in transformations. So we can focus more on refining data than data modeling in our ETLs. On the other hand, SQL databases enforce schema and focus more on structure. NoSQL allows a range of languages while in SQL extraction, we can only standard sql language. In SQL ETLs, data extraction is more ACID compliant and more consistent while in NoSQL, it is less. Data load in NoSQL must accommodate schema evolution while in SQL, data loads are of fixed schema.

  Yes, I have experience working with commodities data for AI forecasting.

  Differences and Challenges:

  1- Schema Evolution is the main challenge to cater for new schema changes in NoSQL. One has to write new ETLs if data is non-overlapping for new requirements.

2- Query variety and flexibility in NoSQL comes with the challenge of using a new query for a new NoSQL database. While SQL uses only one standard language for every use case.

3- Large Data Volumes cause SQL ETLs to be more costly due to the underlying structure management and data consistency. NoSQL ETLs can handle big data in a better way with lesser costs but data optimization becomes a bit complex when dealing with big data.

- **What techniques would you use for efficiently processing large chunks of data? What are the technologies of your choice when it comes to this, and explain briefly the reasoning behind choosing them, while also giving some examples of usage from your own experience.**

  I often use Azure Synapse(underlying **Apache Spark)** for its fast, in-memory processing capabilities, making it ideal for both batch and stream processing. For example, while working on a data migration project using Azure Synapse, the underlying Spark Architecture helped me to handle big datasets efficiently and to optimize nightly ETL jobs, significantly reducing processing time.

  I have also used AWS GLUE for building ETLs, S3 and Data Lake and Redshift for DataWarehousing. Data processing and transformations are written in python.

  Choosing the right tool depends on the specific needs: whether it's real-time processing, batch jobs, or interactive queries. These technologies have helped me handle diverse data challenges efficiently, ensuring performance and scalability.

- **What is data integrity to you, and how do you maintain & measure data integrity in such ecosystem that is ever changing, and what are the most important metrics that come to mind that should be sought after? Please provide examples of ways that you have measured, the methods you have used to maintain those, and any techniques you have employed to mitigate any of these issues if they came up.**

  To me, data integrity defines how accurate the data is. During the development, data formatting and consistency rules should be implemented. And after every data project completion, Quality Assurance is a way to maintain and measure data integrity.

  Metrics:

  Accuracy, Consistency, Completeness and Availability are few metrics that should be sought after.

Examples:

In my experience, maintaining data integrity involves a combination of preventive measures and continuous monitoring during development. And after development, I generally get the data validated and checked by some Quality Assurance Engineer through building Data Comparison reports or checking against some defined rules. We also use data quality dashboards to monitor key metrics, ensuring any deviations were quickly identified and addressed.