*Please explain in detail the following and elaborate on your Data Engineering knowledge, it will good if there are specific used cases you want to use for example.*

1. What is your Data platform stack, ETL / ELT and Visualisation?
   1. For ETL, I mainly use Python mainly using Pandas (I’m planning to explore Polars as some saying it processes data much faster). For Visualization within Python environment, I like to use Plotly for their super pleasing UIs and have interactive hover features. For visualization outside Python, mainly I use MS Power BI to develop Reporting Dashboard for stakeholders.
2. What is the entire lifecycle to operate datalake end-to-end?
   1. My past project’s entire lifecycle includes:
      1. Data Ingestion – Source data arrives in various formats, usually Excel or CSV for legacy systems. I often automate ingestion using Python scripts scheduled on Azure.
      2. Staging Layer – Raw data is stored in a staging table (e.g., cbm.DATA).
      3. Delta Load Processing – Only new or changed data are captured and appended into the staging table, avoiding full refresh unless explicitly needed. I developed my own delta loading process using python.
      4. Transformation – A cleaned, structured dataset (fact table) is built from the staging table using custom logic based on composite keys and business rules.
      5. Data Validation – Validation checks are done via cross-reference with Master data set available from the DB.
      6. Serving Layer – Data is made available via APIs or directly exposed to Power BI for visualization.
3. What did you work on across the lifecycle of the data and what are the tools that you used?
   1. I’ve touched all stages of the data lifecycle:
      1. Ingestion: Python scripts to process Excel and CSV files from engineers extracted directly from Sharepoint using client id and client secret obtained from ICT.
      2. Staging: SQL tables to store raw data processed using Python pandas DataFrame.
      3. Cleaning & Transformation: Extensive Pandas work, including handling NULLs, renaming inconsistent values, custom Regular Expressions mapping, and mapping reference data.
      4. Validation: Joined against master tables or external mappings (some managed in Excel and loaded back into the DB).
      5. Serving: Created APIs with FastAPI for front-end or Power BI to consume.
      6. Automation: Azure VM + Task Scheduler; exploring Azure Data Factory and Airflow for future-proofing (however, I was stopped from further using Airflow by my manager as they are planning to migrate their pipelines to databrick’s Spark Jobs for data orchestration).
      7. Deployment: Azure DevOps for source control and CI/CD pipelines.
4. What is a rather complex workflow you’ve worked on and its targeted use case?
   1. My previous project’s objective is to unify siloed data within Upstream into a single data platform. My previous project had me do QC on the entire Upstream data coming to our data platform, particularly on maintaining the Reference & Master data for Asset Code, Field, Terminals and Platform across sources. As different sources had different glossaries (some record didn’t even exists in the master set), I had to craft a plan to handle them. For data that’s only having different separators (e.g.: -,<whitespace> etc.) require simpler approach of removing all available separator and do a left join on source against master set. For other records with unfamiliar naming, I dump into an Excel sheet and communicate with data source focals to identify their mapping against master set. My script will also capture the mapping stored in the Excel sheet to populate my data frame before I push into the database in a mapping table for out data platform to use. The objective is so that the Data Platform will display records from multiple sources with a standardized data while complying with the master set.
5. Data Processing knowledge (data warehousing)
   1. I’ve designed staging and fact tables in SQL with SHA-256’ hashrows to support my delta processing. I use:
      1. Staging Tables: Preserve full raw data.
      2. Fact Tables: Contain only the latest version of each logical record (e.g., based hashrow id).
      3. Dimension Tables: Master and reference sets, cleaned via mapping.
      4. I use SQL to create views that simplify reporting and reduce joins in Power BI.
6. Data orchestration tool (airflow is what the team uses)
   1. While I haven’t deployed production pipelines on Airflow yet, I understand DAG structuring, operator usage (Bash, Python, SQL), and backfill logic. Currently, my pipeline scheduling is managed using Windows Task, but I’m exploring Airflow for more robust automation and dependency management.
7. Data Platform knowledge (DevOps experience)
   1. We use Azure DevOps for VCS and deployment pipelines. I use DevOps to:
      1. Store Python scripts and API code.
      2. Deploy FastAPI applications to Azure VMs via PM2.
8. Large data and dynamic data handling (thinking beyond and considering data size and performance
   1. As we deploy our pipeline on a virtual machine server setup on Azure where we installed Python (not utilizing more modern approach like Databricks Spark Jobs etc. yet), I usually process large data by chunking them into smaller & manageable sizes to avoid kernel crashes and add on process logs for me to view and estimate how long a pipeline needs to take to finish a run. Also, I ensure that transformations are vectorized and avoid loops where possible.