QUESTION 1: Describe different types of data sources used in ETL with suitable examples.

Data sources generally fall into three categories based on their organization:

- Structured Sources: Highly organized data in rows and columns.
    - *Example:* Relational databases like MySQL, PostgreSQL, or SQL Server.
- Semi-Structured Sources: Data that doesn't reside in a relational database but has some organizational properties (like tags).
    - *Example:* JSON files, XML files, or NoSQL databases like MongoDB.
- Unstructured Sources: Data that lacks a predefined data model.
    - *Example:* PDFs, images, emails, or social media posts.

QUESTION 2: What is data extraction? Explain its role in the ETL pipeline.

Data Extraction is the first step of the ETL process. It involves retrieving data from various source systems (like databases or APIs) and moving it into a staging area.

- **Role:** It acts as the "entry point." Without extraction, the subsequent transformation and loading phases cannot happen. Its primary goal is to gather all necessary raw data without impacting the performance of the source system.

QUESTION 3:Explain the difference between CSV and Excel in terms of extraction and ETL usage.

Feature	CSV (Comma Separated Values)	Excel (.xlsx)
Structure	Plain text; very simple.	Binary format; complex.
Compatibility	Universally compatible with most systems, programming languages, and databases due to its simple, standard format.	Requires Microsoft Excel or compatible software to open and edit.
Automation	Highly suitable for automated processing and scripting (e.g., in Python or R) because it is machine-readable and has a simple structure.	More difficult to programmatically manipulate due to its complex, proprietary binary structure, requiring specific libraries that can be slower.
Functionality	Stores raw, flat tabular data only; does not support multiple sheets, formulas, charts, or macros.	Rich in features like multiple worksheets, cell formatting, formulas, charts, and pivot tables, designed for interactive data manipulation.

QUESTION 4: Explain the steps involved in extracting data from a relational database.

- **Establish Connection:** Use a driver (like JDBC/ODBC) to connect to the database.
- **Identify Change Data:** Use timestamps or logs to decide if you need a "Full Load" or an "Incremental Load."
- **Execute Queries:** Run SQL commands (e.g., SELECT * FROM table WHERE last_updated > 'date') to fetch data.
- **Data Validation:** Perform a quick count or schema check to ensure the data is correct.
- **Stage Data:** Move the retrieved data to a temporary storage area for transformation.

QUESTION 5: Explain three common challenges faced during data extraction.

- **Data Security/Access:** Getting the right permissions to access sensitive or siloed data without violating privacy laws.
- **Schema Evolution:** When the source system changes its table structure (e.g., a column is renamed), it can break the extraction script.
- **System Performance:** Extracting massive amounts of data during peak hours can slow down the source application for end-users.

QUESTION 6: What are APIs? Explain how APIs help in real-time data extraction.

APIs (Application Programming Interfaces) allow two software programs to communicate.

- **Real-time Role:** Unlike batch files (which run once a day), APIs can use **Webhooks** or **Streaming** to push data to the ETL pipeline the moment an event occurs (e.g., a customer makes a purchase). This enables "Real-time ETL" or "Continuous Integration."

QUESTION 7: Why are databases preferred for enterprise-level data extraction?

- **Scalability:** Databases can handle millions of records efficiently.
- **Data Integrity:** They enforce rules (schemas) that ensure the data extracted is consistent and clean.

- **Querying Power:** You can filter and aggregate data *before* it even leaves the source, reducing the volume of data moved across the network.

QUESTION 8: What steps should an ETL developer take when extracting data from large CSV files (1GB+)?

1. **Chunking/Batching:** Read the file in smaller "chunks" (e.g., 50,000 rows at a time) rather than loading the whole 1GB into RAM.
2. **Parallel Processing:** Use multiple CPU cores to process different parts of the file simultaneously.
3. **Schema Inference:** Explicitly define data types (e.g., telling the system a column is an "Integer") to save memory.
4. **Use Optimized Libraries:** Use tools like Dask or Polars (in Python) or specialized ETL tools (like Informatica or Talend) designed for high-volume file handling.