Data Extraction in ETL

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

Answer :

Different types of data sources used in ETL :

1. Categorization by Data Structure
2.Categorization by Origin/System Type

Categorization by Data Structure:

a.Structured Data Sources
b.Semi-Structured Data Sources
c.Unstructured Data Sources

a.Structured Data Sources :

- The data is organized into tables with rows and columns, making it easy to query using SQL.
- Highly organized; requires the least amount of cleaning before loading.
- **Example: Relational Databases (RDBMS)** like MySQL, Oracle, or SQL Server.
- A retail bank's core banking system that stores account numbers, transaction amounts, and customer names in neat tables.

b.Semi-Structured Data Sources :

- They offer more flexibility than structured data but require a "flattening" process during ETL to turn them into rows and columns.
- Example: JSON and XML files.
- An e-commerce website using a REST API to pull product details from a supplier. The data arrives as a JSON object: `{"item_id": 101, "price": 25.99}`.

c.Unstructured Data Sources :

- It Includes text, audio, and video.
- Example: Social Media Posts, Emails, or PDFs.
- A company extracting customer feedback from thousands of emails or scanned PDF invoices to perform sentiment analysis or cost tracking.

Categorization by Origin/System Type :

- Simple text-based files used for bulk data transfer between systems.
- Data hosted by third-party providers, accessed via web APIs.
- Non-relational databases that store data as documents or key-value pairs.
- Real-time "feeds" that provide a continuous flow of information.

- Example : CSV, Excel,Google Cloud Storage,**Apache Kafka** or IoT sensor data etc .

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

Answer :
Data Extraction is the foundational step where raw data is retrieved or "copied" from various source systems and moved into a temporary staging area for further required processing .

Role of Extraction in the ETL Pipeline :

- Most organizations store data in different "databases"—for example, sales data in a SQL database, marketing leads in a SaaS app like Salesforce, and website logs in flat files. Extraction pulls these disparate pieces together so they can be analyzed as a single, unified dataset.
- Performing heavy analysis directly on a live production database (like the one running a bank's ATM or an e-commerce checkout) can slow it down. The extraction process pulls the data out to a separate **Staging Area**, ensuring that subsequent heavy cleaning and transformation tasks don't interfere with the daily operations of the business.

- To save time and resources, extraction doesn't always pull every single record. It often identifies only what has changed since the last run.
    1. **Full Extraction:** Pulls everything (good for small tables or initial setups).
    2. **Incremental Extraction:** Pulls only "new" or "updated" rows based on timestamps or logs.
- Extraction is often the first "gatekeeper." Instead of pulling every column from a source, the ETL developer can choose to extract only the specific fields needed for the business goal .

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

Feature	**CSV (Comma-Separated Values)**	**Excel (.xlsx)**
Data Structure	Plain text; flat file (one table per file).	Binary/XML; workbook (multiple sheets, tabs).

Extraction Speed	**High.** Minimal CPU/RAM overhead for parsing.	**Low.** Requires decompression and XML parsing.
Data Integrity	High. Values are literal text; no auto-formatting.	Lower. Prone to auto-converting IDs to dates/scientific notation.
Memory Usage	Can be streamed line-by-line (low memory).	Usually requires loading the file into memory (high memory).
Volume Limit	No row limit (limited only by disk space).	Hard limit of **1,048,576 rows** per sheet.

Automation	Highly compatible with CLI, SQL `COPY`, and Python.	Requires specific libraries/drivers (OpenPyXL, ODBC).
Metadata	None. Headers are just text.	Includes formatting, formulas, and cell comments.

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

- Use a database driver (JDBC/ODBC) and credentials (host, port, username, password) to connect to the source system.
- Identify the specific tables, columns, or schemas required for the extraction.
- Write a SQL `SELECT` statement to filter or join data at the source.
- Run the query and fetch the result set. For large volumes, data is often "streamed" in batches to prevent memory overflow.

- Capture data types and schema definitions to ensure consistency during transformation.
- Save the extracted data into a temporary staging area (like a CSV file or S3 bucket) before moving to the next ETL phase.

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

- Managing missing values like NULLs, duplicates, and "dirty" data (e.g., misspelled names or "N/A" strings).
- Dates stored as strings (`VARCHAR`) rather than date types. Sources use mixed formats like `DD-MM-YYYY` vs `MM-DD-YYYY`. It's very difficult to set date format in sql workbench .
- Sudden changes in the source system, such as a renamed column, a new field, or a changed data type (e.g., an ID changing from `INT` to `STRING`).
- If you accidentally execute the "create table" code in SQL Workbench multiple times , multiple tables will be generated.
- Databases like MySQL often use `0000-00-00` for empty fields instead of `NULL` for date format .

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

An **API (Application Programming Interface)** is a set of rules and protocols that allows two software applications to communicate. In extraction, it acts as a "messenger" that takes a request from your ETL tool to a source system and returns the specific data requested.

- Unlike batch files (CSV/Excel) that must be exported and moved, APIs allow us to pull data the moment it is created in the source system.
- We can request a single specific record instead of downloading a whole database table, making the extraction lightweight and fast.
- Many APIs use "push" technology. Instead of checking for updates, the source system automatically "pushes" data to your ETL pipeline the second an event occurs .
- APIs enable continuous streaming. Tools can query an API every few seconds to keep a Data Warehouse perfectly synced with a live application.

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

For large-scale operations, databases are superior to files (like CSV or Excel) due to their structural robustness and performance capabilities.

- Databases use Indexes to locate specific rows instantly. Extracting a single record from a 1TB database takes milliseconds, whereas a 1TB file requires a slow, full-file scan.
- Multiple ETL pipelines can extract data simultaneously without locking the system. In file systems, if one process opens a file, others are often blocked.
- Enforcing **Foreign Keys** and **Constraints**, ensure that the data being extracted is consistent and "clean" .
- Databases provide Role-Based Access Control (RBAC). You can restrict an ETL tool to extract only specific columns (masking sensitive data like PII) and log exactly who accessed what.
- Features like Change Data Capture (CDC) or timestamps allow enterprises to extract only *new* or *changed* rows since the last run, rather than re-processing the entire dataset.

- Data remains valid even during system crashes or power failures, preventing the "partial/corrupt data" issues common in file transfers.

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

- Instead of loading the entire file into RAM,we can read the file in small dataset or chunks using native file streams.
- Manually define data types (e.g., `float`, `int`) before reading to reduce memory consumption by up to **80%** compared to auto-detection.
- We can Split the file into smaller segments and process them simultaneously using multi-threading or multi-processing
- Using "Column Projection" to read only the necessary columns, ignoring irrelevant data to save memory.
- We can perform basic data cleaning (removing NULLs or fixing date formats) *during* the read process before the data reaches the transformation level .