**Python:**

**1.Exceptional Handling**

Exception handling is used to handle errors in Python without crashing the program.

Key Components:

try → The block where you write code that might cause an error.  
except → The block that handles the error when it occurs.  
raise → Used to manually trigger an exception.  
finally →This block always runs, no matter what

**2.Multi threading and Multiprocessing**

Multithreading: Used for I/O-bound tasks (e.g., file I/O, API calls). Runs on a single core with multiple threads.

Multiprocessing: Used for CPU-bound tasks (e.g., data processing, computations). Runs on multiple cores.

**Pyspark:**

**1.Lazy Evaluation**

PySpark does not execute transformations immediately. It waits until an action like show(), count(), or write() is triggered. This helps optimize the process by reducing unnecessary computations.

**2. Lakehouse Architecture :**

Lakehouse Architecture - modern way of storing and managing data.It stores and processes all types of data – structured, semi-structured, and unstructured.  
It combines the best features of a Data Lake and a Data Warehouse into a single system. Built on open formats like Parquet & Delta Lake, allowing flexibility

Lakehouse architecture = Data Lake and Data Warehouse

Earlier, companies used:  
Data Warehouses – Structured, fast queries, ACID Transanctions ,but expensive and hard to scale.  
Data Lakes – Cheap, scalable, but slow and lacked data management.

These two systems created problems:

* Data Duplication ,Complexity ,High Costs

The Solution is– Lakehouse Architecture

**3. Performance Optimization While Writing Silver Layer Data to Gold Layer**

Use efficient file formats (Parquet over CSV).

Optimize partitions.

Cache/persist when needed.

Broadcast joins for small datasets.

Increase parallelism using repartition()**.**

**4.Upsert in Deltalakehouse**

Upsert = Insert + Update  
Delta Lake allows upserting using MERGE to update existing records and insert new ones.

**5.Pivot and Unpivot**

* Pivot: Converts rows into columns. (Used for summarizing data)
* Unpivot: Converts columns into rows. (Used for reshaping data)

**When to Use?**

Pivot - When you need row values to become columns

Unpivot - When you need columns to become rows

**6. Write Mode Types in PySpark**

When saving a DataFrame, PySpark provides different write modes:

* overwrite – Deletes old data and writes new data.
* append – Adds new data without deleting existing data.
* ignore – Does nothing if the file/table already exists.
* error (default) – Throws an error if data already exists.

**7. Joins in PySpark**

PySpark supports different types of joins:

* **Inner Join** – Returns matching rows from both tables.
* **Left Join** – Returns all rows from the left table, with matching data from the right.
* **Right Join** – Returns all rows from the right table, with matching data from the left.
* **Full Outer Join** – Returns all rows from both tables.
* **Left Semi Join** – Returns only left table rows that have a match in the right table.
* **Left Anti Join** – Returns left table rows that do NOT have a match in the right table.

**8**. **Persist and When to Use**

* persist() stores the data in memory/disk to avoid recomputation.
* Used when a dataset is used multiple times in a pipeline.

**9.Replace where:**

ReplaceWhere is used in Delta Lake when you want to overwrite specific partitions instead of the entire dataset. This helps in optimizing writes by updating only relevant partitions.

**10.Partitioning:**

Partitioning divides data into smaller, manageable parts (directories) based on column values. It improves query performance by reducing the amount of data scanned.

**SQL**

1.Query using GROUP BY and HAVING

SELECT department, COUNT(\*) as emp\_count

FROM employees

GROUP BY department

HAVING COUNT(\*) > 5;

2.Rank vs Dense Rank

* RANK(): Skips ranking for duplicates.
* DENSE\_RANK(): Doesn’t skip rankings.

SELECT name, salary, RANK() OVER (ORDER BY salary DESC) as rnk,

DENSE\_RANK() OVER (ORDER BY salary DESC) as dense\_rnk FROM employees;

3.Group By vs Window Functions

* GROUP BY: Aggregates data at a group level.
* WINDOW FUNCTION: Works on individual rows while considering multiple rows.

SELECT name, department, SUM(salary) OVER (PARTITION BY department) as total\_salary FROM employees;

4.Window Functions - ROW\_NUMBER(), RANK(), DENSE\_RANK()

ROW\_NUMBER(): Assigns a unique sequential number to each row.

RANK(): Assigns ranks with gaps when there are duplicates.

DENSE\_RANK(): Assigns ranks without gaps.

5.WHERE vs HAVING

* WHERE filters before aggregation.
* HAVING filters after aggregation.

6.Primary vs Unique Key

* Primary Key: Unique + Not Null.
* Unique Key: Only unique, can have NULL values.

Azure Data Factory :

1.Until vs ForEach Activity

* Until: Loops until a condition is met.
* ForEach: Iterates over a collection.

Databricks

1.Medallion Architecture

1. Bronze – Raw data.
2. Silver – Cleaned data.
3. Gold – Aggregated, business-ready data.

2.Unity Catalog:

Unity Catalog - is a tool in Databricks that helps manage data security, access, and organization in one place. It stores metadata (information about data) and makes it easier to control who can access or change data.

Before Unity Catalog (Using Hive Metastore):

No central control – Each workspace had its own rules, making management hard.Difficult access control– Setting permissions for multiple users was complicated.  
No automatic tracking – Couldn’t easily see who used or changed the data.

With Unity Catalog:

Manages all workspaces together – No need for separate setups.Better security – Controls access at the table, row, and column levels.  
Tracks data usage automatically – Shows who accessed and modified the data.  
Shares data across workspaces easily – No need to copy data.  
Works with all file types – Supports Parquet, Delta, CSV, JSON, etc.

3. Control Plane (Managed by Databricks)  
This layer handles management and orchestration and does not process or store user data.

* Manages notebooks, jobs, and cluster configurations.
* Schedules and monitors job executions.
* Controls access and security, such as Unity Catalog.
* Stores metadata and notebook source code.

Data Plane (Runs in Your Cloud Account)  
This is where the actual data processing happens, inside your cloud environment like AWS, Azure, or GCP.

* Runs Apache Spark jobs on Databricks clusters.
* Reads and writes data from cloud storage such as Delta Lake, Parquet, and CSV.
* Handles ETL, machine learning, and analytics workloads.

1.Join Fact and 4 Dimension Tables

SELECT f.\*, d1.\*, d2.\*, d3.\*, d4.\*

FROM fact\_table f

JOIN dim1 d1 ON f.key = d1.key

JOIN dim2 d2 ON f.key = d2.key

JOIN dim3 d3 ON f.key = d3.key

JOIN dim4 d4 ON f.key = d4.key;

1. Unmatched Records from Two DataFrames

df1.join(df2, "id", "left\_anti").show()