**1.What is Data Profiling?**

Data profiling is the process of analyzing, reviewing, and summarizing data to understand its structure, quality, and consistency. It helps in identifying duplicates, missing values, data types, patterns, and inconsistencies in a dataset.

**When Should You Use Data Profiling?**

🔹 Before Data Ingestion → To assess data quality before loading it into a system.

🔹 Before Data Transformation → To identify inconsistencies that may affect processing.

🔹 During Data Migration → To ensure data integrity when moving between systems.

🔹 For Data Quality Checks → To detect nulls, duplicates, incorrect formats, or missing values.

🔹 For Compliance & Governance → To verify sensitive data (PII) handling and ensure regulatory compliance.

**2. Slowly Changing Dimensions (SCD) Types**

Slowly Changing Dimensions (SCD) are used to **manage historical data changes** in a dimension table. There are multiple types based on how data is updated.

**SCD Type 0 – Retain Original Data (No Change)**

* **Description**: The dimension table remains unchanged.
* **Use Case**: Static reference data (e.g., country codes, product categories).
* **Example**: A product’s launch date should never change.

**SCD Type 1 – Overwrite the Data (No History)**

Description: The old data is replaced with the new data, and no history is maintained.

Use Case: Correcting spelling mistakes or errors.

Example: A customer updates their email address, and the system replaces the old email.

**SCD Type 2 – Maintain History with Versioning**

Description: Creates a new row with a different version or timestamp for every change.

Use Case: Tracking historical data like address changes or job titles.

Example: When an employee moves to a new department.

**SCD Type 3 – Maintain Limited History (Previous & Current)**

Description: Stores only the previous and current values in two separate columns.

Use Case: Keeping track of only the last change.

Example: Storing the previous address and current address.

**SCD Type 6 – Hybrid of Type 1, 2, and 3**

Description: Uses both Type 1 (overwrite), Type 2 (history), and Type 3 (limited history).

Use Case: When you need both historical tracking and easy access to recent changes.

Example: Keeping the current and previous department, while maintaining full history.

**3.Optimization Techniques in Spark**

**Partitioning:** Optimize data distribution by using appropriate partitioning (e.g., repartition(), coalesce()).

**Caching & Persistence:** Use. cache () or .persist() for reusing DataFrames in iterative operations.

**Broadcast Joins:** Improve performance for small lookup tables using broadcast().

**File Format Selection:** Use optimized formats like Parquet or Delta Lake instead of CSV.

**Predicate Pushdown:** Reduce data scanning by applying filters early in transformations.

**Optimized Shuffle Operations:** Minimize shuffle operations using partition-aware transformations.

**4.** **Schedule Trigger in ADF**

Used for running pipelines at a fixed time or interval (e.g., daily, hourly).

Example: Run a pipeline every day at 2 AM.

**Steps to create:**

Go to Azure Data Factory > Triggers

Select New > Schedule Trigger

Define Start Time, Frequency (daily, hourly), and Time Zone.

**5.Event-Based Trigger in ADF**

Triggers a pipeline when a specific event occurs (e.g., a file lands in a storage account).

Example: Run a pipeline when a new file is uploaded to Azure Blob Storage.

Go to ADF > Triggers

Select New > Event Trigger

Choose Storage Account and configure Container & Event Type (Blob created/deleted)

**6. what is different between fillna and dropna**

|  |  |  |
| --- | --- | --- |
| **Feature** | **fillna()** | **dropna()** |
| **Purpose** | Replaces NULL (or NaN) values with a specified. | Removes rows or columns containing NULL (or NaN) values. |
| **Effect on Data** | Retains all rows but fills missing values | Removes rows or columns containing NULL values |
| **Arguments** | fillna(value, subset) | dropna(how, thresh, subset) |
| **Use Case** | When you want to replace missing values. | When you want to remove rows or columns with missing values |

**Print Armstrong numbers from 1 to 1000**

def pal(x,y):

lst=[]

for i in range (x,y):

total =0

for j in str(i):

total +=(int(j)\*\*3)

if total == i:

lst.append(i)

return lst

print(pal(1,1000))