ACE - Technical Case Study:
SAP Data Modeling and PySpark Coding

Introduction

Congratulations and welcome to the technical boot camp for data engineers! In this exercise, you will practice data modeling and coding in PySpark to build data products that provide insights into our supply chain. Your goal is to help us build a connected world where everyone makes smart decisions and acts with impact. As a data engineer, your mission is to:

- Map real-world data into connected data products through business-driven data modeling.
- Support decision-making by creating a data model that depicts entities, events, alerts, and their relationships.
- Empower thousands of users to leverage insights and act with impact.

Exercise Overview

You will work with data from **two SAP systems** that have similar data sources. Your task is to implement and integrate this data to provide a unified view for supply chain insights. The exercise involves:

- Processing **Local Material** data.
- Processing **Process Order** data.
- Ensuring both datasets have the **same schema** for harmonization across systems.
- Writing modular, reusable code with proper documentation.
- Following best-in-class principles for flexibility and maintainability.

Table Overview

Hint: You find more information in SAP documentation

SAP Table	Local Material	Process Order
MARA	X	X
MARC	X	
MBEW	X	
T001W	X	
T001K	X	
T001	X	
AFKO		X
AFPO		X

AUFK		X

Important: You will create at least two Python scripts (local_material.py and process_order.py) for **each system**, resulting in a total of four scripts or even more if you decide to introduce more modules. You have to design a proper code project structure to maintain your pipeline accordingly.

Expectation

Technical requirements:

- The data modeling should be captured via an exported architecture diagram from https://app.diagrams.net/
- Code is written in PySpark: https://spark.apache.org/docs/latest/api/python/index.html
- If any additional packages are required > please add a requirements.txt to your submission.

Submission:

- **The project should be zipped with a README to provide additional instruction on how to get started.**
- **The code should run without error.**
- **The code should provide high-quality standards.**

++ If you see improvements beyond the given instruction. Please add them and highlight your thoughts!

Task 1: Process Local Material Data

Overview

You will process local material data by preparing and integrating data from various SAP tables. The goal is to create a comprehensive dataset that provides insights into materials at different plants, their valuations (i.e. how important are certain materials), and related company information.

Steps

1. Prepare General Material Data

- **Function: prep_general_material_data**
- **Input Data: SAP MARA table (General Material Data)**
- **Fields Needed:**
  - MANDT: Client
  - MATNR: Material Number
  - MEINS: Base Unit of Measure
  - **Global Material Number: Column name may vary between systems (e.g., ZZMDGM, ZZA_TEXT4)**
- **Parameters:**

    - col_mara_global_material_number: Specify the global material number column name for each system.
    - check_old_material_number_is_valid (Boolean): Set to True to filter out invalid old material numbers.
    - check_material_is_not_deleted (Boolean): Set to True to exclude materials flagged for deletion.
- **Transformation:**
    - **Filter materials:**
        - Old Material Number (BISMT) is not in ["ARCHIVE", "DUPLICATE", "RENUMBERED"] or is null.
        - Deletion flag (LVORM) is null or empty.
    - **Select the required columns.**
    - **Rename the global material number column to a consistent name, if necessary.**
- **Hints:**
    - Ensure that col_mara_global_material_number correctly references the global material number in each system.
- Example of handling dynamic column names:

```
sap_mara = sap_mara.select(
    "MANDT",
    "MATNR",
    "MEINS",
    F.col(col_mara_global_material_number).alias("global_material_number")
)
```

2. Prepare Material Valuation Data

- **Function: prep_material_valuation**
- **Input Data: SAP MBEW table (Material Valuation)**
- **Fields Needed:**
    - MANDT: Client
    - MATNR: Material Number
    - BWKEY: Valuation Area
    - VPRSV: Price Control Indicator
    - VERPR: Moving Average Price
    - STPRS: Standard Price
    - PEINH: Price Unit
    - BKLAS: Valuation Class
- **Transformation:**
    - **Filter out materials that are flagged for deletion (LVORM is null).**
    - **Filter for entries with BWTAR (Valuation Type) as null to exclude split valuation materials.**
    - **Deduplicate records:**
        - Rule take the record having highest evaluated price LAEPR (Last Evaluated Price) at MATNR and BWKEY level
- Keep the first record per group.
    - **Select the required columns.**
    - **Drop Duplicates.**
- **Hints:**

- Use Window.partitionBy() and F.row_number() to implement the deduplication logic.
- Example:

```
window_spec = Window.partitionBy(
    "MATNR", "BWKEY"
).orderBy(
    F.col("LAEPR").desc()
)

data = data.withColumn(
   "row_number", F.row_number().over(window_spec)
)

data = data.filter(F.col("row_number") == 1).drop("row_number")
```

3. Prepare Plant Data for Material

- **Function: prep_plant_data_for_material**
- **Input Data: SAP MARC table (Plant Data for Material)**
- **Fields Needed:**
  - SOURCE_SYSTEM_ERP: Source ERP system identifier
  - MATNR: Material Number
  - WERKS: Plant
  - Additional fields as required (e.g., PLIFZ, DZEIT, DISLS, etc.)
- **Parameters:**
  - check_deletion_flag_is_null (Boolean): Set to True to exclude records where LVORM is not null.
  - drop_duplicate_records (Boolean): Set to False unless deduplication is necessary.
- **Transformation:**
  - **Filter records where the deletion flag (LVORM) is null.**
  - **Select the required columns.**
  - **Drop Duplicates if drop_duplicate_records is True.**
- **Hints:**
  - Be cautious with the drop_duplicate_records parameter; only use it if duplicates are causing issues.
  - Ensure you're selecting all necessary fields for downstream joins.

4. Prepare Plant and Branches Data

- **Function: prep_plant_and_branches**
- **Input Data: SAP T001W table (Plant/Branch)**
- **Fields Needed:**
  - MANDT: Client
  - WERKS: Plant
  - BWKEY: Valuation Area
  - NAME1: Name of Plant/Branch

- **Transformation:**
  - **Select the required columns.**
- **Hint:**
  - This dataset provides plant names, which can be useful for reporting and analytics.

5. Prepare Valuation Area Data

- **Function: prep_valuation_area**
- **Input Data: SAP T001K table (Valuation Area)**
- **Fields Needed:**
  - MANDT: Client
  - BWKEY: Valuation Area
  - BUKRS: Company Code
- **Transformation:**
  - **Select the required columns.**
  - **Drop Duplicates to ensure uniqueness.**
- **Hint:**
  - The BWKEY (Valuation Area) is used to link materials to company codes.

6. Prepare Company Codes Data

- **Function: prep_company_codes**
- **Input Data: SAP T001 table (Company Codes)**
- **Fields Needed:**
  - MANDT: Client
  - BUKRS: Company Code
  - WAERS: Currency Key
- **Transformation:**
  - **Select the required columns.**
- **Hint:**
  - This data is important for financial reporting and currency conversions.

7. Integrate Data

- **Function: integration**
- **DataFrames to Integrate:**
  - **General Material Data (sap_mara)**
  - **Material Valuation Data (sap_mbew)**
  - **Plant Data for Material (sap_marc)**
  - **Valuation Area Data (sap_t001k)**
  - **Plant and Branches Data (sap_t001w)**
  - **Company Codes Data (sap_t001)**

- **Transformation:**
  - **Join operations:**
    - sap_marc **left join** sap_mara on MATNR.
    - **left join sap_t001w on MANDT and WERKS.**
    - **left join sap_mbew on MANDT, MATNR, and BWKEY.**
    - **left join sap_t001k on MANDT and BWKEY.**
    - **left join sap_t001 on MANDT and BUKRS.**
- **Hint:**
  - Ensure that all joins are **left joins** to preserve all records from the main dataset (sap_marc).
  - Keep track of the columns resulting from each join to avoid conflicts.

8. Post-Process Local Material Data

- **Function: post_prep_local_material**
- **Input Data: Resulting DataFrame from the integration step.**
- **Transformation:**
  - **Create mtl_plant_emd: Concatenate WERKS and NAME1 with a hyphen.**
  - **Assign global_mtl_id from MATNR or the global material number, as appropriate.**
  - **Derive two Derive the following keys (intra represents the primary key of one system and inter the primary key the harmonized view.**
    - **Primary Key (primary_key_intra): Concatenate MATNR and WERKS.**
    - **Primary Key (primary_key_inter): Concatenate SOURCE_SYSTEM_ERP, MATNR, and WERKS.**
  - **Handle Duplicates:**
    - **Add a temporary column no_of_duplicates indicating duplicate counts.**
    - **Drop Duplicates based on SOURCE_SYSTEM_ERP, MATNR, and WERKS.**
- **Hints:**
  - For deduplication, consider adding a sequence number or timestamp to determine which records to keep.
  - Use the derive_intra_and_inter_primary_key utility function to generate primary keys.

9. Write the Output DataFrame

- **Write the final local_material DataFrame to the designated output path.**
- **Hint:**
  - Verify that all required columns are present and correctly formatted.
  - Columns should be mapped to readable names

Expected Output

- A DataFrame named local_material with the processed local material data.
- The DataFrame should have the same schema across both systems for harmonization.

Task 2: Process Order Data

Overview

You will process process order data by preparing and integrating data from various SAP tables. The goal is to create a comprehensive dataset that provides insights into process orders, their statuses, and related material information.

Steps

1. Prepare Order Header Data

- **Function: prep_sap_order_header_data**
- **Input Data: SAP AFKO table (Order Header Data)**
- **Fields Needed:**
    - SOURCE_SYSTEM_ERP: Source ERP system identifier
    - MANDT: Client
    - AUFNR: Order Number
    - Various date fields (e.g., GLTRP, GSTRP, FTRMS, GLTRS, GSTRS, GSTRI, GETRI, GLTRI, FTRMI, FTRMP)
    - Additional fields (e.g., DISPO, FEVOR, PLGRP, FHORI, AUFPL)
- **Transformation:**
    - **Select the required columns.**
    - **Createstart_date and finish_date:**
        - Format GSTRP as 'yyyy-MM'.
        - If GSTRP is null, use the current date.
        - **Concatenate'-01' to form full dates.**
        - **Convert to date format.**
- **Hints:**
    - Use F.date_format() and F.concat_ws() to manipulate date strings.
    - Handle null values carefully to avoid errors.
    - Example:

```
data = data.withColumn(
"start_date", F.when(
            F.col("GSTRP").isNull(),
            F.date_format(F.current_date(), "yyyy-MM"),
          ).otherwise(F.date_format("GSTRP", "yyyy-MM")))
data = data.withColumn("start_date", F.concat_ws("-", F.col("start_date"), F.lit("01")))
```

2. Prepare Order Item Data

- **Function: prep_sap_order_item**
- **Input Data: SAP AFPO table (Order Item Data)**
- **Fields Needed:**
    - AUFNR: Order Number
    - POSNR: Order Item Number
    - DWERK: Plant

    - MATNR: Material Number
    - Additional fields (e.g., MEINS, KDAUF, KDPOS, etc.)
- **Transformation:**
    - **Select the required columns.**
- **Hint:**
    - These records represent items within each order; ensure that AUFNR and POSNR are included for uniqueness.

3. Prepare Order Master Data

- **Function: prep_sap_order_master_data**
- **Input Data: SAP AUFK table (Order Master Data)**
- **Fields Needed:**
    - AUFNR: Order Number
    - OBJNR: Object Number
    - ERDAT: Creation Date
    - ERNAM: Created By
    - AUART: Order Type
    - ZZGLTRP_ORIG: Original Basic Finish Date
    - ZZPRO_TEXT: Project Text
- **Transformation:**
    - **Select the required columns.**
- **Hint:**
    - The OBJNR is important for linking to other datasets or change documents.

4. Prepare General Material Data

- **Function: prep_sap_general_material_data**
- **Input Data: SAP MARA table (General Material Data)**
- **Fields Needed:**
    - MATNR: Material Number
    - **Global Material Number: Column may vary between systems**
    - NTGEW: Net Weight
    - MTART: Material Type
- **Parameters:**
    - col_global_material: Specify the global material number column name for each system.
- **Transformation:**
    - **Filter materials:**
        - Old Material Number (BISMT) is not in ["ARCHIVE", "DUPLICATE", "RENUMBERED"] or is null.
        - Deletion flag (LVORM) is null or empty.
    - **Select the required columns.**
    - **Rename the global material number column to a consistent name, if necessary.**

- **Hints:**
  - Similar to Task 1, ensure consistent handling of the global material number.
  - Use the .filter() method to apply multiple conditions.

6. Integrate Data

- **Function: integration**
- **DataFrames to Integrate:**
  - **Order Header Data (sap_afko)**
  - **Order Item Data (sap_afpo)**
  - **Order Master Data (sap_aufk)**
  - **General Material Data (sap_mara)**
- **Transformation:**
  - **Join operations:**
    - sap_afko **left join** sap_afpo on AUFNR.
    - Result **left join** sap_aufk on AUFNR.
    - Result **left join** sap_mara on MATNR.
    - If sap_cdpos is provided, result **left join** sap_cdpos on OBJNR.
  - **Handle Missing Values:**
    - Use ZZGLTRP_ORIG if available; otherwise, use GLTRP.
- **Hints:**
  - Use F.coalesce() to handle nulls in ZZGLTRP_ORIG.
  - Ensure all joins are correctly on their respective keys.

7. Post-Process Process Order Data

- **Function: post_prep_process_order**
- **Input Data: Resulting DataFrame from the integration step.**
- **Transformation:**
  - Derive the following keys (intra represents the primary key of one system and inter the primary key the harmonized view.
    - **Intra Primary Key (primary_key_intra): Concatenate AUFNR, POSNR, and DWERK.**
    - **Inter Primary Key (primary_key_inter): Concatenate SOURCE_SYSTEM_ERP, AUFNR, POSNR, and DWERK.**
  - **Calculate On-Time Flag:**
    - **Set on_time_flag to:**
      - 1 if ZZGLTRP_ORIG >= LTRMI.
      - 0 if ZZGLTRP_ORIG < LTRMI.
      - null if dates are missing.
  - **Calculate On-Time Deviation and Late Delivery Bucket:**
    - **Compute actual_on_time_deviation as ZZGLTRP_ORIG - LTRMI.**

        - **Categorize late_delivery_bucket based on deviation days.**
    - **Ensure ZZGLTRP_ORIG is Present:**
        - **Add ZZGLTRP_ORIG with null values if it's not in the DataFrame.**
    - **Derive MTO vs. MTS Flag:**
        - **Set mto_vs_mts_flag to:**
            - "MTO" if KDAUF (Sales Order Number) is not null.
            - "MTS" otherwise.
    - **Convert Dates to Timestamps:**
        - **Create order_finish_timestamp from LTRMI.**
        - **Create order_start_timestamp from GSTRI.**
- **Hints:**
    - Use F.datediff() to calculate date differences.
    - Use F.when() for conditional logic.
    - Ensure data types are compatible when performing comparisons.

8. Write the Output DataFrame

- **Write the final process_order DataFrame to the designated output path.**
- **Hint:**
    - Verify that all required columns are present and correctly formatted.
    - Columns should be mapped to readable names

Expected Output

- A DataFrame named process_order with the processed process order data.
- The DataFrame should have the same schema across both systems for harmonization.

Task 3: Harmonize Data Across Systems

Overview

After processing, ensure that both datasets (local_material and process_order) from each SAP system have the same schema. This will allow you to harmonize the data and provide a unified view across systems.

Steps

1. **Compare Schemas:**
    a. **Print the schema of the local_material and process_order DataFrames from both systems.**
    b. **Verify that the column names and data types match.**
2. **Adjust Transformations if Necessary:**
    a. If there are discrepancies:
        i. **Rename columns to achieve consistency.**
        ii. **Cast columns to the same data types.**
        iii. **Add Missing Columns with default or null values where appropriate.**

3. **Document Schema Alignment:**
    a. **Comment in your code any changes made to align the schemas.**
    b. **Ensure that these changes do not affect data integrity.**

- **Hints:**
    - Use DataFrame.printSchema() to inspect schemas.
    - When adding missing columns, ensure the data types match.
    - Use .withColumn() to add missing columns with null values.

Additional Instructions

- **Code Structure:**
    - Use the provided code structures as references for your scripts.
    - Organize your code into functions for better readability and reusability.
- **Documentation:**
    - Include docstrings for all your functions, explaining their purpose, parameters, and returns.
    - Use inline comments to explain complex logic or transformations.
- **Testing:**
    - Write test cases or assertions within your code to check for expected outcomes.
    - Validate that the output DataFrames meet the requirements (e.g., primary keys are unique).
- **Best Practices:**
    - Follow PEP 8 style guidelines for Python code.
    - Handle exceptions and edge cases gracefully.
    - Optimize performance by avoiding unnecessary actions (e.g., avoid multiple shuffles).
- **Data Privacy:**
    - Ensure sensitive data is handled securely.
    - Do not include any real customer or employee data in your scripts.

++ If you see improvements beyond the given instruction. Please add them and highlight your thought process!

By completing this exercise, you will gain hands-on experience in data modeling and transformation using PySpark, specifically in integrating and harmonizing data from SAP systems. This will help you develop the skills to support data-driven decision-making and build impactful data products.

Good luck!