Hands-on Lab: Final Project

Estimated Effort: 75 minutes

Introduction

In this final project, you will apply all the knowledge gained throughout this course using a real-world project scenario and data that simulates the various tasks that a data engineer must perform. Consider the following scenario.

Scenario

You are a data engineer hired by a European online retail company to design a data workflow for their operations. You are required to perform all of the following tasks for them:

- Propose a detailed data architecture for the whole data process.
- Propose a detailed data warehouse schema and design its entity relationship diagram (ERD).
- Propose the infrastructure requirements for the required data architecture.
- Create an ETL pipeline to clean, process, and load the data to an SQL server for analysis. Test the pipeline on a sample database.
- Query the SQL database to access data from the server.
- Implement data analysis and data mining strategies on the final data.

Data set

This lab uses the Online Retail data set available in the UCI ML library, available publically under the CC BY 4.0 license.

Test environment

The testing environment for this project lab is linked and available within the course immediately following this lab link. Please open the lab environment in a separate window to the side and complete the lab setup process.

You can test the code generated in this lab within the testing interface.

Data architecture

To propose a data architecture for the retail system, you will set up the following guidelines based on the client specifications.

The client does not want to go for cloud based processing resources. They want an SQL-based central data repository that their employees from multiple countries can access for their use.

You can use the GenAI model to propose a data architecture. Try to create a prompt that will give you the expected response.

▼ Click here for a sample prompt

```
Create a data architecture, clearly defining each step of the process, for an online retail company.
Please take note of the following specifications.
- The company does not want to use cloud platforms for storage or processing.
-  They want an SQL-based central data repository that can be accessed by any employee as per requirements.
```

▼ Click here for a sample response

1. Define data sources: Identify all the sources of data within the company, such as sales transactions, customer information, inventory data, and so on.
2. Data collection: Set up processes to collect data from the various sources and store it in the on-premises SQL server. This can involve setting up data pipelines to automate the extraction and loading of data into the central repository.
3. Data transformation: Clean and transform the data to ensure consistency and quality. This may involve data cleansing, normalization, and standardization to make the data usable for analysis.
4. Data modeling: Design the database schema and data model to organize the data in a structured and efficient manner. Define relationships between different data entities to facilitate data retrieval and analysis.
5. Data access: Implement access controls and permissions to regulate who can access and modify the data in the central repository. This ensures data security and compliance with privacy regulations.
6. Data analysis: Provide tools and interfaces for employees to access and analyze the data stored in the central repository. This can include dashboards, reporting tools, and data visualization techniques to make data insights easily accessible.
7. Data backup and recovery: Establish regular data backup procedures to prevent data loss in case of system failures or disasters. Implement disaster recovery plans to ensure quick recovery of data in case of emergencies.

8. Data governance: Implement policies and procedures to govern the use and management of data within the company. This includes data quality standards, data retention policies, and data lifecycle management practices.

By following these steps, you can create a comprehensive data architecture for the online retail company that meets their specifications of not using cloud platforms and having an SQL-based central data repository accessible to all employees.

Data warehouse schema and ERD

First, you need to set up the data warehouse schema and its ERD diagram. For that task, you need clearly defined requirements from the client as to the kind of data they want recorded. Assume that you received the following information from the client.

1. The client wants to record customer information, seller information, inventory information, and transaction invoice information.

2. The client wants the final data prepared such that the final record of sales invoices displayes the headers `InvoiceNo`, `StockCode`, `Description`, `Quantity`, `InvoiceDate`, `UnitPrice`, `CustomerID` and `Country`.

To define the schema of the data warehouse that meets these requirements, you can now write a prompt on the GenAI platform.

▶ Click here for a sample prompt
▶ Click here for a sample response

You can further use the generative AI platform to generate the SQL codes for creating this warehouse, and use this code to create the ERD on the DbDiagram interface, as explained earlier in the course.

▶ Click here for a sample prompt
▶ Click here for a sample response
▶ Click here for a sample ERD

Infrastructure requirements

You now need to define the infrastructure requirements for such a setup. You can make use of the same chat for making this prompt as the GenAI will draw context from the previous responses and give you a tailored response.

You will frame a prompt that can use the context from the previous steps and create the infrastructure requirements for your design.

▼ Click here for a sample prompt

```
Propose a set of infrastructure requirements for different stages of the data workflow based on the data architecture discussed her.
```

▼ Click here for a sample response

Based on the data architecture discussed for the online retail company, the infrastructure requirements for different stages of the data workflow can be outlined as follows:

1. Data collection:

    - Hardware: Servers or devices to collect data from various sources and transfer the data to the central SQL server.
    - Software: Data collection tools and scripts to automate the extraction and load the data into the central repository.
    - Networking: Reliable network connections to ensure the seamless data transfer between sources and the central repository.

2. Data transformation:

    - Hardware: Sufficient processing power and memory to handle data transformation tasks efficiently.
    - Software: Data transformation tools and ETL (Extract, Transform, Load) processes to clean, normalize, and standardize the data.
    - Networking: Fast and stable network connections to support data transformation processes.

3. Data modeling:

    - Hardware: Hardware resources to support database schema design and data modeling activities.
    - Software: Database management software to define relationships between data entities and optimize data storage.
    - Networking: Network connectivity to access and manage the database schema.

4. Data access:

    - Hardware: Devices and systems to access the central data repository for analysis and reporting.
    - Software: Data visualization tools, reporting software, and query interfaces for employees to access and analyze data.
    - Networking: Secure network connections to ensure data access controls and permissions are enforced.

5. Data backup and recovery:

    - Hardware: Backup storage devices or systems to store copies of data for disaster recovery purposes.
    - Software: Backup and recovery software to automate data backup processes and ensure data integrity.
    - Networking: Network connections to facilitate data backup and recovery operations.

6. Data security:

    - Hardware: Security appliances or devices to protect the data warehouse from unauthorized access.
    - Software: Encryption software, access control mechanisms, and data security tools to secure sensitive data.
    - Networking: Secure network configurations, firewalls, and monitoring tools to detect and prevent security breaches.

By addressing these infrastructure requirements at each stage of the data workflow, the online retail company can ensure the efficiency, security, and reliability of their data architecture and operations.

The ETL workflow

At this point, you can assume that the central data collection has taken place and the invoice details are available as a CSV file on remote server. You need to create an ETL pipeline that reads this file, cleans the data based on the client specifications and loads the data to a SQL server. Following specifications for this task have been shared.

1. The recorded data is available on the following URL:

   ```
   https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-AI0273EN-SkillsNetwork/labs/v1/m3/data/Project_
   ```

2. InvoiceNo starting with the character `C` is a credit entry and should be removed from the record before analysis.
3. StockCode values of `C2`, `D`, `M`, and `POST` correspond to Carraige, Discount, Manual and Postage entries, all of which are not required for our analysis.
4. There are a few entries where the CustomerID is missing. Such entries can be removed from the data before your analysis.
5. Load the final transaction record to an SQLite3 database `Invoice_Records` under the table `Purchase_transactions`.

Create a prompt and use it on the generative AI platform to generate a Python code that can create the required data processing pipeline.

▼ Click here for a sample prompt

```
For the transactional data table, create a python code that does the following.
1. Read the CSV data from a URL using Pandas
2. Remove entries where
   a. InvoiceNo starts with the character 'C'
   b. StockCode is either 'M', 'D', 'C2' or 'POST'
   c. CustomerID is missing.
   Print the length of the dataset before and after removal.
3. Load the final data to an SQLIte3 database named 'Invoice_Records' as a table 'Purchase_transactions'.
4. Run a sample query to display the first 5 rows of the table.
```

▼ Click here for a sample response

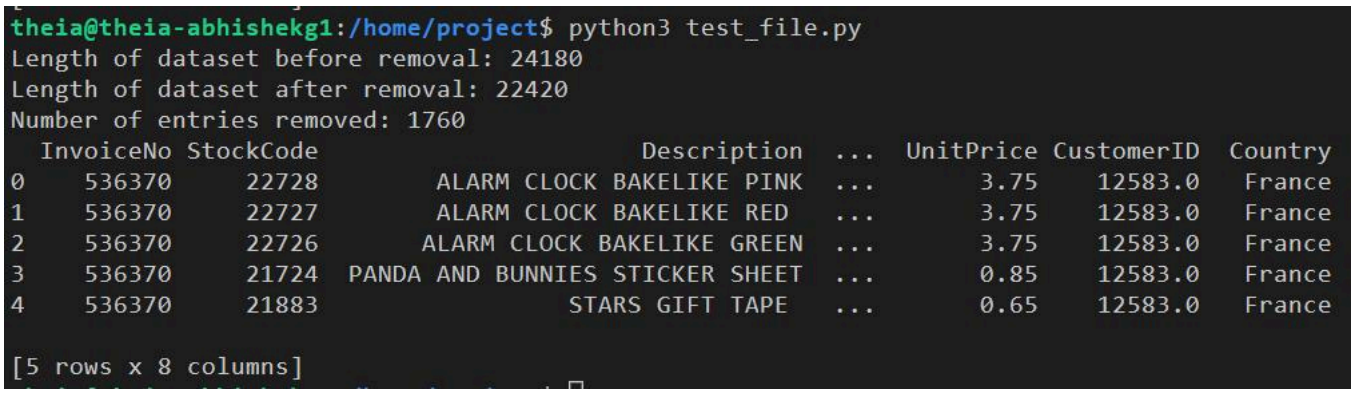
```
import pandas as pd
import sqlite3
# Read the CSV data from a URL
url = "insert_your_csv_url_here"
data = pd.read_csv(url)
# Print the length of the dataset before removal
print("Length of dataset before removal:", len(data))
# Remove entries that meet the specified conditions
initial_length = len(data)
data = data[~data['InvoiceNo'].str.startswith('C')]
data = data[~data['StockCode'].isin(['M', 'D', 'C2', 'POST'])]
data = data.dropna(subset=['CustomerID'])
# Print the length of the dataset after removal
print("Length of dataset after removal:", len(data))
print("Number of entries removed:", initial_length - len(data))
# Load the final data to an SQLite3 database
conn = sqlite3.connect('Invoice_Records.db')
data.to_sql('Purchase_transactions', conn, if_exists='replace', index=False)
# Run a sample query to display the first 5 rows of the table
query_result = pd.read_sql_query("SELECT * FROM Purchase_transactions LIMIT 5;", conn)
print(query_result)
conn.close()
```

▼ Sample content of testing code

```
import pandas as pd
import sqlite3
# Read the CSV data from a URL
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-AI0273EN-SkillsNetwork/labs/v1/m3/data/Pr
data = pd.read_csv(url)
# Print the length of the dataset before removal
print("Length of dataset before removal:", len(data))
# Remove entries that meet the specified conditions
```

```
initial_length = len(data)
data = data[~data['InvoiceNo'].str.startswith('C')]
data = data[~data['StockCode'].isin(['M', 'D', 'C2', 'POST'])]
data = data.dropna(subset=['CustomerID'])
# Print the length of the dataset after removal
print("Length of dataset after removal:", len(data))
print("Number of entries removed:", initial_length - len(data))
# Load the final data to an SQLite3 database
conn = sqlite3.connect('Invoice_Records.db')
data.to_sql('Purchase_transactions', conn, if_exists='replace', index=False)
# Run a sample query to display the first 5 rows of the table
query_result = pd.read_sql_query("SELECT * FROM Purchase_transactions LIMIT 5;", conn)
print(query_result)
conn.close()
```

▼ Click here for a sample output



```
theia@theia-abhishekg1:/home/project$ python3 test_file.py
Length of dataset before removal: 24180
Length of dataset after removal: 22420
Number of entries removed: 1760
  InvoiceNo StockCode                          Description  ...  UnitPrice CustomerID  Country
0    536370     22728       ALARM CLOCK BAKELIKE PINK   ...       3.75    12583.0   France
1    536370     22727       ALARM CLOCK BAKELIKE RED    ...       3.75    12583.0   France
2    536370     22726      ALARM CLOCK BAKELIKE GREEN   ...       3.75    12583.0   France
3    536370     21724  PANDA AND BUNNIES STICKER SHEET  ...       0.85    12583.0   France
4    536370     21883                 STARS GIFT TAPE   ...       0.65    12583.0   France

[5 rows x 8 columns]
```

Querying the database

After the data is available on the central data repository, you can use SQL queries to extract the data directly into your Python coding interface. For the next part of your project, you are required to extract the data of a specific country–let's use Germany. Use the generative AI model to create a code snippet that you can add to your previous code in the testing interface and that will run a SQL query on the data to extract the details transactions for the specified country.

▼ Click here for a sample prompt

```
Create a python code that will extract all records from the table 'Purchase_transactions' under the database 'Invoice_records', such
```

▼ Click here for a sample response

To extract all records from the 'Purchase_transactions' table under the 'Invoice_records' database where the Country parameter is set to Germany and save the resulting records in a dataframe in Python, you can use the following code snippet:

```
import sqlite3
import pandas as pd
# Connect to the database
conn = sqlite3.connect('Invoice_records.db')
# Execute a query to select all records from the 'Purchase_transactions' table with Country set to Germany
query = "SELECT * FROM Purchase_transactions WHERE Country = 'Germany'"
df = pd.read_sql(query, conn)
# Print the dataframe
print(df)
# Close the connection
conn.close()
```

This code snippet specifically filters the records where the Country is set to Germany and saves the results in a pandas dataframe `df`. Ensure you have the pandas library installed in your Python environment to run this code successfully.

▼ Sample code for testing

```
import pandas as pd
import sqlite3
# Read the CSV data from a URL
```

```
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-AI0273EN-SkillsNetwork/labs/v1/m3/data/Pr
data = pd.read_csv(url)
# Print the length of the dataset before removal
print("Length of dataset before removal:", len(data))
# Remove entries that meet the specified conditions
initial_length = len(data)
data = data[~data['InvoiceNo'].str.startswith('C')]
data = data[~data['StockCode'].isin(['M', 'D', 'C2', 'POST'])]
data = data.dropna(subset=['CustomerID'])
# Print the length of the dataset after removal
print("Length of dataset after removal:", len(data))
print("Number of entries removed:", initial_length - len(data))
# Load the final data to an SQLite3 database
conn = sqlite3.connect('Invoice_Records.db')
data.to_sql('Purchase_transactions', conn, if_exists='replace', index=False)
# Run a sample query to display the first 5 rows of the table
query_result = pd.read_sql_query("SELECT * FROM Purchase_transactions LIMIT 5;", conn)
print(query_result)
query = "SELECT * FROM Purchase_transactions WHERE Country IN ('Germany')"
records = pd.read_sql(query, conn)
# Print the extracted records
print(records)
conn.close()
```

▼ Click here for a sample output

```
theia@theia-abhishekg1:/home/project$ python3 test_file.py
Length of dataset before removal: 24180
Length of dataset after removal: 22420
Number of entries removed: 1760
  InvoiceNo StockCode                          Description  Quantity         InvoiceDate  UnitPrice  Cu
0    536370     22728       ALARM CLOCK BAKELIKE PINK         24  01-12-2010 08:45       3.75
1    536370     22727       ALARM CLOCK BAKELIKE RED          24  01-12-2010 08:45       3.75
2    536370     22726      ALARM CLOCK BAKELIKE GREEN         12  01-12-2010 08:45       3.75
3    536370     21724  PANDA AND BUNNIES STICKER SHEET        12  01-12-2010 08:45       0.85
4    536370     21883                STARS GIFT TAPE          24  01-12-2010 08:45       0.65
     InvoiceNo StockCode                          Description  Quantity         InvoiceDate  UnitPr
0       536527     22809             SET OF 6 T-LIGHTS SANTA         6  01-12-2010 13:04       2
1       536527     84347  ROTATING SILVER ANGELS T-LIGHT HLDR         6  01-12-2010 13:04       2
2       536527     84945   MULTI COLOUR SILVER T-LIGHT HOLDER        12  01-12-2010 13:04       0
3       536527     22242        5 HOOK HANGER MAGIC TOADSTOOL        12  01-12-2010 13:04       1
4       536527     22244           3 HOOK HANGER MAGIC GARDEN        12  01-12-2010 13:04       1
...        ...       ...                                  ...       ...               ...
8654    581578     22993         SET OF 4 PANTRY JELLY MOULDS        12  09-12-2011 12:16       1
8655    581578     22907     PACK OF 20 NAPKINS PANTRY DESIGN        12  09-12-2011 12:16       0
8656    581578     22908        PACK OF 20 NAPKINS RED APPLES        12  09-12-2011 12:16       0
8657    581578     23215     JINGLE BELL HEART ANTIQUE SILVER        12  09-12-2011 12:16       2
8658    581578     22736          RIBBON REEL MAKING SNOWMEN         10  09-12-2011 12:16       1

[8659 rows x 8 columns]
```

Data analysis and data mining

One of the most relevant and important techniques for analyzing transactional data is association rule mining. You are required to implement an Apriori algorithm to mine association rules from the data extracted in the previous step. As a result, you will be able to identify the items that have the most likelihood to be purchased together. This information is necessary for the company to efficiently develop their marketing and advertising strategies.

You can use generative AI to create the code for implementation. You can write a prompt that creates code for the implementation of Apriori algorithm for association rule mining on the extracted data frame.

Consider the following prompt that describes the different steps involved in implementing an Apriori algorithm.

```
For the data frame extracted here, write a python code to execute the apriori algorithm and extract association rules for the given
1. Group the records by InvoiceID and Description, along with their total quantities
2. Unpack the data into a table, making the InvoiceNo, the row indexes, Unique descriptions as column indexes and the total quantiti
3. Apply one-hot encoding on this table, making the value True if the Item description existed in the invoice and False if it didn't
4. Perfrom Apriori algorithm on this data and extract the required association rules.
```

You can expect to see a respnese similar to the following sample response:

▼ Click here for a sample response

To execute the Apriori algorithm and extract association rules for the set of invoices extracted from the 'Purchase_transactions' table under the 'Invoice_records' database, you can follow these steps:

1. Group the records by InvoiceID and Description, along with their total quantities:

```
import sqlite3
import pandas as pd
# Connect to the database
conn = sqlite3.connect('Invoice_records.db')
# Execute a query to select all records from the 'Purchase_transactions' table
query = "SELECT InvoiceID, Description, SUM(Quantity) AS TotalQuantity FROM Purchase_transactions GROUP BY InvoiceID, Description"
df_grouped = pd.read_sql_query(query, conn)
# Close the connection
conn.close()
```

2. Unpack the data into a table with the InvoiceNo" items as row indexes, Unique descriptions as column indexes, and total quantities as values:

```
df_pivot = df_grouped.pivot(index='InvoiceNo', columns='Description', values='TotalQuantity').fillna(0)
```

3. Apply one-hot encoding on this table:

```
df_encoded = df_pivot.applymap(lambda x: 1 if x > 0 else 0)
```

4. Perform the Apriori algorithm using this data and extract the association rules:

```
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
# Apply the Apriori algorithm to find frequent itemsets
frequent_itemsets = apriori(df_encoded, min_support=0.1, use_colnames=True)
# Generate association rules from the frequent itemsets
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
# Print the association rules
print(rules)
```

You can comment out all previous print statements and combine these code snippets with the existing test code file, making completed code for reference.

▼ Click here for sample final code

```
import pandas as pd
import sqlite3
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
# Read the CSV data from a URL
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-AI0273EN-SkillsNetwork/labs/v1/m3/data/Pr
data = pd.read_csv(url)
# Print the length of the dataset before removal
#print("Length of data set before removal:", len(data))
# Remove entries that meet the specified conditions
initial_length = len(data)
data = data[~data['InvoiceNo'].str.startswith('C')]
data = data[~data['StockCode'].isin(['M', 'D', 'C2', 'POST'])]
data = data.dropna(subset=['CustomerID'])
# Print the length of the data set after removal
#print("Length of data set after removal:", len(data))
#print("Number of entries removed:", initial_length - len(data))
# Load the final data to an SQLite3 database
conn = sqlite3.connect('Invoice_Records.db')
data.to_sql('Purchase_transactions', conn, if_exists='replace', index=False)
```

```
# Run a sample query to display the first 5 rows of the table
query_result = pd.read_sql_query("SELECT * FROM Purchase_transactions LIMIT 5;", conn)
#print(query_result)
query = "SELECT * FROM Purchase_transactions WHERE Country IN ('Germany')"
records = pd.read_sql(query, conn)
# Print the extracted records
#print(records)
# Execute a query to select all records from the 'Purchase_transactions' table
query = "SELECT InvoiceNo, Description, SUM(Quantity) AS TotalQuantity FROM Purchase_transactions GROUP BY InvoiceNo, Description"
df_grouped = pd.read_sql(query, conn)
df_pivot = df_grouped.pivot(index='InvoiceNo', columns='Description', values='TotalQuantity').fillna(0)
df_encoded = df_pivot.applymap(lambda x: 1 if x > 0 else 0)
# Apply the Apriori algorithm to find frequent itemsets
frequent_itemsets = apriori(df_encoded, min_support=0.05, use_colnames=True)
# Generate association rules from the frequent itemsets
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
# Sort the association rules in descending order of confidence
rules = rules.sort_values(by='confidence', ascending=False)
# Print the association rules
print(rules[['antecedents','consequents','confidence']])
conn.close()
```

▼ Click here for sample output

```
theia@theia-abhishekg1:/home/project$ python3 test_file.py
/home/project/test_file.py:43: FutureWarning: DataFrame.applymap has been deprecated. Use DataFra
  df_encoded = df_pivot.applymap(lambda x: 1 if x > 0 else 0)
/home/theia/.local/lib/python3.10/site-packages/mlxtend/frequent_patterns/fpcommon.py:109: Deprec
es result in worse computationalperformance and their support might be discontinued in the future
  warnings.warn(
                                    antecedents                                consequents  conf
7  (SET/6 RED SPOTTY PAPER CUPS, SET/20 RED RETRO...        (SET/6 RED SPOTTY PAPER PLATES)    0.
6  (SET/6 RED SPOTTY PAPER PLATES, SET/20 RED RET...          (SET/6 RED SPOTTY PAPER CUPS)    0.
2                   (SET/6 RED SPOTTY PAPER PLATES)          (SET/6 RED SPOTTY PAPER CUPS)    0.
3                     (SET/6 RED SPOTTY PAPER CUPS)        (SET/6 RED SPOTTY PAPER PLATES)    0.
4  (PLASTERS IN TIN SPACEBOY, PLASTERS IN TIN CIR...     (PLASTERS IN TIN WOODLAND ANIMALS)    0.
0               (ROUND SNACK BOXES SET OF 4 FRUITS )  (ROUND SNACK BOXES SET OF4 WOODLAND )    0.
5  (SET/6 RED SPOTTY PAPER PLATES, SET/6 RED SPOT...  (SET/20 RED RETROSPOT PAPER NAPKINS )    0.
1                   (SET/6 RED SPOTTY PAPER PLATES)  (SET/20 RED RETROSPOT PAPER NAPKINS )    0.
```

You can infer that if an item from the antecedent column is picked, then it can be said with the shown value of confidence that the corresponding consequent is also going to be picked for purchase in the same invoice.

Conclusion

Congratulations on completing this project!

You are now trained in using generative AI for end-to-end data engineering applications, including but not limited to:

- Data architecture design
- Data warehouse and schema design
- Infrastructure requirements determination
- ETL pipeline integration
- Querying databases
- Data analysis and mining

Author(s)

Abhishek Gagneja