**Semi-Structure Data Analytics**

The provided code demonstrates a data pipeline that processes sales data. Let's break down each step and explain its purpose.

## Ingestion

The first step in any data pipeline is to ingest or load the data. In this code, the data is ingested using the json module:

import json  
  
with open("sample\_supplies.sales.json", "r") as file:  
 data = json.load(file)

This code reads a JSON file named sample\_supplies.sales.json and loads its content into the data variable.

## Transformation

Once the data is ingested, it often needs to be transformed to be suitable for analysis. The code uses various transformations to extract insights from the sales data.

## Code for Each Query with Explanations

### 1. Top 10 Products by Sales

from collections import defaultdict  
  
product\_sales = defaultdict(float)  
  
for record in data:  
 for item in record['items']:  
 product\_sales[item['name']] += float(item['price']['$numberDecimal']) \* item['quantity']  
  
top\_10\_products = sorted(product\_sales.items(), key=lambda x: x[1], reverse=True)[:10]

This code calculates the total sales for each product. It uses a defaultdict to accumulate sales for each product. The final list, top\_10\_products, contains the top 10 products sorted by their total sales.

### 2. Number of Unique Products

unique\_products = set(item['name'] for record in data for item in record['items'])  
num\_unique\_products = len(unique\_products)

This code calculates the number of unique products in the dataset. It uses a set to store unique product names and then calculates the length of this set.

### 3. Top 3 Products by Store

store\_product\_sales = defaultdict(lambda: defaultdict(float))  
  
for record in data:  
 location = record['storeLocation']  
 for item in record['items']:  
 store\_product\_sales[location][item['name']] += float(item['price']['$numberDecimal']) \* item['quantity']  
  
top\_3\_products\_by\_store = {}  
for location, products in store\_product\_sales.items():  
 top\_3\_products\_by\_store[location] = sorted(products.items(), key=lambda x: x[1], reverse=True)[:3]

This code calculates the top 3 products by sales for each store. It uses nested defaultdicts to accumulate sales for each product in each store. The final dictionary, top\_3\_products\_by\_store, contains the top 3 products for each store location.

### 4. Store Rankings by Sales

store\_sales = defaultdict(float)  
  
for record in data:  
 location = record['storeLocation']  
 for item in record['items']:  
 store\_sales[location] += float(item['price']['$numberDecimal']) \* item['quantity']  
  
store\_rankings = sorted(store\_sales.items(), key=lambda x: x[1], reverse=True)

This code calculates the total sales for each store and then ranks the stores based on their total sales. The final list, store\_rankings, contains stores sorted by their total sales.

### 5. Purchases by Gender and Method

purchase\_by\_gender = defaultdict(lambda: defaultdict(int))  
  
for record in data:  
 gender = record['customer']['gender']  
 method = record['purchaseMethod']  
 purchase\_by\_gender[gender][method] += 1  
  
purchase\_method\_table = {  
 "Online": {  
 "Male": purchase\_by\_gender['M']['Online'],  
 "Female": purchase\_by\_gender['F']['Online']  
 },  
 "In store": {  
 "Male": purchase\_by\_gender['M']['In store'],  
 "Female": purchase\_by\_gender['F']['In store']  
 }  
}  
  
df = pd.DataFrame(purchase\_method\_table)

This code calculates the number of purchases made by each gender using different purchase methods (Online or In store). The final DataFrame, df, represents the purchase method distribution across genders.

### 6. Monthly Sales

from datetime import datetime  
  
monthly\_sales = defaultdict(float)  
  
for record in data:  
 date = datetime.fromisoformat(record['saleDate']['$date'][:-1])  
 month\_key = f"{date.year}-{date.month:02}"  
 for item in record['items']:  
 monthly\_sales[month\_key] += float(item['price']['$numberDecimal']) \* item['quantity']  
  
sorted\_monthly\_sales = sorted(monthly\_sales.items(), key=lambda x: x[0])

This code calculates the total sales for each month. It uses the datetime module to extract the month and year from the sale date and then accumulates sales for each month. The final list, sorted\_monthly\_sales, contains monthly sales sorted by month.

## Preparation

Before processing the data, it's essential to prepare it by installing necessary libraries and importing required modules. In this code, the pandas library is installed using !pip install pandas and then imported to process the data into a DataFrame.

## Processing

The main processing in this code involves aggregating sales data based on various criteria, such as product, store location, purchase method, and date. The defaultdict from the collections module is extensively used to facilitate this aggregation.

## Result

The final result of the data pipeline is a series of insights derived from the sales data, such as the top-selling products, store rankings, and monthly sales trends. These insights are represented in various data structures, including lists, dictionaries, and DataFrames.