**Data Analyst Task**

Data Sources: Customer.xls, Orders.csv, Shipping.json

Initial Observations Made:

* The first name column in customer dataset has special characters in it.
* There are no date columns like order\_date or shipping\_date in any of the data sets.
* The referential integrity is maintained across all three data sets.
* All amount and age values are non-negative and fall within a reasonable range.
* There are no duplicates in any of the PK columns of the given datasets.
* Same item in order dataset has 2 different prices. (Mousepad: 250, 200)
* Around 94 orders are not having shipping information in shipping dataset.

Objectives:

**Step 1: Process to check accuracy, completeness and reliability of the data sources**

import pandas as pd

df\_shipping = pd.read\_json('/Users/akshayadhanraj/Python/DataAnalystTask/Shipping.json')

df\_order = pd.read\_csv('/Users/akshayadhanraj/Python/DataAnalystTask/Order.csv')

df\_customer = pd.read\_excel('/Users/akshayadhanraj/Python/DataAnalystTask/Customer.xls', sheet\_name='atkoe-u250m')

#print(df\_shipping.info(),df\_order.info(),df\_customer.info())

print('\n--- Step 1: Completeness ---\n')

# Check NULL values in any of the fields

print('Order missing values:\n', df\_order.isnull().sum())

print('Customer missing values:\n', df\_customer.isnull().sum())

print('Shipping missing values:\n', df\_shipping.isnull().sum())

print('\n--- Step 2: Accuracy ---\n')

# Amounts should be positive

print('Orders with non-positive amount: ', df\_order[df\_order['Amount'] <= 0].count())

print('Customers with non-positive age: ', df\_customer[df\_customer['Age'] <= 0].count())

# Age field should not be more than 100

print('Customer age greater than 100: ', df\_customer[df\_customer['Age'] > 100].count())

# Records containing special characters in First & Last name

print('Customer first name with special characters: \n', df\_customer[df\_customer['First'].str.contains(r'[^a-zA-Z0-9\s]', regex=True)])

print('Customer last name with special characters: \n', df\_customer[df\_customer['Last'].str.contains(r'[^a-zA-Z0-9\s]', regex=True)])

# Check foreign key consistency

print('Orders with missing customer info: ', df\_order[~df\_order['Customer\_ID'].isin(df\_customer['Customer\_ID'])].count())

print('Shipping with missing customer info: ', df\_shipping[~df\_shipping['Customer\_ID'].isin(df\_customer['Customer\_ID'])].count())

print('\n--- Step 3: Reliability ---\n')

# Check for duplicate records

print('Orders with duplicate values: ',df\_order.duplicated().sum())

print('Customer with duplicate values: ',df\_customer.duplicated().sum())

print('Shipping with duplicate values: ',df\_shipping.duplicated().sum())

**Step 2: Create Data Model**

A star schema is created to relate the tables.

* Dim\_Customer is derived from the source file Customer.xls.
  + The First and Last column are mapped as first\_name and last\_name
  + Age is transformed as date column with the formula current\_date – age in years.
  + Housekeeping columns created\_date and modified\_date is added.
* Dim\_Shipping is derived from source file Shipping.json
  + The shipping status is a direct pull from source.
  + The shipping\_date created can be considered as a place holder for future or it can be implemented as the date at which the file arrived.
  + Housekeeping columns created\_date and modified\_date is added.
* Dim Product is derived from the source file Order.csv
  + The product\_id is surrogate key.
  + Item column is mapped as product\_name.
  + Housekeeping columns created\_date and modified\_date is added.
* Dim Date is one one-time derived table
  + The date\_id is PK derived from full date.
  + Useful for timeseries analysis.
* Fact Order is derived from the source file Order.csv
  + Foreign keys from other tables are populated
  + The order\_date created can be considered as a place holder for future or it can be implemented as the date at which the file arrived.
  + Housekeeping columns created\_date and modified\_date is added.

A diagram of a data flow

Description automatically generated with medium confidence

**Step 3: Technical story to implemented by a Data & QA Engineer**

*Data Engineer User Story:*

As a Data Engineer, I want to ingest customer data from the source file Customer.xls into DimCustomer table so it is available for downstream cleansing and reporting process.

Target DDL:

CREATE TABLE DimCustomer (

customer\_id VARCHAR (50),

first\_name VARCHAR (100),

last\_name VARCHAR (100),

date\_of\_birth DATE,

age\_category VARCHAR (10),

country VARCHAR (100),

created\_date DATETIME DEFAULT GETDATE ()

modified\_date DATETIME DEFAULT GETDATE ()

)

Transformation logic:

* Direct pull from source to target.
* Rename source columns first as first\_name and last as last\_name before mapping
* Transform age column as date\_of\_birth column by subtracting age with current date
* Transform age column to age\_category by splitting it as >=30 and <30
* Add housekeeping column – created\_date & modified\_date

*QA Engineer User Story:*

As a QA Engineer, I want to validate the completeness, accuracy and reliability of the data loaded into target system.

Acceptance Criteria:

* Row counts must match between source and target.
* Check for NULLs in mandatory fields - customer\_id
* Check for duplicate rows
* Check for special characters in name field
* Verify transformation rules

**Business Reporting Requirements**

1. *The total amount spent and the country for the Pending delivery status for each country.*

SELECT c.country, SUM(o.amount) AS total\_amount\_spent FROM fact\_order o

LEFT JOIN dim\_customer c ON o.customer\_id = c.customer\_id

LEFT JOIN dim\_shipping s ON o.shipping\_id = s.shipping\_id

WHERE status = 'Pending' GROUP BY country;

1. *The total number of transactions, total quantity sold, and total amount spent for each customer, along with the product details.*

WITH customer\_item\_agg AS (

SELECT

o.customer\_id,

p.product\_name,

COUNT(\*) AS total\_transactions,

COUNT(\*) AS total\_quantity\_sold,

SUM(o.amount) AS total\_amount\_spent

FROM fact\_order o

LEFT JOIN dim\_product p ON o.product\_id = p.product\_id

GROUP BY customer\_id, product\_name

)

SELECT

c.customer\_id,

c.first\_name,

c.last\_name,

cia.product\_name,

cia.total\_transactions,

cia.total\_quantity\_sold,

cia.total\_amount\_spent

FROM customer\_item\_agg cia

JOIN dim\_customer c ON c.customer\_id = cia.customer\_id

ORDER BY c.customer\_id, cia.product\_name;

1. *The maximum product purchased for each country.*

SELECT country, product\_name

FROM (

SELECT c.country, p.product\_name, COUNT(o.customer\_id) AS total\_quantity,

DENSE\_RANK() OVER (PARTITION BY c.country ORDER BY COUNT(o.customer\_id) DESC) AS rn

FROM fact\_order o

LEFT JOIN dim\_customer c ON o.customer\_id = c.customer\_id

LEFT JOIN dim\_product p ON o.product\_id = p.product\_id

GROUP BY c.country, p.product\_name

) a WHERE rn = 1;

1. *The most purchased product based on the age category less than 30 and above 30.*

WITH customer\_age AS (

SELECT

c.customer\_id,

c.age\_category,

p.product\_name

FROM fact\_order o

JOIN dim\_customer c ON o.customer\_id = c.customer\_id

LEFT JOIN dim\_product p ON o.product\_id = p.product\_id

),

product\_quantity AS (

SELECT

age\_category,

product\_name,

COUNT(\*) AS total\_quantity\_sold

FROM customer\_age

GROUP BY age\_category, product\_name

),

ranked\_products AS (

SELECT

\*,

ROW\_NUMBER() OVER (PARTITION BY age\_category ORDER BY total\_quantity\_sold DESC) AS rn

FROM product\_quantity

)

SELECT age\_category, product\_name, total\_quantity\_sold

FROM ranked\_products

WHERE rn = 1;

1. *The country that had minimum transactions and sales amount*

WITH country\_totals AS (

SELECT

c.country,

COUNT(\*) AS total\_transactions,

SUM(o.amount) AS total\_sales

FROM fact\_order o

JOIN dim\_customer c ON o.customer\_id = c.customer\_id

GROUP BY c.country

)

SELECT \*

FROM country\_totals

ORDER BY total\_transactions ASC, total\_sales ASC

LIMIT 1;