TASK 9

September 15, 2025

[1]:
```
import pandas as pd
from glob import glob
```

[4]:
```
import os
print(os.getcwd())
```

```
/home/01e8c0d6-55b3-4404-abea-629433cfc2c8
```

[7]:
```
%pip install pandas numpy matplotlib seaborn scikit-learn plotly missingno␣
↪--quiet

import os, glob
print("Working directory:", os.getcwd())
print("CSV files found here:")
for f in sorted(glob.glob("*.csv")):
    print("  ", f)
```

```
Note: you may need to restart the kernel to use updated packages.
Working directory: /home/01e8c0d6-55b3-4404-abea-629433cfc2c8
CSV files found here:
   gender.csv
   olist_customers_dataset.csv
   olist_geolocation_dataset.csv
   olist_order_items_dataset.csv
   olist_order_payments_dataset.csv
   olist_order_reviews_dataset.csv
   olist_orders_dataset.csv
   olist_products_dataset.csv
   olist_sellers_dataset.csv
   onlineretail.csv
   product_category_name_translation.csv
   test.csv
   train.csv
```

[8]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
import glob, os

sns.set_style("whitegrid")
pd.set_option('display.max_columns', 80)
pd.set_option('display.width', 120)
```

[9]:
```
orders = pd.read_csv("olist_orders_dataset.csv", low_memory=False)
order_items = pd.read_csv("olist_order_items_dataset.csv", low_memory=False)
products = pd.read_csv("olist_products_dataset.csv", low_memory=False)
customers = pd.read_csv("olist_customers_dataset.csv", low_memory=False)
sellers = pd.read_csv("olist_sellers_dataset.csv", low_memory=False)
payments = pd.read_csv("olist_order_payments_dataset.csv", low_memory=False)
reviews = pd.read_csv("olist_order_reviews_dataset.csv", low_memory=False)


if os.path.exists("olist_geolocation_dataset.csv"):
    geolocation = pd.read_csv("olist_geolocation_dataset.csv", low_memory=False)
if os.path.exists("product_category_name_translation.csv"):
    category_translation = pd.read_csv("product_category_name_translation.csv",␣
 ↪low_memory=False)

print("orders:", orders.shape, "order_items:", order_items.shape, "products:",␣
 ↪products.shape)
orders.head(2)
```

```
orders: (99441, 8) order_items: (112650, 7) products: (32951, 9)
```

[9]:
```
                          order_id                         customer_id
order_status order_purchase_timestamp  \
0  e481f51cbdc54678b7cc49136f2d6af7  9ef432eb6251297304e76186b10a928d
delivered      2017-10-02 10:56:33
1  53cdb2fc8bc7dce0b6741e2150273451  b0830fb4747a6c6d20dea0b8c802d7ef
delivered      2018-07-24 20:41:37

     order_approved_at order_delivered_carrier_date
order_delivered_customer_date order_estimated_delivery_date
0  2017-10-02 11:07:15          2017-10-04 19:55:00           2017-10-10
21:25:13           2017-10-18 00:00:00
1  2018-07-26 03:24:27          2018-07-26 14:31:00           2018-08-07
15:27:45           2018-08-13 00:00:00
```

[10]:
```
print(orders.info())
print(order_items.info())
print(products.columns.tolist())

date_cols =␣
 ↪["order_purchase_timestamp","order_approved_at","order_delivered_carrier_date",
```

```
            "order_delivered_customer_date","order_estimated_delivery_date"]
for c in date_cols:
    if c in orders.columns:
        orders[c] = pd.to_datetime(orders[c], errors="coerce")

if 'order_status' in orders.columns:
    print(orders['order_status'].value_counts())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99441 entries, 0 to 99440
Data columns (total 8 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   order_id                       99441 non-null  object
 1   customer_id                    99441 non-null  object
 2   order_status                   99441 non-null  object
 3   order_purchase_timestamp       99441 non-null  object
 4   order_approved_at              99281 non-null  object
 5   order_delivered_carrier_date   97658 non-null  object
 6   order_delivered_customer_date  96476 non-null  object
 7   order_estimated_delivery_date  99441 non-null  object
dtypes: object(8)
memory usage: 6.1+ MB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 112650 entries, 0 to 112649
Data columns (total 7 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   order_id             112650 non-null  object
 1   order_item_id        112650 non-null  int64
 2   product_id           112650 non-null  object
 3   seller_id            112650 non-null  object
 4   shipping_limit_date  112650 non-null  object
 5   price                112650 non-null  float64
 6   freight_value        112650 non-null  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 6.0+ MB
None
['product_id', 'product_category_name', 'product_name_lenght',
'product_description_lenght', 'product_photos_qty', 'product_weight_g',
'product_length_cm', 'product_height_cm', 'product_width_cm']
order_status
delivered      96478
shipped         1107
canceled         625
unavailable      609
```

```
invoiced         314
processing       301
created            5
approved           2
Name: count, dtype: int64
```

[11]:
```
orders.columns = orders.columns.str.strip().str.lower()
order_items.columns = order_items.columns.str.strip().str.lower()
products.columns = products.columns.str.strip().str.lower()
customers.columns = customers.columns.str.strip().str.lower()
payments.columns = payments.columns.str.strip().str.lower()
reviews.columns = reviews.columns.str.strip().str.lower()
sellers.columns = sellers.columns.str.strip().str.lower()


orders = orders.drop_duplicates()
order_items = order_items.drop_duplicates()


print("Missing in orders:\n", orders.isnull().sum().
 ↪sort_values(ascending=False).head(8))
```

```
Missing in orders:
 order_delivered_customer_date    2965
order_delivered_carrier_date     1783
order_approved_at                 160
order_id                            0
order_purchase_timestamp            0
order_status                        0
customer_id                         0
order_estimated_delivery_date       0
dtype: int64
```

[12]:
```
order_totals = order_items.groupby("order_id").agg(
    n_items = ("order_item_id","count"),
    items_value = ("price","sum"),
    freight_value = ("freight_value","sum")
).reset_index()
order_totals["order_revenue"] = order_totals["items_value"] +␣
 ↪order_totals["freight_value"]


payments_by_order = payments.groupby("order_id").agg(
    payment_count = ("payment_type","count"),
    payment_value_sum = ("payment_value","sum"),
    payment_types = ("payment_type", lambda x: ", ".join(x.unique()))
).reset_index()
```

```
reviews_by_order = reviews.groupby("order_id").agg(
    review_score = ("review_score","mean"),
    review_count = ("review_comment_message","count")
).reset_index()


df = orders.merge(order_totals, on="order_id", how="left")
df = df.merge(customers, on="customer_id", how="left")
df = df.merge(payments_by_order, on="order_id", how="left")
df = df.merge(reviews_by_order, on="order_id", how="left")


df[['order_id','order_purchase_timestamp','order_status','order_revenue','payment_value_sum','
 ↪head()
```

[12]:
```
                          order_id order_purchase_timestamp order_status
order_revenue  payment_value_sum  \
0  e481f51cbdc54678b7cc49136f2d6af7      2017-10-02 10:56:33    delivered
38.71              38.71
1  53cdb2fc8bc7dce0b6741e2150273451      2018-07-24 20:41:37    delivered
141.46             141.46
2  47770eb9100c2d0c44946d9cf07ec65d      2018-08-08 08:38:49    delivered
179.12             179.12
3  949d5b44dbf5de918fe9c16f97b45f8a      2017-11-18 19:28:06    delivered
72.20              72.20
4  ad21c59c0840e6cb83a9ceb5573f8159      2018-02-13 21:18:39    delivered
28.62              28.62

   review_score
0           4.0
1           4.0
2           5.0
3           5.0
4           5.0
```

[13]:
```
if 'order_delivered_customer_date' in df.columns and 'order_purchase_timestamp'␣
 ↪in df.columns:
    df['delivery_time_days'] = (df['order_delivered_customer_date'] -␣
 ↪df['order_purchase_timestamp']).dt.days


if 'order_approved_at' in df.columns:
    df['days_to_approve'] = (df['order_approved_at'] -␣
 ↪df['order_purchase_timestamp']).dt.days
```

```
if 'order_status' in df.columns and 'delivered' in df['order_status'].unique():
    df_sales = df[df['order_status'] == 'delivered'].copy()
else:
    df_sales = df.copy()

df_sales['order_month'] = df_sales['order_purchase_timestamp'].dt.to_period('M')
df_sales.head(2)
```

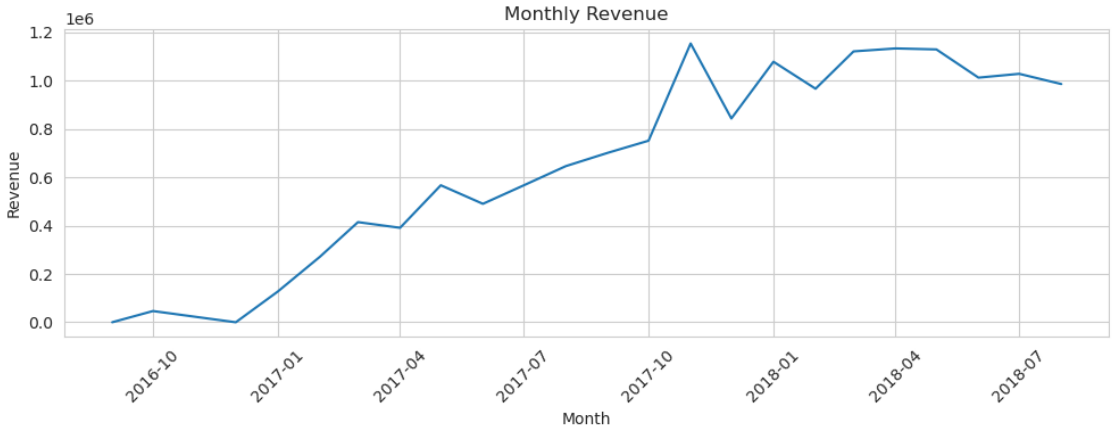
[13]:
```
                           order_id                        customer_id
order_status order_purchase_timestamp  \
0  e481f51cbdc54678b7cc49136f2d6af7  9ef432eb6251297304e76186b10a928d
delivered      2017-10-02 10:56:33
1  53cdb2fc8bc7dce0b6741e2150273451  b0830fb4747a6c6d20dea0b8c802d7ef
delivered      2018-07-24 20:41:37

    order_approved_at order_delivered_carrier_date order_delivered_customer_date
order_estimated_delivery_date  \
0 2017-10-02 11:07:15          2017-10-04 19:55:00           2017-10-10 21:25:13
2017-10-18
1 2018-07-26 03:24:27          2018-07-26 14:31:00           2018-08-07 15:27:45
2018-08-13

   n_items  items_value  freight_value  order_revenue
customer_unique_id  customer_zip_code_prefix  \
0      1.0        29.99           8.72          38.71
7c396fd4830fd04220f754e42b4e5bff                       3149
1      1.0       118.70          22.76         141.46
af07308b275d755c9edb36a90c618231                      47813

  customer_city customer_state  payment_count  payment_value_sum
payment_types  review_score  review_count  \
0     sao paulo             SP            3.0              38.71  credit_card,
voucher           4.0           1.0
1     barreiras             BA            1.0             141.46
boleto           4.0           1.0

   delivery_time_days  days_to_approve order_month
0                 8.0              0.0     2017-10
1                13.0              1.0     2018-07
```

[14]:
```
oi = order_items.merge(products[['product_id','product_category_name']],␣
 ↪on='product_id', how='left')
top_products = oi.groupby("product_id").agg(revenue=("price","sum"),␣
 ↪n_orders=("order_id","nunique")).reset_index()
top_products = top_products.sort_values("revenue", ascending=False).head(10)
top_products
```

[14]:

	product_id	revenue	n_orders
24086	bb50f2e236e5eea0100680137654686c	63885.00	187
14068	6cdd53843498f92890544667809f1595	54730.20	151
27613	d6160fb7873f184099d9bc95e30376af	48899.34	35
27039	d1c427060a0f73f6b889a5c7c61f2ac4	47214.51	323
19742	99a4788cb24856965c36a24e339b6058	43025.56	467
8051	3dd2a17168ec895c781a9191c1e95ad7	41082.60	255
4996	25c38557cf793876c5abdd5931f922db	38907.32	38
12351	5f504b3a1c75b73d6151be81eb05bdc9	37733.90	63
10867	53b36df67ebb7c41585e8d54d6772e08	37683.42	306
22112	aca2eb7d00ea1a7b8ebd4e68314663af	37608.90	431

[15]:
```
monthly = df_sales.groupby(df_sales['order_purchase_timestamp'].dt.
  ↪to_period('M')).agg(month_revenue=("order_revenue","sum")).reset_index()
monthly['order_purchase_timestamp'] = monthly['order_purchase_timestamp'].dt.
  ↪to_timestamp()
plt.figure(figsize=(10,4))
plt.plot(monthly['order_purchase_timestamp'], monthly['month_revenue'])
plt.title("Monthly Revenue")
plt.ylabel("Revenue")
plt.xlabel("Month")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[16]:
```
cust_orders = df_sales.groupby("customer_id").agg(
    n_orders=("order_id","nunique"),
    lifetime_value=("order_revenue","sum")
).reset_index()
repeat_rate = (cust_orders['n_orders'] > 1).mean()
print(f"Repeat purchase rate: {repeat_rate:.2%}")
print("Average Order Value (AOV):", df_sales['order_revenue'].mean())
```

```
Repeat purchase rate: 0.00%
Average Order Value (AOV): 159.82683876116837
```

[17]:
```
seller_revenue = order_items.
 ↪merge(products[['product_id','product_category_name']], on='product_id',␣
 ↪how='left')
seller_revenue = seller_revenue.groupby("seller_id").
 ↪agg(revenue=("price","sum")).reset_index().sort_values("revenue",␣
 ↪ascending=False).head(10)
seller_revenue
```

[17]:

	seller_id	revenue
857	4869f7a5dfa277a7dca6462dcf3b52b2	229472.63
1013	53243585a1d6dc2643021fd1853d8905	222776.05
881	4a3ca9315b744ce9f8e9374361493884	200472.92
3024	fa1c13f2614d7b5c4749cbc52fecda94	194042.03
1535	7c67e1448b00f6e969d365cea6b010ab	187923.89
1560	7e93a43ef30c4f03f38b393420bc753a	176431.87
2643	da8622b14eb17ae2831f4ac5b9dab84a	160236.57
1505	7a67c85e85bb2ce8582c35f2203ad736	141745.53
192	1025f0e2d44d7041d6cf58b6550e0bfa	138968.55
1824	955fee9216a65b617aa5c0531780ce60	135171.70

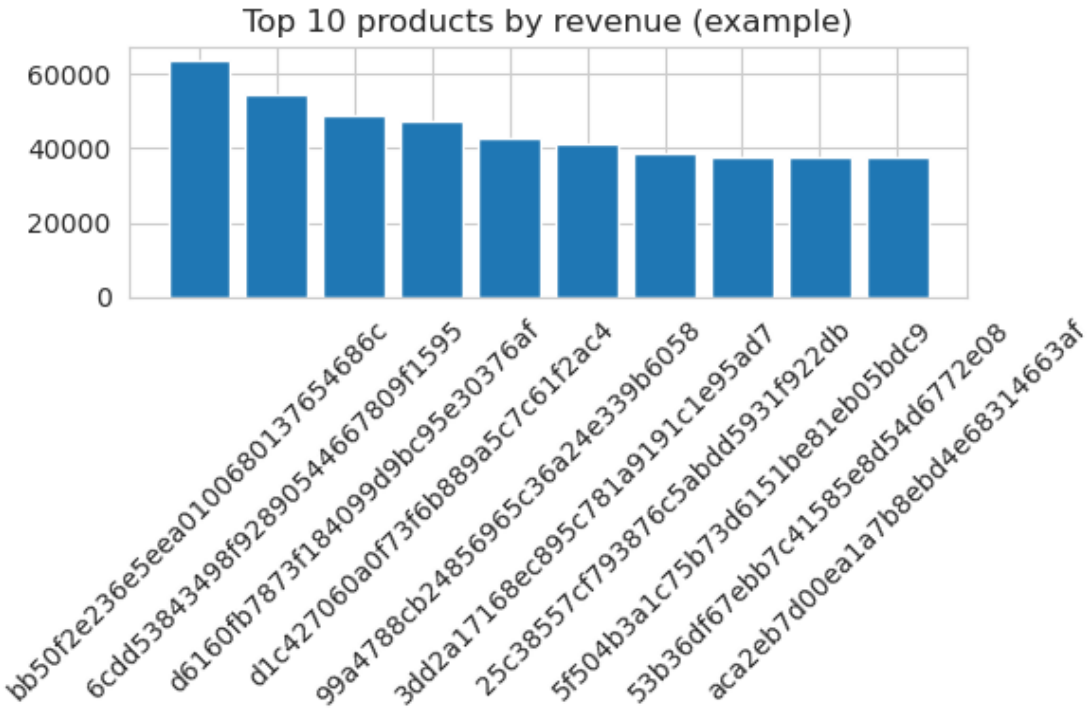
[20]:
```
snapshot_date = df_sales['order_purchase_timestamp'].max() + pd.
 ↪Timedelta(days=1)

rfm = df_sales.groupby("customer_id").agg(
    recency_days = ("order_purchase_timestamp", lambda x: (snapshot_date - x.
 ↪max()).days),
    frequency = ("order_id", "nunique"),
    monetary = ("order_revenue", "sum")
).reset_index()

# Create simple quartile scores (1-4)
rfm['r_quartile'] = pd.qcut(rfm['recency_days'], 4, labels=[4,3,2,1]).
 ↪astype(int)   # smaller recency -> higher score
rfm['f_quartile'] = pd.qcut(rfm['frequency'].rank(method='first'), 4,␣
 ↪labels=[1,2,3,4]).astype(int)
rfm['m_quartile'] = pd.qcut(rfm['monetary'], 4, labels=[1,2,3,4]).astype(int)
rfm['rfm_score'] = rfm['r_quartile'].astype(str) + rfm['f_quartile'].
 ↪astype(str) + rfm['m_quartile'].astype(str)
rfm.head()
```

[20]:

	customer_id	recency_days	frequency	monetary	r_quartile	f_quartile	m_quartile	rfm_score
0	00012a2ce6f8dcda20d059ce98491703	288	1	114.74	2	1	3	213

```
1  000161a058600d5901f007fab4c27140           410          1     67.41
1           1           2       112
2  0001fd6190edaaf884bcaf3d49edf079           548          1    195.42
1           1           4       114
3  0002414f95344307404f0ace7a26f1d5           379          1    179.35
1           1           4       114
4  000379cdec625522490c315e70c7a9fb           150          1    107.01
3           1           3       313
```

[22]:
```
df_sales.to_csv("olist_cleaned_orders.csv", index=False)
rfm.to_csv("olist_customer_rfm.csv", index=False)


plt.figure(figsize=(6,4))
plt.bar(top_products['product_id'].astype(str).head(10),␣
 ↪top_products['revenue'].head(10))
plt.title("Top 10 products by revenue (example)")
plt.xticks(rotation=45)
plt.tight_layout()
plt.savefig("plots_top_products.png")
plt.show()
```



[ ]: