ulugbek_st125457_assignment4

August 31, 2024

1 Load the libraries (0.5 mark)

[ ]:
```
# Code here
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

import os
```

2 Load the datasets (0.5 marks)

[ ]:
```
# Code here
all_data = []

dir_path = '.\datasets'
# for file_name in [name for name in os.listdir(dir_path) if name.
 ↪startswith('olist')]:
for file_name in os.listdir(dir_path):
    add = False
    file = os.path.join(dir_path, file_name)

    if 'product_category_name_translation' in file:
        add = True

    df = pd.read_csv(file)
    # Testing Purposes

    # if 'olist_orders_dataset' in file:
    #     df = df.drop(['order_delivered_customer_date',␣
 ↪'order_delivered_carrier_date', 'order_approved_at'], axis=1)

    # if 'reviews' in file:
    #     df = df.drop(['review_comment_title', 'review_comment_message'],␣
 ↪axis=1)

    # if 'products' in file:
```

```
    #     df = df.dropna(subset=['product_category_name'])

    print(df.isna().sum())
    all_data.append((add, df.copy()))


print([data.shape for cond, data in all_data])
print(*[f'{data.columns}\n' for cond, data in all_data])
```

```
customer_id                 0
customer_unique_id          0
customer_zip_code_prefix    0
customer_city               0
customer_state              0
dtype: int64
order_id                            0
customer_id                         0
order_status                        0
order_purchase_timestamp            0
order_approved_at                 160
order_delivered_carrier_date     1783
order_delivered_customer_date    2965
order_estimated_delivery_date       0
dtype: int64
order_id               0
order_item_id          0
product_id             0
seller_id              0
shipping_limit_date    0
price                  0
freight_value          0
dtype: int64
review_id                      0
order_id                       0
review_score                   0
review_comment_title       87656
review_comment_message     58247
review_creation_date           0
review_answer_timestamp        0
dtype: int64
product_id                    0
product_category_name       610
product_name_lenght         610
product_description_lenght  610
product_photos_qty          610
product_weight_g              2
product_length_cm             2
product_height_cm             2
```

```
product_width_cm                2
dtype: int64
seller_id                  0
seller_zip_code_prefix     0
seller_city                0
seller_state               0
dtype: int64
product_category_name            0
product_category_name_english    0
dtype: int64
[(99441, 5), (99441, 8), (112650, 7), (99224, 7), (32951, 9), (3095, 4), (71,
2)]
Index(['customer_id', 'customer_unique_id', 'customer_zip_code_prefix',
       'customer_city', 'customer_state'],
      dtype='object')
 Index(['order_id', 'customer_id', 'order_status', 'order_purchase_timestamp',
       'order_approved_at', 'order_delivered_carrier_date',
       'order_delivered_customer_date', 'order_estimated_delivery_date'],
      dtype='object')
 Index(['order_id', 'order_item_id', 'product_id', 'seller_id',
       'shipping_limit_date', 'price', 'freight_value'],
      dtype='object')
 Index(['review_id', 'order_id', 'review_score', 'review_comment_title',
       'review_comment_message', 'review_creation_date',
       'review_answer_timestamp'],
      dtype='object')
 Index(['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'],
      dtype='object')
 Index(['seller_id', 'seller_zip_code_prefix', 'seller_city', 'seller_state'],
dtype='object')
 Index(['product_category_name', 'product_category_name_english'],
dtype='object')
```

2.1 Merge the required datasets to get the dataset which will be used to get the requierd recommendation systems (1 marks)

[ ]:
```python
# Code here
# hint: make use of appropriate key columns
_, df_all = all_data.pop(0)
for cond, df in all_data:
    try:
        if not cond:
            df_all = pd.merge(df_all, df)
        else:
```

```
            df_all = pd.merge(df_all, df)# , how='left',␣
 ↪on='product_category_name')
    except Exception as ex:
        pass

print(df_all.shape)
df_all.columns
```

```
(110750, 36)
```

[ ]:
```
Index(['customer_id', 'customer_unique_id', 'customer_zip_code_prefix',
       'customer_city', 'customer_state', 'order_id', 'order_status',
       'order_purchase_timestamp', 'order_approved_at',
       'order_delivered_carrier_date', 'order_delivered_customer_date',
       'order_estimated_delivery_date', 'order_item_id', 'product_id',
       'seller_id', 'shipping_limit_date', 'price', 'freight_value',
       'review_id', 'review_score', 'review_comment_title',
       'review_comment_message', 'review_creation_date',
       'review_answer_timestamp', 'product_category_name',
       'product_name_lenght', 'product_description_lenght',
       'product_photos_qty', 'product_weight_g', 'product_length_cm',
       'product_height_cm', 'product_width_cm', 'seller_zip_code_prefix',
       'seller_city', 'seller_state', 'product_category_name_english'],
      dtype='object')
```

[ ]:
```
df_all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110750 entries, 0 to 110749
Data columns (total 36 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   customer_id                    110750 non-null  object
 1   customer_unique_id             110750 non-null  object
 2   customer_zip_code_prefix       110750 non-null  int64
 3   customer_city                  110750 non-null  object
 4   customer_state                 110750 non-null  object
 5   order_id                       110750 non-null  object
 6   order_status                   110750 non-null  object
 7   order_purchase_timestamp       110750 non-null  object
 8   order_approved_at              110736 non-null  object
 9   order_delivered_carrier_date   109605 non-null  object
 10  order_delivered_customer_date  108457 non-null  object
 11  order_estimated_delivery_date  110750 non-null  object
 12  order_item_id                  110750 non-null  int64
 13  product_id                     110750 non-null  object
 14  seller_id                      110750 non-null  object
 15  shipping_limit_date            110750 non-null  object
```

```
 16  price                          110750 non-null  float64
 17  freight_value                  110750 non-null  float64
 18  review_id                      110750 non-null  object
 19  review_score                   110750 non-null  int64
 20  review_comment_title           13347 non-null   object
 21  review_comment_message         46923 non-null   object
 22  review_creation_date           110750 non-null  object
 23  review_answer_timestamp        110750 non-null  object
 24  product_category_name          110750 non-null  object
 25  product_name_lenght            110750 non-null  float64
 26  product_description_lenght     110750 non-null  float64
 27  product_photos_qty             110750 non-null  float64
 28  product_weight_g               110749 non-null  float64
 29  product_length_cm              110749 non-null  float64
 30  product_height_cm              110749 non-null  float64
 31  product_width_cm               110749 non-null  float64
 32  seller_zip_code_prefix         110750 non-null  int64
 33  seller_city                    110750 non-null  object
 34  seller_state                   110750 non-null  object
 35  product_category_name_english  110750 non-null  object
dtypes: float64(9), int64(4), object(23)
memory usage: 30.4+ MB
```

[ ]:
```
df_all.isna().sum().sort_values(ascending=False)
```

[ ]:
```
review_comment_title            97403
review_comment_message          63827
order_delivered_customer_date    2293
order_delivered_carrier_date     1145
order_approved_at                  14
product_width_cm                    1
product_height_cm                   1
product_length_cm                   1
product_weight_g                    1
seller_zip_code_prefix              0
seller_city                         0
seller_state                        0
product_photos_qty                  0
product_description_lenght          0
product_name_lenght                 0
product_category_name               0
review_answer_timestamp             0
review_creation_date                0
customer_id                         0
review_id                           0
review_score                        0
customer_unique_id                  0
```

```
freight_value                    0
price                            0
shipping_limit_date              0
seller_id                        0
product_id                       0
order_item_id                    0
order_estimated_delivery_date    0
order_purchase_timestamp         0
order_status                     0
order_id                         0
customer_state                   0
customer_city                    0
customer_zip_code_prefix         0
product_category_name_english    0
dtype: int64
```

[ ]:
```
# Data Cleaning - I am going to drop entire column of review_comment_title &␣
 ↪review_comment_message - no need for those details in our recommendation␣
 ↪system - we only need score
# Same for order_delivered_customer_date, order_delivered_carrier_date,␣
 ↪order_approved_at - no need the information regarding deliverance date and␣
 ↪approved at
# And we can simply dropna for left columns
df_all = df_all.drop(['review_comment_title', 'review_comment_message',␣
 ↪'order_delivered_customer_date', 'order_delivered_carrier_date',␣
 ↪'order_approved_at'], axis=1)
# df = df.drop([], axis=1)

df_all = df_all.dropna(subset=['product_category_name_english'])
```

[ ]:
```
df_all.isna().sum().sort_values(ascending=False)
```

[ ]:
```
product_width_cm                 1
product_height_cm                1
product_length_cm                1
product_weight_g                 1
customer_id                      0
review_score                     0
seller_state                     0
seller_city                      0
seller_zip_code_prefix           0
product_photos_qty               0
product_description_lenght       0
product_name_lenght              0
product_category_name            0
review_answer_timestamp          0
review_creation_date             0
```

```
review_id                          0
customer_unique_id                 0
freight_value                      0
price                              0
shipping_limit_date                0
seller_id                          0
product_id                         0
order_item_id                      0
order_estimated_delivery_date      0
order_purchase_timestamp           0
order_status                       0
order_id                           0
customer_state                     0
customer_city                      0
customer_zip_code_prefix           0
product_category_name_english      0
dtype: int64
```

[ ]:
```
df_all.shape
```

[ ]:
```
(110750, 31)
```

3 Location Recommendation System

3.0.1 For the location Sao Paolo find the top 5 product category sold (in English name) by total order price and by order count - 5 marks

Top catgory with most order price for "Sao Paolo" – 2.5 marks

3.0.2 Note: So the thing is, I think, ordering based on seller_city is more accurate if we are considering top products sold by city (where seller is actually located), but the customer_city might be also considered - which creates a bias. Therefore, I will have shown for both cases to meet requirements, at least.

[ ]:
```
# # Variant 1
# df = df_all[['seller_city', 'price', 'product_category_name_english']]
# mask = df['seller_city'] == 'sao paulo'

# df[mask].groupby('product_category_name_english').price.sum().reset_index().
  ↪sort_values(by='price', ascending=False).head()

# Variant 2
df = df_all[['customer_city', 'price', 'product_category_name_english']]
mask = df['customer_city'] == 'sao paulo'

df[mask].groupby('product_category_name_english').price.sum().reset_index().
  ↪sort_values(by='price', ascending=False).head()
```

[ ]:

	product_category_name_english	price
43	health_beauty	188264.85
7	bed_bath_table	171760.02
69	watches_gifts	166317.45
15	computers_accessories	145324.36
64	sports_leisure	145295.92

Top catgory with most order number for "Sao Paolo" – 2.5 marks

[ ]:
```
# # Variant 1
# df = df_all[['order_id', 'seller_city', 'product_category_name_english']]
# mask = df['seller_city'] == 'sao paulo'

# df[mask].groupby('product_category_name_english').order_id.count().
 ↪reset_index(name='order_count').sort_values(by='order_count',␣
 ↪ascending=False).head()

# Variant 2
df = df_all[['order_id', 'customer_city', 'product_category_name_english']]
mask = df['customer_city'] == 'sao paulo'

df[mask].groupby('product_category_name_english').size().
 ↪reset_index(name='order_count').sort_values(by='order_count',␣
 ↪ascending=False).head()
```

[ ]:

	product_category_name_english	order_count
7	bed_bath_table	2005
43	health_beauty	1740
64	sports_leisure	1404
49	housewares	1313
39	furniture_decor	1259

4 Product Category Recommendation System

4.0.1 For the category "Electronics" find the top 5 city where the product is most sold based on price and orde count - 5 marks

Top city by order count – 2.5 marks

[ ]:
```
# # Variant 1
# df = df_all[['order_id', 'seller_city', 'product_category_name_english']]
# mask = df['product_category_name_english'] == 'electronics'

# df[mask].groupby('seller_city').order_id.count().
 ↪reset_index(name='order_count').sort_values(by='order_count',␣
 ↪ascending=False).head()

# Variant 2
df = df_all[['order_id', 'customer_city', 'product_category_name_english']]
```

```
mask = df['product_category_name_english'] == 'electronics'

df[mask].groupby('customer_city').order_id.count().
 ↪reset_index(name='order_count').sort_values(by='order_count',␣
 ↪ascending=False).head()
```

[ ]:

	customer_city	order_count
651	sao paulo	355
576	rio de janeiro	241
67	belo horizonte	66
87	brasilia	51
590	salvador	50

Top city by order price – 2.5 marks

[ ]:
```
# Variant 1
# df = df_all[['seller_city', 'price', 'product_category_name_english']]
# mask = df['product_category_name_english'] == 'electronics'

# df[mask].groupby('seller_city').price.sum().reset_index().
 ↪sort_values(by='price', ascending=False).head()

# Variant 2
df = df_all[['customer_city', 'price', 'product_category_name_english']]
mask = df['product_category_name_english'] == 'electronics'

df[mask].groupby('customer_city').price.sum().reset_index().
 ↪sort_values(by='price', ascending=False).head()
```

[ ]:

	customer_city	price
651	sao paulo	18628.92
576	rio de janeiro	14612.11
67	belo horizonte	3874.00
597	santa luzia	2638.38
613	santo antonio de posse	2484.15

5 Review Monitoring System – 8 marks

5.0.1 Find the top reviewed categories – 4 marks

[ ]:
```
# Code here
# hint: use mean
df_review = df_all[['product_category_name_english', 'review_score']]

df_review.groupby('product_category_name_english').review_score.mean().
 ↪reset_index(name='top_reviews').sort_values(by='top_reviews',␣
 ↪ascending=False).head()
```

[ ]:

	product_category_name_english	top_reviews
11	cds_dvds_musicals	4.642857
29	fashion_childrens_clothes	4.500000
8	books_general_interest	4.446266
22	costruction_tools_tools	4.444444
35	flowers	4.419355

5.0.2 Find the customers who has given the most and least number of reviews – 4 marks

[ ]:
```
# Code here
df = df_all[['customer_unique_id', 'review_id']]

review_counts = df.groupby('customer_unique_id').review_id.count().
 ↪reset_index(name='review_count').sort_values(by='review_count',␣
 ↪ascending=False)

print("The most reviews: ", *review_counts.loc[review_counts.review_count.
 ↪idxmax()], "\nThe least reviews: ", *review_counts.loc[review_counts.
 ↪review_count.idxmin()])
```

```
The most reviews:  c8460e4251689ba205045f3ea17884a1 24
The least reviews:  a1a0841d89f84138975671f3d0c5842e 1
```