Capstone Project (E-commerce Domain)

```
Project Name: Olist Marketplace Sales Data Analysis
Prepared by: Bishowjith Ghosh
Date: September 2024
Master’s Program in Data Science
```

In [1]:
```
# Importing necessary libraries
```

In [2]:
```
import pandas as pd
```

Load all Olist datasets

In [3]:
```
orders = pd.read_csv('olist_orders_dataset.csv')
customers = pd.read_csv('olist_customers_dataset.csv')
geolocation = pd.read_csv('olist_geolocation_dataset.csv')
order_items = pd.read_csv('olist_order_items_dataset.csv')
order_payments = pd.read_csv('olist_order_payments_dataset.csv')
order_reviews = pd.read_csv('olist_order_reviews_dataset.csv')
products = pd.read_csv('olist_products_dataset.csv')
sellers = pd.read_csv('olist_sellers_dataset.csv')
product_categories = pd.read_csv('product_category_name_translation.csv')
```

Preview the datasets

In [4]:
```
print(orders.head())
print(customers.head())
print(geolocation.head())
print(order_items.head())
print(order_payments.head())
print(order_reviews.head())
print(products.head())
print(sellers.head())
print(product_categories.head())
```

```
                           order_id                       customer_id  \
0  e481f51cbdc54678b7cc49136f2d6af7  9ef432eb6251297304e76186b10a928d
1  53cdb2fc8bc7dce0b6741e2150273451  b0830fb4747a6c6d20dea0b8c802d7ef
2  47770eb9100c2d0c44946d9cf07ec65d  41ce2a54c0b03bf3443c3d931a367089
3  949d5b44dbf5de918fe9c16f97b45f8a  f88197465ea7920adcdbec7375364d82
4  ad21c59c0840e6cb83a9ceb5573f8159  8ab97904e6daea8866dbdbc4fb7aad2c

  order_status order_purchase_timestamp    order_approved_at  \
0    delivered      2017-10-02 10:56:33  2017-10-02 11:07:15
1    delivered      2018-07-24 20:41:37  2018-07-26 03:24:27
2    delivered      2018-08-08 08:38:49  2018-08-08 08:55:23
3    delivered      2017-11-18 19:28:06  2017-11-18 19:45:59
4    delivered      2018-02-13 21:18:39  2018-02-13 22:20:29

  order_delivered_carrier_date order_delivered_customer_date  \
0          2017-10-04 19:55:00           2017-10-10 21:25:13
1          2018-07-26 14:31:00           2018-08-07 15:27:45
2          2018-08-08 13:50:00           2018-08-17 18:06:29
3          2017-11-22 13:39:59           2017-12-02 00:28:42
4          2018-02-14 19:46:34           2018-02-16 18:17:02

  order_estimated_delivery_date
0           2017-10-18 00:00:00
1           2018-08-13 00:00:00
2           2018-09-04 00:00:00
3           2017-12-15 00:00:00
4           2018-02-26 00:00:00
                        customer_id                customer_unique_id  \
0  06b8999e2fba1a1fbc88172c00ba8bc7  861eff4711a542e4b93843c6dd7febb0
1  18955e83d337fd6b2def6b18a428ac77  290c77bc529b7ac935b93aa66c333dc3
2  4e7b3e00288586ebd08712fdd0374a03  060e732b5b29e8181a18229c7b0b2b5e
3  b2b6027bc5c5109e529d4dc6358b12c3  259dac757896d24d7702b9acbbff3f3c
4  4f2d8ab171c80ec8364f7c12e35b23ad  345ecd01c38d18a9036ed96c73b8d066

   customer_zip_code_prefix          customer_city customer_state
0                     14409                 franca             SP
1                      9790  sao bernardo do campo             SP
2                      1151              sao paulo             SP
3                      8775        mogi das cruzes             SP
4                     13056               campinas             SP
   geolocation_zip_code_prefix  geolocation_lat  geolocation_lng  \
0                         1037       -23.545621       -46.639292
1                         1046       -23.546081       -46.644820
2                         1046       -23.546129       -46.642951
3                         1041       -23.544392       -46.639499
4                         1035       -23.541578       -46.641607

  geolocation_city geolocation_state
0        sao paulo                SP
1        sao paulo                SP
2        sao paulo                SP
3        sao paulo                SP
4        sao paulo                SP
                           order_id  order_item_id  \
0  00010242fe8c5a6d1ba2dd792cb16214              1
1  00018f77f2f0320c557190d7a144bdd3              1
2  000229ec398224ef6ca0657da4fc703e              1
3  00024acbcdf0a6daa1e931b038114c75              1
4  00042b26cf59d7ce69dfabb4e55b4fd9              1

                         product_id                         seller_id  \
0  4244733e06e7ecb4970a6e2683c13e61  48436dade18ac8b2bce089ec2a041202
1  e5f2d52b802189ee658865ca93d83a8f  dd7ddc04e1b6c2c614352b383efe2d36
2  c777355d18b72b67abbeef9df44fd0fd  5b51032eddd242adc84c38acab88f23d
3  7634da152a4610f1595efa32f14722fc  9d7a1d34a5052409006425275ba1c2b4
4  ac6c3623068f30de03045865e4e10089  df560393f3a51e74553ab94004ba5c87

   shipping_limit_date   price  freight_value
0  2017-09-19 09:45:35   58.90          13.29
1  2017-05-03 11:05:13  239.90          19.93
2  2018-01-18 14:48:30  199.00          17.87
3  2018-08-15 10:10:18   12.99          12.79
4  2017-02-13 13:57:51  199.90          18.14
                           order_id  payment_sequential payment_type  \
0  b81ef226f3fe1789b1e8b2acac839d17                   1  credit_card
1  a9810da82917af2d9aefd1278f1dcfa0                   1  credit_card
2  25e8ea4e93396b6fa0d3dd708e76c1bd                   1  credit_card
3  ba78997921bbcdc1373bb41e913ab953                   1  credit_card
4  42fdf880ba16b47b59251dd489d4441a                   1  credit_card

   payment_installments  payment_value
0                     8          99.33
1                     1          24.39
2                     1          65.71
3                     8         107.78
4                     2         128.45
                          review_id                          order_id  \
0  7bc2406110b926393aa56f80a40eba40  73fc7af87114b39712e6da79b0a377eb
1  80e641a11e56f04c1ad469d5645fdfde  a548910a1c6147796b98fdf73dbeba33
2  228ce5500dc1d8e020d8d1322874b6f0  f9e4b658b201a9f2ecdecbb34bed034b
```

```
3  e64fb393e7b32834bb789ff8bb30750e  658677c97b385a9be170737859d3511b
4  f7c4243c7fe1938f181bec41a392bdeb  8e6bfb81e283fa7e4f11123a3fb894f1

   review_score review_comment_title  \
0             4                  NaN
1             5                  NaN
2             5                  NaN
3             5                  NaN
4             5                  NaN

                              review_comment_message review_creation_date  \
0                                                NaN  2018-01-18 00:00:00
1                                                NaN  2018-03-10 00:00:00
2                                                NaN  2018-02-17 00:00:00
3              Recebi bem antes do prazo estipulado.  2017-04-21 00:00:00
4  Parabéns lojas lannister adorei comprar pela I...  2018-03-01 00:00:00

  review_answer_timestamp
0     2018-01-18 21:46:59
1     2018-03-11 03:05:13
2     2018-02-18 14:36:24
3     2017-04-21 22:02:06
4     2018-03-02 10:26:53
                         product_id  product_category_name  \
0  1e9e8ef04dbcff4541ed26657ea517e5             perfumaria
1  3aa071139cb16b67ca9e5dea641aaa2f                  artes
2  96bd76ec8810374ed1b65e291975717f          esporte_lazer
3  cef67bcfe19066a932b7673e239eb23d                  bebes
4  9dc1a7de274444849c219cff195d0b71  utilidades_domesticas

   product_name_lenght  product_description_lenght  product_photos_qty  \
0                 40.0                       287.0                 1.0
1                 44.0                       276.0                 1.0
2                 46.0                       250.0                 1.0
3                 27.0                       261.0                 1.0
4                 37.0                       402.0                 4.0

   product_weight_g  product_length_cm  product_height_cm  product_width_cm
0             225.0               16.0               10.0              14.0
1            1000.0               30.0               18.0              20.0
2             154.0               18.0                9.0              15.0
3             371.0               26.0                4.0              26.0
4             625.0               20.0               17.0              13.0
                          seller_id  seller_zip_code_prefix  \
0  3442f8959a84dea7ee197c632cb2df15                   13023
1  d1b65fc7debc3361ea86b5f14c68d2e2                   13844
2  ce3ad9de960102d0677a81f5d0bb7b2d                   20031
3  c0f3eea2e14555b6faeea3dd58c1b1c3                    4195
4  51a04a8a6bdcb23deccc82b0b80742cf                   12914

         seller_city seller_state
0           campinas           SP
1         mogi guacu           SP
2     rio de janeiro           RJ
3          sao paulo           SP
4  braganca paulista           SP
    product_category_name product_category_name_english
0            beleza_saude                 health_beauty
1  informatica_acessorios          computers_accessories
2              automotivo                          auto
3          cama_mesa_banho                bed_bath_table
4         moveis_decoracao               furniture_decor
```

Cheecking Datatypes for all dataset

In [5]:
```
# Check data types for the orders dataset
print(orders.dtypes)

# Check data types for all other datasets
print(customers.dtypes)
print(geolocation.dtypes)
print(order_items.dtypes)
print(order_payments.dtypes)
print(order_reviews.dtypes)
print(products.dtypes)
print(sellers.dtypes)
print(product_categories.dtypes)
```

```
order_id                         object
customer_id                      object
order_status                     object
order_purchase_timestamp         object
order_approved_at                object
order_delivered_carrier_date     object
order_delivered_customer_date    object
order_estimated_delivery_date    object
dtype: object
customer_id                 object
customer_unique_id          object
customer_zip_code_prefix     int64
customer_city               object
customer_state              object
dtype: object
geolocation_zip_code_prefix      int64
geolocation_lat                float64
geolocation_lng                float64
geolocation_city                object
geolocation_state               object
dtype: object
order_id                object
order_item_id            int64
product_id              object
seller_id               object
shipping_limit_date     object
price                  float64
freight_value          float64
dtype: object
order_id                 object
payment_sequential        int64
payment_type             object
payment_installments      int64
payment_value           float64
dtype: object
review_id                  object
order_id                   object
review_score                int64
review_comment_title       object
review_comment_message     object
review_creation_date       object
review_answer_timestamp    object
dtype: object
product_id                     object
product_category_name          object
product_name_lenght           float64
product_description_lenght    float64
product_photos_qty            float64
product_weight_g              float64
product_length_cm             float64
product_height_cm             float64
product_width_cm              float64
dtype: object
seller_id                 object
seller_zip_code_prefix     int64
seller_city               object
seller_state              object
dtype: object
product_category_name            object
product_category_name_english    object
dtype: object
```

Data Types Need Corrections

1. Orders Dataset:

Columns : order_purchase_timestamp, order_approved_at, order_delivered_customer_date, order_delivered_carrier_date

Reason: These are date columns, and it’s essential to convert them into the datetime data type to perform any time-based analysis.

2. Order Items Dataset:

Columns: shipping_limit_dat

Reason: shipping_limit_date should be converted from object to datetime to handle shipping deadlines properly.

3. Order Reviews Dataset:

Columns: review_creation_date, review_answer_timestamp,

Reason: These columns are timestamps indicating when reviews were created and answered. They should be converted to the datetime data type for any time-based analysis, such as tracking review response times.

Correct Data Types

In [6]:
```
# 1. Orders Dataset : Columns

orders['order_purchase_timestamp'] = pd.to_datetime(orders['order_purchase_timestamp'])
orders['order_approved_at'] = pd.to_datetime(orders['order_approved_at'])
orders['order_delivered_carrier_date'] = pd.to_datetime(orders['order_delivered_carrier_date'])
orders['order_delivered_customer_date'] = pd.to_datetime(orders['order_delivered_customer_date'])
orders['order_estimated_delivery_date'] = pd.to_datetime(orders['order_estimated_delivery_date'])

# 2. Order Items Dataset: Columns
order_items['shipping_limit_date'] = pd.to_datetime(order_items['shipping_limit_date'])

# 3. Order Reviews Dataset: Columns

order_reviews['review_creation_date'] = pd.to_datetime(order_reviews['review_creation_date'])
order_reviews['review_answer_timestamp'] = pd.to_datetime(order_reviews['review_answer_timestamp'])


# Checking corrected data types
print(orders.dtypes)
print(order_items.dtypes)
print(order_reviews.dtypes)
```

```
order_id                                  object
customer_id                               object
order_status                              object
order_purchase_timestamp          datetime64[ns]
order_approved_at                 datetime64[ns]
order_delivered_carrier_date      datetime64[ns]
order_delivered_customer_date     datetime64[ns]
order_estimated_delivery_date     datetime64[ns]
dtype: object
order_id                          object
order_item_id                      int64
product_id                        object
seller_id                         object
shipping_limit_date       datetime64[ns]
price                            float64
freight_value                    float64
dtype: object
review_id                                object
order_id                                 object
review_score                              int64
review_comment_title                     object
review_comment_message                   object
review_creation_date             datetime64[ns]
review_answer_timestamp          datetime64[ns]
dtype: object
```

Checking for missing values in all datasets

In [7]:
```
print("Orders Missing Values:\n", orders.isnull().sum())
print("Customers Missing Values:\n", customers.isnull().sum())
print("Geolocation Missing Values:\n", geolocation.isnull().sum())
print("Order Items Missing Values:\n", order_items.isnull().sum())
print("Order Payments Missing Values:\n", order_payments.isnull().sum())
print("Order Reviews Missing Values:\n", order_reviews.isnull().sum())
print("Products Missing Values:\n", products.isnull().sum())
print("Sellers Missing Values:\n", sellers.isnull().sum())
print("Product Categories Missing Values:\n", product_categories.isnull().sum())
```

```
Orders Missing Values:
 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
Customers Missing Values:
 customer_id                 0
customer_unique_id          0
customer_zip_code_prefix    0
customer_city               0
customer_state              0
dtype: int64
Geolocation Missing Values:
 geolocation_zip_code_prefix    0
geolocation_lat                0
geolocation_lng                0
geolocation_city               0
geolocation_state              0
dtype: int64
Order Items Missing Values:
 order_id               0
order_item_id          0
product_id             0
seller_id              0
shipping_limit_date    0
price                  0
freight_value          0
dtype: int64
Order Payments Missing Values:
 order_id                0
payment_sequential      0
payment_type            0
payment_installments    0
payment_value           0
dtype: int64
Order Reviews Missing Values:
 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
Products Missing Values:
 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
Sellers Missing Values:
 seller_id                 0
seller_zip_code_prefix    0
seller_city               0
seller_state              0
dtype: int64
Product Categories Missing Values:
 product_category_name            0
product_category_name_english    0
dtype: int64
```

Handling Missing Values For All Datasets

Missing values in Orders Dataset,Order Reviews Dataset, Products Dataset.

In [8]:
```
# 1. Orders Dataset:

# order_approved_at:        Fill in the missing order_approved_at values by using the order_purchase_timestamp
#                           Or you can drop these rows if approval date is crucial for further analysis.

orders['order_approved_at'].fillna(orders['order_purchase_timestamp'], inplace=True)


# Drop rows where order_delivered_carrier_date or order_delivered_customer_date is missing
orders.dropna(subset=['order_delivered_carrier_date', 'order_delivered_customer_date'], inplace=True)
```

```
# Check for remaining missing values
orders.isnull().sum()
```

Out[8]:
```
order_id                         0
customer_id                      0
order_status                     0
order_purchase_timestamp         0
order_approved_at                0
order_delivered_carrier_date     0
order_delivered_customer_date    0
order_estimated_delivery_date    0
dtype: int64
```

In [9]:
```
# 2. Order Reviews Dataset:

# review_comment_title, review_comment_message:
#                      These are textual fields that are not critical for numerical analysis. Fill with placeholders
#                      like "No Title" or "No Comment". Drop rows with missing values if not needed.

order_reviews['review_comment_title'].fillna('No Title', inplace=True)
order_reviews['review_comment_message'].fillna('No Comment', inplace=True)


order_reviews.isnull().sum()
```

Out[9]:
```
review_id                  0
order_id                   0
review_score               0
review_comment_title       0
review_comment_message     0
review_creation_date       0
review_answer_timestamp    0
dtype: int64
```

In [10]:
```
# 3. products Dataset

# product_category_name: Since these are categorical fields, fill with a placeholder like "Unknown".

products['product_category_name'].fillna('Unknown', inplace=True)


# product_name_length, product_description_length, product_photos_qty, product_weight_g, product_length_cm,
# product_height_cm, product_width_cm:   These are Numerical fields.Using mean or median to fill missing numerical va


# Fill missing numerical values with the mean
products['product_name_lenght'].fillna(products['product_name_lenght'].mean(), inplace=True)
products['product_description_lenght'].fillna(products['product_description_lenght'].mean(), inplace=True)
products['product_photos_qty'].fillna(products['product_photos_qty'].mean(), inplace=True)
products['product_weight_g'].fillna(products['product_weight_g'].mean(), inplace=True)
products['product_length_cm'].fillna(products['product_length_cm'].mean(), inplace=True)
products['product_height_cm'].fillna(products['product_height_cm'].mean(), inplace=True)
products['product_width_cm'].fillna(products['product_width_cm'].mean(), inplace=True)


products.isnull().sum()
```

Out[10]:
```
product_id                    0
product_category_name         0
product_name_lenght           0
product_description_lenght    0
product_photos_qty            0
product_weight_g              0
product_length_cm             0
product_height_cm             0
product_width_cm              0
dtype: int64
```

Check for Duplicates

In [11]:
```
print( "orders duplicated:", orders.duplicated().sum())
print("customers duplicated:", customers.duplicated().sum())
print("geolocation duplicated:", geolocation.duplicated().sum())
print("order_items duplicated:", order_items.duplicated().sum())
print("order_payments duplicated:", order_payments.duplicated().sum())
print("order_reviews duplicatedn", order_reviews.duplicated().sum())
print("products duplicated", products.duplicated().sum())
print("sellers duplicated:", sellers.duplicated().sum())
```

```
orders duplicated: 0
customers duplicated: 0
geolocation duplicated: 261831
order_items duplicated: 0
order_payments duplicated: 0
order_reviews duplicatedn 0
products duplicated 0
sellers duplicated: 0
```

Remove Duplicates in the Geolocation Dataset

In [12]:
```
# Remove duplicates from the geolocation dataset
geolocation_cleaned = geolocation.drop_duplicates()

# Confirm the duplicates have been removed
print("geolocation_cleaned.duplicated :", geolocation_cleaned.duplicated().sum())
```

```
geolocation_cleaned.duplicated : 0
```

Data Cleaning,Trimming Whitespaces and Handling Case Sensitivity

Check for Whitespace:

In [13]:
```
# Check for rows with leading/trailing whitespaces in the customer_id column
print("Orders - Customer ID (leading/trailing whitespaces):")
print(orders[orders['customer_id'].str.contains('^\s+|\s+$', regex=True)]['customer_id'].head())

# Check for rows with leading/trailing whitespaces in the product_id column
print("Order Items - Product ID (leading/trailing whitespaces):")
print(order_items[order_items['product_id'].str.contains('^\s+|\s+$', regex=True)]['product_id'].head())
```

```
Orders - Customer ID (leading/trailing whitespaces):
Series([], Name: customer_id, dtype: object)
Order Items - Product ID (leading/trailing whitespaces):
Series([], Name: product_id, dtype: object)
```

Check for Case Sensitivity:

In [14]:
```
# Check for case inconsistencies in customer_id (uppercase letters)
print("Orders - Customer ID (case inconsistencies):")
print(orders[orders['customer_id'].str.contains('[A-Z]')]['customer_id'].head())

# Check for case inconsistencies in product_id (uppercase letters)
print("Order Items - Product ID (case inconsistencies):")
print(order_items[order_items['product_id'].str.contains('[A-Z]')]['product_id'].head())
```

```
Orders - Customer ID (case inconsistencies):
Series([], Name: customer_id, dtype: object)
Order Items - Product ID (case inconsistencies):
Series([], Name: product_id, dtype: object)
```

Check, Trimming and converting to lowercase

In [15]:
```
# Display a sample of the key columns before trimming and converting to lowercase
print("Before cleaning:")
print(orders['customer_id'].head())
print(customers['customer_id'].head())

print(products['product_id'].head())
print(order_items['product_id'].head())

print(sellers['seller_id'].head())
print(order_items['seller_id'].head())
```

```
Before cleaning:
0    9ef432eb6251297304e76186b10a928d
1    b0830fb4747a6c6d20dea0b8c802d7ef
2    41ce2a54c0b03bf3443c3d931a367089
3    f88197465ea7920adcdbec7375364d82
4    8ab97904e6daea8866dbdbc4fb7aad2c
Name: customer_id, dtype: object
0    06b8999e2fba1a1fbc88172c00ba8bc7
1    18955e83d337fd6b2def6b18a428ac77
2    4e7b3e00288586ebd08712fdd0374a03
3    b2b6027bc5c5109e529d4dc6358b12c3
4    4f2d8ab171c80ec8364f7c12e35b23ad
Name: customer_id, dtype: object
0    1e9e8ef04dbcff4541ed26657ea517e5
1    3aa071139cb16b67ca9e5dea641aaa2f
2    96bd76ec8810374ed1b65e291975717f
3    cef67bcfe19066a932b7673e239eb23d
4    9dc1a7de274444849c219cff195d0b71
Name: product_id, dtype: object
0    4244733e06e7ecb4970a6e2683c13e61
1    e5f2d52b802189ee658865ca93d83a8f
2    c777355d18b72b67abbeef9df44fd0fd
3    7634da152a4610f1595efa32f14722fc
4    ac6c3623068f30de03045865e4e10089
Name: product_id, dtype: object
0    3442f8959a84dea7ee197c632cb2df15
1    d1b65fc7debc3361ea86b5f14c68d2e2
2    ce3ad9de960102d0677a81f5d0bb7b2d
3    c0f3eea2e14555b6faeea3dd58c1b1c3
4    51a04a8a6bdcb23deccc82b0b80742cf
Name: seller_id, dtype: object
0    48436dade18ac8b2bce089ec2a041202
1    dd7ddc04e1b6c2c614352b383efe2d36
2    5b51032eddd242adc84c38acab88f23d
3    9d7a1d34a5052409006425275ba1c2b4
4    df560393f3a51e74553ab94004ba5c87
Name: seller_id, dtype: object
```

In [16]:

```python
# Trim whitespaces and ensure consistent case in key columns
orders['customer_id'] = orders['customer_id'].str.strip().str.lower()
customers['customer_id'] = customers['customer_id'].str.strip().str.lower()

products['product_id'] = products['product_id'].str.strip().str.lower()
order_items['product_id'] = order_items['product_id'].str.strip().str.lower()

sellers['seller_id'] = sellers['seller_id'].str.strip().str.lower()
order_items['seller_id'] = order_items['seller_id'].str.strip().str.lower()


# Display the same key columns after trimming and converting to lowercase
print("After cleaning:")
print(orders['customer_id'].head())
print(customers['customer_id'].head())

print(products['product_id'].head())
print(order_items['product_id'].head())

print(sellers['seller_id'].head())
print(order_items['seller_id'].head())
```

```
After cleaning:
0    9ef432eb6251297304e76186b10a928d
1    b0830fb4747a6c6d20dea0b8c802d7ef
2    41ce2a54c0b03bf3443c3d931a367089
3    f88197465ea7920adcdbec7375364d82
4    8ab97904e6daea8866dbdbc4fb7aad2c
Name: customer_id, dtype: object
0    06b8999e2fba1a1fbc88172c00ba8bc7
1    18955e83d337fd6b2def6b18a428ac77
2    4e7b3e00288586ebd08712fdd0374a03
3    b2b6027bc5c5109e529d4dc6358b12c3
4    4f2d8ab171c80ec8364f7c12e35b23ad
Name: customer_id, dtype: object
0    1e9e8ef04dbcff4541ed26657ea517e5
1    3aa071139cb16b67ca9e5dea641aaa2f
2    96bd76ec8810374ed1b65e291975717f
3    cef67bcfe19066a932b7673e239eb23d
4    9dc1a7de274444849c219cff195d0b71
Name: product_id, dtype: object
0    4244733e06e7ecb4970a6e2683c13e61
1    e5f2d52b802189ee658865ca93d83a8f
2    c777355d18b72b67abbeef9df44fd0fd
3    7634da152a4610f1595efa32f14722fc
4    ac6c3623068f30de03045865e4e10089
Name: product_id, dtype: object
0    3442f8959a84dea7ee197c632cb2df15
1    d1b65fc7debc3361ea86b5f14c68d2e2
2    ce3ad9de960102d0677a81f5d0bb7b2d
3    c0f3eea2e14555b6faeea3dd58c1b1c3
4    51a04a8a6bdcb23deccc82b0b80742cf
Name: seller_id, dtype: object
0    48436dade18ac8b2bce089ec2a041202
1    dd7ddc04e1b6c2c614352b383efe2d36
2    5b51032eddd242adc84c38acab88f23d
3    9d7a1d34a5052409006425275ba1c2b4
4    df560393f3a51e74553ab94004ba5c87
Name: seller_id, dtype: object
```

Merge/Join the Datasets

After cleaning the data and removing duplicates, merge the datasets based on keys like order_id, customer_id, product_id, etc. Let's merge the relevant datasets step by step. You can start with merging the orders, customers, and order_items datasets.

In [17]:

```python
# Step 1: Merge orders and customers datasets

# Need customer information to understand customer behavior.
# Using a left join keeps all orders, even if some orders may not have associated customer details

merged_data = pd.merge(orders, customers, on='customer_id', how='left')

# Step 2: Merge the result with order_items

# Each order may have one or more items. To analyze what items were purchased in each order join on order_id.
# Left join keeps all orders, even if some orders don't have items.

merged_data = pd.merge(merged_data, order_items, on='order_id', how='left')

# Step 3: Merge with products dataset

#  need to enrich the dataset with product information.
# Each order item is linked to a specific product by product_id
# You use a left join to ensure that all items from the order_items table are retained, even if some items have missi
# f a matching product_id is not found, the product-related columns will be filled with NaN

merged_data = pd.merge(merged_data, products, on='product_id', how='left')

# Step 4: Merge with geolocation dataset

# need geolocation data (latitude, longitude) to perform geographic analyses on where customers are locate
# join the customer_zip_code_prefix in orders with the geolocation_zip_code_prefix in the geolocation dataset.

merged_data = pd.merge(merged_data, geolocation_cleaned, left_on='customer_zip_code_prefix', right_on='geolocation_zi

# Step 5: Merge with sellers dataset

# Sellers are crucial in the marketplace model, so need to merge their details.
#  Left join ensures that all orders, including those without seller information.

merged_data = pd.merge(merged_data, sellers, on='seller_id', how='left')

# Step 6: Merge with order_payments and order_reviews if necessary

# Payment and review information is important to understand order success and customer feedback.
# Left join ensures that all orders are kept, even if some orders do not have associated payment or review informatio

merged_data = pd.merge(merged_data, order_payments, on='order_id', how='left')
merged_data = pd.merge(merged_data, order_reviews, on='order_id', how='left')
```

```
# Verify the merged dataset
print(merged_data.head())
```

```
                           order_id                       customer_id  \
0  e481f51cbdc54678b7cc49136f2d6af7  9ef432eb6251297304e76186b10a928d
1  e481f51cbdc54678b7cc49136f2d6af7  9ef432eb6251297304e76186b10a928d
2  e481f51cbdc54678b7cc49136f2d6af7  9ef432eb6251297304e76186b10a928d
3  e481f51cbdc54678b7cc49136f2d6af7  9ef432eb6251297304e76186b10a928d
4  e481f51cbdc54678b7cc49136f2d6af7  9ef432eb6251297304e76186b10a928d

  order_status order_purchase_timestamp   order_approved_at  \
0    delivered      2017-10-02 10:56:33 2017-10-02 11:07:15
1    delivered      2017-10-02 10:56:33 2017-10-02 11:07:15
2    delivered      2017-10-02 10:56:33 2017-10-02 11:07:15
3    delivered      2017-10-02 10:56:33 2017-10-02 11:07:15
4    delivered      2017-10-02 10:56:33 2017-10-02 11:07:15

  order_delivered_carrier_date order_delivered_customer_date  \
0          2017-10-04 19:55:00           2017-10-10 21:25:13
1          2017-10-04 19:55:00           2017-10-10 21:25:13
2          2017-10-04 19:55:00           2017-10-10 21:25:13
3          2017-10-04 19:55:00           2017-10-10 21:25:13
4          2017-10-04 19:55:00           2017-10-10 21:25:13

  order_estimated_delivery_date                customer_unique_id  \
0                    2017-10-18  7c396fd4830fd04220f754e42b4e5bff
1                    2017-10-18  7c396fd4830fd04220f754e42b4e5bff
2                    2017-10-18  7c396fd4830fd04220f754e42b4e5bff
3                    2017-10-18  7c396fd4830fd04220f754e42b4e5bff
4                    2017-10-18  7c396fd4830fd04220f754e42b4e5bff

   customer_zip_code_prefix  ... payment_sequential payment_type  \
0                      3149  ...                1.0  credit_card
1                      3149  ...                3.0      voucher
2                      3149  ...                2.0      voucher
3                      3149  ...                1.0  credit_card
4                      3149  ...                3.0      voucher

   payment_installments payment_value                         review_id  \
0                   1.0         18.12  a54f0611adc9ed256b57ede6b6eb5114
1                   1.0          2.00  a54f0611adc9ed256b57ede6b6eb5114
2                   1.0         18.59  a54f0611adc9ed256b57ede6b6eb5114
3                   1.0         18.12  a54f0611adc9ed256b57ede6b6eb5114
4                   1.0          2.00  a54f0611adc9ed256b57ede6b6eb5114

  review_score  review_comment_title  \
0          4.0              No Title
1          4.0              No Title
2          4.0              No Title
3          4.0              No Title
4          4.0              No Title

                              review_comment_message review_creation_date  \
0  Não testei o produto ainda, mas ele veio corre...           2017-10-11
1  Não testei o produto ainda, mas ele veio corre...           2017-10-11
2  Não testei o produto ainda, mas ele veio corre...           2017-10-11
3  Não testei o produto ainda, mas ele veio corre...           2017-10-11
4  Não testei o produto ainda, mas ele veio corre...           2017-10-11

   review_answer_timestamp
0      2017-10-12 03:43:48
1      2017-10-12 03:43:48
2      2017-10-12 03:43:48
3      2017-10-12 03:43:48
4      2017-10-12 03:43:48

[5 rows x 44 columns]
```

Data Transformation and Feature Engineering

Data transformation involves modifying existing data into a new form that is easier to work with for analysis or visualization. The focus here is on:

Creating new columns based on existing data. Standardizing data types. Handling skewness in the data if required.

In [18]:
```
# Order Processing Time: Calculate the difference between when the order was purchased and when it was approved.

# Calculate order processing time (in days)
orders['order_processing_time'] = (orders['order_approved_at'] - orders['order_purchase_timestamp']).dt.days

# Display the first few rows to check the output
print("Order Processing Time:")
print(orders[['order_purchase_timestamp', 'order_approved_at', 'order_processing_time']].head())
```

```
Order Processing Time:
  order_purchase_timestamp   order_approved_at  order_processing_time
0      2017-10-02 10:56:33 2017-10-02 11:07:15                      0
1      2018-07-24 20:41:37 2018-07-26 03:24:27                      1
2      2018-08-08 08:38:49 2018-08-08 08:55:23                      0
3      2017-11-18 19:28:06 2017-11-18 19:45:59                      0
4      2018-02-13 21:18:39 2018-02-13 22:20:29                      0
```

In [19]:
```
# Total Order Value: Combine the price and freight_value in the order_items dataset to get the total order value.

# Calculate total order value
order_items['total_order_value'] = order_items['price'] + order_items['freight_value']

# Display the first few rows to check the output
print("Total Order Value:")
print(order_items[['price', 'freight_value', 'total_order_value']].head())
```

```
Total Order Value:
    price  freight_value  total_order_value
0   58.90          13.29              72.19
1  239.90          19.93             259.83
2  199.00          17.87             216.87
3   12.99          12.79              25.78
4  199.90          18.14             218.04
```

In [20]:
```
# Shipping Time: Calculate the difference between the estimated delivery date and the actual customer delivery date.

# Calculate shipping time (in days)
orders['shipping_time'] = (orders['order_delivered_customer_date'] - orders['order_estimated_delivery_date']).dt.days

# Display the first few rows to check the output
print("Shipping Time:")
print(orders[['order_delivered_customer_date', 'order_estimated_delivery_date', 'shipping_time']].head())
```

```
Shipping Time:
  order_delivered_customer_date order_estimated_delivery_date  shipping_time
0           2017-10-10 21:25:13                    2017-10-18             -8
1           2018-08-07 15:27:45                    2018-08-13             -6
2           2018-08-17 18:06:29                    2018-09-04            -18
3           2017-12-02 00:28:42                    2017-12-15            -13
4           2018-02-16 18:17:02                    2018-02-26            -10
```

In [21]:
```
# Calculate shipping time (in days)
orders['shipping_time'] = (orders['order_delivered_customer_date'] - orders['order_estimated_delivery_date']).dt.days

# Replace negative and zero shipping times with 1
orders.loc[orders['shipping_time'] < 1, 'shipping_time'] = 1

# Display the first few rows to check the output
print(orders[['order_delivered_customer_date', 'order_estimated_delivery_date', 'shipping_time']].head())
```

```
  order_delivered_customer_date order_estimated_delivery_date  shipping_time
0           2017-10-10 21:25:13                    2017-10-18              1
1           2018-08-07 15:27:45                    2018-08-13              1
2           2018-08-17 18:06:29                    2018-09-04              1
3           2017-12-02 00:28:42                    2017-12-15              1
4           2018-02-16 18:17:02                    2018-02-26              1
```

In [ ]:

In [22]:
```
# Extract Date Components: Extract useful information such as year, month, and day of the week from order_purchase_ti

# Extract year, month, and day of the week from purchase timestamp
orders['purchase_year'] = orders['order_purchase_timestamp'].dt.year
orders['purchase_month'] = orders['order_purchase_timestamp'].dt.month
orders['purchase_day_of_week'] = orders['order_purchase_timestamp'].dt.dayofweek

# Display the first few rows to check the output
print("Date Components (Year, Month, Day of Week):")
print(orders[['order_purchase_timestamp', 'purchase_year', 'purchase_month', 'purchase_day_of_week']].head())
```

```
Date Components (Year, Month, Day of Week):
  order_purchase_timestamp  purchase_year  purchase_month  \
0      2017-10-02 10:56:33           2017              10   
1      2018-07-24 20:41:37           2018               7   
2      2018-08-08 08:38:49           2018               8   
3      2017-11-18 19:28:06           2017              11   
4      2018-02-13 21:18:39           2018               2   

   purchase_day_of_week  
0                     0  
1                     1  
2                     2  
3                     5  
4                     1  
```

Calculate Total Spend per Customer:

In [23]:
```python
# Merge orders with order_items to associate orders with customers
merged_data = pd.merge(orders, order_items, on='order_id', how='left')

# Calculate total spend per customer
total_spend_per_customer = merged_data.groupby('customer_id')['total_order_value'].sum().reset_index()
total_spend_per_customer.columns = ['customer_id', 'total_spend']

# Check the result
print(total_spend_per_customer.head())
```

```
                        customer_id  total_spend
0  00012a2ce6f8dcda20d059ce98491703       114.74
1  000161a058600d5901f007fab4c27140        67.41
2  0001fd6190edaaf884bcaf3d49edf079       195.42
3  0002414f95344307404f0ace7a26f1d5       179.35
4  000379cdec625522490c315e70c7a9fb       107.01
```

Calculate Number of Orders per Customer:

In [24]:
```python
# Calculate number of orders per customer
number_of_orders_per_customer = merged_data.groupby('customer_id')['order_id'].nunique().reset_index()
number_of_orders_per_customer.columns = ['customer_id', 'number_of_orders']

# Check the result
print(number_of_orders_per_customer.head())
```

```
                        customer_id  number_of_orders
0  00012a2ce6f8dcda20d059ce98491703                 1
1  000161a058600d5901f007fab4c27140                 1
2  0001fd6190edaaf884bcaf3d49edf079                 1
3  0002414f95344307404f0ace7a26f1d5                 1
4  000379cdec625522490c315e70c7a9fb                 1
```

Calculate Purchase Frequency:

In [25]:
```python
# Calculate the first and last purchase date for each customer
customer_purchase_times = merged_data.groupby('customer_id').agg(
    first_purchase=('order_purchase_timestamp', 'min'),
    last_purchase=('order_purchase_timestamp', 'max')
).reset_index()

# Merge the number_of_orders and total_spend with the customer_purchase_times dataframe
customer_segmentation = pd.merge(total_spend_per_customer, number_of_orders_per_customer, on='customer_id')
customer_segmentation = pd.merge(customer_segmentation, customer_purchase_times, on='customer_id')

# Calculate purchase frequency (days between first and last purchase divided by number of orders)
customer_segmentation['purchase_frequency'] = (
    (customer_segmentation['last_purchase'] - customer_segmentation['first_purchase']).dt.days
    / customer_segmentation['number_of_orders']
)

# Check the result
print(customer_segmentation.head())
```

```
                        customer_id  total_spend  number_of_orders  \
0  00012a2ce6f8dcda20d059ce98491703       114.74                 1
1  000161a058600d5901f007fab4c27140        67.41                 1
2  0001fd6190edaaf884bcaf3d49edf079       195.42                 1
3  0002414f95344307404f0ace7a26f1d5       179.35                 1
4  000379cdec625522490c315e70c7a9fb       107.01                 1

       first_purchase       last_purchase  purchase_frequency
0 2017-11-14 16:08:26 2017-11-14 16:08:26                 0.0
1 2017-07-16 09:40:32 2017-07-16 09:40:32                 0.0
2 2017-02-28 11:06:43 2017-02-28 11:06:43                 0.0
3 2017-08-16 13:09:20 2017-08-16 13:09:20                 0.0
4 2018-04-02 13:42:17 2018-04-02 13:42:17                 0.0
```

Additional Aggregates (Revenue per Product Category & State):

Total Revenue per Product Category:

In [26]:
```python
# Merge products dataset with order_items to associate products with orders
merged_products = pd.merge(order_items, products, on='product_id', how='left')

# Calculate total revenue per product category
total_revenue_per_category = merged_products.groupby('product_category_name')['total_order_value'].sum().reset_index(
total_revenue_per_category.columns = ['product_category_name', 'total_revenue']

# Check the result
print(total_revenue_per_category.head())
```

```
       product_category_name  total_revenue
0                    Unknown      207705.09
1  agro_industria_e_comercio       78374.07
2                  alimentos       36664.44
3           alimentos_bebidas      19687.47
4                      artes       28247.81
```

Total Revenue per Customer State:

In [27]:
```
# Merge customers dataset with merged_data to get customer state associated with each order
merged_geo = pd.merge(merged_data, customers, on='customer_id', how='left')

# Calculate total revenue per customer state
total_revenue_per_state = merged_geo.groupby('customer_state')['total_order_value'].sum().reset_index()
total_revenue_per_state.columns = ['customer_state', 'total_revenue']

# Check the result
print(total_revenue_per_state.head())
```

```
  customer_state  total_revenue
0             AC       19575.33
1             AL       94172.49
2             AM       27585.47
3             AP       16141.81
4             BA      591137.81
```

Descriptive Statistics

In [28]:
```
# Descriptive Statistics for Sales (Total Order Value)
sales_stats = order_items['total_order_value'].describe()
print("Descriptive Statistics for Sales (Total Order Value):")
print(sales_stats)

# Descriptive Statistics for Freight Value
freight_stats = order_items['freight_value'].describe()
print("Descriptive Statistics for Freight Value:")
print(freight_stats)

# Descriptive Statistics for Customer Segments
total_spend_stats = customer_segmentation['total_spend'].describe()
orders_per_customer_stats = customer_segmentation['number_of_orders'].describe()
purchase_frequency_stats = customer_segmentation['purchase_frequency'].describe()

print("Descriptive Statistics for Total Spend per Customer:")
print(total_spend_stats)
print("Descriptive Statistics for Number of Orders per Customer:")
print(orders_per_customer_stats)
print("Descriptive Statistics for Purchase Frequency:")
print(purchase_frequency_stats)
```

```
Descriptive Statistics for Sales (Total Order Value):
count    112650.000000
mean        140.644059
std         190.724394
min           6.080000
25%          55.220000
50%          92.320000
75%         157.937500
max        6929.310000
Name: total_order_value, dtype: float64
Descriptive Statistics for Freight Value:
count    112650.000000
mean         19.990320
std          15.806405
min           0.000000
25%          13.080000
50%          16.260000
75%          21.150000
max         409.680000
Name: freight_value, dtype: float64
Descriptive Statistics for Total Spend per Customer:
count    96475.000000
mean       159.823264
std        218.797315
min          9.590000
25%         61.850000
50%        105.280000
75%        176.260000
max      13664.080000
Name: total_spend, dtype: float64
Descriptive Statistics for Number of Orders per Customer:
count    96475.0
mean         1.0
std          0.0
min          1.0
25%          1.0
50%          1.0
75%          1.0
max          1.0
Name: number_of_orders, dtype: float64
Descriptive Statistics for Purchase Frequency:
count    96475.0
mean         0.0
std          0.0
min          0.0
25%          0.0
50%          0.0
75%          0.0
max          0.0
Name: purchase_frequency, dtype: float64
```

Export Cleaned Data For Further Analysis

In [29]:

```python
# Export cleaned orders data
orders.to_csv('cleaned_orders.csv', index=False)

# Export cleaned order items data
order_items.to_csv('cleaned_order_items.csv', index=False)

# Export cleaned customers data
customers.to_csv('cleaned_customers.csv', index=False)

# Export cleaned products data
products.to_csv('cleaned_products.csv', index=False)

# Export customer segmentation data (if you have this created)
customer_segmentation.to_csv('customer_segmentation.csv', index=False)

# You can print a confirmation message after the exports
print("All datasets have been successfully exported as CSV files.")
```

```
All datasets have been successfully exported as CSV files.
```

In [30]:

```python
# Export cleaned order_reviews data
order_reviews.to_csv('cleaned_order_reviews.csv', index=False)

# Export cleaned order_payments data
order_payments.to_csv('cleaned_order_payments.csv', index=False)

# Export cleaned sellers data
sellers.to_csv('cleaned_sellers.csv', index=False)

# Export cleaned geolocation data
geolocation_cleaned.to_csv('cleaned_geolocation.csv', index=False)

print("All additional datasets have been successfully exported as CSV files.")
```

```
All additional datasets have been successfully exported as CSV files.
```