# Assignment -Week6 Project Milestone-3 Customer Segmentation EDA analysis

Course: DSC630 - Predictive Analytics

Instructor: Fadi Alsaleem

Team project

Team Members: Karthikeyan Chellamuthu, Subhashini Natarajan

In [1]:

```
# Import the required libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
# Load and inspect orders data
order_df =pd.read_csv("/Users/LENOVO/Desktop/BU/DSC 630/Assignment Submitted/olist_o
order_df.head()
```

Out[2]:

| | order_id | customer_id | order_status | order_purcha |
|---|---|---|---|---|
| 0 | e481f51cbdc54678b7cc49136f2d6af7 | 9ef432eb6251297304e76186b10a928d | delivered | 2017 |
| 1 | 53cdb2fc8bc7dce0b6741e2150273451 | b0830fb4747a6c6d20dea0b8c802d7ef | delivered | 2018 |
| 2 | 47770eb9100c2d0c44946d9cf07ec65d | 41ce2a54c0b03bf3443c3d931a367089 | delivered | 2018 |
| 3 | 949d5b44dbf5de918fe9c16f97b45f8a | f88197465ea7920adcdbec7375364d82 | delivered | 2017 |
| 4 | ad21c59c0840e6cb83a9ceb5573f8159 | 8ab97904e6daea8866dbdbc4fb7aad2c | delivered | 2018 |

In [3]:

```
#Load and inspect customer data
customer_df = pd.read_csv("/Users/LENOVO/Desktop/BU/DSC 630/Assignment Submitted/oli
customer_df.head()
```

Out[3]:

| | customer_id | customer_unique_id | customer_zip_code_prefix |
|---|---|---|---|
| 0 | 06b8999e2fba1a1fbc88172c00ba8bc7 | 861eff4711a542e4b93843c6dd7febb0 | 14409 |
| 1 | 18955e83d337fd6b2def6b18a428ac77 | 290c77bc529b7ac935b93aa66c333dc3 | 9790 |
| 2 | 4e7b3e00288586ebd08712fdd0374a03 | 060e732b5b29e8181a18229c7b0b2b5e | 1151 |
| 3 | b2b6027bc5c5109e529d4dc6358b12c3 | 259dac757896d24d7702b9acbbff3f3c | 8775 |
| 4 | 4f2d8ab171c80ec8364f7c12e35b23ad | 345ecd01c38d18a9036ed96c73b8d066 | 13056 |

In [4]:
```
#Load and inspect customer data
order_review_df = pd.read_csv("/Users/LENOVO/Desktop/BU/DSC 630/Assignment Submitted
order_review_df.head()
```

Out[4]:

| | review_id | order_id | review_score | review_com |
|---|---|---|---|---|
| **0** | 7bc2406110b926393aa56f80a40eba40 | 73fc7af87114b39712e6da79b0a377eb | 4 | |
| **1** | 80e641a11e56f04c1ad469d5645fdfde | a548910a1c6147796b98fdf73dbeba33 | 5 | |
| **2** | 228ce5500dc1d8e020d8d1322874b6f0 | f9e4b658b201a9f2ecdecbb34bed034b | 5 | |
| **3** | e64fb393e7b32834bb789ff8bb30750e | 658677c97b385a9be170737859d3511b | 5 | |
| **4** | f7c4243c7fe1938f181bec41a392bdeb | 8e6bfb81e283fa7e4f11123a3fb894f1 | 5 | |

In [5]:
```
#Load and inspect product data

product_df = pd.read_csv("/Users/LENOVO/Desktop/BU/DSC 630/Assignment Submitted/olis
product_df.head()
```

Out[5]:

| | product_id | product_category_name | product_name_lenght | product_descri |
|---|---|---|---|---|
| **0** | 1e9e8ef04dbcff4541ed26657ea517e5 | perfumaria | 40.0 | |
| **1** | 3aa071139cb16b67ca9e5dea641aaa2f | artes | 44.0 | |
| **2** | 96bd76ec8810374ed1b65e291975717f | esporte_lazer | 46.0 | |
| **3** | cef67bcfe19066a932b7673e239eb23d | bebes | 27.0 | |
| **4** | 9dc1a7de274444849c219cff195d0b71 | utilidades_domesticas | 37.0 | |

In [6]:
```
# Load and inspect order items data

order_items_df = pd.read_csv("/Users/LENOVO/Desktop/BU/DSC 630/Assignment Submitted/
order_items_df.head()
```

Out[6]:

| | order_id | order_item_id | product_id | |
|---|---|---|---|---|
| **0** | 00010242fe8c5a6d1ba2dd792cb16214 | 1 | 4244733e06e7ecb4970a6e2683c13e61 | 48436dade |
| **1** | 00018f77f2f0320c557190d7a144bdd3 | 1 | e5f2d52b802189ee658865ca93d83a8f | dd7ddc04e |
| **2** | 000229ec398224ef6ca0657da4fc703e | 1 | c777355d18b72b67abbeef9df44fd0fd | 5b51032ed |
| **3** | 00024acbcdf0a6daa1e931b038114c75 | 1 | 7634da152a4610f1595efa32f14722fc | 9d7a1d34a |
| **4** | 00042b26cf59d7ce69dfabb4e55b4fd9 | 1 | ac6c3623068f30de03045865e4e10089 | df560393f |

In [7]:
```
# Merge customer and order data based on customer id

olist_df1 = pd.merge(
    customer_df,
    order_df,
    how="inner",
    on= ["customer_id"]
)

olist_df1.head()
```

Out[7]:

| | customer_id | customer_unique_id | customer_zip_code_prefix |
|---|---|---|---|
| **0** | 06b8999e2fba1a1fbc88172c00ba8bc7 | 861eff4711a542e4b93843c6dd7febb0 | 14409 |
| **1** | 18955e83d337fd6b2def6b18a428ac77 | 290c77bc529b7ac935b93aa66c333dc3 | 9790 |
| **2** | 4e7b3e00288586ebd08712fdd0374a03 | 060e732b5b29e8181a18229c7b0b2b5e | 1151 |
| **3** | b2b6027bc5c5109e529d4dc6358b12c3 | 259dac757896d24d7702b9acbbff3f3c | 8775 |
| **4** | 4f2d8ab171c80ec8364f7c12e35b23ad | 345ecd01c38d18a9036ed96c73b8d066 | 13056 |

In [8]:

```
# Merge order review data with the above based on order id
olist_df2 = pd.merge(
    olist_df1,
    order_review_df,
    how="inner",
    on= ["order_id"]
)

olist_df2.head()
```

Out[8]:

| | customer_id | customer_unique_id | customer_zip_code_prefix |
|---|---|---|---|
| **0** | 06b8999e2fba1a1fbc88172c00ba8bc7 | 861eff4711a542e4b93843c6dd7febb0 | 14409 |
| **1** | 18955e83d337fd6b2def6b18a428ac77 | 290c77bc529b7ac935b93aa66c333dc3 | 9790 |
| **2** | 4e7b3e00288586ebd08712fdd0374a03 | 060e732b5b29e8181a18229c7b0b2b5e | 1151 |
| **3** | b2b6027bc5c5109e529d4dc6358b12c3 | 259dac757896d24d7702b9acbbff3f3c | 8775 |
| **4** | 4f2d8ab171c80ec8364f7c12e35b23ad | 345ecd01c38d18a9036ed96c73b8d066 | 13056 |

In [9]:

```
# Merge order items data based on order id
olist_df3=pd.merge(
    olist_df2,
    order_items_df,
    how="inner",
    on= ["order_id"]
)
```

```
olist_df3.head()
```

Out[9]:

| | customer_id | customer_unique_id | customer_zip_code_prefix |
|---|---|---|---|
| **0** | 06b8999e2fba1a1fbc88172c00ba8bc7 | 861eff4711a542e4b93843c6dd7febb0 | 14409 |
| **1** | 18955e83d337fd6b2def6b18a428ac77 | 290c77bc529b7ac935b93aa66c333dc3 | 9790 |
| **2** | 4e7b3e00288586ebd08712fdd0374a03 | 060e732b5b29e8181a18229c7b0b2b5e | 1151 |
| **3** | b2b6027bc5c5109e529d4dc6358b12c3 | 259dac757896d24d7702b9acbbff3f3c | 8775 |
| **4** | 4f2d8ab171c80ec8364f7c12e35b23ad | 345ecd01c38d18a9036ed96c73b8d066 | 13056 |

5 rows × 24 columns

In [10]:

```
# Merge product data based on product id

olist_df4 = pd.merge(
    olist_df3,
    product_df,
    how="inner",
    on= ["product_id"]
)

olist_df4.head()
```

Out[10]:

| | customer_id | customer_unique_id | customer_zip_code_prefix |
|---|---|---|---|
| **0** | 06b8999e2fba1a1fbc88172c00ba8bc7 | 861eff4711a542e4b93843c6dd7febb0 | 14409 |
| **1** | 8912fc0c3bbf1e2fbf35819e21706718 | 9eae34bbd3a474ec5d07949ca7de67c0 | 68030 |
| **2** | 8912fc0c3bbf1e2fbf35819e21706718 | 9eae34bbd3a474ec5d07949ca7de67c0 | 68030 |
| **3** | f0ac8e5a239118859b1734e1087cbb1f | 3c799d181c34d51f6d44bbbc563024db | 92480 |
| **4** | 6bc8d08963a135220ed6c6d098831f84 | 23397e992b09769faf5e66f9e171a241 | 25931 |

5 rows × 32 columns

In [11]:

```
# Create a final dataframe with only the required attributes

olist_df = olist_df4[['customer_id','customer_unique_id', 'order_id', 'product_id','
olist_df.head()
```

Out[11]:

| | customer_id | customer_unique_id |
|---|---|---|

| | customer_id | customer_unique_id | |
|---|---|---|---|
| **0** | 06b8999e2fba1a1fbc88172c00ba8bc7 | 861eff4711a542e4b93843c6dd7febb0 | 00e7ee1b050b8499577073a |
| **1** | 8912fc0c3bbf1e2fbf35819e21706718 | 9eae34bbd3a474ec5d07949ca7de67c0 | c1d2b34febe9cd269e37811 |
| **2** | 8912fc0c3bbf1e2fbf35819e21706718 | 9eae34bbd3a474ec5d07949ca7de67c0 | c1d2b34febe9cd269e37811 |
| **3** | f0ac8e5a239118859b1734e1087cbb1f | 3c799d181c34d51f6d44bbbc563024db | b1a5d5365d330d10485e020 |
| **4** | 6bc8d08963a135220ed6c6d098831f84 | 23397e992b09769faf5e66f9e171a241 | 2e604b3614664aa66867856 |

In [12]:

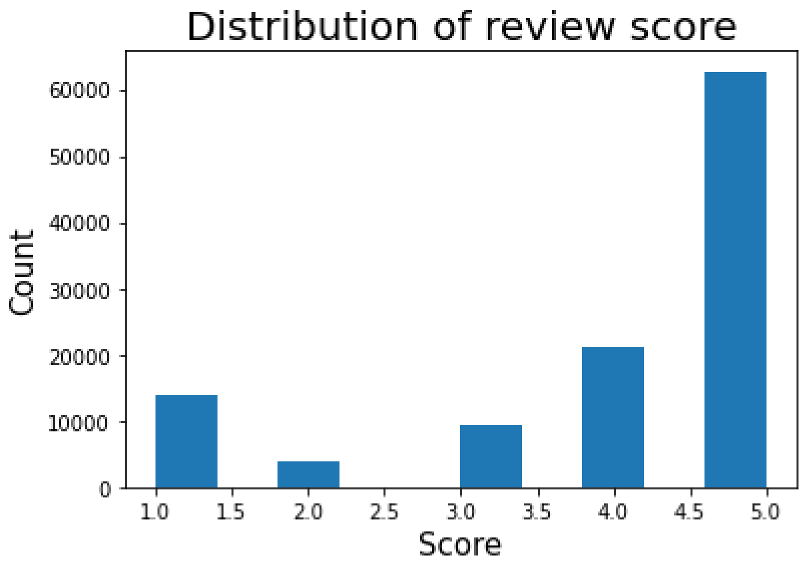
```
#Check the number of records

len(olist_df.customer_unique_id.unique())
```

Out[12]:

```
94721
```

In [13]:

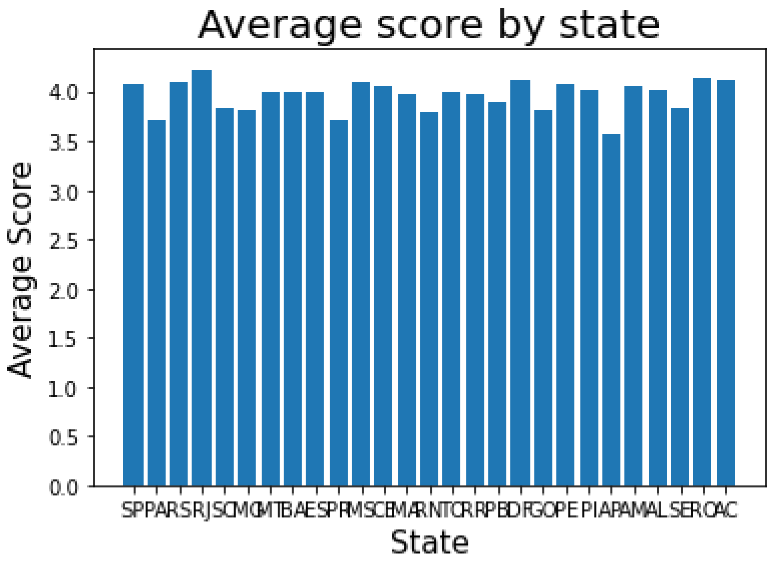
```
# Obtain the distribution of review score

olist_df=olist_df.dropna()
plt.hist(olist_df.review_score)
plt.xlabel("Score", fontsize=15)
plt.ylabel("Count", fontsize=15)
plt.title("Distribution of review score", fontsize=20)
plt.show()
```



The above distribution shows that number of records with higher ratings are quite high.

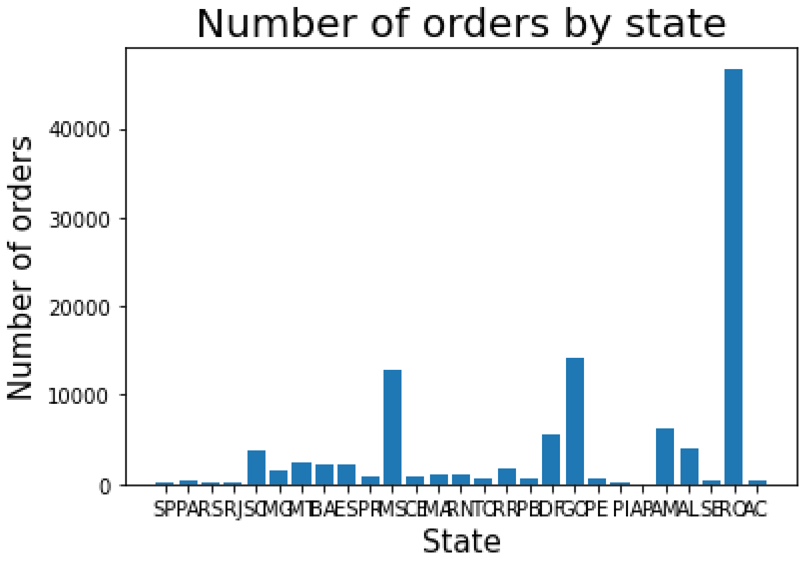
In [14]:

```
# Obtain distribution of review score by state

x = olist_df.customer_state.unique()
y = olist_df.groupby("customer_state")['review_score'].mean()

plt.bar(x,y)
plt.ylabel("Average Score", fontsize=15)
plt.xlabel("State", fontsize=15)
plt.title("Average score by state", fontsize=20)

plt.show()
```



In [15]:

```
# Obtain count of orders by state

x1 = olist_df.customer_state.unique()
y1 = olist_df.groupby("customer_state")['customer_unique_id'].count()

plt.bar(x1,y1)
plt.xlabel("State", fontsize=15)
plt.ylabel("Number of orders", fontsize=15)
plt.title("Number of orders by state", fontsize=20)
plt.show()
```



In [16]:

```
# Obtain data type info for every field
olist_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 110774 entries, 0 to 112371
Data columns (total 14 columns):
 #   Column                    Non-Null Count   Dtype 
---  ------                    --------------   ----- 
 0   customer_id               110774 non-null  object
 1   customer_unique_id        110774 non-null  object
 2   order_id                  110774 non-null  object
 3   product_id                110774 non-null  object
 4   review_id                 110774 non-null  object
 5   order_purchase_timestamp  110774 non-null  object
 6   customer_city             110774 non-null  object
 7   customer_state            110774 non-null  object
 8   product_category_name     110774 non-null  object
 9   review_score              110774 non-null  int64
 10  order_item_id             110774 non-null  int64
 11  review_creation_date      110774 non-null  object
 12  price                     110774 non-null  float64
 13  freight_value             110774 non-null  float64
dtypes: float64(2), int64(2), object(10)
memory usage: 12.7+ MB
```

In [21]:

```
# Change the data type of order purchase timestamp from object to datatime

olist_df['order_purchase_timestamp']=pd.to_datetime(olist_df['order_purchase_timesta

olist_df.sort_values(by=['order_purchase_timestamp'], inplace=True)
```

In [25]:

```
# Obtain the number of orders by year month

order_by_month = olist_df[['order_purchase_timestamp']].groupby(olist_df['order_purc
order_by_month.head()
order_by_month = order_by_month.rename(columns = {'order_purchase_timestamp': 'num_o
order_by_month = order_by_month.reset_index()
order_by_month['month_year'] = order_by_month['order_purchase_timestamp'].dt.strftim
order_by_month.head()
```

Out[25]:

| | order_purchase_timestamp | num_of_orders | month_year |
|---|---|---|---|
| **0** | 2016-09 | 6 | 2016-Sep |
| **1** | 2016-10 | 359 | 2016-Oct |
| **2** | 2016-12 | 1 | 2016-Dec |
| **3** | 2017-01 | 942 | 2017-Jan |
| **4** | 2017-02 | 1893 | 2017-Feb |

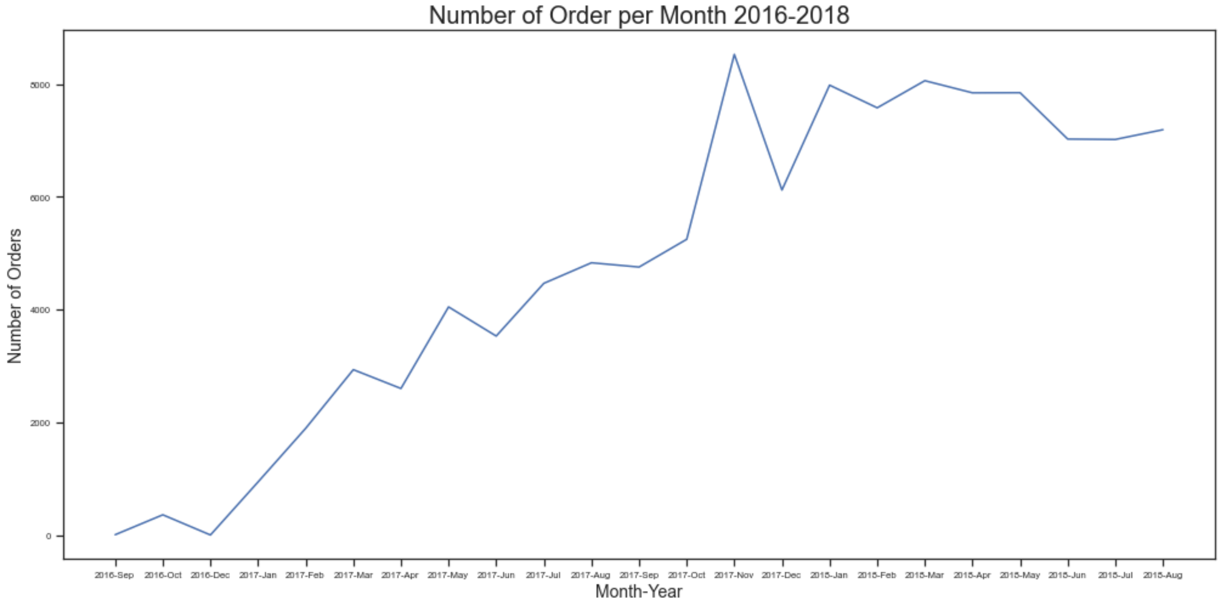
In [23]:

```
order_by_month.head(40)
order_by_month = order_by_month[order_by_month['month_year']!='2018-Sep']
```

An outlier observed for sep-2018, which was removed.

In [24]:

```
# Plot the trend of orders


plt.figure(figsize=(17,8))

sns.set_theme(style="ticks", font_scale=0.7)

order_by_month_trend=sns.lineplot(x = 'month_year', y = 'num_of_orders', data = orde

order_by_month_trend.set_title('Number of Order per Month 2016-2018',fontsize = 20)

order_by_month_trend.set_xlabel('Month-Year',fontsize = 14)

order_by_month_trend.set_ylabel('Number of Orders',fontsize = 14)

plt.show()
```



The number of orders from the plot above can be seen with steady increase apart from few occasional dips and the latest data at the end of 2018 is well above 70000 orders.

In [27]:

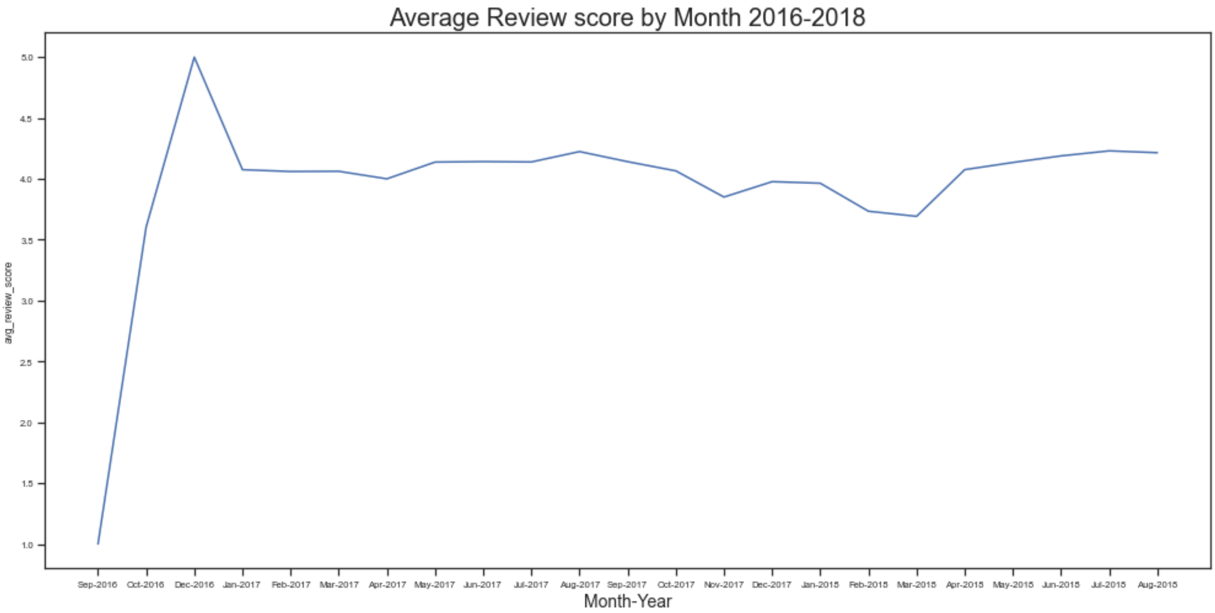
```
# Obtain the average review score by year month

reviewscore_by_month = olist_df[['review_score']].groupby(olist_df['order_purchase_t
reviewscore_by_month.head()
reviewscore_by_month = reviewscore_by_month.rename(columns = {'review_score': 'avg_r
reviewscore_by_month = reviewscore_by_month.reset_index()
reviewscore_by_month['month_year'] = reviewscore_by_month['order_purchase_timestamp'
reviewscore_by_month = reviewscore_by_month[reviewscore_by_month['month_year']!='Sep
reviewscore_by_month.tail()
```

Out[27]:

| | order_purchase_timestamp | avg_review_score | month_year |
|---|---|---|---|
| **18** | 2018-04 | 4.075481 | Apr-2018 |
| **19** | 2018-05 | 4.134481 | May-2018 |
| **20** | 2018-06 | 4.188523 | Jun-2018 |
| **21** | 2018-07 | 4.229903 | Jul-2018 |
| **22** | 2018-08 | 4.214773 | Aug-2018 |

In [28]:

```
# Plot the average review score by year month

plt.figure(figsize=(17,8))
sns.set_theme(style="ticks", font_scale=0.7)
reviewscore_by_month=sns.lineplot(x = 'month_year', y = 'avg_review_score', data = r
reviewscore_by_month.set_title('Average Review score by Month 2016-2018',fontsize = 
reviewscore_by_month.set_xlabel('Month-Year',fontsize = 14)
order_by_month_trend.set_ylabel('Average Review Score',fontsize = 14)
plt.show()
```



The average review score for the products from the plot above can be seen as pretty constant and above 4, except for a brief dip between Jan 2018 - April 2018.

In [29]:

```
# Obtain the number of orders placed by every customer
num_of_orders = olist_df[['order_id']].groupby(olist_df['customer_unique_id']).agg({
num_of_orders = num_of_orders.rename(columns = {'order_id': 'order_count'})
num_of_orders = num_of_orders.reset_index()
num_of_orders.order_count.unique()
```

Out[29]:

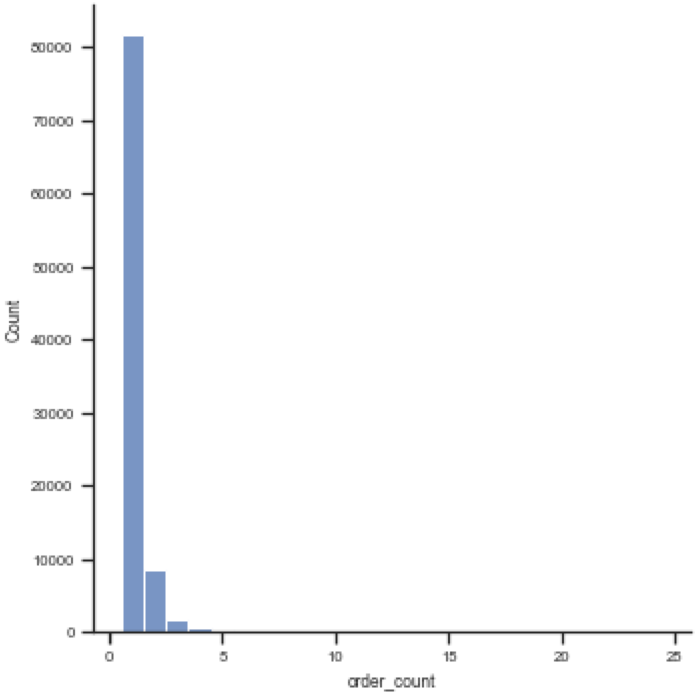
```
array([ 1,  2,  4,  3,  7,  5,  6, 12, 10,  8, 18, 15,  9, 14, 21, 11, 13,
       20, 24], dtype=int64)
```

In [30]:

```
# Obtain distribution of number of orders by a customer

sns.displot(num_of_orders, x="order_count",discrete=True)
```

Out[30]:

```
<seaborn.axisgrid.FacetGrid at 0x1759d0492b0>
```



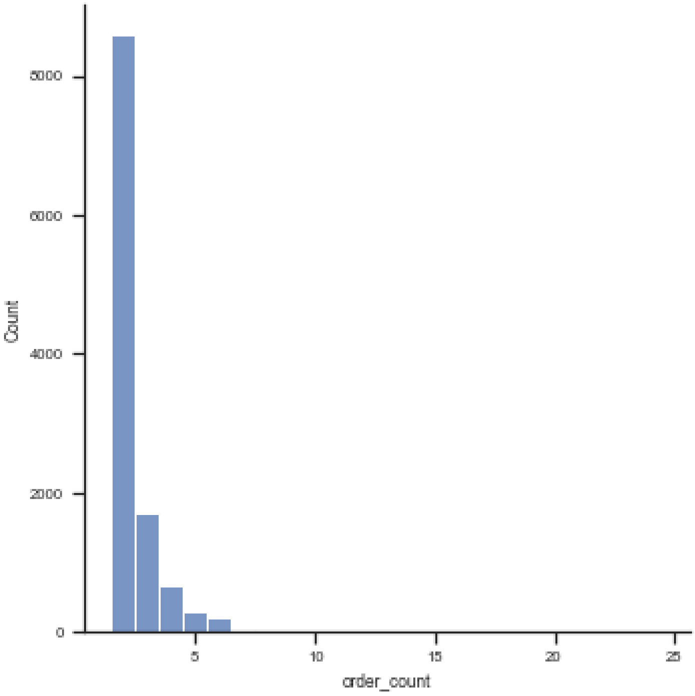
The number of one time orders is the highest, followed by twice and thrice.

In [31]:

```
# Obtain number of multiple orders by the customer and plot the distribution

multiple_orders = num_of_orders[num_of_orders.order_count>1]
sns.displot(multiple_orders, x="order_count",discrete=True)
```

Out[31]:

```
<seaborn.axisgrid.FacetGrid at 0x175a062d220>
```



Number of three orders and more, by the same customer is lesser than 20,000.

In [ ]: