```
# import required packages to work with
import numpy as np
import pandas as pd
import datetime as dt

# import required packages for data visuals
import matplotlib.pyplot as plt
import seaborn as sns
```

Import datasets

```
orders = pd.read_csv("C://Users//SANAM KANDAR//OneDrive//Desktop//Case Study//PYTHON//Python Foundation End to End Case Study E-Commerce
order_payment  = pd.read_csv("C://Users//SANAM KANDAR//OneDrive//Desktop//Case Study//PYTHON//Python Foundation End to End Case Study E-C
order_items  = pd.read_csv("C://Users//SANAM KANDAR//OneDrive//Desktop//Case Study//PYTHON//Python Foundation End to End Case Study E-Com
order_review  = pd.read_csv("C://Users//SANAM KANDAR//OneDrive//Desktop//Case Study//PYTHON//Python Foundation End to End Case Study E-Co
customer = pd.read_csv("C://Users//SANAM KANDAR//OneDrive//Desktop//Case Study//PYTHON//Python Foundation End to End Case Study E-Commerc
seller = pd.read_csv("C://Users//SANAM KANDAR//OneDrive//Desktop//Case Study//PYTHON//Python Foundation End to End Case Study E-Commerce
product = pd.read_csv("C://Users//SANAM KANDAR//OneDrive//Desktop//Case Study//PYTHON//Python Foundation End to End Case Study E-Commerce
geo_location = pd.read_csv("C://Users//SANAM KANDAR//OneDrive//Desktop//Case Study//PYTHON//Python Foundation End to End Case Study E-Com
```

DATA PREPRATION | DATA CLEANING | DATA WRANGLING

```
# convert into date columns
orders.order_purchase_timestamp = pd.to_datetime(orders.order_purchase_timestamp, format="%m/%d/%Y %H:%M")
orders.order_approved_at = pd.to_datetime(orders.order_approved_at, format="%m/%d/%Y %H:%M")
orders.order_delivered_carrier_date = pd.to_datetime(orders.order_delivered_carrier_date, format="%m/%d/%Y %H:%M")
orders.order_delivered_customer_date = pd.to_datetime(orders.order_delivered_customer_date, format="%m/%d/%Y %H:%M")
orders.order_estimated_delivery_date = pd.to_datetime(orders.order_estimated_delivery_date, format="%m/%d/%Y %H:%M")
```

```
order_items.shipping_limit_date = pd.to_datetime(order_items.shipping_limit_date, format="%m/%d/%Y %H:%M")
```

```
order_review.review_creation_date = pd.to_datetime(order_review.review_creation_date, format="%m/%d/%Y %H:%M")
order_review.review_answer_timestamp = pd.to_datetime(order_review.review_answer_timestamp, format="%m/%d/%Y %H:%M")
```

```
######################################################################
```

Merge all table - 360 dataset

```
tb = pd.merge(left=orders, right=customer, how="inner", on="customer_id" )
tb1 =pd.merge(left=tb, right=order_items, how="inner", on="order_id")
tb2= pd.merge(left=tb1 , right=order_payment, how="inner", on="order_id")
tb3 = pd.merge(left=tb2 , right=order_review, how="inner", on="order_id")
tb4 = pd.merge(left=tb3 , right=product, how="inner", on="product_id")
final_order = pd.merge(left=tb4 , right=seller, how="inner", on="seller_id")

final_order
```

	order_id	customer_id	order_status	order_purchase_timestamp	order_approved_
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-10-02 10:56:00	2017-10-02 11:07:(
1	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-10-02 10:56:00	2017-10-02 11:07:(
2	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-10-02 10:56:00	2017-10-02 11:07:(
3	128e10d95713541c87cd1a2e48201934	a20e8105f23924cd00833fd87daa0831	delivered	2017-08-15 18:29:00	2017-08-15 20:05:(
4	0e7e841ddf8f8f2de2bad69267ecfbcf	26c7ac168e1433912a51b924fbd34d34	delivered	2017-08-02 18:24:00	2017-08-02 18:43:(
...	...	...	...	...	
118310	1ab38815794efa43d269d62b98dae815	a0b67404d84a70ef420a7f99ad6b190a	delivered	2018-07-01 10:23:00	2018-07-05 16:17:(
118311	b159d0ce7cd881052da94fa165617b05	e0c3bc5ce0836b975d6b2a8ce7bb0e3e	canceled	2017-03-11 19:51:00	2017-03-11 19:51:(
118312	735dce2d574afe8eb87e80a3d6229c48	d531d01affc2c55769f6b9ed410d8d3c	delivered	2018-07-24 09:46:00	2018-07-24 11:24:(
118313	25d2bfa43663a23586afd12f15b542e7	9d8c06734fde9823ace11a4b5929b5a7	delivered	2018-05-22 21:13:00	2018-05-22 21:35:(
118314	1565f22aa9452ff278638e87cc895678	56772dfbcbe7df908a284ff0d53adf7d	delivered	2018-05-15 17:41:00	2018-05-16 03:35:(

118315 rows × 37 columns

```
final_order.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 118315 entries, 0 to 118314
Data columns (total 37 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   order_id                       118315 non-null  object
 1   customer_id                    118315 non-null  object
 2   order_status                   118315 non-null  object
 3   order_purchase_timestamp       118315 non-null  datetime64[ns]
 4   order_approved_at              118300 non-null  datetime64[ns]
 5   order_delivered_carrier_date   117061 non-null  datetime64[ns]
 6   order_delivered_customer_date  115727 non-null  datetime64[ns]
 7   order_estimated_delivery_date  118315 non-null  datetime64[ns]
 8   customer_unique_id             118315 non-null  object
 9   customer_zip_code_prefix       118315 non-null  int64
 10  customer_city                  118315 non-null  object
 11  customer_state                 118315 non-null  object
 12  order_item_id                  118315 non-null  int64
 13  product_id                     118315 non-null  object
 14  seller_id                      118315 non-null  object
 15  shipping_limit_date            118315 non-null  datetime64[ns]
 16  price                          118315 non-null  float64
 17  freight_value                  118315 non-null  float64
 18  payment_sequential             118315 non-null  int64
 19  payment_type                   118315 non-null  object
 20  payment_installments           118315 non-null  int64
 21  payment_value                  118315 non-null  float64
 22  review_id                      118315 non-null  object
 23  review_score                   118315 non-null  int64
 24  review_creation_date           118315 non-null  datetime64[ns]
 25  review_answer_timestamp        118315 non-null  datetime64[ns]
 26  product_category_name          116581 non-null  object
 27  product_name_lenght            116606 non-null  float64
 28  product_description_lenght     116606 non-null  float64
 29  product_photos_qty             116606 non-null  float64
 30  product_weight_g               118295 non-null  float64
 31  product_length_cm              118295 non-null  float64
 32  product_height_cm              118295 non-null  float64
 33  product_width_cm               118295 non-null  float64
 34  seller_zip_code_prefix         118315 non-null  int64
 35  seller_city                    116664 non-null  object
 36  seller_state                   116664 non-null  object
dtypes: datetime64[ns](8), float64(10), int64(6), object(13)
memory usage: 34.3+ MB
```

```
final_order.sort_values(by="order_purchase_timestamp", inplace=True, ignore_index=True)
```

EDA | DATA PROFILING | DATA INSPECTION

1. Perform Detailed exploratory analysis

a. Define & calculate high level metrics like (Total Revenue, Total quantity, Total

products, Total categories, Total sellers, Total locations, Total channels, Total payment methods etc…)

```
Total_Revenue = final_order.loc[::,["order_id" , "product_category_name" , "payment_value"]].payment_value.sum()
Total_Quantity = final_order.loc[final_order.order_status != "canceled",["order_id"]].shape[0]
Total_Products = final_order.product_id.nunique()
Total_Categories = final_order.product_category_name.nunique()
Total_customers = final_order.customer_id.nunique()
Total_sellers = final_order.seller_id.nunique()
Total_location = geo_location.geolocation_zip_code_prefix.nunique()
Total_payment_method = final_order.payment_type.nunique()
Total_order_status = np.round((final_order.order_status.value_counts() / final_order.order_status.count())*100, 2)

print(f"Total revenue generated - {Total_Revenue}")
print(f"Total quantity ordered - {Total_Quantity}")
print(f"Total products we have - {Total_Products}")
print(f"Total categories we have - {Total_Categories}")
print(f"Total no. of seller we have, who help in our business - {Total_sellers}")
print(f"Total no. of location where we have our services - {Total_location}")
print(f"Total no. of customer who are in our business - {Total_customers}")
print(f"Total no. of payment methods we are accepting - {Total_payment_method}")
print("\n")
print(f"Percentage of order status -\n{Total_order_status}")
```

```
Total revenue generated - 20418288.15
Total quantity ordered - 117745
Total products we have - 32951
Total categories we have - 71
Total no. of seller we have, who help in our business - 3095
Total no. of location where we have our services - 19015
Total no. of customer who are in our business - 98665
Total no. of payment methods we are accepting - 4


Percentage of order status -
delivered      97.81
shipped         1.06
canceled        0.48
invoiced        0.32
processing      0.32
unavailable     0.01
approved        0.00
Name: order_status, dtype: float64
```

b. Understanding how many new customers acquired every month

```
cust_data =  final_order.loc[::,["order_id", "customer_id" , "order_purchase_timestamp"]]
```

```
cust_data
```

	order_id	customer_id	order_purchase_timestamp
0	2e7a8482f6fb09756ca50c10d7bfc047	08c5351a6aca1c1589a38f244edeee9d	2016-09-04 21:15:00
1	2e7a8482f6fb09756ca50c10d7bfc047	08c5351a6aca1c1589a38f244edeee9d	2016-09-04 21:15:00
2	e5fa5a7210941f7d56d0208e4e071d35	683c54fc24d40ee9f8a6fc179fd9856c	2016-09-05 00:15:00
3	71303d7e93b399f5bcd537d124c0bcfa	b106b360fe2ef8849fbbd056f777b4d5	2016-10-02 22:07:00
4	3b697a20d9e427646d92567910af6d57	355077684019f7f60a031656bd7262b8	2016-10-03 09:44:00
...	...	...	...
118310	0b223d92c27432930dfe407c6aea3041	e60df9449653a95af4549bbfcb18a6eb	2018-08-29 14:18:00
118311	0b223d92c27432930dfe407c6aea3041	e60df9449653a95af4549bbfcb18a6eb	2018-08-29 14:18:00
118312	03ef5dedbe7492bdae72eec50764c43f	496630b6740bcca28fce9ba50d8a26ef	2018-08-29 14:52:00
118313	35a972d7f8436f405b56e36add1a7140	898b7fee99c4e42170ab69ba59be0a8b	2018-08-29 15:00:00
118314	54282e97f61c23b78330c15b154c867d	4b7decb9b58e2569548b8b4c8e20e8d7	2018-09-03 09:06:00

118315 rows × 3 columns

```
# Get the first purchase date of every customers

first_purchase = cust_data.groupby(by="customer_id").agg({"order_purchase_timestamp": "min"})
first_purchase["order_month"] = first_purchase.order_purchase_timestamp.dt.strftime("%B-%Y") # created for summary purpose
```

```
first_purchase["month_no"] = first_purchase.order_purchase_timestamp.dt.month               # created for sorting purpose
first_purchase["year"] = first_purchase.order_purchase_timestamp.dt.year                     # created for sorting purpose
```

```
first_purchase
```

customer_id	order_purchase_timestamp	order_month	month_no	year
00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:00	November-2017	11	2017
000161a058600d5901f007fab4c27140	2017-07-16 09:40:00	July-2017	7	2017
0001fd6190edaaf884bcaf3d49edf079	2017-02-28 11:06:00	February-2017	2	2017
0002414f95344307404f0ace7a26f1d5	2017-08-16 13:09:00	August-2017	8	2017
000379cdec625522490c315e70c7a9fb	2018-04-02 13:42:00	April-2018	4	2018
...	...	...	...	...
fffcb937e9dd47a13f05ecb8290f4d3e	2018-03-17 00:55:00	March-2018	3	2018
fffecc9f79fd8c764f843e9951b11341	2018-03-29 16:59:00	March-2018	3	2018
fffeda5b6d849fbd39689bb92087f431	2018-05-22 13:36:00	May-2018	5	2018
ffff42319e9b2d713724ae527742af25	2018-06-13 16:57:00	June-2018	6	2018
ffffa3172527f765de70084a7e53aae8	2017-09-02 11:53:00	September-2017	9	2017

98665 rows × 4 columns

```
# got which customer made first purchase in which month, will be consider as a new customer

first_purchase.sort_values(by="order_purchase_timestamp")
```

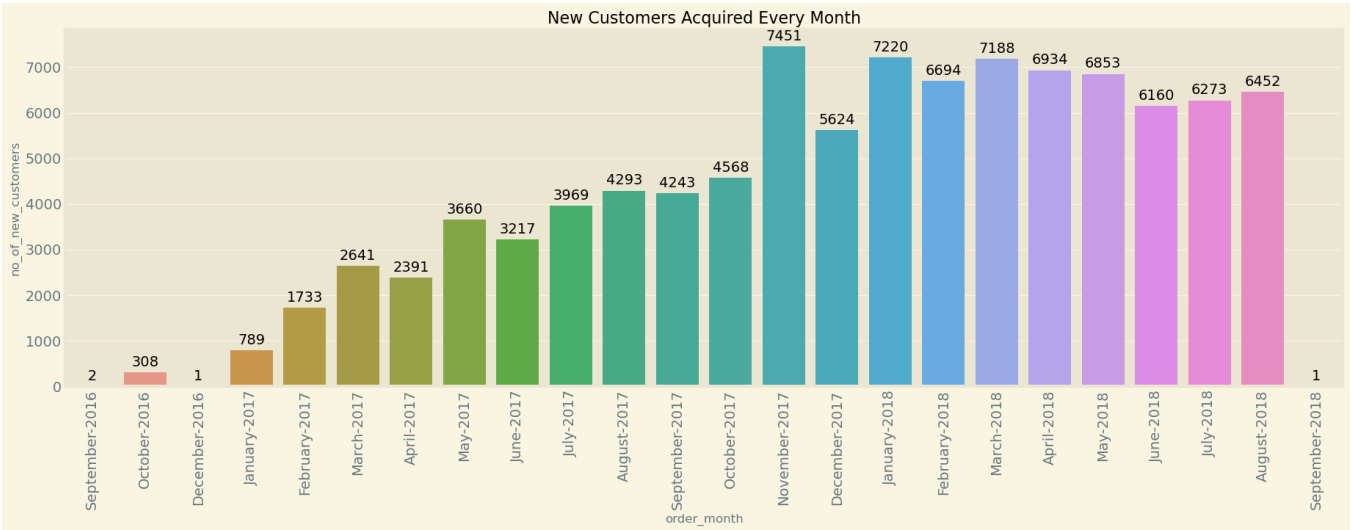
customer_id	order_purchase_timestamp	order_month	month_no	year
08c5351a6aca1c1589a38f244edeee9d	2016-09-04 21:15:00	September-2016	9	2016
683c54fc24d40ee9f8a6fc179fd9856c	2016-09-05 00:15:00	September-2016	9	2016
b106b360fe2ef8849fbbd056f777b4d5	2016-10-02 22:07:00	October-2016	10	2016
355077684019f7f60a031656bd7262b8	2016-10-03 09:44:00	October-2016	10	2016
7ec40b22510fdbea1b08921dd39e63d8	2016-10-03 16:56:00	October-2016	10	2016
...	...	...	...	...
6e353700bc7bcdf6ebc15d6de16d7002	2018-08-29 14:18:00	August-2018	8	2018
e60df9449653a95af4549bbfcb18a6eb	2018-08-29 14:18:00	August-2018	8	2018
496630b6740bcca28fce9ba50d8a26ef	2018-08-29 14:52:00	August-2018	8	2018
898b7fee99c4e42170ab69ba59be0a8b	2018-08-29 15:00:00	August-2018	8	2018
4b7decb9b58e2569548b8b4c8e20e8d7	2018-09-03 09:06:00	September-2018	9	2018

98665 rows × 4 columns

```
# got the numbers of new customers by month-year

new_cust = pd.crosstab(index=[first_purchase.year, first_purchase.month_no, first_purchase.order_month],
                       columns="no_of_new_customers",
                       values=first_purchase.index,
                       aggfunc="count").reset_index()

new_cust[["order_month", "no_of_new_customers"]]
```

col_0	order_month	no_of_new_customers
0	September-2016	2
1	October-2016	308
2	December-2016	1
3	January-2017	789
4	February-2017	1733
5	March-2017	2641
6	April-2017	2391
7	May-2017	3660
8	June-2017	3217
9	July-2017	3969
10	August-2017	4293
11	September-2017	4243
12	October-2017	4568
13	November-2017	7451
14	December-2017	5624
15	January-2018	7220
16	February-2018	6694
17	March-2018	7188
18	April-2018	6934
19	May-2018	6853
20	June-2018	6160
21	July-2018	6273
22	August-2018	6452
23	September-2018	1

```
plt.figure(figsize=(20,6))

linechart = sns.barplot(x=new_cust.order_month, y=new_cust.no_of_new_customers)

linechart.set_xticklabels(labels=new_cust.order_month, rotation=90, horizontalalignment='center')

for i in linechart.containers:
    labels= [int(x.get_height()) for x in i]
    linechart.bar_label(i, labels=labels, padding=3)

plt.title("New Customers Acquired Every Month")
plt.show()
```



c. Understand the retention of customers on month on month basis

```
data = orders.loc[::,["customer_id","order_purchase_timestamp"]]

data
```

	customer_id	order_purchase_timestamp
0	9ef432eb6251297304e76186b10a928d	2017-10-02 10:56:00
1	b0830fb4747a6c6d20dea0b8c802d7ef	2018-07-24 20:41:00
2	41ce2a54c0b03bf3443c3d931a367089	2018-08-08 08:38:00
3	f88197465ea7920adcdbec7375364d82	2017-11-18 19:28:00
4	8ab97904e6daea8866dbdbc4fb7aad2c	2018-02-13 21:18:00
...	...	...
99436	39bd1228ee8140590ac3aca26f2dfe00	2017-03-09 09:54:00
99437	1fca14ff2861355f6e5f14306ff977a7	2018-02-06 12:58:00
99438	1aa71eb042121263aafbe80c1b562c9c	2017-08-27 14:46:00
99439	b331b74b18dc79bcdf6532d51e1637c1	2018-01-08 21:28:00
99440	edb027a75a1449115f6b43211ae02a24	2018-03-08 20:57:00

99441 rows × 2 columns

```
# create invoice months to get year, month

data["Invoice_month"] = data["order_purchase_timestamp"].apply(lambda x: dt.datetime(x.year, x.month, 1))
```

```
#create cohort month in which customer make first purchase

data["cohort_month"] = data.groupby("customer_id")["Invoice_month"].transform("min")
```

```
# func to get day month year from a date columns

def get_date_elements(df, col):
    year = df[col].dt.year
    month = df[col].dt.month
    day  = df[col].dt.day
    return day, month, year
```

```
_, invoice_month, invoice_year = get_date_elements(data, "Invoice_month")
_, FirstPurchase_month, FirstPurchase_year = get_date_elements(data, "cohort_month")
```

```
# cohort_index -For how many months is the customer in this business?

year_diff = invoice_year - FirstPurchase_year
month_diff = invoice_month - FirstPurchase_month
data["cohort_index"] = year_diff*12+month_diff+1
```

```
data
```

	customer_id	order_purchase_timestamp	Invoice_month	cohort_month	cohort_index
0	9ef432eb6251297304e76186b10a928d	2017-10-02 10:56:00	2017-10-01	2017-10-01	1
1	b0830fb4747a6c6d20dea0b8c802d7ef	2018-07-24 20:41:00	2018-07-01	2018-07-01	1
2	41ce2a54c0b03bf3443c3d931a367089	2018-08-08 08:38:00	2018-08-01	2018-08-01	1
3	f88197465ea7920adcdbec7375364d82	2017-11-18 19:28:00	2017-11-01	2017-11-01	1
4	8ab97904e6daea8866dbdbc4fb7aad2c	2018-02-13 21:18:00	2018-02-01	2018-02-01	1
...	...	...	...	...	...
99436	39bd1228ee8140590ac3aca26f2dfe00	2017-03-09 09:54:00	2017-03-01	2017-03-01	1
99437	1fca14ff2861355f6e5f14306ff977a7	2018-02-06 12:58:00	2018-02-01	2018-02-01	1
99438	1aa71eb042121263aafbe80c1b562c9c	2017-08-27 14:46:00	2017-08-01	2017-08-01	1
99439	b331b74b18dc79bcdf6532d51e1637c1	2018-01-08 21:28:00	2018-01-01	2018-01-01	1
99440	edb027a75a1449115f6b43211ae02a24	2018-03-08 20:57:00	2018-03-01	2018-03-01	1

99441 rows × 5 columns

```
# count of the customers with respect to first month and how long cust is in business

cohort_data = data.groupby(by=["cohort_month", "cohort_index"])["customer_id"].nunique().reset_index()

cohort_data.head()
```

	cohort_month	cohort_index	customer_id
0	2016-09-01	1	4
1	2016-10-01	1	324
2	2016-12-01	1	1
3	2017-01-01	1	800
4	2017-02-01	1	1780

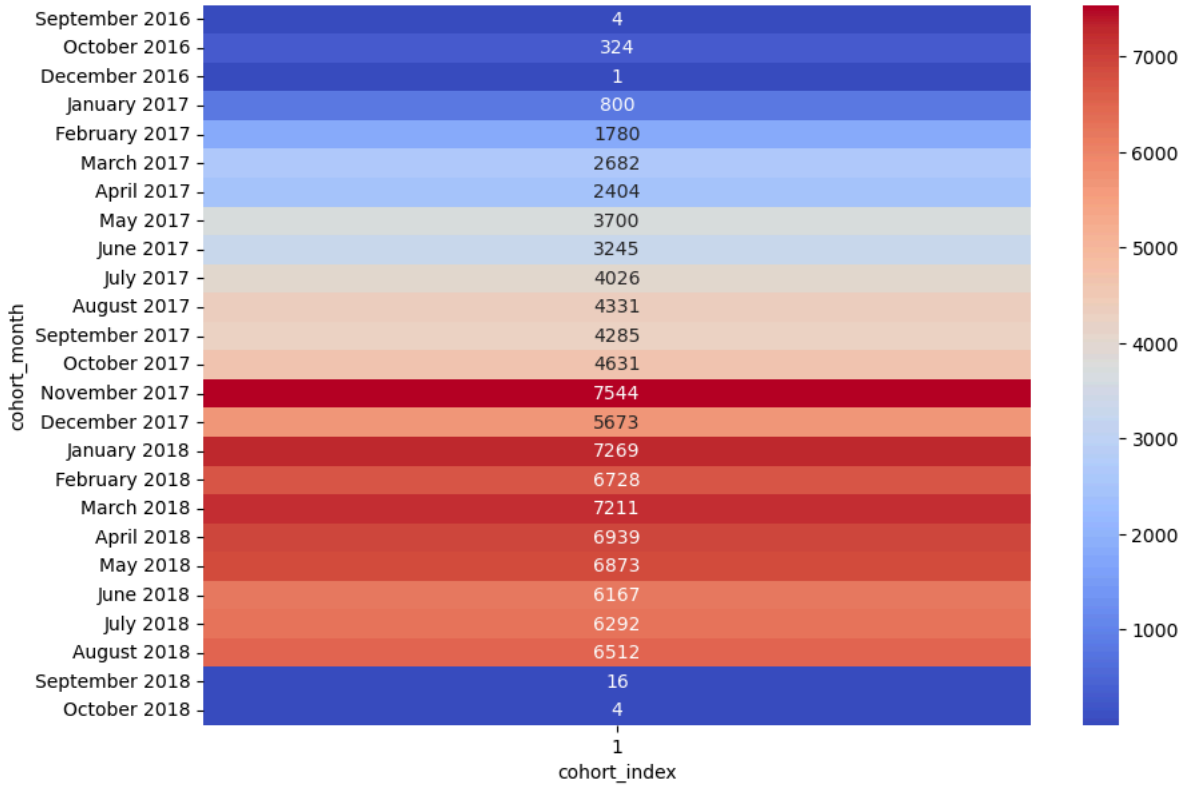
```
# crosstab

cohort_table = pd.pivot(data=cohort_data, index="cohort_month", columns="cohort_index", values="customer_id")

cohort_table.head()
```

cohort_index	1
cohort_month	
2016-09-01	4
2016-10-01	324
2016-12-01	1
2017-01-01	800
2017-02-01	1780

```
# change index name

cohort_table.index = cohort_table.index.strftime("%B %Y")
```

```
# heatmaps

plt.figure(figsize=(10,7))
sns.heatmap(data=cohort_table, annot=True, fmt="0", cmap="coolwarm")
```

```
<AxesSubplot:xlabel='cohort_index', ylabel='cohort_month'>
```



FINDINGS - from the above cohort analysis i say that all the customers retained in business for 1 month only, some customers have single items in their order or multiple items in order but on the same date time and they didn't make any other purchase or transaction on another date.

d. How the revenues from existing/new customers on month on month basis

```
order_item_tb = pd.merge(left=order_items, right=orders, how="inner", on="order_id")

order_item_tb.head()
```

	order_id	order_item_id	product_id	seller_id	shipping_li
0	00010242fe8c5a6d1ba2dd792cb16214	1	4244733e06e7ecb4970a6e2683c13e61	48436dade18ac8b2bce089ec2a041202	2017-09-19
1	00018f77f2f0320c557190d7a144bdd3	1	e5f2d52b802189ee658865ca93d83a8f	dd7ddc04e1b6c2c614352b383efe2d36	2017-05-03
2	000229ec398224ef6ca0657da4fc703e	1	c777355d18b72b67abbeef9df44fd0fd	5b51032eddd242adc84c38acab88f23d	2018-01-18
3	00024acbcdf0a6daa1e931b038114c75	1	7634da152a4610f1595efa32f14722fc	9d7a1d34a5052409006425275ba1c2b4	2018-08-15
4	00042b26cf59d7ce69dfabb4e55b4fd9	1	ac6c3623068f30de03045865e4e10089	df560393f3a51e74553ab94004ba5c87	2017-02-13

```
order_item_tb["revenue"] = order_item_tb.price + order_item_tb.freight_value
```

```
order_item_tb
```

	order_id	order_item_id	product_id	seller_id	shippi
0	00010242fe8c5a6d1ba2dd792cb16214	1	4244733e06e7ecb4970a6e2683c13e61	48436dade18ac8b2bce089ec2a041202	2017
1	00018f77f2f0320c557190d7a144bdd3	1	e5f2d52b802189ee658865ca93d83a8f	dd7ddc04e1b6c2c614352b383efe2d36	2017
2	000229ec398224ef6ca0657da4fc703e	1	c777355d18b72b67abbeef9df44fd0fd	5b51032eddd242adc84c38acab88f23d	2018
3	00024acbcdf0a6daa1e931b038114c75	1	7634da152a4610f1595efa32f14722fc	9d7a1d34a5052409006425275ba1c2b4	2018
4	00042b26cf59d7ce69dfabb4e55b4fd9	1	ac6c3623068f30de03045865e4e10089	df560393f3a51e74553ab94004ba5c87	2017
...	...	...	...	...	
112645	fffc94f6ce00a00581880bf54a75a037	1	4aa6014eceb682077f9dc4bffebc05b0	b8bc237ba3788b23da09c0f1f3a3288c	2018
112646	fffcd46ef2263f404302a634eb57f7eb	1	32e07fd915822b0765e448c4dd74c828	f3c38ab652836d21de61fb8314b69182	2018
112647	fffce4705a9662cd70adb13d4a31832d	1	72a30483855e2eafc67aee5dc2560482	c3cfdc648177fdbbbb35635a37472c53	2017
112648	fffe18544ffabc95dfada21779c9644f	1	9c422a519119dcad7575db5af1ba540e	2b3e4a2a3ea8e01938cabda2a3e5cc79	2017
112649	fffe41c64501cc87c801fd61db3f6244	1	350688d9dc1e75ff97be326363655e01	f7ccf836d21b2fb1de37564105216cc1	2018

112650 rows × 15 columns

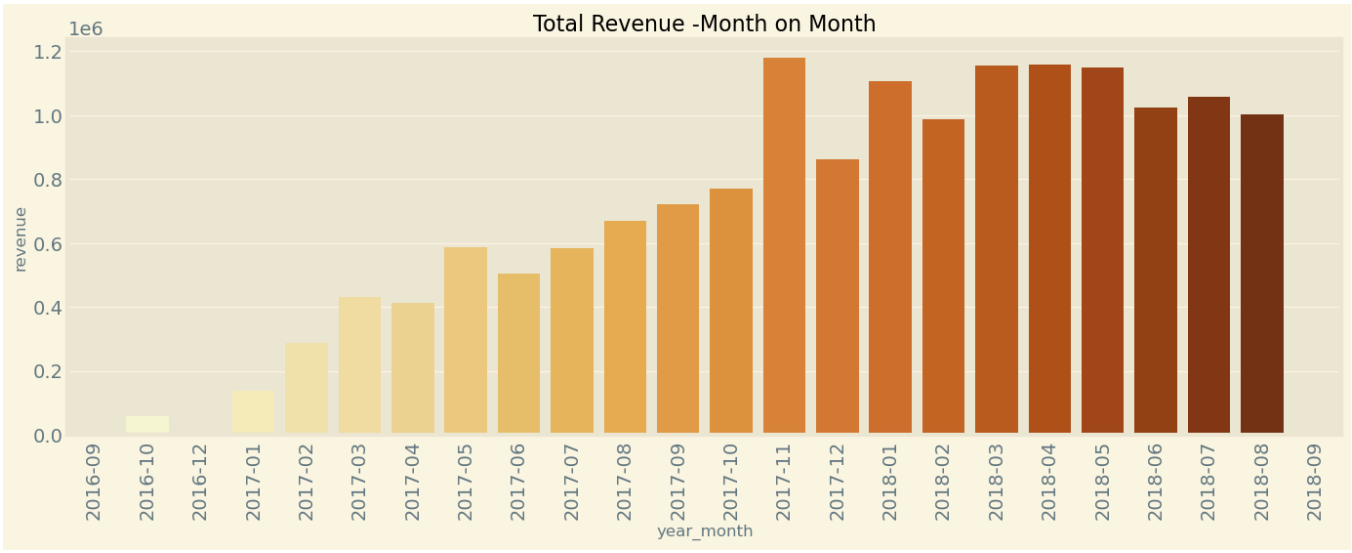
```
order_item_tb["year_month"] = order_item_tb.order_purchase_timestamp.apply(lambda x: dt.datetime.strftime(x, "%Y-%m"))
```

```
Revenue_MoM =order_item_tb.groupby(by="year_month").agg({"revenue":"sum"})

Revenue_MoM
```

year_month	revenue
2016-09	354.75
2016-10	56808.84
2016-12	19.62
2017-01	137188.49
2017-02	286280.62
2017-03	432048.59
2017-04	412422.24
2017-05	586190.95
2017-06	502963.04
2017-07	584971.62
2017-08	668204.60
2017-09	720398.91
2017-10	769312.37
2017-11	1179143.77
2017-12	863547.23
2018-01	1107301.89
2018-02	986908.96
2018-03	1155126.82
2018-04	1159698.04
2018-05	1149781.82
2018-06	1022677.11
2018-07	1058728.03
2018-08	1003308.47
2018-09	166.46

```
plt.figure(figsize=(15,5))
plt.ylabel("Revenue")
plt.title("Total Revenue -Month on Month")

barchart = sns.barplot(x=Revenue_MoM.index, y=Revenue_MoM.revenue, palette="YlOrBr")
```

```
barchart.set_xticklabels(labels=Revenue_MoM.index, rotation=90, horizontalalignment='center')

plt.show()
```



e. Understand the trends/seasonality of sales, quantity by category, location, month, week, day, time, channel, payment method etc…

```
sales_data = pd.merge(left=order_items, right=product, how="inner", on="product_id")
sales_data["months"] = sales_data.shipping_limit_date.apply(lambda x: dt.datetime.strftime(x, "%m") )
sales_data["weekdays"] = sales_data.shipping_limit_date.apply(lambda x: dt.datetime.strftime(x, "%A") )
sales_data["time_in_day"] = sales_data.shipping_limit_date.apply(lambda x: dt.datetime.strftime(x, "%H"))
```

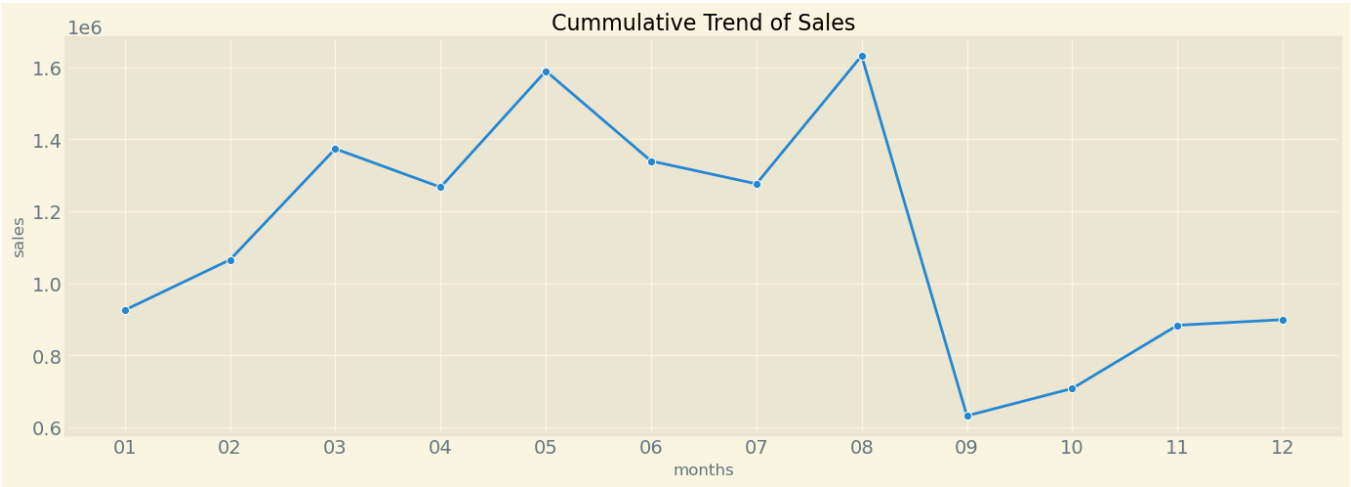
```
sales_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 112650 entries, 0 to 112649
Data columns (total 18 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  datetime64[ns]
 5   price                       112650 non-null  float64
 6   freight_value               112650 non-null  float64
 7   product_category_name       111023 non-null  object
 8   product_name_lenght         111047 non-null  float64
 9   product_description_lenght  111047 non-null  float64
 10  product_photos_qty          111047 non-null  float64
 11  product_weight_g            112632 non-null  float64
 12  product_length_cm           112632 non-null  float64
 13  product_height_cm           112632 non-null  float64
 14  product_width_cm            112632 non-null  float64
 15  months                      112650 non-null  object
 16  weekdays                    112650 non-null  object
 17  time_in_day                 112650 non-null  object
dtypes: datetime64[ns](1), float64(9), int64(1), object(7)
memory usage: 16.3+ MB
```

```
monthly_qty_sales = sales_data.groupby(by="months").agg({"product_id": "count",
                                    "price":"sum"}).reset_index().rename(columns={"product_id": "quantity",
                                                                                  "price":"sales"})
monthly_qty_sales
```

	months	quantity	sales
0	01	8173	925403.92
1	02	9243	1065356.84
2	03	11510	1373770.44
3	04	10003	1267078.00
4	05	12915	1590014.01
5	06	10698	1339737.89
6	07	10788	1276414.18
7	08	13857	1632241.04
8	09	4827	631743.09
9	10	5554	707600.78
10	11	7355	883351.63
11	12	7727	898931.88

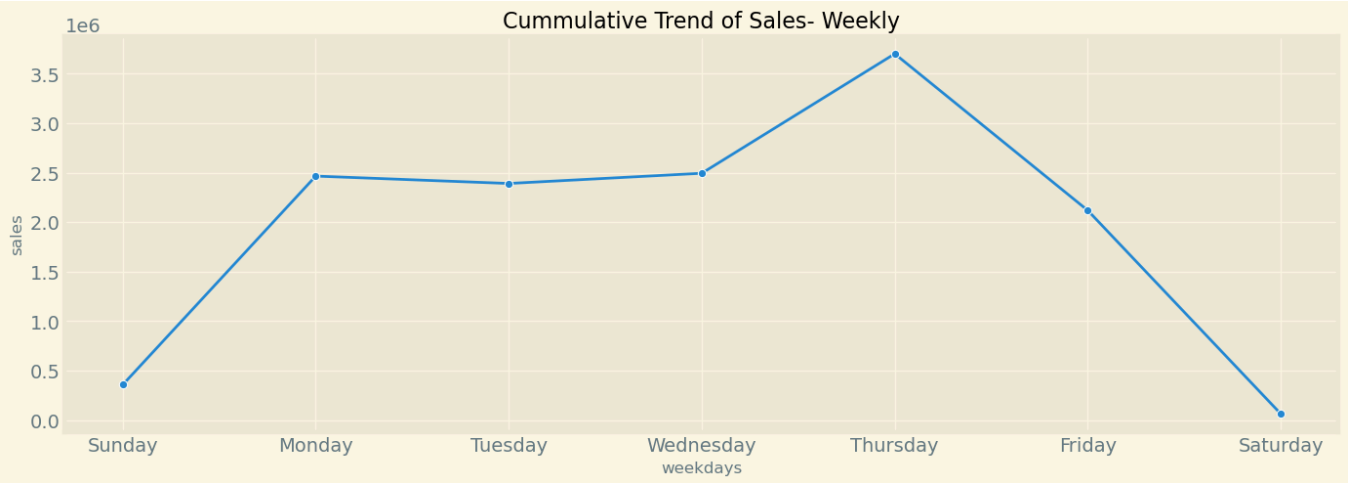
```
plt.figure(figsize=(15,5))
plt.title("Cummulative Trend of Sales")
sns.lineplot(x=monthly_qty_sales.months, y=monthly_qty_sales.sales, marker="o", legend=True)
plt.show()
```




```
weekly_qty_sales = sales_data.groupby(by="weekdays").agg({"product_id": "count",
                                    "price":"sum"}).reset_index().rename(columns={"product_id": "quantity",
                                                                                  "price":"sales"})

weekly_qty_sales["dayofweek_no"] = np.where(weekly_qty_sales.weekdays=="Monday", 1,

                                        np.where(weekly_qty_sales.weekdays=="Tuesday",2,

                                                 np.where(weekly_qty_sales.weekdays=="Wednesday",3,

                                                          np.where(weekly_qty_sales.weekdays=="Thursday",4,

                                                                   np.where(weekly_qty_sales.weekdays=="Friday",5,

                                                                            np.where(weekly_qty_sales.weekdays=="Saturday",6,0)))))

weekly_qty_sales.sort_values(by="dayofweek_no", ascending=True, inplace=True)

weekly_qty_sales
```

	weekdays	quantity	sales	dayofweek_no
3	Sunday	2759	356031.13	0
1	Monday	20750	2465891.41	1
5	Tuesday	20079	2391326.09	2
6	Wednesday	20638	2494393.38	3
4	Thursday	30476	3702324.97	4
0	Friday	17487	2119002.23	5
2	Saturday	461	62674.49	6

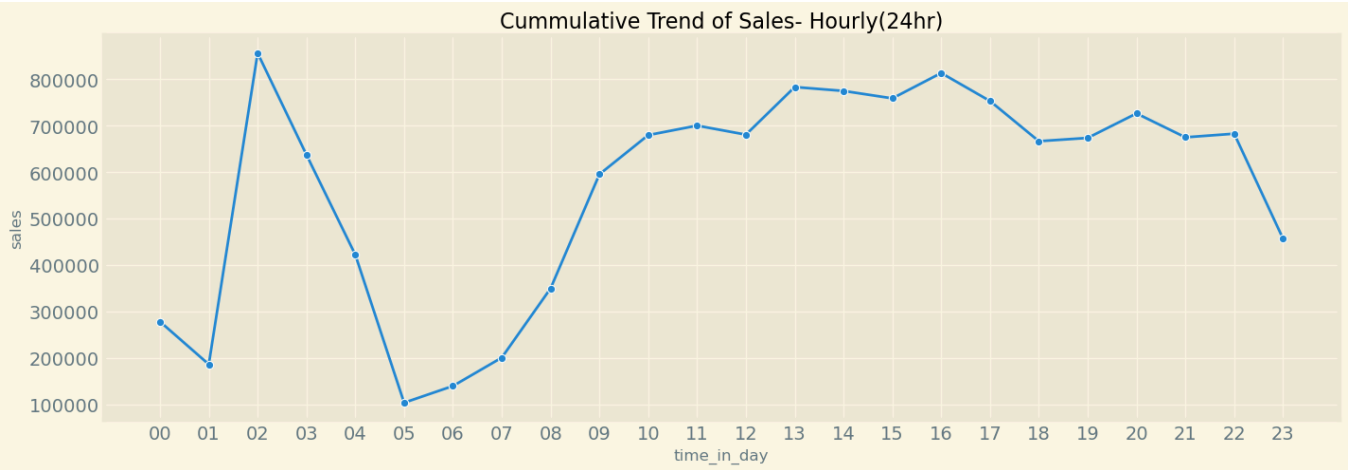
```
plt.figure(figsize=(15,5))
plt.title("Cummulative Trend of Sales- Weekly")
sns.lineplot(x=weekly_qty_sales.weekdays, y=weekly_qty_sales.sales, marker="o",legend=True)
plt.show()
```




```
hourly_qty_sales = sales_data.groupby(by="time_in_day").agg({"product_id": "count",
                                    "price":"sum"}).reset_index().rename(columns={"product_id": "quantity",
                                                                                  "price":"sales"})

hourly_qty_sales
```

	time_in_day	quantity	sales
0	00	2403	277950.78
1	01	1649	185955.99
2	02	8407	856307.77
3	03	6377	637271.49
4	04	4082	422136.33
5	05	1061	103318.70
6	06	1073	139281.80
7	07	1600	200374.96
8	08	2637	349521.84
9	09	4605	595254.96
10	10	5314	679689.28
11	11	5927	700078.45
12	12	5393	680655.12
13	13	6225	783188.84
14	14	6117	774785.31
15	15	6135	758800.22
16	16	6366	813419.63
17	17	6167	753021.84
18	18	5282	666518.84
19	19	5432	673586.80
20	20	5599	726322.66
21	21	5394	674906.45
22	22	5442	682631.48
23	23	3963	456664.16

```
plt.figure(figsize=(15,5))
plt.title("Cummulative Trend of Sales- Hourly(24hr)")
sns.lineplot(x=hourly_qty_sales.time_in_day, y=hourly_qty_sales.sales,marker="o", legend=True)
plt.show()
```




```
category = pd.merge(left=order_items, right=product, how="left", on="product_id")

category.head()
```

	order_id	order_item_id	product_id	seller_id	shipping_li
0	00010242fe8c5a6d1ba2dd792cb16214	1	4244733e06e7ecb4970a6e2683c13e61	48436dade18ac8b2bce089ec2a041202	2017-09-19
1	00018f77f2f0320c557190d7a144bdd3	1	e5f2d52b802189ee658865ca93d83a8f	dd7ddc04e1b6c2c614352b383efe2d36	2017-05-03
2	000229ec398224ef6ca0657da4fc703e	1	c777355d18b72b67abbeef9df44fd0fd	5b51032eddd242adc84c38acab88f23d	2018-01-18
3	00024acbcdf0a6daa1e931b038114c75	1	7634da152a4610f1595efa32f14722fc	9d7a1d34a5052409006425275ba1c2b4	2018-08-15
4	00042b26cf59d7ce69dfabb4e55b4fd9	1	ac6c3623068f30de03045865e4e10089	df560393f3a51e74553ab94004ba5c87	2017-02-13

```
sales_by_category = category.groupby(by="product_category_name").agg({"product_id":"count",
                                                                       "price":"sum"}).reset_index().rename(columns={"product_id":"quantity",
                                                                                                                     "price":"sales"})

sales_by_category
```

	product_category_name	quantity	sales
0	Agro_Industry_And_Commerce	212	72530.47
1	Air_Conditioning	297	55024.96
2	Art	209	24202.64
3	Arts_And_Craftmanship	24	1814.01
4	Audio	364	50688.50
...	...	...	...
66	Stationery	2517	230943.23
67	Tablets_Printing_Image	83	7528.41
68	Telephony	4545	323667.53
69	Toys	4117	483946.60
70	Watches_Gifts	5991	1205005.68

71 rows × 3 columns

```
Start coding or generate with AI.
```

```
loc_by_sale = final_order[["order_id", "customer_id", "customer_unique_id",
                           "customer_city", "customer_state",
                           "product_id","product_category_name",
                           "price", "freight_value","seller_id","seller_city","seller_state"]]

loc_by_sale
```

	order_id	customer_id	customer_unique_id	customer_city	custome
0	2e7a8482f6fb09756ca50c10d7bfc047	08c5351a6aca1c1589a38f244edeee9d	b7d76e111c89f7ebf14761390f0f7d17	Dholka	
1	2e7a8482f6fb09756ca50c10d7bfc047	08c5351a6aca1c1589a38f244edeee9d	b7d76e111c89f7ebf14761390f0f7d17	Dholka	
2	e5fa5a7210941f7d56d0208e4e071d35	683c54fc24d40ee9f8a6fc179fd9856c	4854e9b3feff728c13ee5fc7d1547e92	Tuni	Andhra
3	71303d7e93b399f5bcd537d124c0bcfa	b106b360fe2ef8849fbbd056f777b4d5	0eb1ee9dba87f5b36b4613a65074337c	Akkarampalle	Andhra
4	3b697a20d9e427646d92567910af6d57	355077684019f7f60a031656bd7262b8	32ea3bdedab835c3aa6cb68ce66565ef	Akkarampalle	Andhra
...	...	...	...	...	
118310	0b223d92c27432930dfe407c6aea3041	e60df9449653a95af4549bbfcb18a6eb	5c58de6fb80e93396e2f35642666b693	Anakapalle	Andhra
118311	0b223d92c27432930dfe407c6aea3041	e60df9449653a95af4549bbfcb18a6eb	5c58de6fb80e93396e2f35642666b693	Anakapalle	Andhra
118312	03ef5dedbe7492bdae72eec50764c43f	496630b6740bcca28fce9ba50d8a26ef	b701bebbdf478f5500348f03aff62121	Vinukonda	Andhra
118313	35a972d7f8436f405b56e36add1a7140	898b7fee99c4e42170ab69ba59be0a8b	24ac2b4327e25baf39f2119e4228976a	Kharsia	Chh
118314	54282e97f61c23b78330c15b154c867d	4b7decb9b58e2569548b8b4c8e20e8d7	ff22e30958c13ffe219db7d711e8f564	Akkarampalle	Andhra

118315 rows × 12 columns

```
# seller

sales_of_seller = loc_by_sale.groupby(by=["seller_state","seller_city"]).agg({"product_id": "count",
                                            "price":"sum"}).reset_index().rename(columns={"product_id":"quantity",
                                                                                          "price":"sales"})
sales_of_seller
```

	seller_state	seller_city	quantity	sales
0	Andhra Pradesh	Adilabad	685	91205.83
1	Andhra Pradesh	Adoni	1320	206713.85
2	Andhra Pradesh	Akkarampalle	29367	2829456.13
3	Andhra Pradesh	Akkayapalle	426	50240.19
4	Andhra Pradesh	Alwal	1504	195913.45
...	...	...	...	...
529	West Bengal	Jaygaon	4	759.00
530	West Bengal	Khandra	12	496.00
531	West Bengal	Maheshtala	45	2361.06
532	West Bengal	Mal	13	2553.70
533	West Bengal	Siduli	1	98.00

534 rows × 4 columns

```
# customer
revenue_from_cust = loc_by_sale.groupby(by=["customer_state", "customer_city"]).agg({"product_id": "count",
                                            "price":"sum"}).reset_index().rename(columns={"product_id":"quantity",
                                                                                          "price":"sales"})
revenue_from_cust
```

	customer_state	customer_city	quantity	sales
0	Andhra Pradesh	Adilabad	229	25736.07
1	Andhra Pradesh	Adoni	1121	109051.09
2	Andhra Pradesh	Akkarampalle	18728	2021302.58
3	Andhra Pradesh	Akkayapalle	480	55882.43
4	Andhra Pradesh	Alwal	1743	197385.14
...	...	...	...	...
4105	West Bengal	Uttar Latabari	1	385.60
4106	West Bengal	Uttar Mahammadpur	1	95.90
4107	West Bengal	Uttar Pirpur	1	540.00
4108	West Bengal	Uttar Raypur	1	50.90
4109	West Bengal	Uttarpara Kotrung	1	99.90

4110 rows × 4 columns

```
Start coding or generate with AI.
```

```
#sales
payment_method_tb = pd.merge(left=order_items, right=order_payment, how="inner", on="order_id")

payment_method = payment_method_tb.groupby(by="payment_type").agg({"price":"sum",
                                            "product_id":"count"}).reset_index().rename(columns={"product_id":"quantity",
                                                                                          "price":"sales"})
```

```
payment_method
```

	payment_type	sales	quantity
0	UPI	2391525.66	22867
1	credit_card	10974357.30	86769
2	debit_card	183758.74	1691
3	voucher	659473.64	6274

f. Popular Products by month, seller, state, category.

```
# sellers
```

```
PopProd_seller = order_items.groupby(by=["seller_id", "product_id"]).agg({"order_id":"count"}).reset_index().rename(columns={"order_id":"

PopProd_seller.sort_values(by=["seller_id", "count"], ascending=[True, False], inplace=True, ignore_index=True)

PopProd_seller.drop_duplicates(subset="seller_id", keep="first", inplace=True, ignore_index=True)
```

```
#popular products by sellers
PopProd_seller
```

	seller_id	product_id	count
0	0015a82c2db000af6aaaf3ae2ecb0532	a2ff5a97bf95719e38ea2e3b4105bce8	3
1	001cca7ae9ae17fb1caed9dfb1094831	08574b074924071f4e201e151b152b4e	113
2	001e6ad469a905060d959994f1b41e4f	093cd981b714bcdff182b427d87fc8fc	1
3	002100f778ceb8431b7a1020ff7ab48f	158102fe543dbaeb84d87811bfe06d0d	17
4	003554e2dce176b5555353e4f3555ac8	67f36b3689147d882d2b298fd0715d80	1
...	...	...	...
3090	ffcfefa19b08742c5d315f2791395ee5	cea898bfbca0b5b0e7b36cecd350709e	1
3091	ffdd9f82b9a447f6f8d4b91554cc7dd3	ada800a927673ac73cdfbbd2c832331b	5
3092	ffeee66ac5d5a62fe688b9d26f83f534	1347d4320dcd0acd750e37bb3d94a918	10
3093	fffd5413c0700ac820c7069d66d98c89	ebb12274522d82caa7dca657873b2ad7	6
3094	ffff564a4f9085cd26170f4732393726	8f7a3322e1abfed89ac080b0f7364779	2

3095 rows × 3 columns

```
# months
```

```
PopProd_tb = pd.merge(left=orders, right=order_items, how="inner", on="order_id")

PopProd_tb["year_month"] = PopProd_tb.order_purchase_timestamp.apply(lambda x: dt.datetime.strftime(x,"%Y-%m"))

PopProd_month = PopProd_tb.groupby(by=["year_month",
                                       "product_id"]).agg({"order_id":"count"}).reset_index().rename(columns={"order_id":"qty"})
```

```
PopProd_month.sort_values(by=["year_month","qty"], ascending=[True,False], inplace=True, ignore_index=True)

PopProd_month.drop_duplicates(subset="year_month", keep="first", inplace=True, ignore_index=True)
```

```
#popular products by months
PopProd_month
```

	year_month	product_id	qty
0	2016-09	5a6b04657a4c5ee34285d1e4619a96b4	3
1	2016-10	eba7488e1c67729f045ab43fac426f2e	11
2	2016-12	f5d8f4fbc70ca2a0038b9a0010ed5cb0	1
3	2017-01	37eb69aca8718e843d897aa7b82f462d	15
4	2017-02	a703f5ade6e4fae527357132230ea778	13
5	2017-03	7e0dc102074f8285580c9777f79c90cf	45
6	2017-04	99a4788cb24856965c36a24e339b6058	34
7	2017-05	99a4788cb24856965c36a24e339b6058	58
8	2017-06	42a2c92a0979a949ca4ea89ec5c7b934	73
9	2017-07	99a4788cb24856965c36a24e339b6058	60
10	2017-08	f1c7f353075ce59d8a6f3cf58f419c9c	52
11	2017-09	422879e10f46682990de24d770e7f83d	37
12	2017-10	422879e10f46682990de24d770e7f83d	39
13	2017-11	422879e10f46682990de24d770e7f83d	91
14	2017-12	422879e10f46682990de24d770e7f83d	51
15	2018-01	aca2eb7d00ea1a7b8ebd4e68314663af	122
16	2018-02	e53e557d5a159f5aa2c5e995dfdf244b	71
17	2018-03	aca2eb7d00ea1a7b8ebd4e68314663af	65
18	2018-04	53b36df67ebb7c41585e8d54d6772e08	85
19	2018-05	53b36df67ebb7c41585e8d54d6772e08	114
20	2018-06	19c91ef95d509ea33eda93495c4d3481	43
21	2018-07	d285360f29ac7fd97640bf0baef03de0	45
22	2018-08	e7cc48a9daff5436f63d3aad9426f28b	68
23	2018-09	b98992ea80b467987a7fbb88e7f2076a	1

```
#state
```

```
ord_cust = pd.merge(left=orders, right=customer, how="inner", on="customer_id")

ord_items_cust = pd.merge(left=ord_cust, right=order_items, how="inner", on="order_id")

Prod_state_tb = ord_items_cust[["order_id","product_id", "customer_state"]]

PopProd_state = Prod_state_tb.groupby(by=["customer_state",
                          "product_id"]).agg({"order_id":"count"}).reset_index().rename(columns={"order_id":"qty"})

PopProd_state.sort_values(by=["customer_state", "qty"],ascending=[True, False], inplace=True, ignore_index=True)

PopProd_state.drop_duplicates(subset="customer_state", keep="first", inplace=True, ignore_index=True)
```

```
#popular products by states
PopProd_state
```

	customer_state	product_id	qty
0	Andhra Pradesh	aca2eb7d00ea1a7b8ebd4e68314663af	345
1	Arunachal Pradesh	03e1c946c0ddfc58724ff262aef08dff	12
2	Chhattisgarh	368c6c730842d78016ad823897a372db	40
3	Delhi	53759a2ecddad2bb87a079a1f1519f73	22
4	Goa	2a5806f10d0f00e5ad032dd2e3c8806e	1
5	Gujarat	53759a2ecddad2bb87a079a1f1519f73	45
6	Haryana	2b4609f8948be18874494203496bc318	13
7	Himachal Pradesh	422879e10f46682990de24d770e7f83d	6
8	Jammu & Kashmir	89b190a046022486c635022524a974a8	15
9	Karnataka	422879e10f46682990de24d770e7f83d	24
10	Kerala	aca2eb7d00ea1a7b8ebd4e68314663af	13
11	Madhya Pradesh	99a4788cb24856965c36a24e339b6058	17
12	Maharashtra	99a4788cb24856965c36a24e339b6058	13
13	Orissa	aca2eb7d00ea1a7b8ebd4e68314663af	6
14	Punjab	d1c427060a0f73f6b889a5c7c61f2ac4	6
15	Rajasthan	368c6c730842d78016ad823897a372db	7
16	Tamil Nadu	99a4788cb24856965c36a24e339b6058	13
17	Uttar Pradesh	3dd2a17168ec895c781a9191c1e95ad7	8
18	Uttaranchal	53759a2ecddad2bb87a079a1f1519f73	3
19	West Bengal	99a4788cb24856965c36a24e339b6058	6

```
#category
```

```
prod_cat_tb = pd.merge(left=PopProd_tb, right=product, how="inner", on="product_id")
prod_cat_tb = prod_cat_tb[["order_id", "product_id", "product_category_name"]]
```

```
PopProd_category = prod_cat_tb.groupby(by=["product_category_name",
                          "product_id"]).agg({"order_id":"count"}).reset_index().rename(columns={"order_id":"qty"})

PopProd_category.sort_values(by=["product_category_name", "qty"], ascending=[True, False], inplace=True, ignore_index=True)

PopProd_category.drop_duplicates(subset="product_category_name", keep="first", inplace=True, ignore_index=True)
```

```
#popular products by category
PopProd_category
```

	product_category_name	product_id	qty
0	Agro_Industry_And_Commerce	11250b0d4b709fee92441c5f34122aed	22
1	Air_Conditioning	98e91d0f32954dcd8505875bb2b42cdb	17
2	Art	4fe644d766c7566dbc46fb851363cb3b	107
3	Arts_And_Craftmanship	b9976e9c22fb1540bd71d1bcd2989475	5
4	Audio	db5efde3ad0cc579b130d71c4b2db522	48
...	...	...	...
66	Stationery	fb55982be901439613a95940feefd9ee	84
67	Tablets_Printing_Image	6bbe55cf8f85c87b6eebb775a53402f4	33
68	Telephony	e7cc48a9daff5436f63d3aad9426f28b	93
69	Toys	880be32f4db1d9f6e2bec38fb6ac23ab	99
70	Watches_Gifts	53b36df67ebb7c41585e8d54d6772e08	323

71 rows × 3 columns

g. Popular categories by state, month

```
OrdCust = pd.merge(left=orders, right=customer, how="inner", on="customer_id")
```

```
OrdCustItems = pd.merge(left=OrdCust, right=order_items, how="inner", on="order_id")

df = pd.merge(left=OrdCustItems, right=product, how="inner", on="product_id")

final_data = df[["order_id","order_purchase_timestamp", "product_id","product_category_name", "customer_state"]]
```

```
final_data["year_month"] = final_data.order_purchase_timestamp.apply(lambda x: dt.datetime.strftime(x, "%Y-%m"))
```

```
C:\Users\SANAM KANDAR\AppData\Local\Temp\ipykernel_8508\2173596272.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  final_data["year_month"] = final_data.order_purchase_timestamp.apply(lambda x: dt.datetime.strftime(x, "%Y-%m"))
```

```
final_data
```

	order_id	order_purchase_timestamp	product_id	product_category_name	custom
0	e481f51cbdc54678b7cc49136f2d6af7	2017-10-02 10:56:00	87285b34884572647811a353c7ac498a	Housewares	Andhra
1	128e10d95713541c87cd1a2e48201934	2017-08-15 18:29:00	87285b34884572647811a353c7ac498a	Housewares	Andhra
2	0e7e841ddf8f8f2de2bad69267ecfbcf	2017-08-02 18:24:00	87285b34884572647811a353c7ac498a	Housewares	Andhra
3	bfc39df4f36c3693ff3b63fcbea9e90a	2017-10-23 23:26:00	87285b34884572647811a353c7ac498a	Housewares	Andhra
4	53cdb2fc8bc7dce0b6741e2150273451	2018-07-24 20:41:00	595fac2a385ac33a80bd5114aec74eb8	Perfumery	Ch
...	...	...	...	...	
112645	e8fd20068b9f7e6ec07068bb7537f781	2017-08-10 21:21:00	0df37da38a30a713453b03053d60d3f7	Sports_Leisure	Andhra
112646	e8fd20068b9f7e6ec07068bb7537f781	2017-08-10 21:21:00	0df37da38a30a713453b03053d60d3f7	Sports_Leisure	Andhra
112647	cfa78b997e329a5295b4ee6972c02979	2017-12-20 09:52:00	3d2c44374ee42b3003a470f3e937a2ea	Musical_Instruments	Utta
112648	9c5dedf39a927c1b2549525ed64a053c	2017-03-09 09:54:00	ac35486adb7b02598c182c2ff2e05254	Health_Beauty	Andhra
112649	66dea50a8b16d9b4dee7af250b4be1a5	2018-03-08 20:57:00	006619bbed68b000c8ba3f8725d5409e	Health_Beauty	

112650 rows × 6 columns

```
PopProd_SM = final_data.groupby(by=["year_month",
                     "customer_state",
                     "product_category_name"]).agg({"order_id":"count"}).reset_index().rename(columns={"order_id":"qty"})
PopProd_SM.head()
```

	year_month	customer_state	product_category_name	qty
0	2016-09	Andhra Pradesh	Health_Beauty	3
1	2016-09	Andhra Pradesh	Telephony	1
2	2016-09	Gujarat	Furniture_Decor	2
3	2016-10	Andhra Pradesh	Air_Conditioning	3
4	2016-10	Andhra Pradesh	Audio	2

```
PopProd_SM.sort_values(by=["year_month","customer_state","qty"], ascending=[True,True,False], inplace=True, ignore_index=True)

PopProd_SM.drop_duplicates(subset=["year_month","customer_state"], keep="first", inplace=True, ignore_index=True)
```

```
# popular categories table in states with months
PopProd_SM
```

	year_month	customer_state	product_category_name	qty
0	2016-09	Andhra Pradesh	Health_Beauty	3
1	2016-09	Gujarat	Furniture_Decor	2
2	2016-10	Andhra Pradesh	Furniture_Decor	44
3	2016-10	Arunachal Pradesh	Fashion_Male_Clothing	1
4	2016-10	Chhattisgarh	Air_Conditioning	2
...	...	...	...	...
404	2018-08	Tamil Nadu	Health_Beauty	19
405	2018-08	Uttar Pradesh	Health_Beauty	11
406	2018-08	Uttaranchal	Baby	1
407	2018-08	West Bengal	Bed_Bath_Table	2
408	2018-09	Andhra Pradesh	Kitchen_Dining_Laundry_Garden_Furniture	1

409 rows × 4 columns

```
# Popular categories by state, month
PopularCat_Summary =  pd.pivot(data=PopProd_SM, index="year_month",
                               columns="customer_state", values="product_category_name").fillna("----")

PopularCat_Summary
```

customer_state	Andhra Pradesh	Arunachal Pradesh	Chhattisgarh	Delhi	
year_month					
2016-09	Health_Beauty	----	----	----	
2016-10	Furniture_Decor	Fashion_Male_Clothing	Air_Conditioning	Furniture_Decor	
2016-12	Fashion_Bags_Accessories	----	----	----	
2017-01	Furniture_Decor	Furniture_Decor	Health_Beauty	Furniture_Decor	
2017-02	Furniture_Decor	Bed_Bath_Table	Furniture_Decor	Furniture_Decor	
2017-03	Furniture_Decor	Furniture_Decor	Furniture_Decor	Furniture_Decor	
2017-04	Bed_Bath_Table	Bed_Bath_Table	Sports_Leisure	Housewares	
2017-05	Bed_Bath_Table	Bed_Bath_Table	Computers_Accessories	Telephony	
2017-06	Bed_Bath_Table	Cool_Stuff	Housewares	Bed_Bath_Table	
2017-07	Bed_Bath_Table	Toys	Bed_Bath_Table	Bed_Bath_Table	
2017-08	Bed_Bath_Table	Health_Beauty	Sports_Leisure	Furniture_Decor	
2017-09	Bed_Bath_Table	Bed_Bath_Table	Bed_Bath_Table	Furniture_Decor	
2017-10	Bed_Bath_Table	Housewares	Health_Beauty	Computers_Accessories	
2017-11	Bed_Bath_Table	Garden_Tools	Bed_Bath_Table	Furniture_Decor	
2017-12	Bed_Bath_Table	Bed_Bath_Table	Health_Beauty	Furniture_Decor	
2018-01	Bed_Bath_Table	Furniture_Decor	Bed_Bath_Table	Bed_Bath_Table	
2018-02	Computers_Accessories	Sports_Leisure	Computers_Accessories	Computers_Accessories	
2018-03	Bed_Bath_Table	Watches_Gifts	Health_Beauty	Computers_Accessories	
2018-04	Bed_Bath_Table	Furniture_Decor	Sports_Leisure	Health_Beauty	
2018-05	Health_Beauty	Furniture_Decor	Furniture_Decor	Watches_Gifts	
2018-06	Health_Beauty	Bed_Bath_Table	Bed_Bath_Table	Health_Beauty	Construction_Too
2018-07	Health_Beauty	Watches_Gifts	Health_Beauty	Health_Beauty	Fashior
2018-08	Health_Beauty	Health_Beauty	Bed_Bath_Table	Health_Beauty	
2018-09	Kitchen_Dining_Laundry_Garden_Furniture	----	----	----	

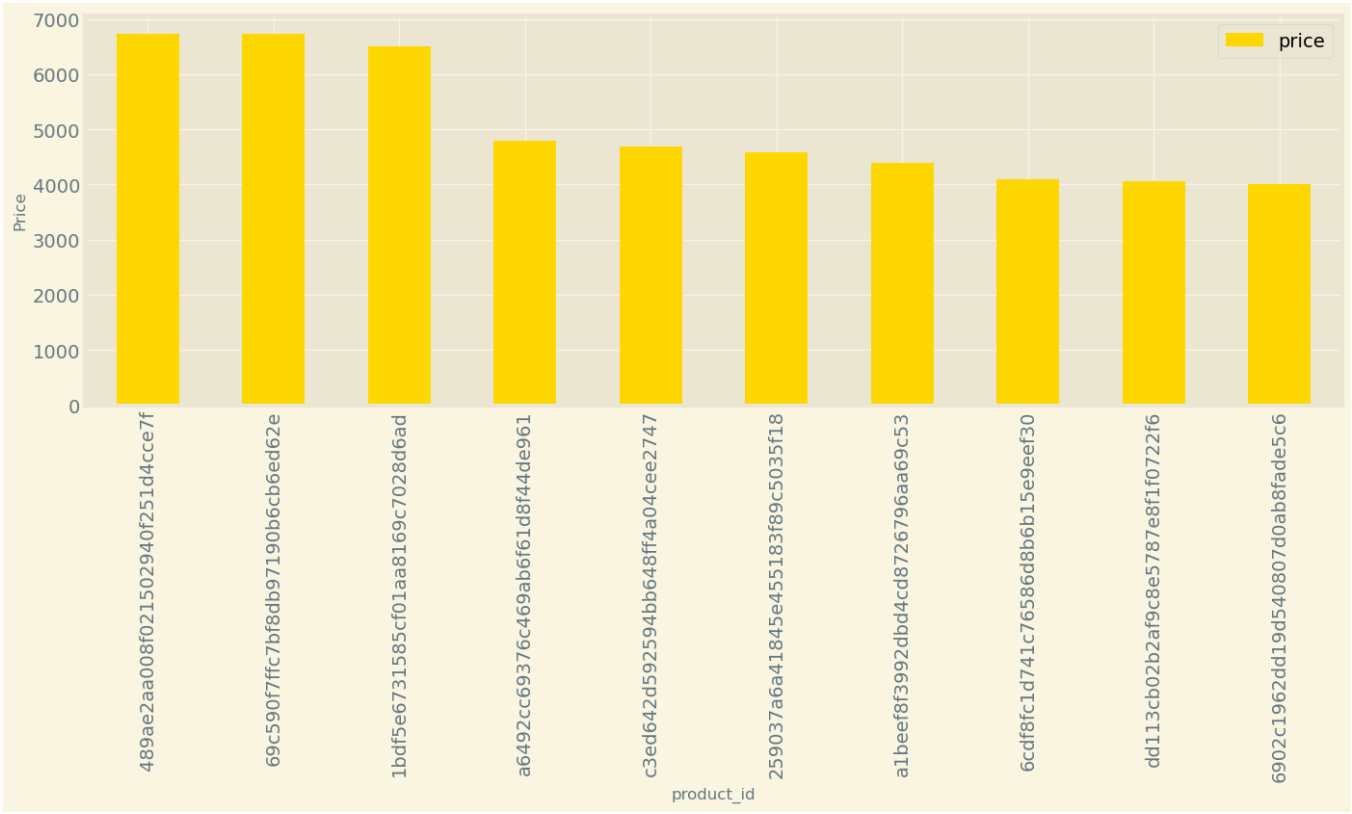
h. List top 10 most expensive products sorted by price

```
ProductPrice_tb = pd.merge(left=order_items, right=product,
                           how="inner", on="product_id")[["product_id","product_category_name", "price"]]

ProductPrice_tb.sort_values(by="price", ascending=False, inplace=True, ignore_index=True)
```

```
Top10ExpensiveProducts = ProductPrice_tb.head(10)

Top10ExpensiveProducts
```

	product_id	product_category_name	price
0	489ae2aa008f021502940f251d4cce7f	Housewares	6735.00
1	69c590f7ffc7bf8db97190b6cb6ed62e	Computers	6729.00
2	1bdf5e6731585cf01aa8169c7028d6ad	Art	6499.00
3	a6492cc69376c469ab6f61d8f44de961	Small_Appliances	4799.00
4	c3ed642d592594bb648ff4a04cee2747	Small_Appliances	4690.00
5	259037a6a41845e455183f89c5035f18	Computers	4590.00
6	a1beef8f3992dbd4cd8726796aa69c53	Musical_Instruments	4399.87
7	6cdf8fc1d741c76586d8b6b15e9eef30	Consoles_Games	4099.99
8	dd113cb02b2af9c8e5787e8f1f0722f6	Sports_Leisure	4059.00
9	6902c1962dd19d540807d0ab8fade5c6	Watches_Gifts	3999.90

```
Top10ExpensiveProducts_chart = Top10ExpensiveProducts.plot(kind="bar", x="product_id",
                                                           figsize=(15,5), color="gold")
plt.ylabel("Price")
plt.show()
```



2. Performing Customers / Sellers Segmentation

a. Divide the customers into groups based on the revenue generated

```
CustSeller_data = pd.merge(left=orders, right=order_items, how="inner", on="order_id")
CustSeller_data["revenue"] = CustSeller_data.price + CustSeller_data.freight_value

CustSeller_data.head()
```

	order_id	customer_id	order_status	order_purchase_timestamp	order_approved_at	or
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-10-02 10:56:00	2017-10-02 11:07:00	
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-07-24 20:41:00	2018-07-26 03:24:00	
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	2018-08-08 08:38:00	2018-08-08 08:55:00	
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	2017-11-18 19:28:00	2017-11-18 19:45:00	
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	2018-02-13 21:18:00	2018-02-13 22:20:00	

```
CustSegmentation = CustSeller_data[["customer_id", "revenue"]]

CustSegmentation.head()
```

	customer_id	revenue
0	9ef432eb6251297304e76186b10a928d	38.71
1	b0830fb4747a6c6d20dea0b8c802d7ef	141.46
2	41ce2a54c0b03bf3443c3d931a367089	179.12
3	f88197465ea7920adcdbec7375364d82	72.20
4	8ab97904e6daea8866dbdbc4fb7aad2c	28.62

```
# segment given to customer based on total revenue(which is spend for customer)

CustSegment_tb = CustSegmentation.groupby(by="customer_id").agg({"revenue":"sum"}).reset_index()

CustSegment_tb["segment"] = pd.cut(CustSegment_tb.revenue, bins=4,labels=["Silver","Gold", "Platinum", "Daimond"])
```

```
CustSegment_tb.sort_values(by="revenue")
```

	customer_id	revenue	segment
64612	a790343ca6f3fee08112d678b43aa7c5	9.59	Silver
9383	184e8e8e48937145eb96c721ef1f0747	10.07	Silver
54747	8e4bd65db637116b6b68109e4df21b84	10.89	Silver
1727	046f890135acc703faff4c1fc0c2d73c	11.56	Silver
81496	d2c63ad286e3ca9dd69218008d61ff81	11.62	Silver
...	...	...	...
24603	3fd6777bbce08a352fddd04e4a7cc8f6	6726.66	Gold
94398	f48d464a0baaea338cb25f816991ab1f	6922.21	Platinum
76948	c6e2731c5b391845f6800c97401a43a9	6929.31	Platinum
91284	ec5b2ba62e574342386871631fafd3fc	7274.88	Platinum
8475	1617b1357756262bfa56ab541c47bc16	13664.08	Daimond

98666 rows × 3 columns

b. Divide the sellers into groups based on the revenue generated

```
SellerSegmentation = CustSeller_data[[ "seller_id", "revenue"]].rename(columns={"revenue":"sales"})

SellerSegmentation
```

	seller_id	sales
0	3504c0cb71d7fa48d967e0e4c94d59d9	38.71
1	289cdb325fb7e7f891c38608bf9e0962	141.46
2	4869f7a5dfa277a7dca6462dcf3b52b2	179.12
3	66922902710d126a0e7d26b0e3805106	72.20
4	2c9e548be18521d1c43cde1c582c6de8	28.62
...	...	...
112645	1f9ab4708f3056ede07124aad39a2554	195.00
112646	d50d79cb34e38265a8649c383dcffd48	271.01
112647	a1043bafd471dff536d0c462352beb48	220.58
112648	a1043bafd471dff536d0c462352beb48	220.58
112649	ececbfcff9804a2d6b40f589df8eef2b	86.86

112650 rows × 2 columns

```
# segment given to seller based on total sales raised by them
segment_tb = SellerSegmentation.groupby(by="seller_id").agg({"sales":"sum"}).reset_index()

segment_tb["segment"] = pd.cut(segment_tb.sales, bins=4,labels=["Silver","Gold", "Platinum", "Daimond"])
```

```
segment_tb
```

	seller_id	sales	segment
0	0015a82c2db000af6aaaf3ae2ecb0532	2748.06	Silver
1	001cca7ae9ae17fb1caed9dfb1094831	33934.17	Silver
2	001e6ad469a905060d959994f1b41e4f	267.94	Silver
3	002100f778ceb8431b7a1020ff7ab48f	2028.16	Silver
4	003554e2dce176b5555353e4f3555ac8	139.38	Silver
...	...	...	...
3090	ffcfefa19b08742c5d315f2791395ee5	79.52	Silver
3091	ffdd9f82b9a447f6f8d4b91554cc7dd3	2828.66	Silver
3092	ffeee66ac5d5a62fe688b9d26f83f534	2259.55	Silver
3093	fffd5413c0700ac820c7069d66d98c89	11896.04	Silver
3094	ffff564a4f9085cd26170f4732393726	1770.86	Silver

3095 rows × 3 columns

3. Cross-Selling (Which products are selling together)

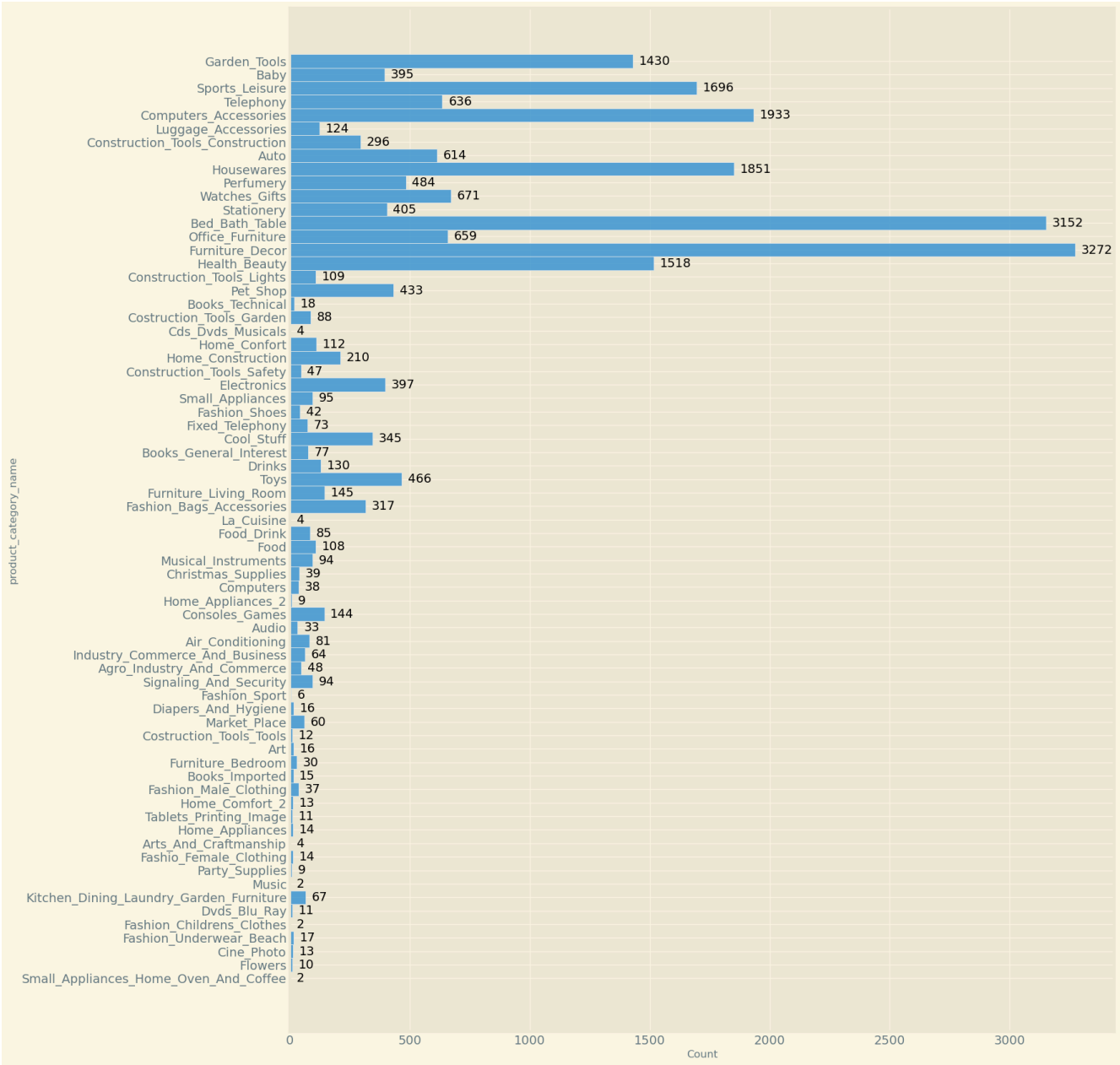
Hint: We need to find which of the top 10 combinations of products are selling together in each transaction. (combination of 2 or 3 buying together)

```
CrossSell_tb = pd.merge(left=orders, right=order_items, how="inner", on="order_id")[["order_id", "customer_id", "order_purchase_timestar

CrossSell_tb = CrossSell_tb[CrossSell_tb.duplicated(subset=["customer_id", "order_purchase_timestamp"], keep=False)]

CrossSell_tb = pd.merge(left=CrossSell_tb, right=product[["product_id", "product_category_name"]], how="inner", on="product_id")

CrossSell_tb
```

	order_id	customer_id	order_purchase_timestamp	product_
0	e6ce16cb79ec1d90b1da9085a6118aeb	494dded5b201313c64ed7f100595b95c	2017-05-16 19:41:00	08574b074924071f4e201e151b152b
1	e6ce16cb79ec1d90b1da9085a6118aeb	494dded5b201313c64ed7f100595b95c	2017-05-16 19:41:00	08574b074924071f4e201e151b152b
2	f585ab48a4d0027744dd2912cb42a2cb	133d446a88638f9aa5aac7f8c5e5164b	2017-10-08 18:04:00	08574b074924071f4e201e151b152b
3	f585ab48a4d0027744dd2912cb42a2cb	133d446a88638f9aa5aac7f8c5e5164b	2017-10-08 18:04:00	08574b074924071f4e201e151b152b
4	b3ef4de1762699288ca5c55a1e574a12	6f967a810363042cfb1dde2417d70324	2017-07-06 20:02:00	08574b074924071f4e201e151b152b
...	...	...	...	
23782	dbd4cbb492a12b99c5224014930acc18	a14f464b2a19d761f7f3f0885d5ea955	2017-03-26 14:12:00	d34efb58e1930773fcde9b951892b1
23783	e8fd20068b9f7e6ec07068bb7537f781	609b9fb8cad4fe0c7b376f77c8ab76ad	2017-08-10 21:21:00	0df37da38a30a713453b03053d60d3
23784	e8fd20068b9f7e6ec07068bb7537f781	609b9fb8cad4fe0c7b376f77c8ab76ad	2017-08-10 21:21:00	0df37da38a30a713453b03053d60d3
23785	9115830be804184b91f5c00f6f49f92d	da2124f134f5dfbce9d06f29bdb6c308	2017-10-04 19:57:00	c982dbea53b864f4d27c1d36f14b60
23786	9115830be804184b91f5c00f6f49f92d	da2124f134f5dfbce9d06f29bdb6c308	2017-10-04 19:57:00	49d2e2460386273b195e7e59b43587

23787 rows × 5 columns

```
plt.figure(figsize=(15,20))
plt.style.use("Solarize_Light2")

cross_sellChart = sns.histplot(y=CrossSell_tb.product_category_name)

for bar in cross_sellChart.containers:
    lables= [x.get_width() for x in bar]
    cross_sellChart.bar_label(container=bar, labels=lables, label_type="edge", padding=7)

plt.show()
```



4. Payment Behaviour

a. How customers are paying?

```
payment_tb = pd.merge(left=orders, right=order_payment, how="inner", on="order_id")

payment_tb.payment_type.value_counts().reset_index().rename(columns={"index":"payment methods"})
```

	payment methods	payment_type
0	credit_card	76795
1	UPI	19784
2	voucher	5775
3	debit_card	1529
4	not_defined	3

b. Which payment channels are used by most customers?

```
payment_tb.isna().sum()
```

```
order_id                           0
customer_id                        0
order_status                       0
order_purchase_timestamp           0
order_approved_at                175
order_delivered_carrier_date    1888
order_delivered_customer_date   3132
order_estimated_delivery_date      0
payment_sequential                 0
payment_type                       0
payment_installments               0
payment_value                      0
dtype: int64
```

```
payment_tb.dropna(axis="columns", inplace=True)
```

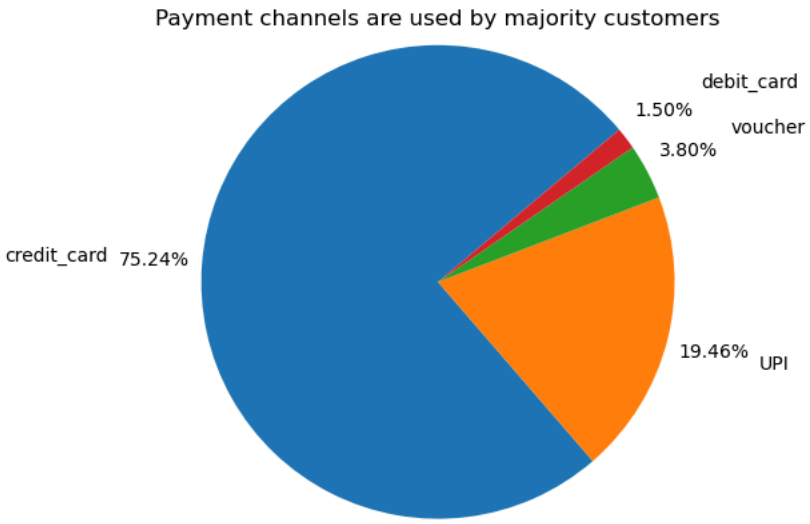
```
cust_payment = payment_tb[["customer_id","payment_type"]]

cust_payment = cust_payment.drop_duplicates()
```

```
CustPaymentUsage =cust_payment.groupby(by="payment_type").agg({"customer_id":"count"}
                                          ).reset_index().rename(columns={"customer_id":"payment_mode_usage"})
# CustPaymentUsage
CustPaymentUsage.payment_mode_usage = np.round((CustPaymentUsage.payment_mode_usage/cust_payment.shape[0])*100 , 2 )
```

```
CustPaymentUsage.sort_values(by="payment_mode_usage", ascending=False, inplace=True, ignore_index=True)

CustPaymentUsage
```

	payment_type	payment_mode_usage
0	credit_card	75.24
1	UPI	19.46
2	voucher	3.80
3	debit_card	1.50
4	not_defined	0.00

```
# plt.figure(figsize=(10,5))
plt.title("Payment channels are used by majority customers")
plt.pie(x=CustPaymentUsage.payment_mode_usage[0:4], labels=CustPaymentUsage.payment_type[0:4],startangle=40 ,
        autopct="%0.2f%%", pctdistance=1.2, labeldistance=1.4, radius=1.2)

plt.show()
```



5. Customer satisfaction towards category & product

a. Which categories (top 10) are maximum rated & minimum rated?

```
A = pd.merge(left=orders, right=order_review, how="left", on="order_id")

B = pd.merge(left=A, right=customer, how="left", on="customer_id")

C = pd.merge(left=B, right=order_items, how="left", on="order_id")

Df = pd.merge(left=C, right=product, how="left", on="product_id")

Df.drop(labels=["order_status","order_purchase_timestamp","order_estimated_delivery_date","review_answer_timestamp",
                "customer_unique_id", "price","freight_value","order_approved_at","order_delivered_carrier_date",
                 "order_delivered_customer_date","customer_zip_code_prefix","customer_city","customer_state",
                 "shipping_limit_date","product_name_lenght","product_description_lenght",
                 "product_photos_qty","product_weight_g","product_length_cm","product_height_cm","product_width_cm"],
        axis=1,
        inplace=True)

Df
```

	order_id	customer_id	review_id	review_score	review_
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	a54f0611adc9ed256b57ede6b6eb5114	4	
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	8d5266042046a06655c8db133d120ba5	4	
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	e73b67b67587f7644d5bd1a52deb1b01	5	
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	359d03e676b3c069f62cadba8dd3f6e8	5	
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	e50934924e227544ba8246aeb3770dd4	5	
...	...	...	...	...	
114095	63943bddc261676b46f01ca7ac2f7bd8	1fca14ff2861355f6e5f14306ff977a7	29bb71b2760d0f876dfa178a76bc4734	4	
114096	83c1379a015df1e13d02aae0204711ab	1aa71eb042121263aafbe80c1b562c9c	371579771219f6db2d830d50805977bb	5	
114097	11c177c8e97725db2631073c19f07b62	b331b74b18dc79bcdf6532d51e1637c1	8ab6855b9fe9b812cd03a480a25058a1	2	
114098	11c177c8e97725db2631073c19f07b62	b331b74b18dc79bcdf6532d51e1637c1	8ab6855b9fe9b812cd03a480a25058a1	2	
114099	66dea50a8b16d9b4dee7af250b4be1a5	edb027a75a1449115f6b43211ae02a24	dc9c59b4688062c25758c2be4cafc523	5	

114100 rows × 9 columns

TOP 10 MAXIMUM RATED CATEGORIES

```
Top10Max_categories = Df.groupby(by="product_category_name").agg({"order_id":"count"}
                                        ).rename(columns={"order_id":"no_of_orders"}
                                                ).sort_values(by="no_of_orders", ascending=False).reset_index().head(10)
```

```
Top10Max_categories
```

	product_category_name	no_of_orders
0	Bed_Bath_Table	11272
1	Health_Beauty	9728
2	Sports_Leisure	8701
3	Furniture_Decor	8416
4	Computers_Accessories	7895
5	Housewares	6989
6	Watches_Gifts	6001
7	Telephony	4550
8	Garden_Tools	4361
9	Auto	4256

```
Cat_MaxReview_tb = pd.merge(left=Top10Max_categories, right=Df, how="left", on="product_category_name")
```

```
print("TOP MAXIMUM RATED CATEGORIES")
Cat_MaxReview_tb.groupby(by="product_category_name").agg({"review_score":"sum"}).sort_values(by="review_score", ascending=False).reset_
```

```
TOP MAXIMUM RATED CATEGORIES
```

	product_category_name	review_score
0	Bed_Bath_Table	43636
1	Health_Beauty	40121
2	Sports_Leisure	35616
3	Furniture_Decor	32716
4	Computers_Accessories	30953
5	Housewares	28235
6	Watches_Gifts	24016
7	Telephony	17907
8	Garden_Tools	17572
9	Auto	17193

TOP 10 MINIMUM RATED CATEGORIES

```
Top10Min_categories = Df.groupby(by="product_category_name").agg({"order_id":"count"}
                                        ).rename(columns={"order_id":"no_of_orders"}
                                                ).sort_values(by="no_of_orders", ascending=True).reset_index().head(10)
```

```
Top10Min_categories
```

	product_category_name	no_of_orders
0	Security_And_Services	2
1	Fashion_Childrens_Clothes	8
2	La_Cuisine	14
3	Cds_Dvds_Musicals	14
4	Arts_And_Craftmanship	24
5	Home_Comfort_2	30
6	Fashion_Sport	31
7	Flowers	33
8	Furniture_Mattress_And_Upholstery	38
9	Music	38

```
Cat_MinReview_tb = pd.merge(left=Top10Min_categories, right=Df, how="left", on="product_category_name")
```

```
print("TOP MINIMUM RATED CATEGORIES")
Cat_MinReview_tb.groupby(by="product_category_name").agg({"review_score":"sum"}).sort_values(by="review_score", ascending=True).reset_in
```

```
TOP MINIMUM RATED CATEGORIES
```

	product_category_name	review_score
0	Security_And_Services	5
1	Fashion_Childrens_Clothes	36
2	La_Cuisine	53
3	Cds_Dvds_Musicals	65
4	Arts_And_Craftmanship	99
5	Home_Comfort_2	101
6	Fashion_Sport	132
7	Flowers	139
8	Furniture_Mattress_And_Upholstery	145
9	Music	160

b. Which products (top10) are maximum rated & minimum rated?

TOP 10 MAXIMUM RATED PRODUCTS

```
Top10Max_products = Df.groupby(by="product_id").agg({"order_id":"count"}
                                        ).rename(columns={"order_id":"no_of_orders"}
                                                ).sort_values(by="no_of_orders", ascending=False).reset_index().head(10)
```

```
Top10Max_products
```

	product_id	no_of_orders
0	aca2eb7d00ea1a7b8ebd4e68314663af	527
1	99a4788cb24856965c36a24e339b6058	491
2	422879e10f46682990de24d770e7f83d	487
3	389d119b48cf3043d311335e499d9c6b	392
4	368c6c730842d78016ad823897a372db	391
5	53759a2ecddad2bb87a079a1f1519f73	375
6	d1c427060a0f73f6b889a5c7c61f2ac4	343
7	53b36df67ebb7c41585e8d54d6772e08	323
8	154e7e31ebfa092203795c972e5804a6	293
9	3dd2a17168ec895c781a9191c1e95ad7	274

```
Product_MaxReview_tb = pd.merge(left=Top10Max_products, right=Df, how="left", on="product_id")
```

```
print("TOP 10 MAXIMUM RATED PRODUCTS")
Product_MaxReview_tb.groupby(by="product_id").agg({"review_score":"sum"}).sort_values(by="review_score", ascending=False).reset_index()
```

```
TOP 10 MAXIMUM RATED PRODUCTS
```

	product_id	review_score
0	aca2eb7d00ea1a7b8ebd4e68314663af	2112
1	422879e10f46682990de24d770e7f83d	1920
2	99a4788cb24856965c36a24e339b6058	1896
3	389d119b48cf3043d311335e499d9c6b	1612
4	368c6c730842d78016ad823897a372db	1531
5	53759a2ecddad2bb87a079a1f1519f73	1448
6	d1c427060a0f73f6b889a5c7c61f2ac4	1432
7	53b36df67ebb7c41585e8d54d6772e08	1348
8	154e7e31ebfa092203795c972e5804a6	1264
9	3dd2a17168ec895c781a9191c1e95ad7	1147