E-Commerce Project

A comprehensive e-commerce data analysis using Python

Presenter: Aileen Kate Fu

Agenda

- Data exploration and data cleaning
- Web analysis
- Sales trend analysis
- Product bucket analysis
- Promotion analysis
- Customer analysis
- Marketing solution

Part 1: Data Exploration and data cleaning

- **Datasets**
  - Customers
  - Orders_items
  - Orders
  - Products_skus
  - Products
  - Traffic
  - transactions
- **Data cleaning**
  - Check duplicate rows
  - Check missing values
  - Check incorrect words
  - Check outliers

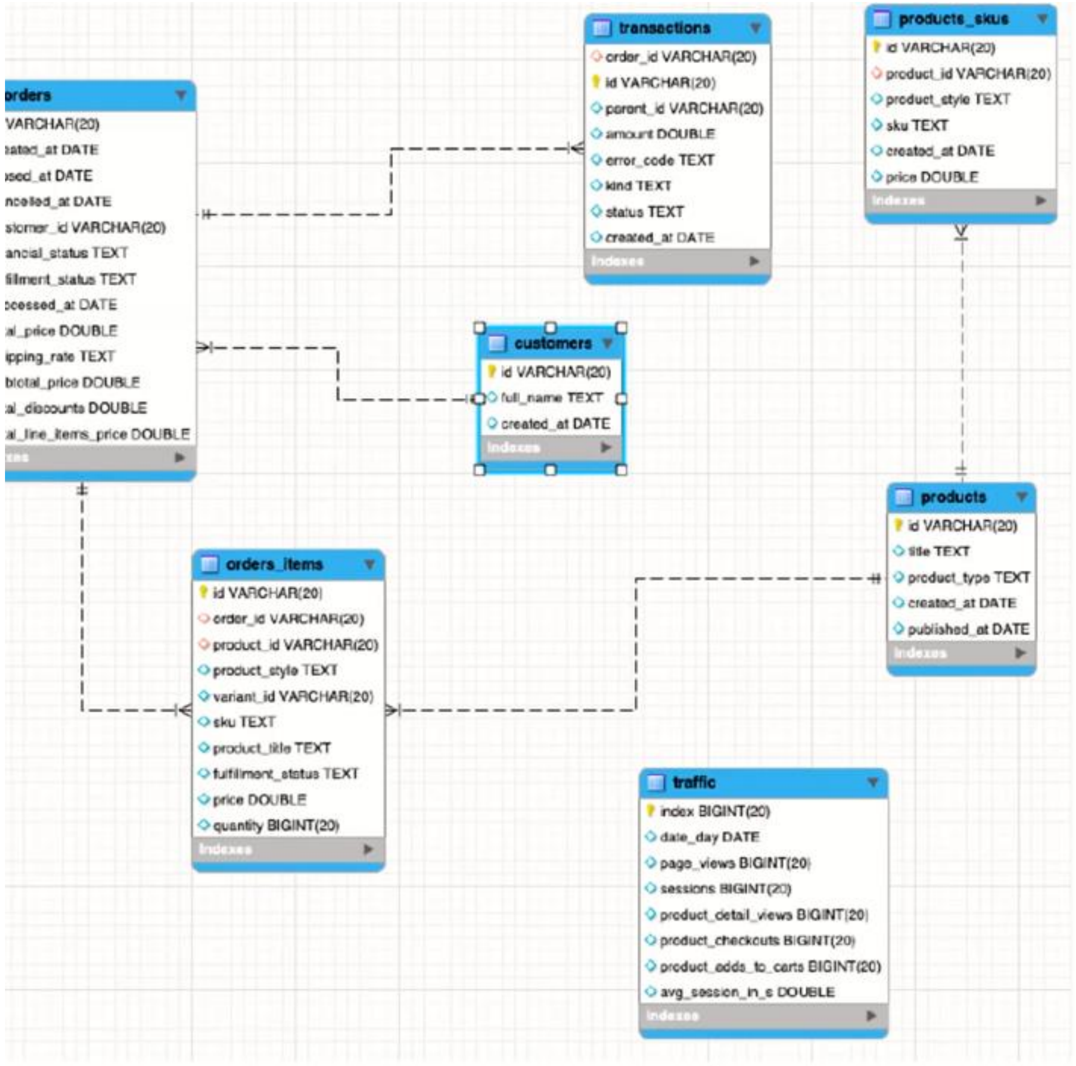
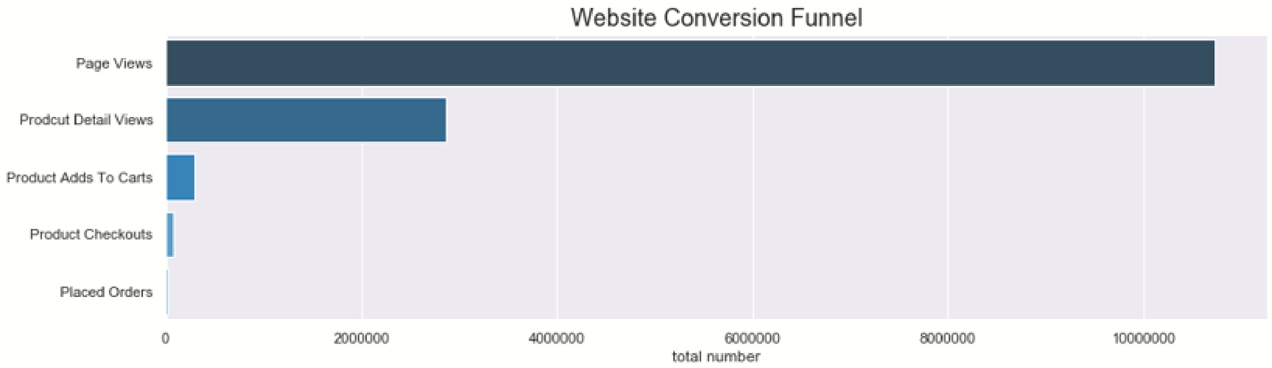


figure 1 EER diagram

Part 2: Web analysis

2.1 Funnel analysis

- The overall conversion rate from page_views to final placed_orders is only 0.19%:
- The conversion rate from product_detail_view to product_adds_to_carts is 10.1%, which is the lowest.

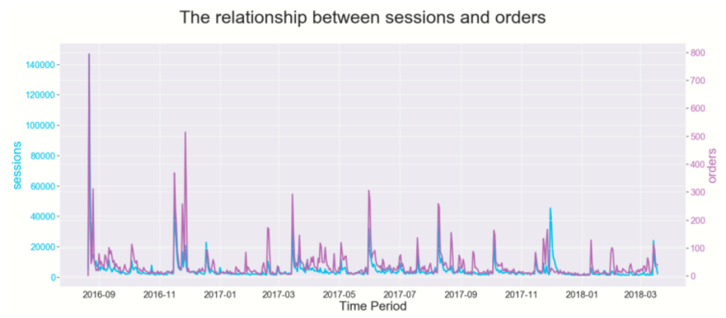


	total_number	conversion_rate_based_on_last_step	conversion_rate_based_on_page_views
Page_Views	10729488	NaN	1.000000
Product_Detail_Views	2869270	0.267419	0.267419
Product_Adds_To_Carts	289106	0.100759	0.026945
Product_Checkouts	84452	0.292114	0.007871
Placed_Orders	20879	0.247229	0.001946

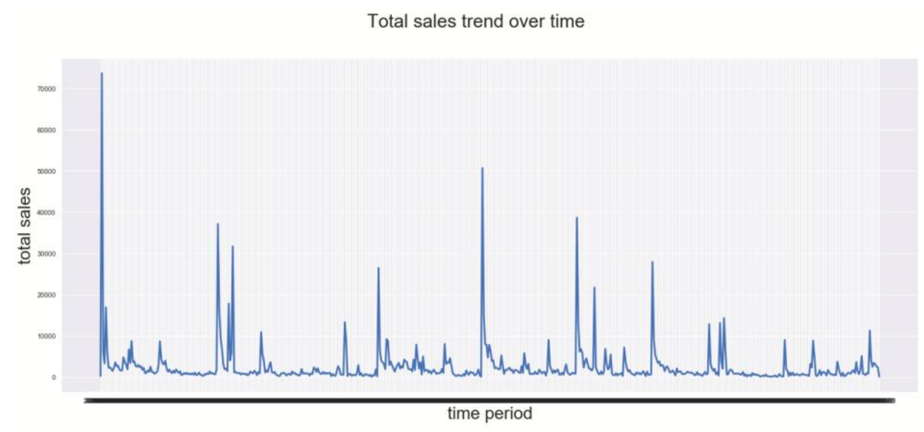
Part 2: Web analysis

2.2 Website traffic analysis

- The result shows a strong positive correlation between page_views, and the number of orders, coefficient is 0.82.
- Also, there is a strong positive correlation between sessions, product_detail_views and the number of orders.
- Therefore, we conclude that the website traffic has a significant correlation to the number of orders.



Part 3: Sales trend analysis



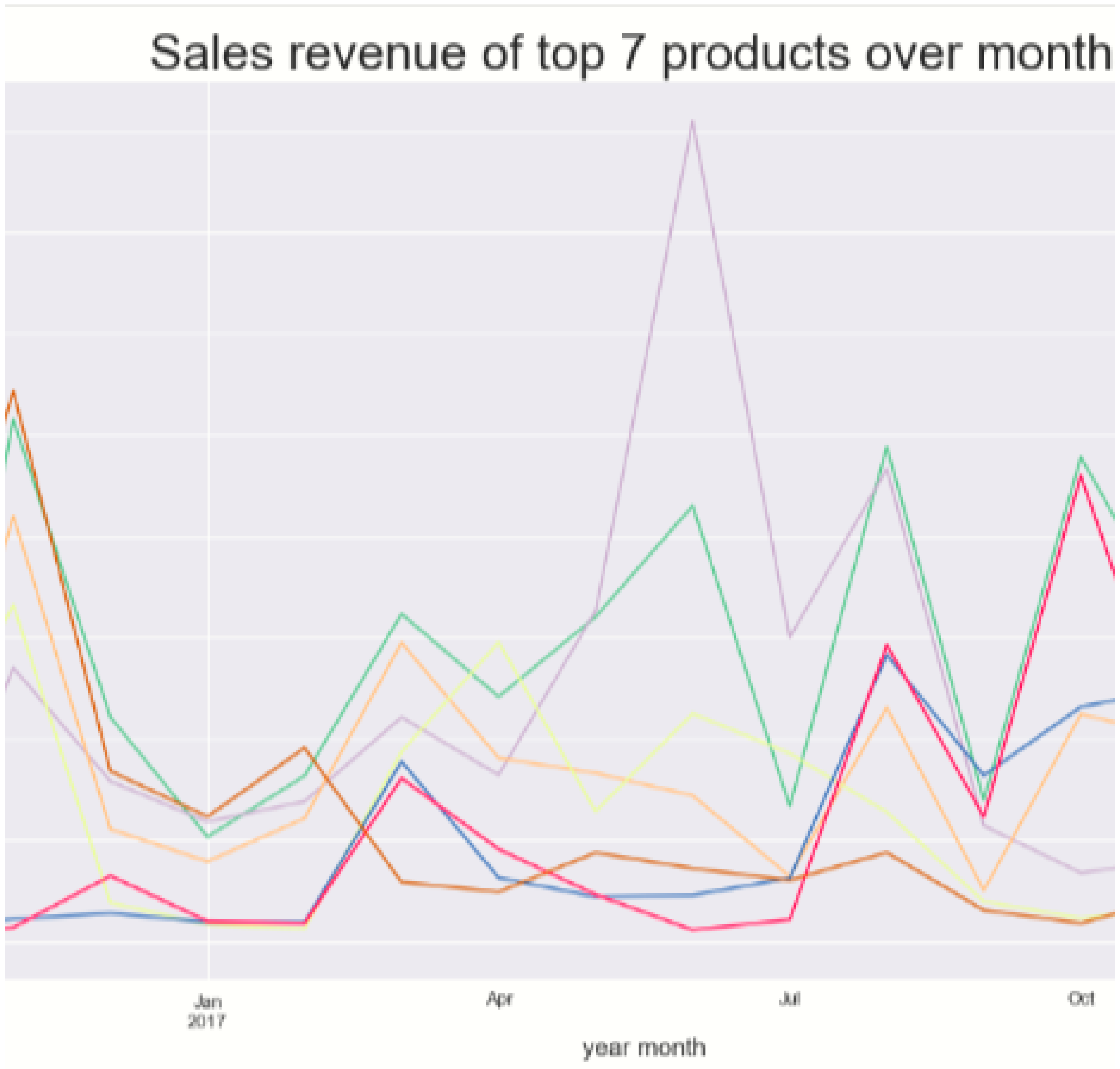
3.1 Total sales trend

- In the long term, the sales are decreasing from August 2016 to March 2018.
- The sales fluctuated with seasons and holidays. For example, the sales are higher during summer than winter (except Nov.).

Part 3: Sales trend analysis

3.2 Sales from different products

- The sales revenue of different product type fluctuated seasonally:
- The revenue of Dress reaches highest in the summer (June 2017).
- The revenue of Sweater reaches highest at the beginning of winter (Oct 2017).
- The revenue of Top is higher in the summer and autumn, and lower in the winter.
- The revenue of Trousers is higher in the spring and autumn, and lower in the winter and summer.



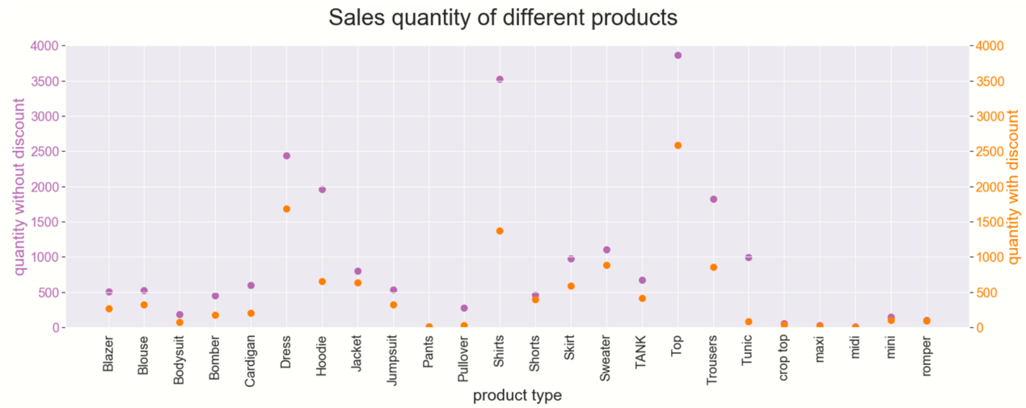
product_id	product_id_purchased_together	purchased_counts	num_orders	percentage_purchased_together
521264076285.0	521264043517.0	287	324	0.8858024691
1007444137469.0	1007444071933.0	21	26	0.8076923077
850182018557.0	850181953021.0	49	70	0.7000000000
850182444541.0	850182411773.0	42	66	0.6363636364
850183165437.0	850183198205.0	15	24	0.6250000000
933949445629.0	933949543933.0	17	28	0.6071428571
1007444071933.0	1007444137469.0	21	35	0.6000000000
850182903293.0	850182968829.0	25	42	0.5952380952
521264043517.0	521264076285.0	287	485	0.5917525773
521263846909.0	521264371197.0	110	186	0.5913978495

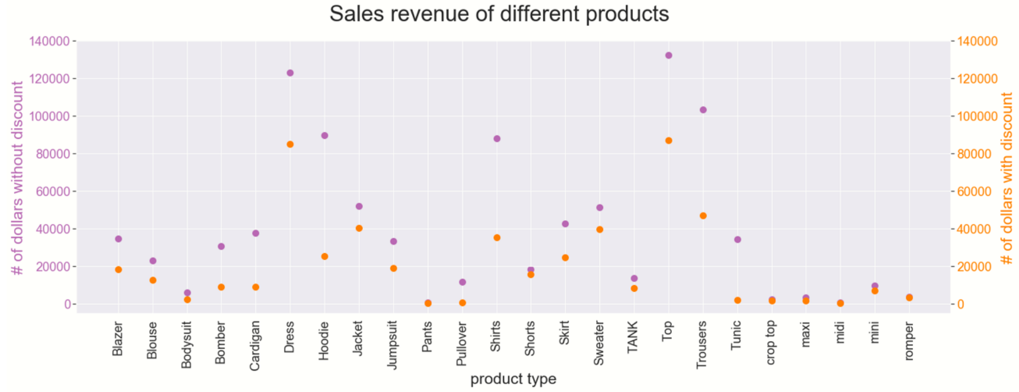
Part 4: Product bucket analysis

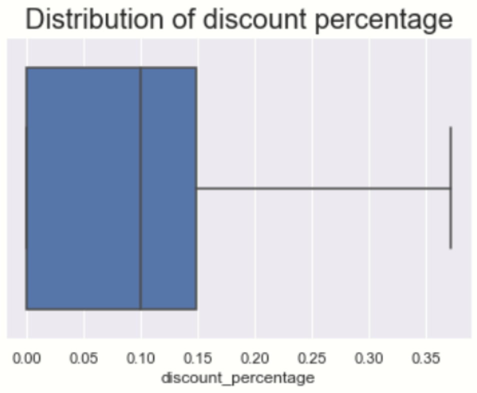
We can see the items that often bought together and with the percentage they were bought together.

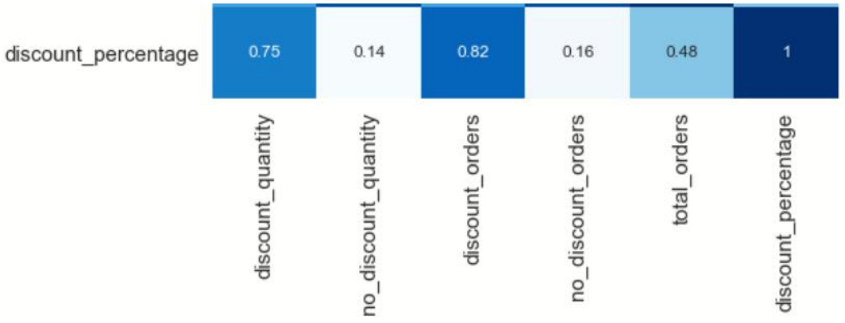
Part 5: Promotion analysis

- The correlation coefficient of discount_percentage and discount_quantity is 0.75.
- The correlation coefficient of discount_percentage and discount_orders is 0.82.
- We can conclude that discount does promote sales.







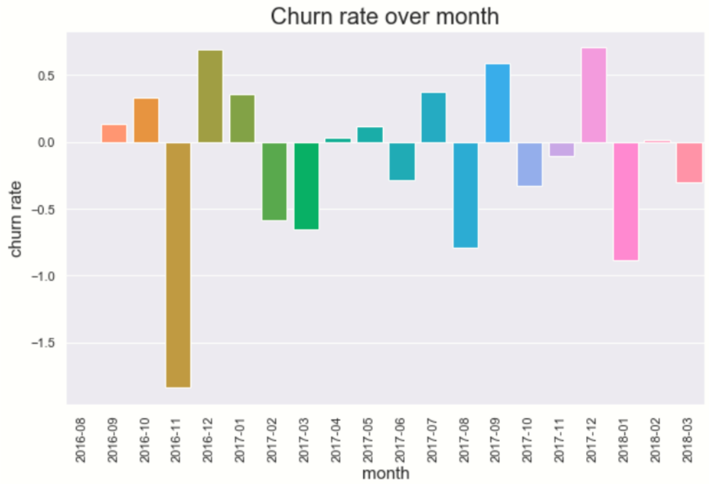


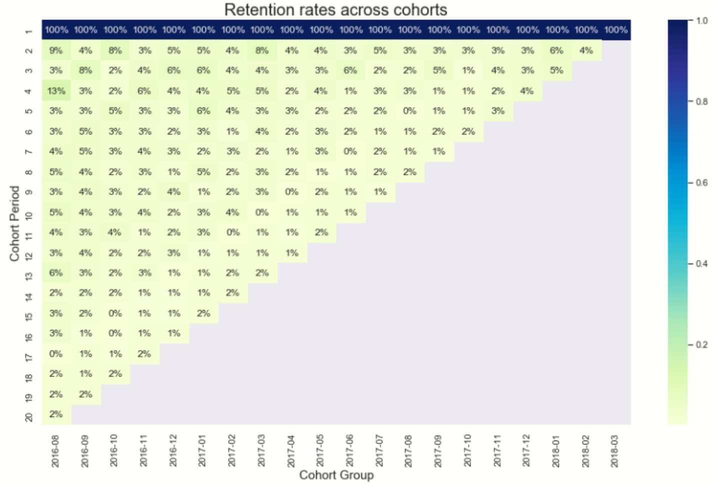
Part 6: Customer analysis

6.1 Churn rate and Retention rate

- The number of unique customers is decreasing overally.

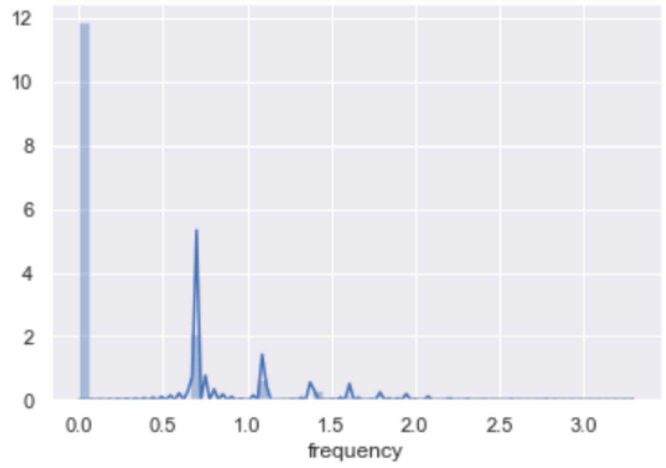
- In the cohort period 2, the retentin rate is less than 10%. After the first purchase, we can see that over 90% customers will not purchase again in the second month. It is not good for this online store if the store seeks for sustainable growth, we should focus on the old customers.

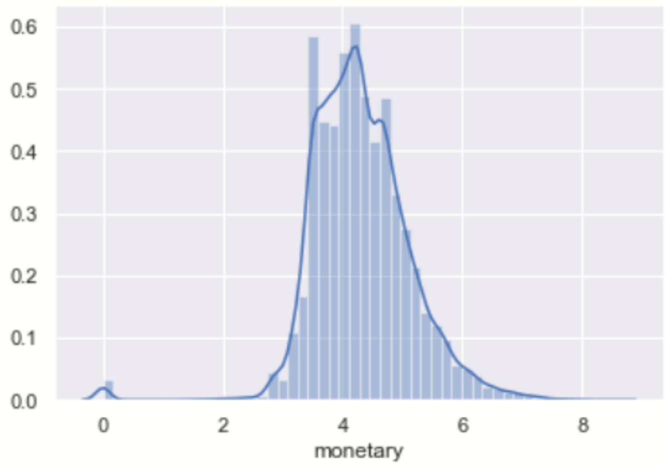


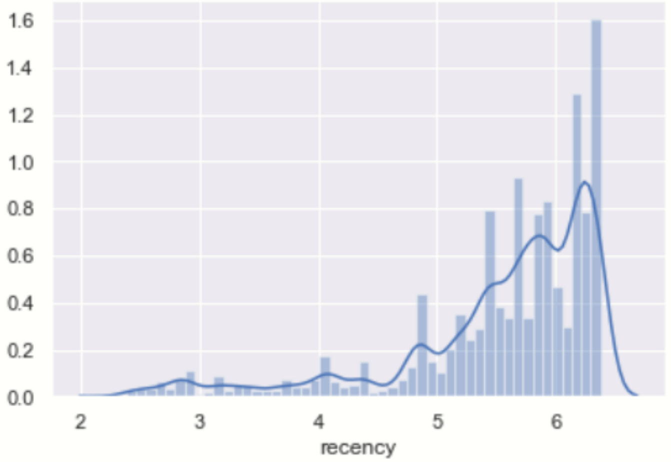


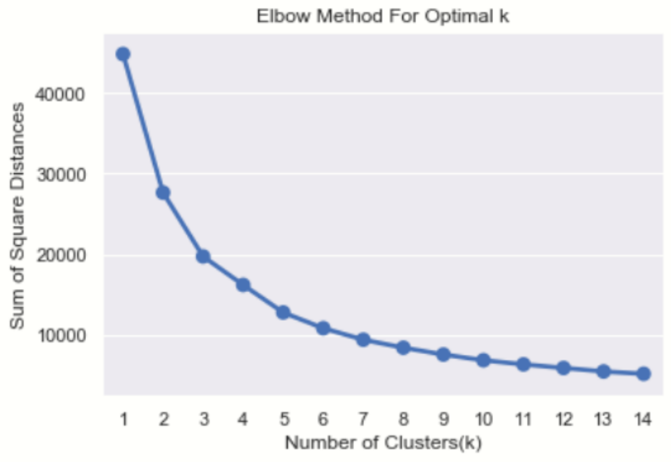
Part 6: Customer analysis

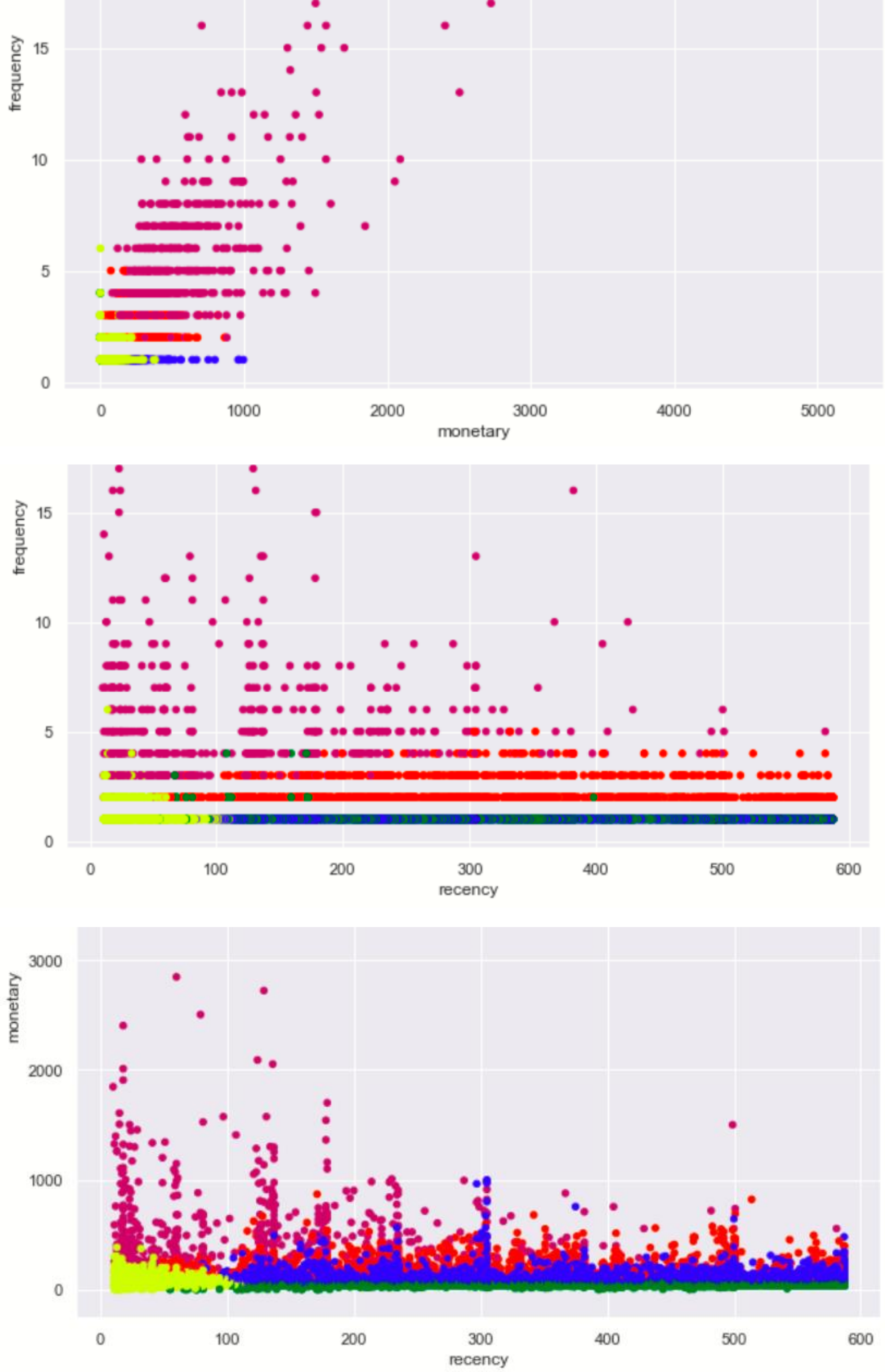
- **6.2 RFM analysis - Process**
  - In order to improve the retention rate, we need to get a better understanding of the old customers.
  - Instead of analyzing the entire customer base as a whole, it's better to segment them into homogeneous groups, understand the traits of each group, and engage them with relevant campaigns rather than segmenting on just customer age or geography.











Part 6: Customer analysis

6.2 RFM analysis - Result

- Cluster 1 **At Risk Customers**

This group of customers who spent big amounts, but they haven't purchased recently.

Send them personalized reactivation campaigns to reconnect, and to offer renewals and helpful products to encourage another purchase.

- Cluster 2 **New Customers**

This group of customers who have a high overall recency score but are not frequent shoppers.

Start building relationships with these customers by providing onboarding support and special offers to increase their visits.

- Cluster 3 **Champions**

This group of customers are the best customers, who bought most recently, most often, and are heavy spenders.

Reward these customers. They can become early adopters for new products and will help promote the brand.

- Cluster 4 **Require Activation**

Poorest performers of our RFM model. They might have stay with our competitors for now and will require a different activation strategy to win them back.

Bring them back with relevant promotions and run surveys to find out what went wrong and avoid losing them to a competitor.

- Cluster 0 **Potential Loyalists**

This group of customers with average frequency and who spent a good amount.

Offer membership or loyalty programs or recommend related products to upsell them and help them become your Loyalists or Champions.

Part 7: Marketing solution

1. Improve website conversion rates

- **Improve product detail page**

Because the conversion from product_detail_view to product_adds_to_carts is low. We need to improve product detail page.

Informative product page would ease decision-making. Check if there are enough images fully display the product, do the images look real, is the description specific and concrete

Personalized and non-personalized recommendation system. User can shop the suits, compare similar items, or browse other items if this one doesn't fit. Non-personalized recommendations can be provided based on the association rules discovered, popular items and highly-reated items. Personalized recommender can be built with algorithms like collaborative filtering.

- **Recover abandoned cart**

Check and remove possible frictions that prevent checkout, e.g., do they hide shipping and tax until the last step, is the checkout process too complex, are there enough payment options etc.

It’s common that people use cart as favorite list, they add items but are just watching with low purchasing intent. Incentivize them to make the purchase by **ad retargeting** (remarketing the items to the user on different platforms) or sending a follow-up email with **time-limited coupon**.

Part 7: Marketing solution

2. Promote popular products at certain seasons and improve recommendation system

- **Improve the variety of popular products at certain seasons**

From the sales trend analysis, we find that the sales revenue and quantity of different product type fluctuated seasonally.

The Top product is the best seller in terms of quantity. The quantity is higher in summer and autumn, and lower in the winter. To help the sales growth, we can increase the variety of Top product, have more promotions for this type of product in summer and autumn.

The quantity of Dress is highest in summer, especially in June. We should increase the variety of Dress, have more promotion and recommendation for this type of product in summer.

The quantity of Sweater is highest in autumn. In order to improve the sales, we should increase the variety of Sweater in autumn.

- **Improve the recommendation system**

From the product bucket analysis, we can see some products were oftenly bought together by the customers. To achieve the sales growth, we should improve the algorithms of recommendation system. Recommend the items to customers that could potentially bought together.

Part 7: Marketing solution

3. Offer discount at larger rate; Offer more discount at holidays

We concluded that discount does promote sales. In order to increase the sales revenue and quantity, we can offer discount at larger rate.

We also observe that the sales is high at holidays, especially during Thanksgiving. To boost the sales, we can have more promotions at holidays.

- ## 4. Build loyalty

Launch or improve current loyalty program.

Do customer satisfaction research and improve customer experience. e.g., Do customer find what they need/like? Are they satisfied with the product/service? Does the company capture the latest fashion trends?

see the marketing strategy which provided at the send of RFM analysis (Part 6).

- ## 5.Create awareness and interests

The pricing, user size and sales volume suggest the company is probably a fast fashion startup. Given that web traffic has significant impact on sales, it would be good for them to focus on creating awareness by boosting traffic. Possible solutions:

Collaborate with designers to provide exclusive editions

Collaborate with influencers/carry out social media campaigns

Ads

Appendix: Python coding snapshot

Part 1: Data exploratory analysis

Merge dataset

In [16]:
```
# rename key joined columns
orders_items = orders_items.rename(columns={'id':'orders_items_id'})
orders = orders.rename(columns={'id':'order_id'})
products = products.rename(columns={'id':'product_id'})
```

In [17]:
```
# merge table 'orders_items' & 'products'
df2_1 = pd.merge(left=orders_items, right=products, how='left', on='produc
```

In [18]:
```
# merge table 'orders'
df2_2 = pd.merge(left=df2_1, right= orders, how='left', on ='order_id')
```

In [19]:
```
# merge table 'transactions'
df2_3 = pd.merge(left=df2_2, right=transactions, how='left', on='order_id'
```

In [20]:
```
df2_3.head()
```

Out[20]:

orders_items_id	order_id	product_id	product_style	
13325125855	7675398239	1.292763e+10	2c259a42d38f5f097274beff811168e2	505
13325125855	7675398239	1.292763e+10	2c259a42d38f5f097274beff811168e2	505
13327045983	7676331935	1.292763e+10	dd804c4025d230467823200aa82e9219	505

Appendix: Python coding snapshot

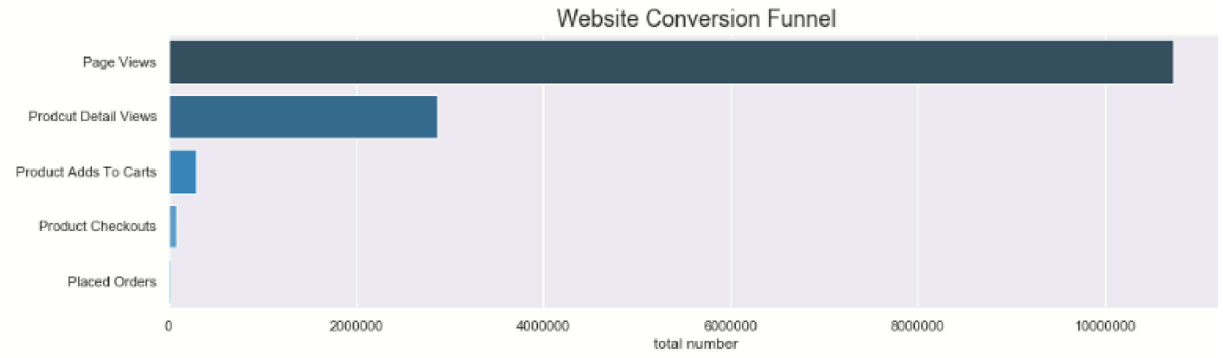
Part 2: Web funnel analysis

In [50]:
```python
dfq1 = dfq1.rename(columns={'order_id':'order_count'})
dfq1.head()
```

Out[50]:

	index	date_day	page_views	sessions	product_detail_views	product_checkouts	product_adds_to_carts	avg_session_in_
0	4	2016-08-21	10276	4946	0	0	0	73.47048
1	5	2016-08-22	625003	146860	175257	5639	10851	142.40783
2	6	2016-08-23	220707	61654	58940	761	1817	106.16144
3	7	2016-08-24	93694	27182	24935	256	638	98.99966
4	8	2016-08-25	63927	15239	19167	901	1826	130.41085

In [51]:
```python
f, ax = plt.subplots(figsize=(15,4))
ax = sns.barplot(y=['Page Views','Prodcut Detail Views', 'Product Adds To Carts', 'Product Checkouts', 'Placed
                x=[dfq1['page_views'].sum(), dfq1['product_detail_views'].sum(),dfq1['product_adds_to_carts'].
                 palette="Blues_d")
ax.set_title('Website Conversion Funnel', fontsize=18)
plt.ticklabel_format(style='plain', axis='x')
plt.xlabel('total number')
plt.show()
```



Appendix: Python coding snapshot

Part 2: Web traffic correlation

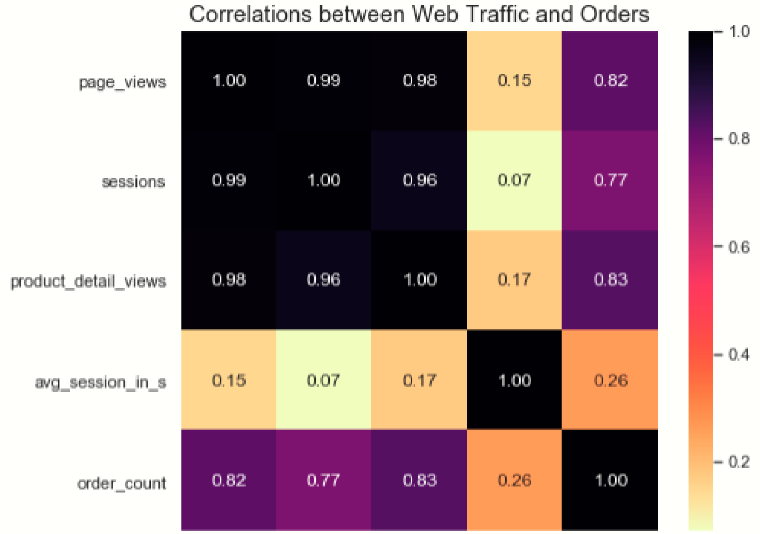
3.2.1 correlation matrix

In [57]:
```
corr = dfq1[['page_views','sessions','product_detail_views','avg_session_in_s','order_count']].corr()
corr
```

Out[57]:

	page_views	sessions	product_detail_views	avg_session_in_s	order_count
page_views	1.000000	0.989081	0.984876	0.150182	0.815809
sessions	0.989081	1.000000	0.959438	0.070872	0.770344
product_detail_views	0.984876	0.959438	1.000000	0.173966	0.828847
avg_session_in_s	0.150182	0.070872	0.173966	1.000000	0.261292
order_count	0.815809	0.770344	0.828847	0.261292	1.000000

In [58]:
```
sns.set(color_codes=True)
plt.figure(figsize=(7,6))
ax.set_title = sns.heatmap(corr, annot=True, fmt='.2f', cmap='magma_r')
plt.title('Correlations between Web Traffic and Orders', fontsize=15)
plt.show()
```



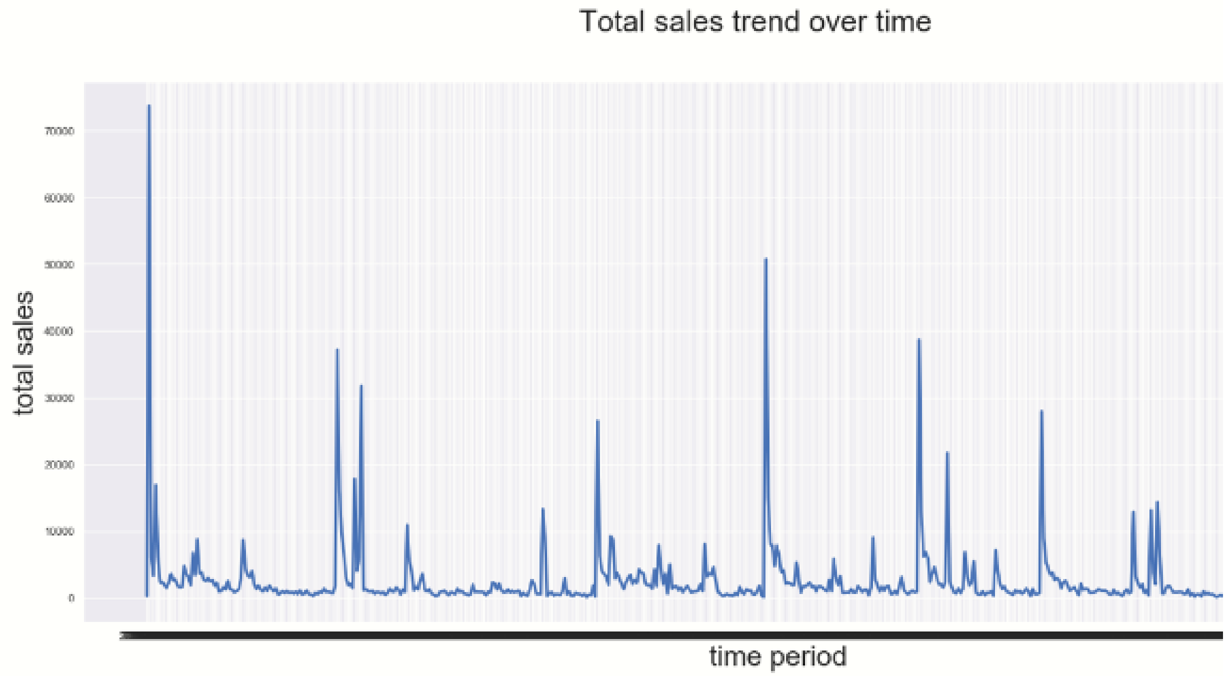
Appendix: Python coding snapshot

Part 3: Sales trend analysis

4.1 Total sales trend

[62]:
```python
dfq4['order_item_sale'] = dfq4['price']*dfq4['quantity']
df4_4 = dfq4[['created_at','order_item_sale']].groupby('created_at').sum()
```

[63]:
```python
fig, ax1 = plt.subplots(figsize=(25,10))
fig.suptitle('Total sales trend over time', fontsize=30)
ax1.set_xlabel('time period', fontsize=28)
ax1.set_ylabel('total sales', fontsize=28)
ax1.plot(df4_4.index, df4_4['order_item_sale'], linewidth=3)
ax1.tick_params(axis='y')
```



Appendix: Python coding snapshot

Part 4: Product bucket analysis

```
purchased['percentage_purchased_together'] = purchased.purchased_counts/purchased.num_orders
pd.set_option('display.precision',10)
purchased.sort_values(by='percentage_purchased_together', ascending=False).head()
```

	product_id	product_id_purchased_together	purchased_counts	num_orders	percentage_purchased
14717	1183022257661.0	1183021831677.0	2	2	1.00
14720	1183022257661.0	1183022159357.0	2	2	1.00
5969	521264076285.0	521264043517.0	287	324	0.88
14770	1183022388733.0	1183022454269.0	5	6	0.83
14452	1007444137469.0	1007444071933.0	21	26	0.80

```
# select number of orderst that is higher than 20
purchased[purchased['num_orders']>20].sort_values(by='percentage_purchased_together', ascending=Fa
```

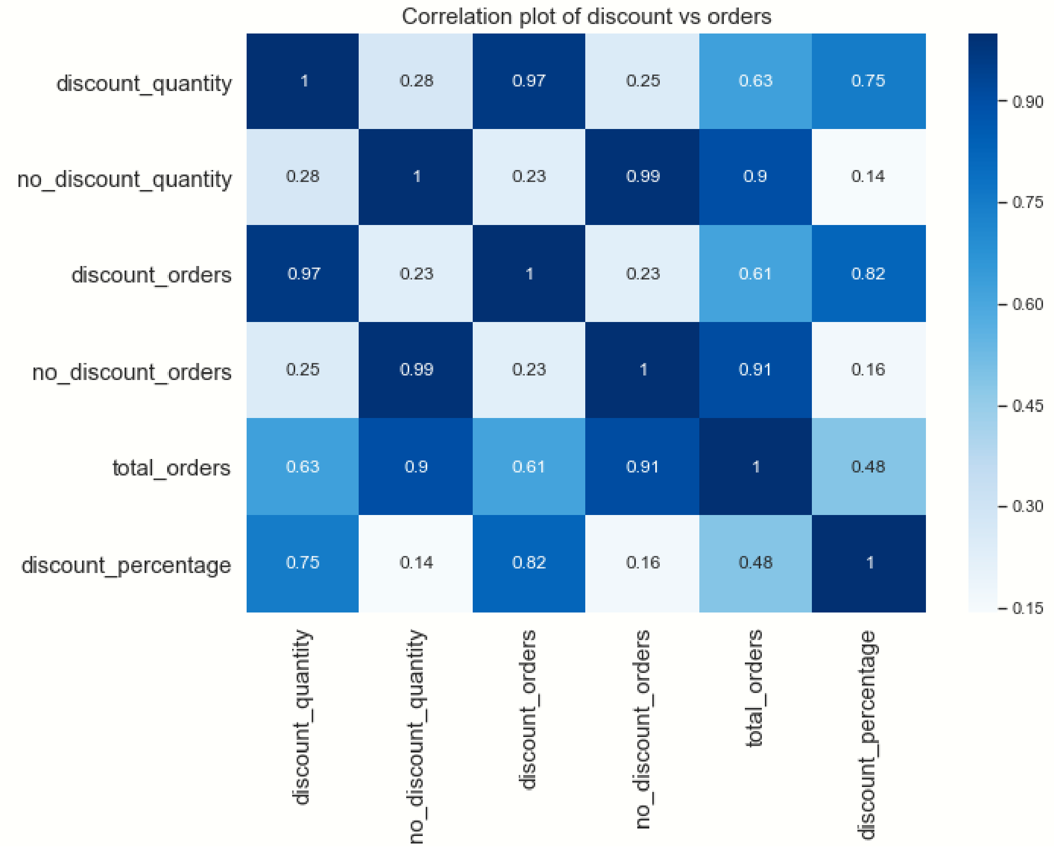
	product_id	product_id_purchased_together	purchased_counts	num_orders	percentage_purchased
5969	521264076285.0	521264043517.0	287	324	0.88
14452	1007444137469.0	1007444071933.0	21	26	0.80
10204	850182018557.0	850181953021.0	49	70	0.70
11002	850182444541.0	850182411773.0	42	66	0.63
11862	850183165437.0	850183198205.0	15	24	0.62
13113	933949445629.0	933949543933.0	17	28	0.60
14427	1007444071933.0	1007444137469.0	21	35	0.60
11505	850182903293.0	850182968829.0	25	42	0.59
5865	521264043517.0	521264076285.0	287	485	0.59
5436	521263846909.0	521264371197.0	110	186	0.59

Appendix: Python coding snapshot

Part 5: Promotion analysis

```
[125]: dfq6_1 = dfq6.groupby('created_at')['discount_quantity','no_discount_quantity','discount_orders','no_discount_orde

[126]: fig, ax = plt.subplots(figsize=(10, 7))
       corr_plot = sns.heatmap(dfq6_1.corr(),cmap='Blues',annot=True)
       ax.tick_params(axis="x", labelsize=15)
       ax.tick_params(axis="y", labelsize=15)
       plt.title("Correlation plot of discount vs orders", fontsize=15,  y=1.08)
       plt.show()
```



Appendix: Python coding snapshot

Part 6: Customer analysis

1) Churn rate ¶

In [130]:
```
df7_1 = dfq4[['month_year','customer_id']].groupby('month_year').agg({'customer_id':pd.Series.nunique})
df7_1 = df7_1.rename(columns={'customer_id':'num_customers'})
df7_1.reset_index('month_year',inplace=True)
```

In [131]:
```
df7_1['last_num_customers'] = df7_1['num_customers'].shift(periods=1)
df7_1['num_churned'] = df7_1['last_num_customers'] - df7_1['num_customers']
df7_1['churn_rate'] = df7_1['num_churned']/df7_1['last_num_customers']
df7_1.drop('last_num_customers',axis=1,inplace=True)
```

In [132]:
```
df7_1
```

Out[132]:

	month_year	num_customers	num_churned	churn_rate
0	2016-08	1393.0	NaN	NaN
1	2016-09	1205.0	188.0	0.1349605169
2	2016-10	812.0	393.0	0.3261410788
3	2016-11	2302.0	-1490.0	-1.8349753695
4	2016-12	718.0	1584.0	0.6880973067
5	2017-01	460.0	258.0	0.3593314763
6	2017-02	731.0	-271.0	-0.5891304348
7	2017-03	1209.0	-478.0	-0.6538987688
8	2017-04	1172.0	37.0	0.0306038048
9	2017-05	1031.0	141.0	0.1203071672
10	2017-06	1327.0	-296.0	-0.2870999030
11	2017-07	834.0	493.0	0.3715146948
12	2017-08	1496.0	-662.0	-0.7937649880
13	2017-09	622.0	874.0	0.5842245989
14	2017-10	826.0	-204.0	-0.3279742765
15	2017-11	916.0	-90.0	-0.1089588378
16	2017-12	271.0	645.0	0.7041484716
17	2018-01	512.0	-241.0	-0.8892988930
18	2018-02	507.0	5.0	0.0097656250

Appendix: Python coding snapshot

Part 6: Customer analysis

2) Retention rate

In [138]:
```
df_orders = orders
```

In [143]:
```
df_orders['order_month'] = df_orders['created_at'].dt.to_period('M')
```

In [144]:
```
df_orders['cohort_month'] = df_orders.groupby('customer_id')['order_month'].transform('min')
```

In [145]:
```
df_orders.head()
```

Out[145]:

	order_id	created_at	closed_at	cancelled_at	customer_id	financial_status	fulfillment_status	processed_at	total_price	s
0	7675398239	2016-08-21	2016-08-25	2016-08-22	8683754719	voided	NaN	2016-08-21	44.57	
1	7676331935	2016-08-22	2016-08-22	NaT	8686224991	refunded	NaN	2016-08-22	124.55	
2	7676363167	2016-08-22	NaT	2016-08-22	8686224991	voided	NaN	2016-08-22	97.68	
3	7676539359	2016-08-22	2016-08-22	NaT	8686915935	paid	fulfilled	2016-08-22	131.10	
4	7676549855	2016-08-22	2016-08-22	NaT	8686924319	paid	fulfilled	2016-08-22	91.12	

In [146]:
```
grouped = df_orders.groupby(['cohort_month', 'order_month'])
```

In [147]:
```
cohorts = grouped.agg({'customer_id': pd.Series.nunique,
                       'order_id': pd.Series.nunique})
cohorts.rename(columns={'customer_id':'total_customers',
                        'order_id':'total_orders'}, inplace=True)
```

In [148]:
```
def cohort_period(df):
    df['cohort_period'] = np.arange(len(df)) + 1
    return df

cohorts = cohorts.groupby(level=0).apply(cohort_period)
cohorts.head()
```

Appendix: Python coding snapshot

Part 6: Customer RFM analysis

```
In [170]: # split into four segments using quantiles
          quantiles = rfm.quantile(q=[0.25,0.5,0.75])
          quantiles = quantiles.to_dict()

In [171]: def Rscore(x, p, d):
              if x <= d[p][0.25]:
                  return 1
              elif x <= d[p][0.50]:
                  return 2
              elif x <= d[p][0.75]:
                  return 3
              else:
                  return 4

          def Fscore(x, p, d):
              if x <= d[p][0.25]:
                  return 4
              elif x <= d[p][0.50]:
                  return 3
              elif x <= d[p][0.75]:
                  return 2
              else:
                  return 1

          def Mscore(x, p, d):
              if x <= d[p][0.25]:
                  return 4
              elif x <= d[p][0.50]:
                  return 3
              elif x <= d[p][0.75]:
                  return 2
              else:
                  return 1

In [172]: rfm['R'] = rfm['recency'].apply(Rscore, args=('recency', quantiles))
          rfm['F'] = rfm['frequency'].apply(Fscore, args=('frequency', quantiles))
          rfm['M'] = rfm['monetary'].apply(Mscore, args=('monetary', quantiles))
```