GROUP-4

CAPSTONE PROJECT-1

Customer Segmentation

- RFM Analysis
- K-Means Modelling
- Cohort Analysis

Contents

Group Members

- C8124-Jack Sellow
- C8125-Mustafa
- C8250-Onur
- C8278-Engin
- C8292-Ken
- C8301-Sam
- C8307-Hakan
- C8315-Halit
- C8399-Benjamin
- C8492-Semih
- C9231-Hüseyin

Main Steps

1. Data Cleaning & EDA
2. RFM Analysis
3. RFM Segmentation
4. K-Means Modelling
5. Cohort Analysis

Schedule

04-06 December 2021	:	Individual study
06 December 2021 13:00 (IST)	:	Data Cleaning & EDA
07 December 2021 20:00 (IST)	:	RFM Analysis & Segmentation
08 December 2021 20:00 (IST)	:	K-Means Modelling & Cohort Analysis
09 December 2021 13:00 (IST)	:	Review

Analysis

EDA

RFM Analysis

K-Means

Cohort Analysis

Methodology

Columns : 8 | Rows : 541.909 | Duplicated: 5.268

#	Column	Non-Null Count	Dtype
0	InvoiceNo	541909 non-null	object
1	StockCode	541909 non-null	object
2	Description	540455 non-null	object
3	Quantity	541909 non-null	int64
4	InvoiceDate	541909 non-null	datetime64[ns]
5	UnitPrice	541909 non-null	float64
6	CustomerID	406829 non-null	float64
7	Country	541909 non-null	object

Descriptive Statistics

	Quantity	UnitPrice	CustomerID
count	541909.00	541909.00	406829.00
mean	9.55	4.61	15287.69
std	218.08	96.76	1713.60
min	-80995.00	-11062.06	12346.00
25%	1.00	1.25	13953.00
50%	3.00	2.08	15152.00
75%	10.00	4.13	16791.00
max	80995.00	38970.00	18287.00

Missing Values:

	Missing_Number	Missing_Percent
CustomerID	135080	0.249267
Description	1454	0.002683

Number of Uniques:

InvoiceNo	25900
StockCode	4070
Description	4223
Quantity	722
InvoiceDate	23260
UnitPrice	1630
CustomerID	4372
Country	38

Descriptive Statistics (Categorical Columns)

	count	unique	top	freq
InvoiceNo	541909	25900	573585	1114
StockCode	541909	4070	85123A	2313
Description	540455	4223	WHITE HANGING HEART T-LIGHT HOLDER	2369
Country	541909	38	United Kingdom	495478

Methodology

Analysis

1. InvoiceNo
2. StockCode
3. Description
4. Quantity
5. InvoiceDate
6. UnitPrice
7. CustomerID
8. Country

25.900 unique invoices

3.836 cancelled invoices

22.064 others

```
df["invoiceno"].str.startswith('C').value_counts(normalize = True)*100
```

```
False    98.286059
True      1.713941
```

Methodology

Analysis

1. InvoiceNo
2. StockCode
3. Description
4. Quantity
5. InvoiceDate
6. UnitPrice
7. CustomerID
8. Country

AMAZONFEE : (32 of the 34 AMAZONFEE invoices have been cancelled)

BANK CHARGES : (25 of the 37 BANK CHARGE invoices have been cancelled)

D : Discount (All of the 77 D invoices have been cancelled)

M / m : Manual (244 of the 572 M invoices have been cancelled)

S : Samples (61 of the 63 S invoices have been cancelled. InvNo : 549684 – 572849)

B : Adjust bad debt (2 of the 3 B invoices have been cancelled)

C2 : Carriage (2 of the 244 C2 invoices have been cancelled)

DOR : DOTCOM POSTAGE (1 of the 710 DOT invoices has been cancelled)

POST : POSTAGE (126 of the 1256 POST invoices have been cancelled)

PADS : PADS TO MATCH ALL CUSHIONS (Any ofthe 4 PADS invoices has not been cancelled)

CRUK : CRUK Commission (All of the 16 CRUK invoices have been cancelled)

Methodology

Analysis

1. InvoiceNo
2. StockCode
3. Description
4. Quantity
5. InvoiceDate
6. UnitPrice
7. CustomerID
8. Country

```
(df["Quantity"] < 0).value_counts()
```

```
False    526054
True      10587
```

A negative quantity indicates canceled invoices or returned items.

```
df[(df["InvoiceNo"].str.find("C").isnull()) & (df["Quantity"] < 0)]["Description"].unique()
```

```
array([nan, '?', 'check', 'damages', 'faulty', 'Dotcom sales',
       'reverse 21/5/10 adjustment', 'mouldy, thrown away.', 'counted',
       'Given away', 'Dotcom', 'label mix up', 'samples/damages',
       'thrown away', 'incorrectly made-thrown away.', 'showroom', 'MIA',
       'Dotcom set', 'wrongly sold as sets', 'Amazon sold sets',
       'dotcom sold sets', 'wrongly sold sets', '? sold as sets?',
       '?sold as sets?', 'Thrown away.', 'damages/display',
       'damaged stock', 'broken', 'throw away', 'wrong barcode (22467)',
       'wrong barcode', 'barcode problem', '?lost',
       "thrown away-can't sell.", "thrown away-can't sell", 'damages?',
       're dotcom quick fix.', "Dotcom sold in 6's", 'sold in set?',
       'cracked', 'sold as 22467', 'Damaged',
       'mystery! Only ever imported 1800',
       'MERCHANT CHANDLER CREDIT ERROR, STO', 'POSSIBLE DAMAGES OR LOST?',
       'damaged', 'DAMAGED', 'Display', 'Missing', 'wrong code?',
       'wrong code', 'adjust', 'crushed', 'damages/showroom etc',
       'samples', 'damages/credits from ASOS.',
       'Not rcvd in 10/11/2010 delivery', 'Thrown away-rusty',
```

Methodology

Analysis

1. InvoiceNo
2. StockCode
3. Description
4. Quantity
5. InvoiceDate
6. UnitPrice
7. CustomerID
8. Country

```
print("Max Date : ", df["InvoiceDate"].max())
print("Min Date : ", df["InvoiceDate"].min())
```

```
Max Date :  2011-12-09 12:50:00
Min Date :  2010-12-01 08:26:00
```

```
df[df["UnitPrice"] < 0]
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
299983	A563186	B	Adjust bad debt	1	2011-08-12 14:51:00	-11062.06	NaN	United Kingdom
299984	A563187	B	Adjust bad debt	1	2011-08-12 14:52:00	-11062.06	NaN	United Kingdom

```
df[df["StockCode"] == "B"]
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
299982	A563185	B	Adjust bad debt	1	2011-08-12 14:50:00	11062.06	NaN	United Kingdom
299983	A563186	B	Adjust bad debt	1	2011-08-12 14:51:00	-11062.06	NaN	United Kingdom
299984	A563187	B	Adjust bad debt	1	2011-08-12 14:52:00	-11062.06	NaN	United Kingdom

Methodology

Analysis

1. InvoiceNo
2. StockCode
3. Description
4. Quantity
5. InvoiceDate
6. UnitPrice
7. CustomerID
8. Country

```
df[df.InvoiceNo.str.contains("C").isnull()].groupby(by="CustomerID").count()["InvoiceNo"].sort_values(ascending=False).head(3)
```

```
CustomerID
17841.0    7676
14911.0    5672
14096.0    5111
Name: InvoiceNo, dtype: int64
```

- **Customer 17841** is the most purchasing customer.

```
df.groupby(by="CustomerID").sum()["TotalCost"].sort_values(ascending=False).head(3)
```

```
CustomerID
14646.0    279489.02
18102.0    256438.49
17450.0    187322.17
Name: TotalCost, dtype: float64
```

- **Customer 14646** is the most revenue customer.

```
print("Number of avarage un-cancelled invoices per customer :")
print(df[df.InvoiceNo.str.contains("C").isnull()]["InvoiceNo"].unique().shape[0] / \
df[df.InvoiceNo.str.contains("C").isnull()]["CustomerID"].unique().shape[0])
```

```
Number of avarage un-cancelled invoices per customer :
5.083179723502304
```

Methodology

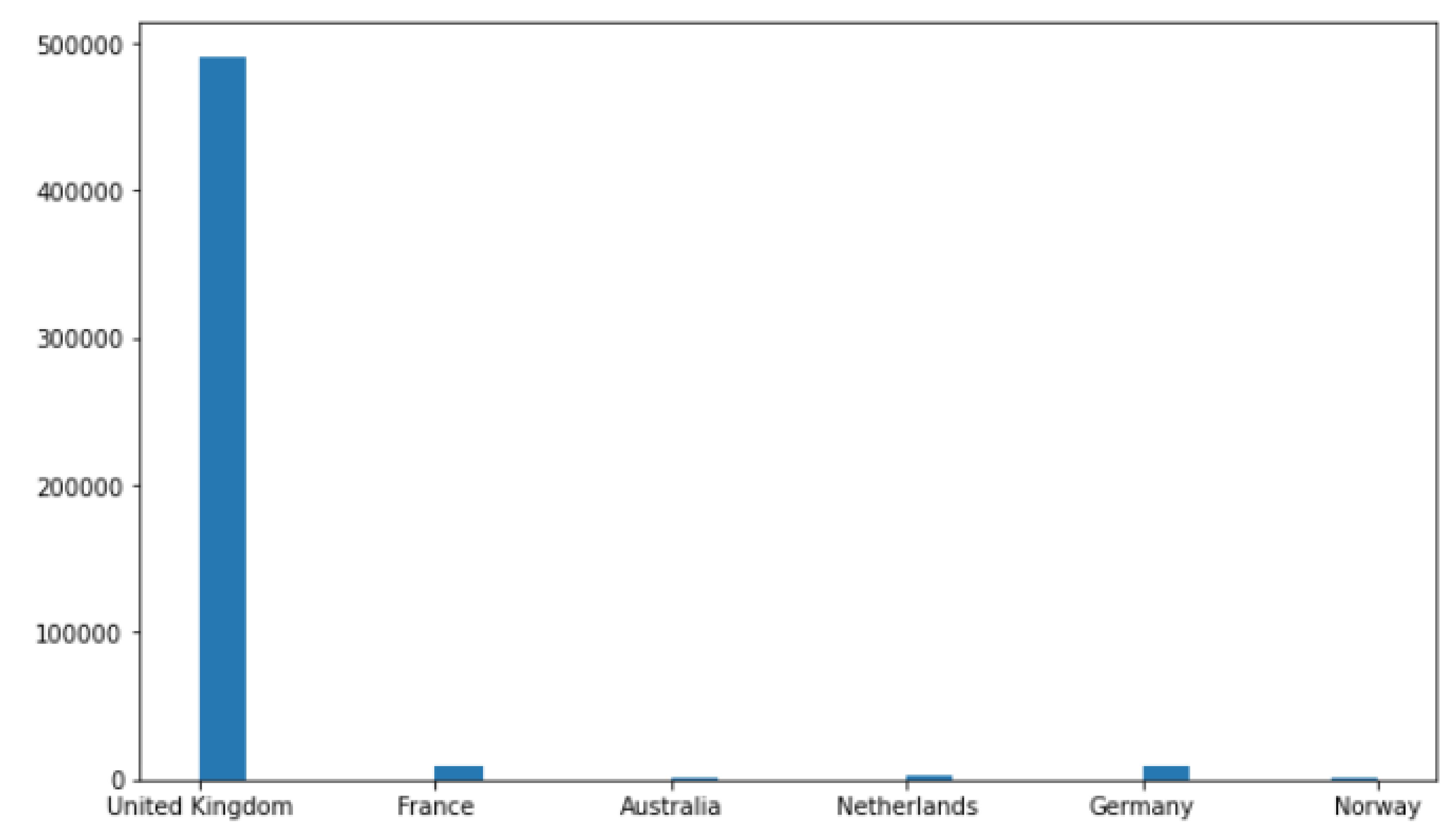
Analysis

1. InvoiceNo
2. StockCode
3. Description
4. Quantity
5. InvoiceDate
6. UnitPrice
7. CustomerID
8. **Country**

```
df["Country"].value_counts()
```

```
United Kingdom      490300
Germany               9480
France                8541
EIRE                  8184
Spain                 2528
Netherlands           2371
Belgium               2069
Switzerland           1994
Portugal              1510
Australia             1258
Norway                1086
Italy                  803
Channel Islands        757
Finland                695
Cyprus                 611
Sweden                 461
Unspecified            442
Austria                401
Denmark                389
Japan                  358
Poland                 341
Israel                 294
USA                    291
Hong Kong              284
Singapore              229
Iceland                182
Canada                 151
Greece                 146
Malta                  127
```

```
plt.hist(data=df, x="Country", bins=25, range=(0,5) );
```



Analysis

EDA

RFM Analysis

K-Means

Cohort Analysis

Methodology

Dropped

- Duplicates
- Cancelled Invoices
- Negative UnitPrices
- Negative Quantities
- Missing CustomerIDs

Rows

541.909 → 392.692

Methodology

Analysis

EDA

RFM Analysis

K-Means

Cohort Analysis

Total revenue per country

```
df.groupby("country")['total_price'].sum().sort_values(ascending=False)
```

```
country
United Kingdom     7285024.644
Netherlands         285446.340
EIRE                265262.460
Germany             228678.400
France              208934.310
Australia           138453.810
Spain                61558.560
Switzerland          56443.950
Belgium              41196.340
```

Total customer per country

```
df.groupby("country")['customerid'].nunique().sort_values(ascending=False)
```

```
country
United Kingdom    3920
Germany             94
France              87
Spain               30
Belgium             25
Switzerland         21
Portugal            19
Italy               14
Finland             12
```

Analysis

EDA

RFM Analysis

K-Means

Cohort Analysis

Methodology

Focused on the UK market

```
df_uk = df[df["country"]=="United Kingdom"]
df_uk.head(1)
```

```
df_uk.shape
```

```
(349203, 12)
```

2. What are the most popular products that are bought in the UK?

```
df_uk.groupby(["stockcode"])["quantity"].sum().sort_values(ascending=False).head(5)
```

```
stockcode
23843     80995
23166     76919
84077     49086
22197     45609
85099B    41878
```

Analysis

EDA

RFM Analysis

K-Means

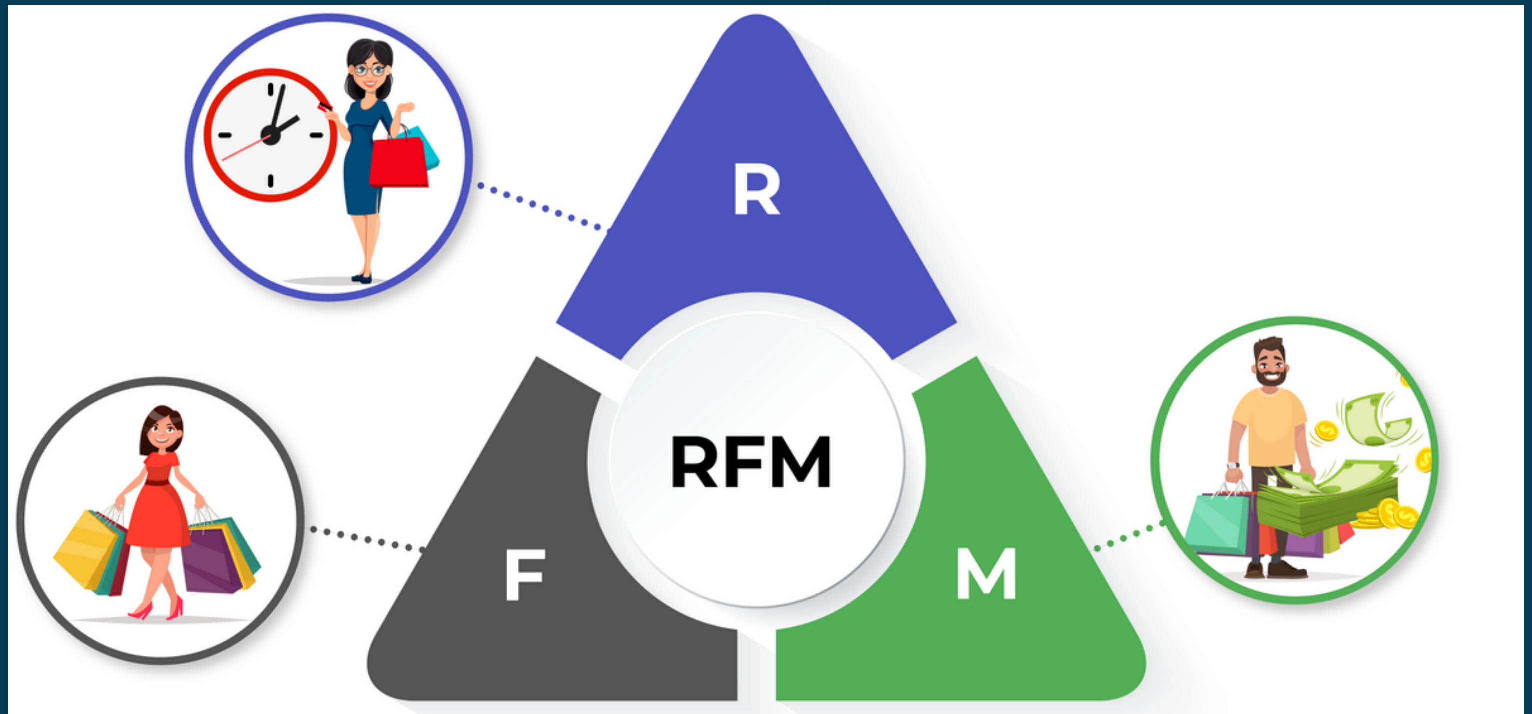
Cohort Analysis

Methodology

Recencey

Frequency

Monetary



RFM Metrics

RECENCY

The freshness of the customer activity, be it purchases or visits

E.g. Time since last order or last engaged with the product

FREQUENCY

The frequency of the customer transactions or visits

E.g. Total number of transactions or average time between transactions/ engaged visits

MONETARY

The intention of customer to spend or purchasing power of customer

E.g. Total or average transactions value

CUSTOMER ID	RECENCY (DAY)	FREQUENCY (NUMBER)	MONETARY (TOTAL)
1	4	6	540
2	6	11	940
3	46	1	35
4	23	3	65
5	15	4	179
6	32	2	56
7	7	3	140
8	50	1	950
9	34	15	2630
10	10	5	191
11	3	8	845
12	1	10	1510
13	27	3	54
14	18	2	40
15	5	1	25

Methodology

Recencey

ref_date = 2011-12-09

```
df_uk["customer_recency"] = df_uk["ref_date"] - df_uk["last_purchase_date"]
```

```
customer_recency = pd.DataFrame(df_uk.groupby('customerid')['recency_value'].min())
```

Frequency

```
customer_frequency = pd.DataFrame(df_uk.groupby('customerid')['invoiceno'].nunique())
```

```
df_uk['customer_frequency'] = df_uk.groupby('customerid')['invoiceno'].transform('count')
```

Monetary

```
customer_monetary = pd.DataFrame(df_uk.groupby('customerid')['total_price'].sum())
```

```
df_uk['customer_monetary'] = df_uk.groupby('customerid')['total_price'].transform('sum')
```

Analysis

EDA

RFM Analysis

K-Means

Cohort Analysis

Methodology

RFM Table

```
customer_rfm = pd.merge(pd.merge(customer_recency, customer_frequency, on='customerid'), customer_monetary, on='customerid')
customer_rfm.head()
```

	customerid	recency	frequency	monetary
0	12346.000	325	1	77183.600
1	12747.000	2	11	4196.010
2	12748.000	0	209	33053.190
3	12749.000	3	5	4090.880
4	12820.000	3	4	942.340

```
customer_rfm.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3920 entries, 0 to 3919
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   customerid  3920 non-null   float64
 1   recency     3920 non-null   int64  
 2   frequency   3920 non-null   int64  
 3   monetary    3920 non-null   float64
dtypes: float64(2), int64(2)
memory usage: 153.1 KB
```

Analysis

EDA

RFM Analysis

K-Means

Cohort Analysis

Methodology

RFM Table

```
quantiles = customer_rfm.quantile(q = [0.25,0.50,0.75])
quantiles
```

	recency	frequency	monetary
0.250	17.000	1.000	298.185
0.500	50.000	2.000	644.975
0.750	142.000	5.000	1571.285

customerid	recency	frequency	monetary	recency_score	frequency_score	monetary_score
12346.000	325	1	77183.600	1	1	4
12747.000	2	11	4196.010	4	4	4
12748.000	0	209	33053.190	4	4	4
12749.000	3	5	4090.880	4	3	4
12820.000	3	4	942.340	4	3	3

```
def RScore(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 FMScore(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
```

Analysis

EDA

RFM Analysis

K-Means

Cohort Analysis

Methodology

RFM Table

customerid	recency	frequency	monetary	recency_score	frequency_score	monetary_score	RFM_score	RFM_level
14498.0	42	5	1957.32	3	3	4	334	10
17692.0	127	4	740.94	2	3	3	233	8
16209.0	88	5	2262.62	2	3	4	234	9
15277.0	46	1	255.90	3	1	1	311	5
18058.0	9	1	170.16	4	1	1	411	6
16567.0	194	2	865.60	1	2	3	123	6
13851.0	95	3	2651.46	2	3	4	234	9
14004.0	43	7	4582.64	3	4	4	344	11
15937.0	64	1	145.35	2	1	1	211	4
13294.0	189	2	873.74	1	2	3	123	6

Analysis

EDA

RFM Analysis

K-Means

Cohort Analysis

Methodology

RFM Segmentation

```python
def segments(df_rfm):
    if df_rfm['RFM_level'] == 12 :
        return 'champion'
    elif (df_rfm['RFM_level'] == 11) or (df_rfm['RFM_level'] == 10 ):
        return 'loyal_customer'
    elif (df_rfm['RFM_level'] == 9) or (df_rfm['RFM_level'] == 8 ):
        return 'promising'
    elif (df_rfm['RFM_level'] == 7) or (df_rfm['RFM_level'] == 6 ):
        return 'need_attention'
    elif (df_rfm['RFM_level'] == 5) or (df_rfm['RFM_level'] == 4 ):
        return 'hibernating'
    else:
        return 'almost_lost'
```

	RFM_level	General_Segment
Bronz	6.477	776
Gold	10.462	649
Platinium	12.000	423
Silver	8.526	775
Steel	4.497	902
Trash	3.000	395

customerid	recency	frequency	monetary	recency_score	frequency_score	monetary_score	RFM_score	RFM_level	customer_segment
15125.0	25	15	11528.48	3	4	4	344	11	loyal_customer
12962.0	7	2	266.39	4	2	1	421	7	need_attention
16078.0	283	1	79.20	1	1	1	111	3	almost_lost
15776.0	133	1	241.62	2	1	1	211	4	hibernating
15537.0	163	1	110.92	1	1	1	111	3	almost_lost

Analysis

EDA

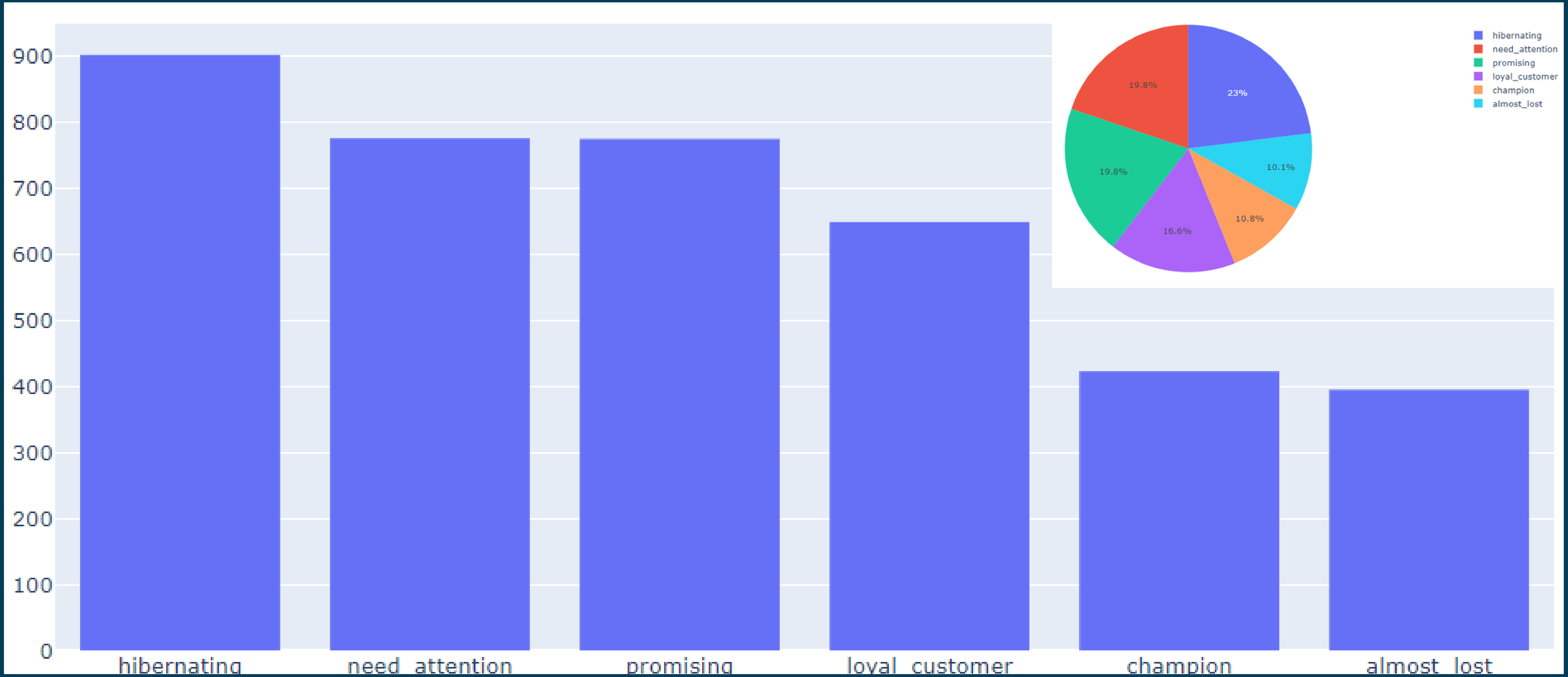
RFM Analysis

K-Means

Cohort Analysis

Methodology

RFM Segmentation



Methodology

Analysis

EDA

RFM Analysis

K-Means

Cohort Analysis

```
skew_vals = customer_rfm.skew().sort_values(ascending=False)
skew_vals
```

```
monetary            20.217581
frequency           10.751932
recency              1.244993
frequency_score      0.153764
rfm_level            0.115653
monetary_score       0.000000
rfm_score           -0.005586
recency_score       -0.009948
```

```
rfm_log = customer_rfm[skew_cols.index].copy()
for col in skew_cols.index.values:
    rfm_log[col] = rfm_log[col].apply(np.log1p)
print(rfm_log[skew_cols.index].skew())
```

```
monetary     0.373135
frequency    1.182249
recency     -0.463737
```

```
rfm_before_trans = customer_rfm[skew_cols.index].copy()
pt = PowerTransformer(method='yeo-johnson')
trans= pt.fit_transform(rfm_before_trans)
rfm_after_trans = pd.DataFrame(trans, columns =skew_cols.index)
print(rfm_after_trans.skew())
```

```
monetary    -0.013749
frequency    0.215681
recency     -0.063910
```

Methodology

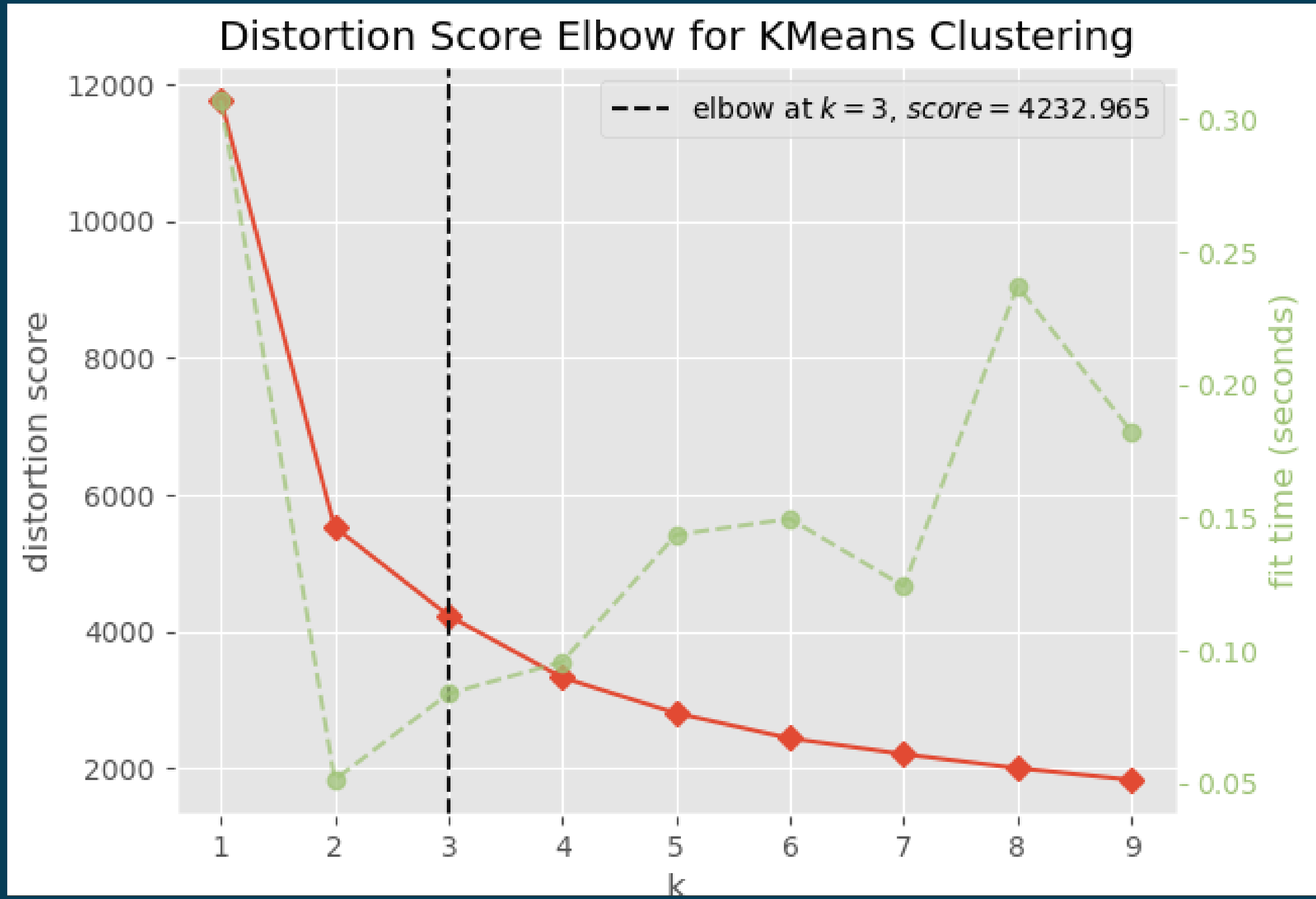
Analysis

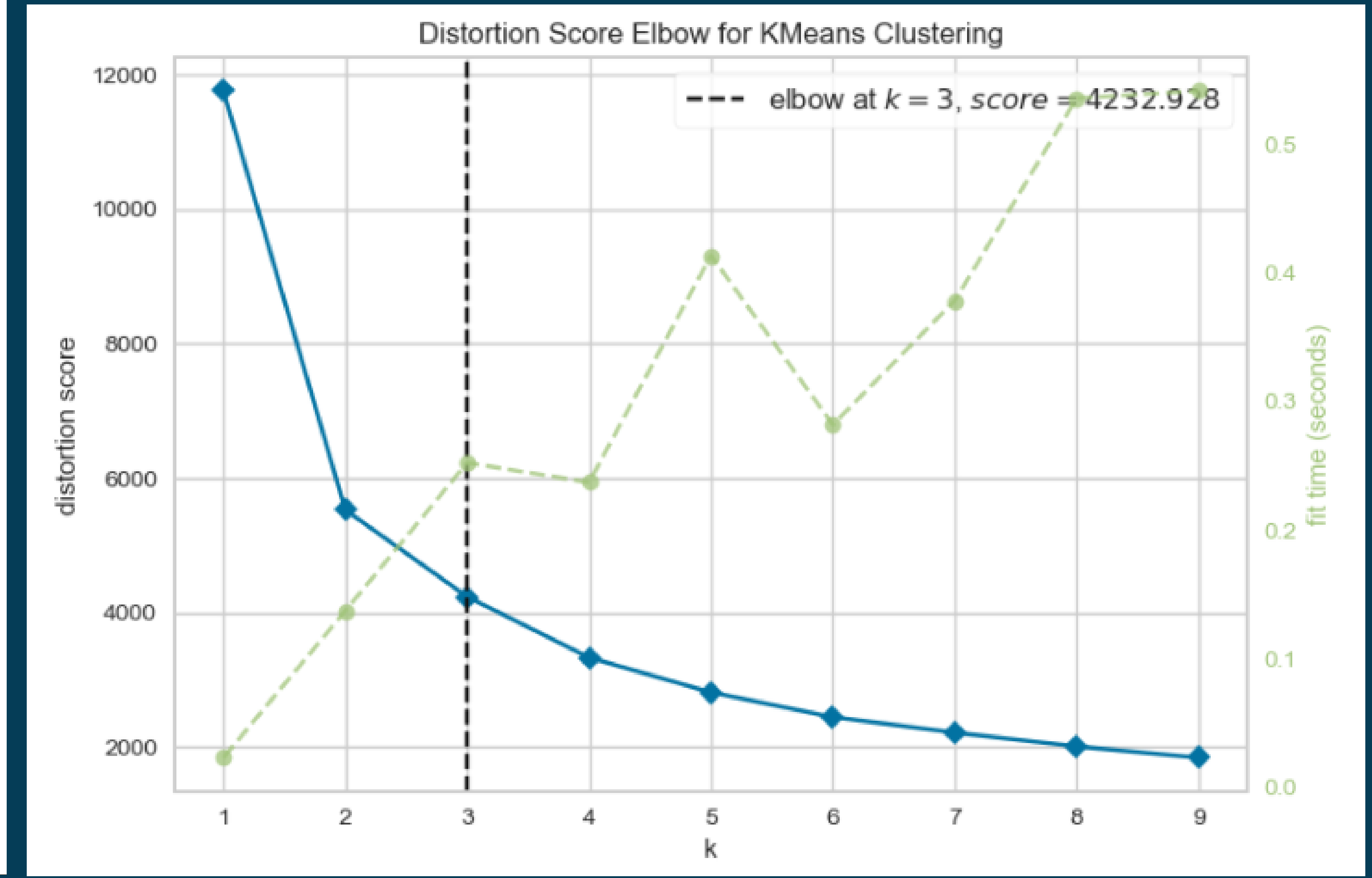
EDA

RFM Analysis

K-Means

Cohort Analysis






```
k_means_model = KMeans(n_clusters = 3, random_state = 42)
k_means_model.fit_predict(rfm_scaled)
labels = k_means_model.labels_
rfm_trans['predicted_clusters'] = labels
rfm_trans
```

Analysis

EDA

RFM Analysis

K-Means

Cohort Analysis

Methodology

App Launched ↓ % Active users after App Launches →

Cohort	Users	Day 0	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10
Jan 25	1,098	100%	33.9%	23.5%	18.7%	15.9%	16.3%	14.2%	14.5%	Retention over user lifetime		12.1%
Jan 26	1,358	100%	31.1%	18.6%	14.3%	16.0%	14.9%	13.2%	12.9%			
Jan 27	1,257	100%	27.2%	19.6%	14.5%	12.9%	13.4%	13.0%	10.8%	11.4%		
Jan 28	1,587	100%	26.6%	17.9%	14.6%	14.8%	14.9%	13.7%	11.9%			
Jan 29	1,758	100%	26.2%	20.4%	16.9%	14.3%	12.7%	12.5%				
Jan 30	1,624	100%	26.4%	18.1%	13.7%	15.4%	11.8%					
Jan 31	1,541	100%	23.9%	19.6%	15.0%	14.8%						
Feb 01	868	100%	24.7%	16.9%	15.8%							
Feb 02	1,143	Retention over product lifetime		18.5%								
Feb 03	1,253											
All Users	**13,487**	**100%**	**27.0%**	**19.2%**	**15.4%**	**14.9%**	**14.0%**	**13.3%**	**12.5%**	**13.1%**	**12.2%**	**12.1%**

- **Acquisition Cohorts:** divide users by when they signed up first for your product. For your app users, you might break down your cohorts by the day, the week or the month they launched an app, and thereby track daily, weekly or monthly cohorts.

- **Behavioral Cohorts:** divide users by the behaviors they have (or haven't) taken in your app within a given time period. These could be any number of discrete actions that a user can perform – App Install, App Launch, App Uninstall, Transaction or Charged, or any combination of these actions / events.

Analysis

EDA

RFM Analysis

K-Means

Cohort Analysis

Methodology

ref_date	date	last_purchase_date	customer_recency	recency_value	customer_frequency	customer_monetary	invoice_month	cohort_month	cohort_index
2011-12-09	2010-12-01	2010-12-02	372 days	372	297	5391.21	2010-12-01	2010-12-01	1
2011-12-09	2010-12-01	2010-12-02	372 days	372	297	5391.21	2010-12-01	2010-12-01	1
2011-12-09	2010-12-01	2010-12-02	372 days	372	297	5391.21	2010-12-01	2010-12-01	1
2011-12-09	2010-12-01	2010-12-02	372 days	372	297	5391.21	2010-12-01	2010-12-01	1

cohort_index	1	2	3	4	5	6	7	8	9	10	11	12	13
cohort_month													
2010-12-01	815.0	289.0	263.0	304.0	293.0	323.0	291.0	278.0	289.0	325.0	299.0	405.0	218.0
2011-01-01	358.0	76.0	93.0	84.0	119.0	99.0	90.0	87.0	108.0	117.0	127.0	43.0	NaN
2011-02-01	340.0	64.0	66.0	97.0	98.0	86.0	87.0	96.0	90.0	104.0	25.0	NaN	NaN
2011-03-01	419.0	64.0	109.0	83.0	94.0	69.0	111.0	96.0	119.0	38.0	NaN	NaN	NaN
2011-04-01	277.0	58.0	56.0	60.0	56.0	61.0	61.0	73.0	20.0	NaN	NaN	NaN	NaN
2011-05-01	256.0	48.0	44.0	44.0	53.0	58.0	68.0	23.0	NaN	NaN	NaN	NaN	NaN
2011-06-01	214.0	38.0	31.0	51.0	51.0	69.0	21.0	NaN	NaN	NaN	NaN	NaN	NaN
2011-07-01	169.0	30.0	33.0	39.0	47.0	18.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-08-01	141.0	32.0	32.0	34.0	17.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-09-01	276.0	63.0	83.0	32.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

cohort_index	1	2	3	4	5	6	7	8	9	10	11	12	13
cohort_month													
2010-12-01	100.0	35.5	32.3	37.3	36.0	39.6	35.7	34.1	35.5	39.9	36.7	49.7	26.7
2011-01-01	100.0	21.2	26.0	23.5	33.2	27.7	25.1	24.3	30.2	32.7	35.5	12.0	NaN
2011-02-01	100.0	18.8	19.4	28.5	28.8	25.3	25.6	28.2	26.5	30.6	7.4	NaN	NaN
2011-03-01	100.0	15.3	26.0	19.8	22.4	16.5	26.5	22.9	28.4	9.1	NaN	NaN	NaN
2011-04-01	100.0	20.9	20.2	21.7	20.2	22.0	22.0	26.4	7.2	NaN	NaN	NaN	NaN
2011-05-01	100.0	18.8	17.2	17.2	20.7	22.7	26.6	9.0	NaN	NaN	NaN	NaN	NaN
2011-06-01	100.0	17.8	14.5	23.8	23.8	32.2	9.8	NaN	NaN	NaN	NaN	NaN	NaN
2011-07-01	100.0	17.8	19.5	23.1	27.8	10.7	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-08-01	100.0	22.7	22.7	24.1	12.1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-09-01	100.0	22.8	30.1	11.6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN