RFM ile Müşteri Segmentasyonu (Customer Segmentation with RFM)

İş Problemi (Business Problem)

FLO müşterilerini segmentlere ayırıp bu segmentlere göre pazarlama stratejileri belirlemek istiyor.

Buna yönelik olarak müşterilerin davranışları tanımlanacak ve bu davranış öbeklenmelerine göre gruplar oluşturulacak.

Veri seti son alışverişlerini 2020 - 2021 yıllarında OmniChannel(hem online hem offline alışveriş yapan) olarak yapan müşterilerin geçmiş alışveriş davranışlarından elde edilen bilgilerden oluşmaktadır.

master_id: Eşsiz müşteri numarası
order_channel : Alışveriş yapılan platforma ait hangi kanalın kullanıldığı (Android, ios, Desktop, Mobile, Offline)
last_order_channel : En son alışverişin yapıldığı kanal
first_order_date : Müşterinin yaptığı ilk alışveriş tarihi
last_order_date : Müşterinin yaptığı son alışveriş tarihi
last_order_date_online : Muşterinin online platformda yaptığı son alışveriş tarihi
last_order_date_offline : Muşterinin offline platformda yaptığı son alışveriş tarihi
order_num_total_ever_online : Müşterinin online platformda yaptığı toplam alışveriş sayısı
order_num_total_ever_offline : Müşterinin offline'da yaptığı toplam alışveriş sayısı
customer_value_total_ever_offline : Müşterinin offline alışverişlerinde ödediği toplam ücret
customer_value_total_ever_online : Müşterinin online alışverişlerinde ödediği toplam ücret
interested_in_categories_12 : Müşterinin son 12 ayda alışveriş yaptığı kategorilerin listesi

In [3]:
```
import pandas as pd
pd.set_option("display.max_columns", None)
pd.set_option("display.float_format", lambda x: "%.3f" % x)

df_ = pd.read_csv("/Users/umutaykanat/Desktop/ÇALIŞMALAR/Miuul/CRM/FLOMuste
df = df_.copy()
```

In [4]:
```
# 2. Veri setinde
        # a. İlk 10 gözlem,
        # b. Değişken isimleri,
        # c. Boyut,
        # d. Betimsel istatistik,
```

```
        # e. Boş değer,
        # f. Değişken tipleri, incelemesi yapınız.
df.head(10)
df.columns
df.describe().T # numerik değişkenlere dair istatistikler
df.isnull().sum() # boş değer bulunmamaktadır.
df.dtypes
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19945 entries, 0 to 19944
Data columns (total 12 columns):
 #   Column                             Non-Null Count  Dtype  
---  ------                             --------------  -----  
 0   master_id                          19945 non-null  object 
 1   order_channel                      19945 non-null  object 
 2   last_order_channel                 19945 non-null  object 
 3   first_order_date                   19945 non-null  object 
 4   last_order_date                    19945 non-null  object 
 5   last_order_date_online             19945 non-null  object 
 6   last_order_date_offline            19945 non-null  object 
 7   order_num_total_ever_online        19945 non-null  float64
 8   order_num_total_ever_offline       19945 non-null  float64
 9   customer_value_total_ever_offline  19945 non-null  float64
 10  customer_value_total_ever_online   19945 non-null  float64
 11  interested_in_categories_12        19945 non-null  object 
dtypes: float64(4), object(8)
memory usage: 1.8+ MB
```

In [5]:
```
# 3. Omnichannel müşterilerin hem online'dan hemde offline platformlardan a
# Herbir müşterinin toplam alışveriş sayısı ve harcaması için yeni değişken
df["order_num_total"] = df["order_num_total_ever_online"] + df["order_num_to
df["customer_value_total"] = df["customer_value_total_ever_offline"] + df["
```

In [6]:
```
# 4. Değişken tiplerini inceleyiniz. Tarih ifade eden değişkenlerin tipini d
date_columns = df.columns[df.columns.str.contains("date")]
df[date_columns] = df[date_columns].apply(pd.to_datetime)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19945 entries, 0 to 19944
Data columns (total 14 columns):
 #   Column                             Non-Null Count  Dtype         
---  ------                             --------------  -----         
 0   master_id                          19945 non-null  object        
 1   order_channel                      19945 non-null  object        
 2   last_order_channel                 19945 non-null  object        
 3   first_order_date                   19945 non-null  datetime64[ns]
 4   last_order_date                    19945 non-null  datetime64[ns]
 5   last_order_date_online             19945 non-null  datetime64[ns]
 6   last_order_date_offline            19945 non-null  datetime64[ns]
 7   order_num_total_ever_online        19945 non-null  float64       
 8   order_num_total_ever_offline       19945 non-null  float64       
 9   customer_value_total_ever_offline  19945 non-null  float64       
 10  customer_value_total_ever_online   19945 non-null  float64       
 11  interested_in_categories_12        19945 non-null  object        
 12  order_num_total                    19945 non-null  float64       
 13  customer_value_total               19945 non-null  float64       
dtypes: datetime64[ns](4), float64(6), object(4)
memory usage: 2.1+ MB
```

In [7]:
```
# 5. Alışveriş kanallarındaki müşteri sayısının, toplam alınan ürün sayısı v
df.groupby("order_channel").agg({"master_id": "count",
```

```
                                    "order_num_total": "sum",
                                    "customer_value_total": "sum"})
```

Out[7]:

order_channel	master_id	order_num_total	customer_value_total
Android App	9495	52269.000	7819062.760
Desktop	2735	10920.000	1610321.460
Ios App	2833	15351.000	2525999.930
Mobile	4882	21679.000	3028183.160

In [8]:
```
# 6. En fazla kazancı getiren ilk 10 müşteriyi sıralayınız.
df.sort_values("customer_value_total",ascending=False)[:10]["master_id"]
```

Out[8]:
```
11150    5d1c466a-9cfd-11e9-9897-000d3a38a36f
4315     d5ef8058-a5c6-11e9-a2fc-000d3a38a36f
7613     73fd19aa-9e37-11e9-9897-000d3a38a36f
13880    7137a5c0-7aad-11ea-8f20-000d3a38a36f
9055     47a642fe-975b-11eb-8c2a-000d3a38a36f
7330     a4d534a2-5b1b-11eb-8dbd-000d3a38a36f
8068     d696c654-2633-11ea-8e1c-000d3a38a36f
163      fef57ffa-aae6-11e9-a2fc-000d3a38a36f
7223     cba59206-9dd1-11e9-9897-000d3a38a36f
18767    fc0ce7a4-9d87-11e9-9897-000d3a38a36f
Name: master_id, dtype: object
```

In [9]:
```
# 7. En fazla siparişi veren ilk 10 müşteriyi sıralayınız.
df.sort_values("order_num_total", ascending=False)[:10]
```

Out[9]:

	master_id	order_channel	last_order_channel	first_order_date	last_order_date
11150	5d1c466a-9cfd-11e9-9897-000d3a38a36f	Android App	Desktop	2013-10-11	2021-04-30
7223	cba59206-9dd1-11e9-9897-000d3a38a36f	Android App	Android App	2013-02-21	2021-05-09
8783	a57f4302-b1a8-11e9-89fa-000d3a38a36f	Android App	Offline	2019-08-07	2020-11-04
2619	fdbe8304-a7ab-11e9-a2fc-000d3a38a36f	Android App	Offline	2018-10-18	2020-06-30
6322	329968c6-a0e2-11e9-a2fc-000d3a38a36f	Ios App	Ios App	2019-02-14	2021-04-05
7613	73fd19aa-9e37-11e9-9897-000d3a38a36f	Ios App	Offline	2014-01-14	2021-05-18
9347	44d032ee-a0d4-11e9-a2fc-000d3a38a36f	Mobile	Mobile	2019-02-11	2021-02-11
10954	b27e241a-a901-11e9-a2fc-000d3a38a36f	Mobile	Mobile	2015-09-12	2021-04-01
8068	d696c654-2633-11ea-8e1c-000d3a38a36f	Ios App	Ios App	2017-05-10	2021-04-13
7330	a4d534a2-5b1b-11eb-8dbd-000d3a38a36f	Desktop	Desktop	2020-02-16	2021-04-30

In [10]:
```
# 8. Veri ön hazırlık sürecini fonksiyonlaştırınız.
def data_prep(dataframe):
    dataframe["order_num_total"] = dataframe["order_num_total_ever_online"]
    dataframe["customer_value_total"] = dataframe["customer_value_total_ever
    date_columns = dataframe.columns[dataframe.columns.str.contains("date")]
    dataframe[date_columns] = dataframe[date_columns].apply(pd.to_datetime)
    return df
```

In [11]:
```
df = df_.copy()
data_prep(df)
```

Out[11]:

	master_id	order_channel	last_order_channel	first_order_date	last_order_date
0	cc294636-19f0-11eb-8d74-000d3a38a36f	Android App	Offline	2020-10-30	2021-02-26
1	f431bd5a-ab7b-11e9-a2fc-000d3a38a36f	Android App	Mobile	2017-02-08	2021-02-16
2	69b69676-1a40-11ea-941b-000d3a38a36f	Android App	Android App	2019-11-27	2020-11-27
3	1854e56c-491f-11eb-806e-000d3a38a36f	Android App	Android App	2021-01-06	2021-01-17
4	d6ea1074-f1f5-11e9-9346-000d3a38a36f	Desktop	Desktop	2019-08-03	2021-03-07
...	...	...	...	...	...
19940	727e2b6e-ddd4-11e9-a848-000d3a38a36f	Android App	Offline	2019-09-21	2020-07-05
19941	25cd53d4-61bf-11ea-8dd8-000d3a38a36f	Desktop	Desktop	2020-03-01	2020-12-22
19942	8aea4c2a-d6fc-11e9-93bc-000d3a38a36f	Ios App	Ios App	2019-09-11	2021-05-24
19943	e50bb46c-ff30-11e9-a5e8-000d3a38a36f	Android App	Android App	2019-03-27	2021-02-13
19944	740998d2-b1f7-11e9-89fa-000d3a38a36f	Android App	Android App	2019-09-03	2020-06-06

19945 rows × 14 columns

GÖREV 2: RFM Metriklerinin Hesaplanması

In [12]:
```
import datetime as dt
```

In [13]:
```
# Veri setindeki en son alışverişin yapıldığı tarihten 2 gün sonrasını anali
df["last_order_date"].max() # 2021-05-30
analysis_date = dt.datetime(2021,6,1)
```

In [14]:
```
# customer_id, recency, frequnecy ve monetary değerlerinin yer aldığı yeni b
rfm = pd.DataFrame()
```

```
rfm["customer_id"] = df["master_id"]
rfm["recency"] = (analysis_date - df["last_order_date"]).astype('timedelta64
rfm["frequency"] = df["order_num_total"]
rfm["monetary"] = df["customer_value_total"]

rfm.head()
```

Out[14]:

	customer_id	recency	frequency	monetary
0	cc294636-19f0-11eb-8d74-000d3a38a36f	95.000	5.000	939.370
1	f431bd5a-ab7b-11e9-a2fc-000d3a38a36f	105.000	21.000	2013.550
2	69b69676-1a40-11ea-941b-000d3a38a36f	186.000	5.000	585.320
3	1854e56c-491f-11eb-806e-000d3a38a36f	135.000	2.000	121.970
4	d6ea1074-f1f5-11e9-9346-000d3a38a36f	86.000	2.000	209.980

GÖREV 3: RF ve RFM Skorlarının Hesaplanması (Calculating RF and RFM Scores)

In [15]:

```
rfm["recency_score"] = pd.qcut(rfm["recency"], 5, labels=[5,4,3,2,1])
rfm["frequency_score"] = pd.qcut(rfm["frequency"].rank(method="first"), 5,
rfm["monetary_score"] = pd.qcut(rfm["monetary"], 5, labels=[1,2,3,4,5])

rfm.head()
```

Out[15]:

	customer_id	recency	frequency	monetary	recency_score	frequency_score	monetary
0	cc294636-19f0-11eb-8d74-000d3a38a36f	95.000	5.000	939.370	3	4	
1	f431bd5a-ab7b-11e9-a2fc-000d3a38a36f	105.000	21.000	2013.550	3	5	
2	69b69676-1a40-11ea-941b-000d3a38a36f	186.000	5.000	585.320	2	4	
3	1854e56c-491f-11eb-806e-000d3a38a36f	135.000	2.000	121.970	3	1	
4	d6ea1074-f1f5-11e9-9346-000d3a38a36f	86.000	2.000	209.980	3	1	

In [16]:

```
# recency_score ve frequency_score'u tek bir değişken olarak ifade edilmesi
rfm["RF_SCORE"] = (rfm['recency_score'].astype(str) + rfm['frequency_score']

# 3. recency_score ve frequency_score ve monetary_score'u tek bir değişken
rfm["RFM_SCORE"] = (rfm['recency_score'].astype(str) + rfm['frequency_score

rfm.head()
```

Out[16]:

	customer_id	recency	frequency	monetary	recency_score	frequency_score	monetary
0	cc294636-19f0-11eb-8d74-000d3a38a36f	95.000	5.000	939.370	3	4	
1	f431bd5a-ab7b-11e9-a2fc-000d3a38a36f	105.000	21.000	2013.550	3	5	
2	69b69676-1a40-11ea-941b-000d3a38a36f	186.000	5.000	585.320	2	4	
3	1854e56c-491f-11eb-806e-000d3a38a36f	135.000	2.000	121.970	3	1	
4	d6ea1074-f1f5-11e9-9346-000d3a38a36f	86.000	2.000	209.980	3	1	

GÖREV 4: RF Skorlarının Segment Olarak Tanımlanması

In [17]:
```
# Oluşturulan RFM skorların daha açıklanabilir olması için segment tanımlama
seg_map = {
    r'[1-2][1-2]': 'hibernating',
    r'[1-2][3-4]': 'at_Risk',
    r'[1-2]5': 'cant_loose',
    r'3[1-2]': 'about_to_sleep',
    r'33': 'need_attention',
    r'[3-4][4-5]': 'loyal_customers',
    r'41': 'promising',
    r'51': 'new_customers',
    r'[4-5][2-3]': 'potential_loyalists',
    r'5[4-5]': 'champions'
}

rfm['segment'] = rfm['RF_SCORE'].replace(seg_map, regex=True) # regular expr

rfm.head()
```

Out[17]:

	customer_id	recency	frequency	monetary	recency_score	frequency_score	monetary
0	cc294636-19f0-11eb-8d74-000d3a38a36f	95.000	5.000	939.370	3	4	
1	f431bd5a-ab7b-11e9-a2fc-000d3a38a36f	105.000	21.000	2013.550	3	5	
2	69b69676-1a40-11ea-941b-000d3a38a36f	186.000	5.000	585.320	2	4	
3	1854e56c-491f-11eb-806e-000d3a38a36f	135.000	2.000	121.970	3	1	
4	d6ea1074-f1f5-11e9-9346-000d3a38a36f	86.000	2.000	209.980	3	1	

1. Segmentlerin recency, frequnecy ve monetary ortalamalarını inceleyiniz.

In [44]:
```
rfm[["segment", "recency", "frequency", "monetary"]].groupby("segment").agg(
```

Out[44]:

	recency		frequency		monetary	
	mean	count	mean	count	mean	count
segment						
about_to_sleep	113.785	1629	2.401	1629	359.009	1629
at_Risk	241.607	3131	4.472	3131	646.610	3131
cant_loose	235.444	1200	10.698	1200	1474.468	1200
champions	17.107	1932	8.934	1932	1406.625	1932
hibernating	247.950	3604	2.394	3604	366.267	3604
loyal_customers	82.595	3361	8.375	3361	1216.819	3361
need_attention	113.829	823	3.728	823	562.143	823
new_customers	17.918	680	2.000	680	339.956	680
potential_loyalists	37.156	2938	3.304	2938	533.184	2938
promising	58.921	647	2.000	647	335.673	647

2. RFM analizi yardımı ile 2 case için ilgili profildeki müşterileri bulunuz ve müşteri id'lerini csv ye kaydediniz.

a. FLO bünyesine yeni bir kadın ayakkabı markası dahil ediyor. Dahil ettiği markanın ürün fiyatları genel müşteri tercihlerinin üstünde. Bu nedenle markanın tanıtımı ve ürün satışları için ilgilenecek profildeki müşterilerle özel olarak iletişime geçilmek isteniliyor. Bu müşterilerin sadık ve kadın kategorisinden alışveriş yapan kişiler olması planlandı. Müşterilerin id

numaralarını csv dosyasına yeni_marka_hedef_müşteri_id.cvs olarak kaydediniz.

b. Erkek ve Çoçuk ürünlerinde %40'a yakın indirim planlanmaktadır. Bu indirimle ilgili kategorilerle ilgilenen geçmişte iyi müşterilerden olan ama uzun süredir alışveriş yapmayan ve yeni gelen müşteriler özel olarak hedef alınmak isteniliyor. Uygun profildeki müşterilerin id'lerini csv dosyasına indirim_hedef_müşteri_ids.csv olarak kaydediniz.

Sadık kişiler filtrelemesini oluşturduğum *rfm* dataframeinden elde edeceğim. Daha sonra bu id'lere sahip kadın müşterileri *df* adlı dataframe'den filtreleyeceğim.

In [25]:
```
rfm["segment"].unique()
```

Out[25]:
```
array(['loyal_customers', 'at_Risk', 'about_to_sleep',
       'potential_loyalists', 'hibernating', 'champions', 'cant_loose',
       'need_attention', 'promising', 'new_customers'], dtype=object)
```

In [26]:
```
target_segments_customer_ids = rfm[rfm["segment"].isin(["champions","loyal_
```

In [31]:
```
cust_ids = df[(df["master_id"].isin(target_segments_customer_ids)) & df["int
```

In [32]:
```
cust_ids.to_csv("yeni_marka_hedef_müşteri_id.cvs", index=False)
```

In [33]:
```
cust_ids.shape # hedeflenen müşteri kitlesinde 2497 adet müşteri bulunmaktad
```

Out[33]:
```
(2497,)
```

In [ ]:
```

```

In [35]:
```
target_segments_customes_ids2 = rfm[rfm["segment"].isin(["at_Risk","hibernat
```

In [39]:
```
cust_ids2 = df[df["master_id"].isin(target_segments_customes_ids2) & df["int
```

In [42]:
```
df[df["master_id"].isin(target_segments_customes_ids2) & df["interested_in_c
```

Out[42]:

	master_id	order_channel	last_order_channel	first_order_date	last_order_date
1	f431bd5a-ab7b-11e9-a2fc-000d3a38a36f	Android App	Mobile	2017-02-08	2021-02-16
2	69b69676-1a40-11ea-941b-000d3a38a36f	Android App	Android App	2019-11-27	2020-11-27
3	1854e56c-491f-11eb-806e-000d3a38a36f	Android App	Android App	2021-01-06	2021-01-17
7	3f1b4dc8-8a7d-11ea-8ec0-000d3a38a36f	Mobile	Offline	2020-05-15	2020-08-12
8	cfbda69e-5b4f-11ea-aca7-000d3a38a36f	Android App	Android App	2020-01-23	2021-03-07
...	...	...	...	...	...
19934	9777eb76-bed4-11ea-958c-000d3a38a36f	Desktop	Offline	2020-07-05	2020-12-02
19936	1982ac0e-9f4c-11e9-9897-000d3a38a36f	Android App	Android App	2019-02-08	2020-06-28
19937	515ca2d8-afdc-11e9-9757-000d3a38a36f	Ios App	Ios App	2019-03-10	2020-08-27
19938	2427ef66-a410-11e9-a2fc-000d3a38a36f	Android App	Android App	2019-04-12	2020-12-14
19940	727e2b6e-ddd4-11e9-a848-000d3a38a36f	Android App	Offline	2019-09-21	2020-07-05

8432 rows × 14 columns

In [41]:
```
cust_ids2.to_csv("indirim_hedef_müşteri_ids.csv", index=False)
```

In [ ]:
```
```

In [36]:
```
df.head()
```

Out[36]:

	master_id	order_channel	last_order_channel	first_order_date	last_order_date	last_
0	cc294636-19f0-11eb-8d74-000d3a38a36f	Android App	Offline	2020-10-30	2021-02-26	
1	f431bd5a-ab7b-11e9-a2fc-000d3a38a36f	Android App	Mobile	2017-02-08	2021-02-16	
2	69b69676-1a40-11ea-941b-000d3a38a36f	Android App	Android App	2019-11-27	2020-11-27	
3	1854e56c-491f-11eb-806e-000d3a38a36f	Android App	Android App	2021-01-06	2021-01-17	
4	d6ea1074-f1f5-11e9-9346-000d3a38a36f	Desktop	Desktop	2019-08-03	2021-03-07	

In [45]:
```
rfm.head()
```

Out[45]:

	customer_id	recency	frequency	monetary	recency_score	frequency_score	monetary
0	cc294636-19f0-11eb-8d74-000d3a38a36f	95.000	5.000	939.370	3	4	
1	f431bd5a-ab7b-11e9-a2fc-000d3a38a36f	105.000	21.000	2013.550	3	5	
2	69b69676-1a40-11ea-941b-000d3a38a36f	186.000	5.000	585.320	2	4	
3	1854e56c-491f-11eb-806e-000d3a38a36f	135.000	2.000	121.970	3	1	
4	d6ea1074-f1f5-11e9-9346-000d3a38a36f	86.000	2.000	209.980	3	1	

In [ ]:

📌 Proje Çıktıları, Bulgular ve Kazanımlar

🔍 Proje Özeti

Bu projede müşteri davranışlarını daha iyi anlamak ve aksiyon alınabilir segmentler oluşturmak amacıyla RFM (Recency, Frequency, Monetary) analizi uygulanmıştır. Çalışma boyunca veri temizleme, özellik mühendisliği, segmentasyon ve iş odaklı yorumlama adımları uçtan uca ele alınmıştır.

📊 Temel Bulgular

Müşteriler, satın alma zamanı, sıklığı ve harcama tutarına göre anlamlı segmentlere ayrılmıştır.

Champions ve Loyal Customers gibi yüksek değerli müşteri grupları net şekilde ayrıştırılmıştır.

At Risk ve Hibernating segmentleri, elde tutma (retention) ve geri kazanım (re-activation) çalışmaları için kritik olarak belirlenmiştir.

Segment bazlı analizler, tekil müşteri skorlarından daha güçlü içgörüler sunmuştur.

🎯 İş Değeri ve Aksiyon Önerileri

Yüksek değerli segmentler için kişiselleştirilmiş kampanyalar ve sadakat programları önerilmektedir.

Düşüşte olan segmentler için hedefli indirimler ve hatırlatma kampanyaları planlanabilir.

Pazarlama bütçesinin, segment bazlı optimize edilmesi mümkün hale gelmiştir.

TEŞEKKÜRLER

UMUT AYKANAT

In [ ]:

In [ ]: