БИБЛИОТЕКИ И ОБЗОР ДАННЫХ

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from datetime import datetime, date
from google.colab import files
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
#подключение библиотек
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
```

```
df = pd.read_csv('/content/drive/MyDrive/hse/3rd_year/minor/bank_transactions
```

```
df.head() #смотрим "шапку" таблицы
```

	TransactionID	CustomerID	CustomerDOB	CustGender	CustLocation	CustAcc
0	T1	C5841053	10/1/94	F	JAMSHEDPUR	
1	T2	C2142763	4/4/57	M	JHAJJAR	
2	T3	C4417068	26/11/96	F	MUMBAI	
3	T4	C5342380	14/9/73	F	MUMBAI	

```
df #смотрим таблицу
```

	TransactionID	CustomerID	CustomerDOB	CustGender	CustLocation
0	T1	C5841053	10/1/94	F	JAMSHEDPUR
1	T2	C2142763	4/4/57	M	JHAJJAR
2	T3	C4417068	26/11/96	F	MUMBAI

```
df.info() #
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048567 entries, 0 to 1048566
Data columns (total 9 columns):
 #   Column                  Non-Null Count    Dtype  
---  ------                  --------------    -----  
 0   TransactionID           1048567 non-null  object 
 1   CustomerID              1048567 non-null  object 
 2   CustomerDOB             1045170 non-null  object 
 3   CustGender              1047467 non-null  object 
 4   CustLocation            1048416 non-null  object 
 5   CustAccountBalance      1046198 non-null  float64
 6   TransactionDate         1048567 non-null  object 
 7   TransactionTime         1048567 non-null  int64  
 8   TransactionAmount (INR) 1048567 non-null  float64
dtypes: float64(2), int64(1), object(6)
memory usage: 72.0+ MB
```

ЧИСТКА: ПРЕОБРАЗОВАНИЯ CustomerDOB в AGE

```
df.describe().T #статистика для преобразований
```

	count	mean	std	min	25%	
CustAccountBalance	1046198.0	115403.540056	846485.380601	0.0	4721.76	1679
TransactionTime	1048567.0	157087.529393	51261.854022	0.0	124030.00	16422
TransactionAmount	1048567.0	1574.335003	6574.742978	0.0	161.00	45

```
df.isna().sum() #Discover how many nulls in each column
```

```
TransactionID               0
CustomerID                  0
CustomerDOB              3397
CustGender               1100
CustLocation              151
CustAccountBalance       2369
TransactionDate             0
TransactionTime             0
TransactionAmount (INR)     0
dtype: int64
```

```
df = df.dropna() #Drop all rows that have nulls
```

```
df.isna().sum()
```

```
TransactionID             0
CustomerID                0
CustomerDOB               0
CustGender                0
CustLocation              0
CustAccountBalance        0
TransactionDate           0
TransactionTime           0
TransactionAmount (INR)   0
dtype: int64
```

Все нули удалены

```
df['CustomerDOB'].value_counts() #сколько раз повторяются даты
```

```
1/1/1800    56292
1/1/89        809
1/1/90        784
6/8/91        698
1/1/91        665
             ...
2/12/51         1
20/3/52         1
26/9/47         1
4/10/41         1
24/10/44        1
Name: CustomerDOB, Length: 17233, dtype: int64
```

```
df = df.drop(df[df['CustomerDOB'] == '1/1/1800'].index,axis = 0) #1 января 18
```

```
df['CustomerDOB'].value_counts() #проверка повторения
```

```
1/1/89      809
1/1/90      784
6/8/91      698
1/1/91      665
1/1/92      631
           ...
23/2/05       1
28/11/42      1
23/9/49       1
14/3/40       1
24/10/44      1
Name: CustomerDOB, Length: 17232, dtype: int64
```

```
df.CustomerDOB = pd.to_datetime(df.CustomerDOB) #меняем тип данных со строки
```

```
df #вывод
```

```
df.loc[df.CustomerDOB.dt.year >= 2022, 'CustomerDOB'] = df.loc[df.CustomerDOB
df.head() #Substitute any year more than or equal 2022 by 100
```

```
def get_age(birthdate):
    return date.today().year - birthdate.year #функция преобразования
```

```
df['age'] = [get_age(x) for x in df['CustomerDOB']] #запись новой переменной
```

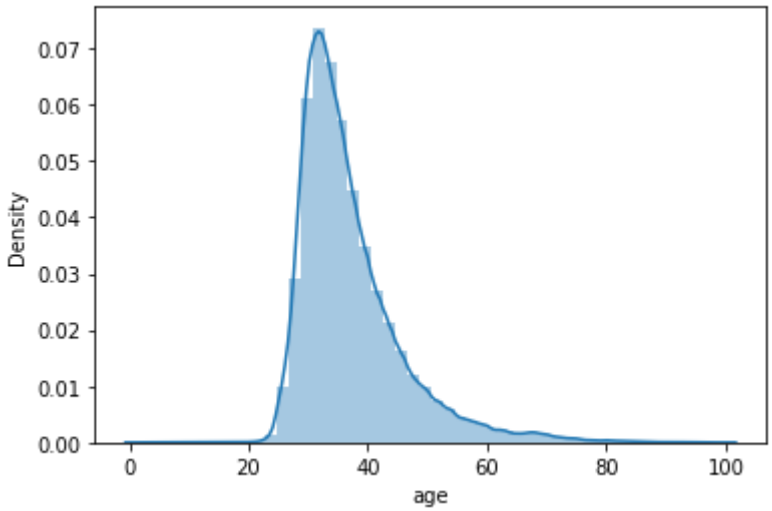
```
df #вывод
```

ОПИСАТЕЛЬНАЯ СТАТИСТИКА

```
df['age'].describe().T
```

```
sns.distplot(df['age'])
```

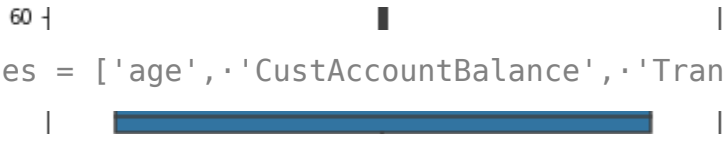
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: Futur
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fdde859bf10>
```




```
sns.boxplot(y=df['age'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdde8b69dd0>
```

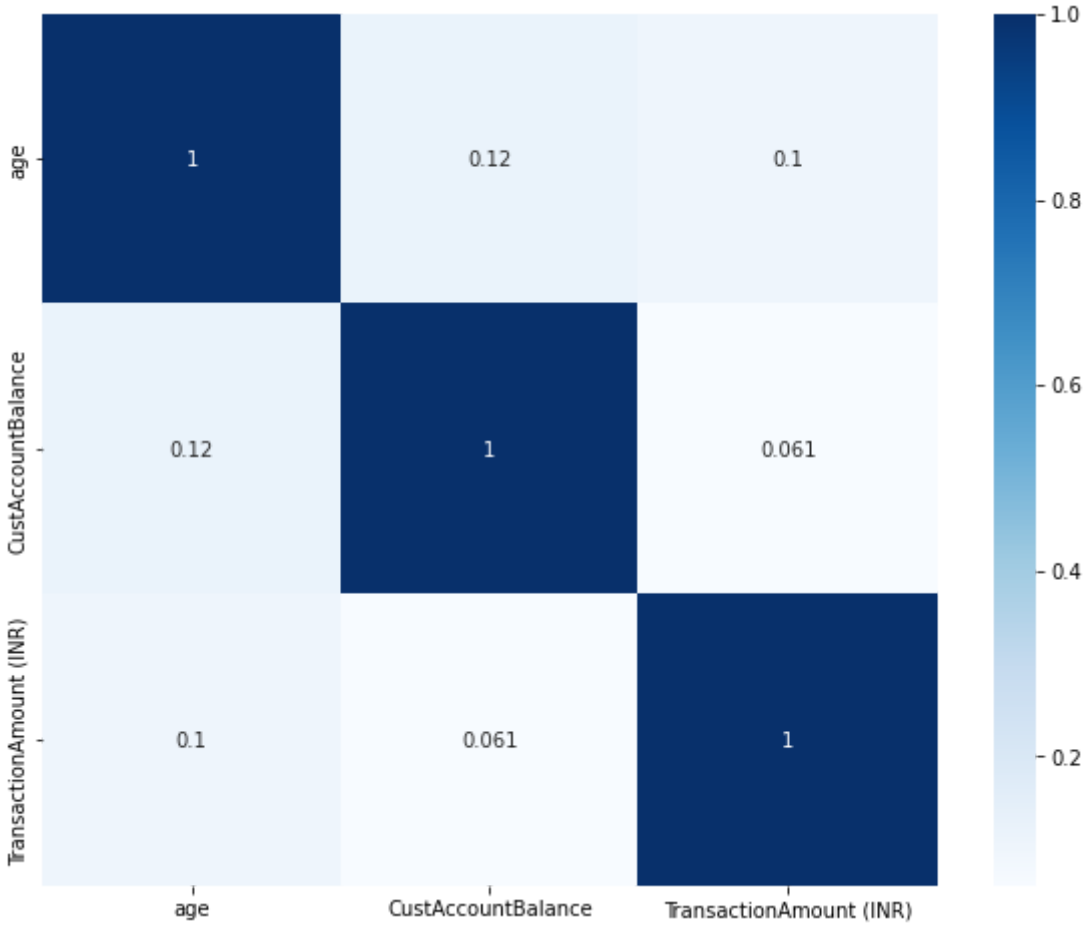
Кластеризация




```
variables = ['age',·'CustAccountBalance',·'TransactionAmount·(INR)']
```

```
plt.figure(figsize=(10, 8))
sns.heatmap(df[variables].corr(), annot=True, cmap='Blues')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdde55e24d0>
```

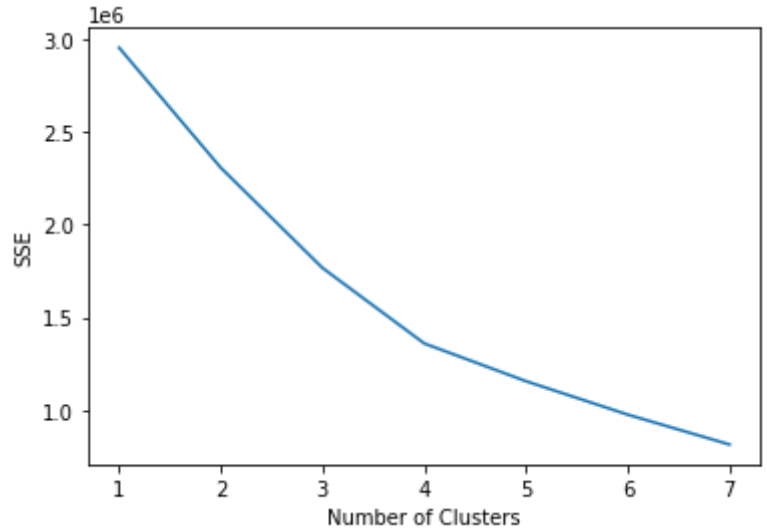



```
scaler = StandardScaler()
features = scaler.fit_transform(df[variables])


sse = []

for k in range(1, 8):
    # Создаем экземпляр класса KMeans иизадаем количество кластеров, равное и
    kmeans = KMeans(n_clusters=k)
    # Запустим алгоритм k-means на наших стандартизированных данных
    kmeans.fit(features)
    # В массив sse кладем наименьшее значение суммы квадратов ошибок
    sse.append(kmeans.inertia_)
```

```
plt.plot(range(1, 8), sse)
plt.xticks(range(1, 8))
plt.xlabel('Number of Clusters')
plt.ylabel('SSE')
plt.show()
```




```
kmeans = KMeans(n_clusters=4)
kmeans.fit(features)
sse.append(kmeans.inertia_)
```

```
df.head(5)
```

	TransactionID	CustomerID	CustomerDOB	CustGender	CustLocation	CustAcc
0	T1	C5841053	1994-10-01	F	JAMSHEDPUR	
1	T2	C2142763	1957-04-04	M	JHAJJAR	
2	T3	C4417068	1996-11-26	F	MUMBAI	
3	T4	C5342380	1973-09-14	F	MUMBAI	

```
sns.scatterplot(x='CustAccountBalance', y='age', data=df, hue=kmeans.labels_,
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdde81b2790>
```


```
kmeans = KMeans(n_clusters=3)
kmeans.fit(features)
sse.append(kmeans.inertia_)
```

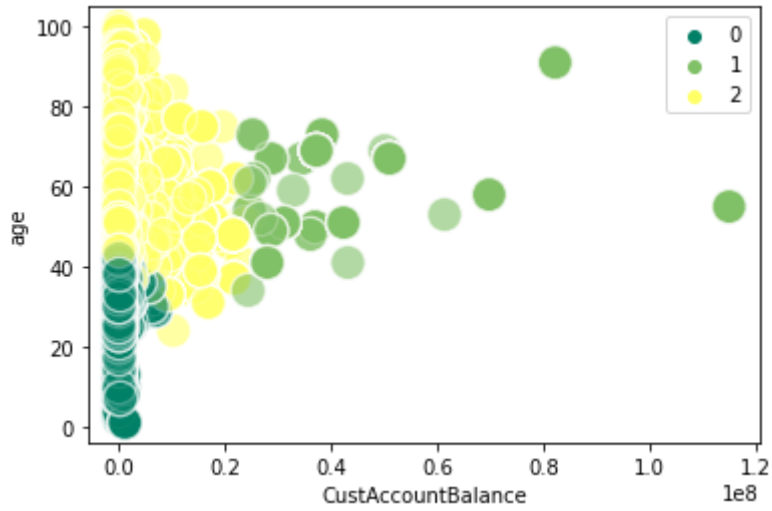

```
df.head(5)
```

	TransactionID	CustomerID	CustomerDOB	CustGender	CustLocation	CustAcc
0	T1	C5841053	1994-10-01	F	JAMSHEDPUR	
1	T2	C2142763	1957-04-04	M	JHAJJAR	
2	T3	C4417068	1996-11-26	F	MUMBAI	
3	T4	C5342380	1973-09-14	F	MUMBAI	

Кластеризация по трём кластерам

```
sns.scatterplot(x='CustAccountBalance', y='age', data=df, hue=kmeans.labels_,
```

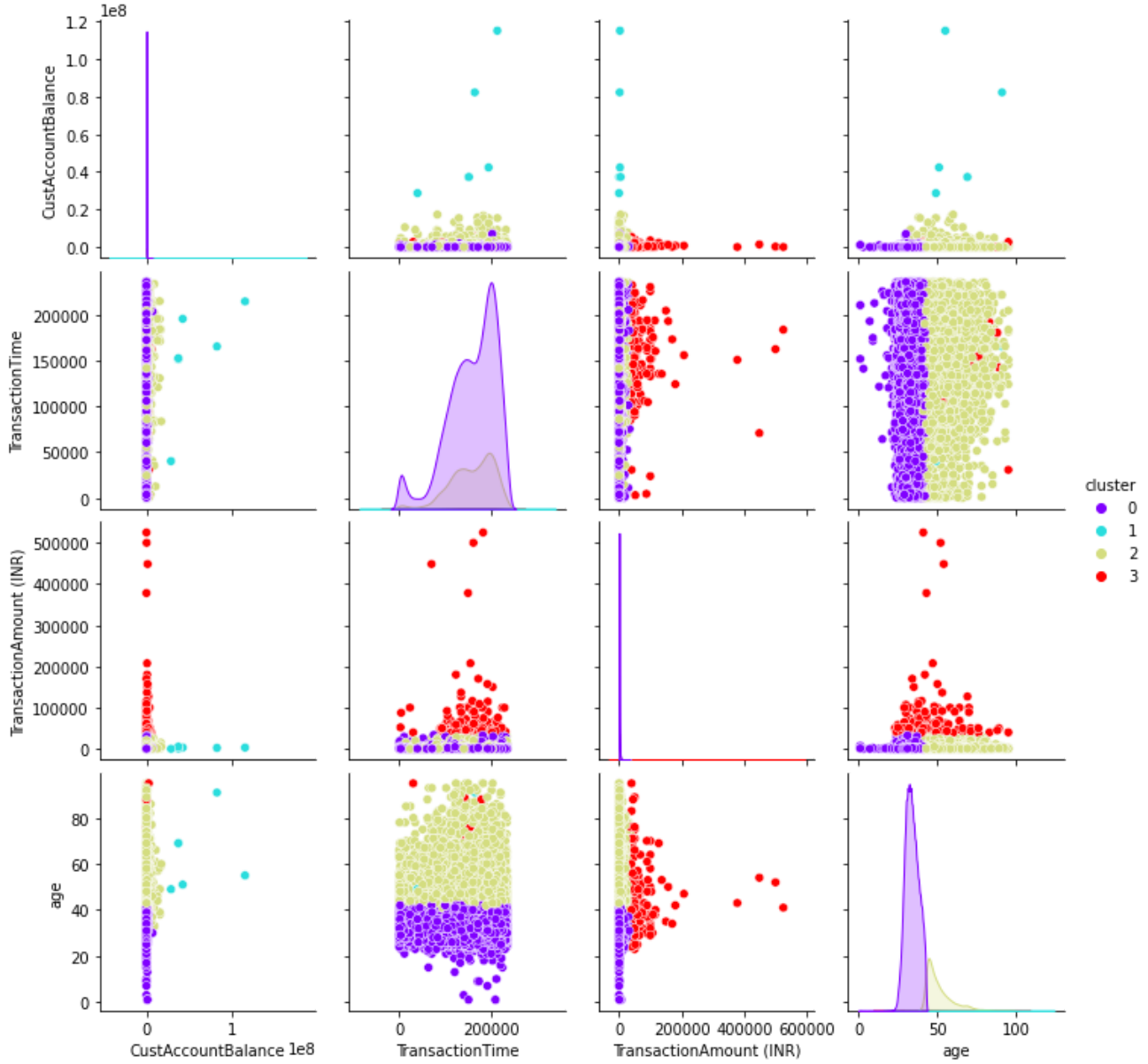
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdde7607790>
```




```
df['cluster'] = kmeans.labels_
```

```
sns.pairplot(df.sample(n=50000),hue='cluster',palette='rainbow')
```

```
<seaborn.axisgrid.PairGrid at 0x7fdde7b9ffd0>
```



Cohort analysis

Colab paid products - Cancel contracts here

✓ 1m 5s completed at 10:18 AM