Name: LIU,HONGYANG

Matric Number: 17201091/1

1. You are required to write code to implement either time-series clustering or density-based clustering model using the above dataset (Question 1). If you select density-based clustering approach to achieve the task, you are going to cover the following steps:
   - Importing required libraries
   - Load the dataset (Question 1) into a DataFrame object
   - Visualize the data, use only two of these attributes at the time
   - You may need to normalise the attribute if necessary
   - Show positive correlation between attributes if necessary
   - Construct a density-based clustering model and extract cluster labels and outliers to plot your results.

In [170]:

```
# Importing required libraries
```

In [171]:

```
import pandas as pd
from sklearn.cluster import KMeans
import seaborn as sns, numpy as np

import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn.datasets import make_moons
```

In [172]:

```
# Load the dataset (Question 1)  into a DataFrame object
```

In [173]:

```
customer=pd.read_csv("customer.csv",header=0,index_col=0)
transactions=pd.read_csv("transactions.csv",header=0,index_col=0)
product=pd.read_csv("product.csv",header=0,index_col=0)
```

In [174]:

```
product=product.rename(columns={"pro_cat_code":"prod_cat_code"})
prod_tran = pd.merge(left=transactions, right=product,on=["prod_cat_code","prod_
subcat_code"],how="left")
```

In [175]:

```
#2 merge "prod_tran" and "customer" table and create the table "df"
df= pd.merge(left=prod_tran, right=customer,right_on="customer_id", left_on="cus
tomer_id", how="left").drop_duplicates()
```

In [176]:

```
len(df)
```

Out[176]:

```
23040
```

In [177]:

```
df['transaction_date'] = pd.to_datetime(df['transaction_date'], errors='coerce')
df.insert(loc=3, column='Tran_year', value= df.transaction_date.dt.year)


df['DOB'] = pd.to_datetime(df['DOB'], errors='coerce')
df.insert(loc=4, column='Birth_year', value= df.DOB.dt.year)



df['Tran_year']=df['Tran_year'].astype(int)
df['Birth_year']=df['Birth_year'].astype(int)

df["age"]=df['Tran_year'] -df['Birth_year']

df= df.drop(['Tran_year','Birth_year'],axis=1)
```

In [178]:

```
df.head()
```

Out[178]:

	transaction_id	customer_id	transaction_date	prod_cat_code	prod_subcat_code	Quantity
0	80712190438	270351	2014-02-28	1	1	-5
1	29258453508	270384	2014-02-27	5	3	-5
2	51750724947	273420	2014-02-24	6	5	-2
3	93274880719	271509	2014-02-24	11	6	-3
4	51750724947	273420	2014-02-23	6	5	-2

In [179]:

```
# data preprocessing
df = df.dropna()
len(df)
```

Out[179]:

```
5952
```

In [180]:

```
# Visualize the data, use only two of these attributes at the time
```

In [198]:

```
from sklearn.preprocessing import StandardScaler
#fautures:

Features=["customer_id","Gender","Total_amount","prod_cat_code","prod_subcat_cod
e","Quantity","Price","Tax","Store_type","City","age"]


X= pd.get_dummies(df[Features])


x = StandardScaler().fit_transform(X.values)
```

In [199]:

```
# ax = sns.pairplot(X)
```

In [200]:

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['principal component 1', 'principal component 2'])
```

In [201]:

```
principalDf.head()
```

Out[201]:

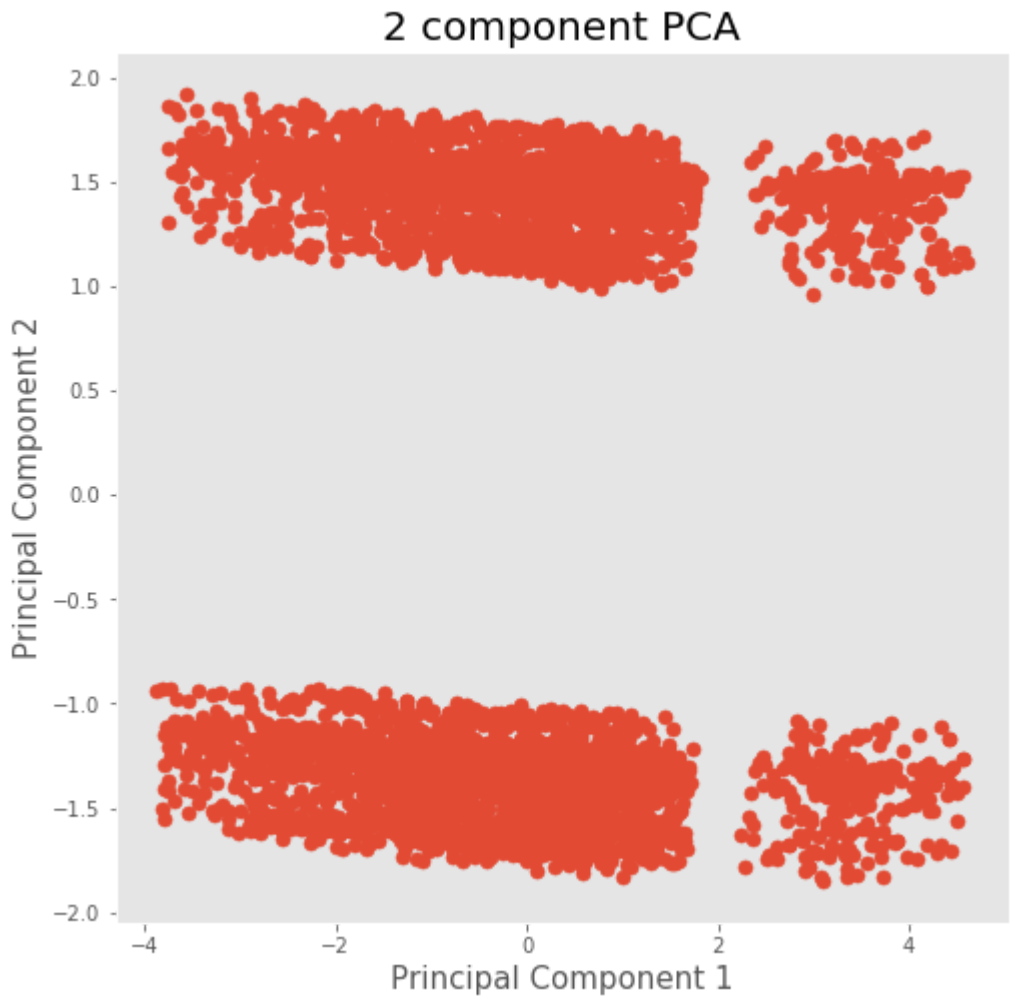
	principal component 1	principal component 2
0	3.908875	-1.369639
1	4.613643	1.117297
2	-3.742446	1.303831
3	-1.352817	1.640442
4	0.260002	1.351724

In [202]:

```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)


ax.scatter(principalDf.loc[:, 'principal component 1']
               , principalDf.loc[:, 'principal component 2']
               , s = 50)

ax.grid()
```



In [203]:

```
# Construct a density-based clustering model and extract cluster labels and outl
iers to plot your results.
```

In [204]:

```
plt.style.use('ggplot')
%matplotlib inline
```

In [205]:

```
model = DBSCAN(eps=0.2, min_samples=20).fit(principalDf)

print(model)
```

```
DBSCAN(algorithm='auto', eps=0.2, leaf_size=30, metric='euclidean',
       metric_params=None, min_samples=20, n_jobs=None, p=None)
```

In [206]:

```
model.labels_
```

Out[206]:

```
array([ 0, -1, -1, ...,  2,  1,  2])
```

In [207]:

```
model.core_sample_indices_
```

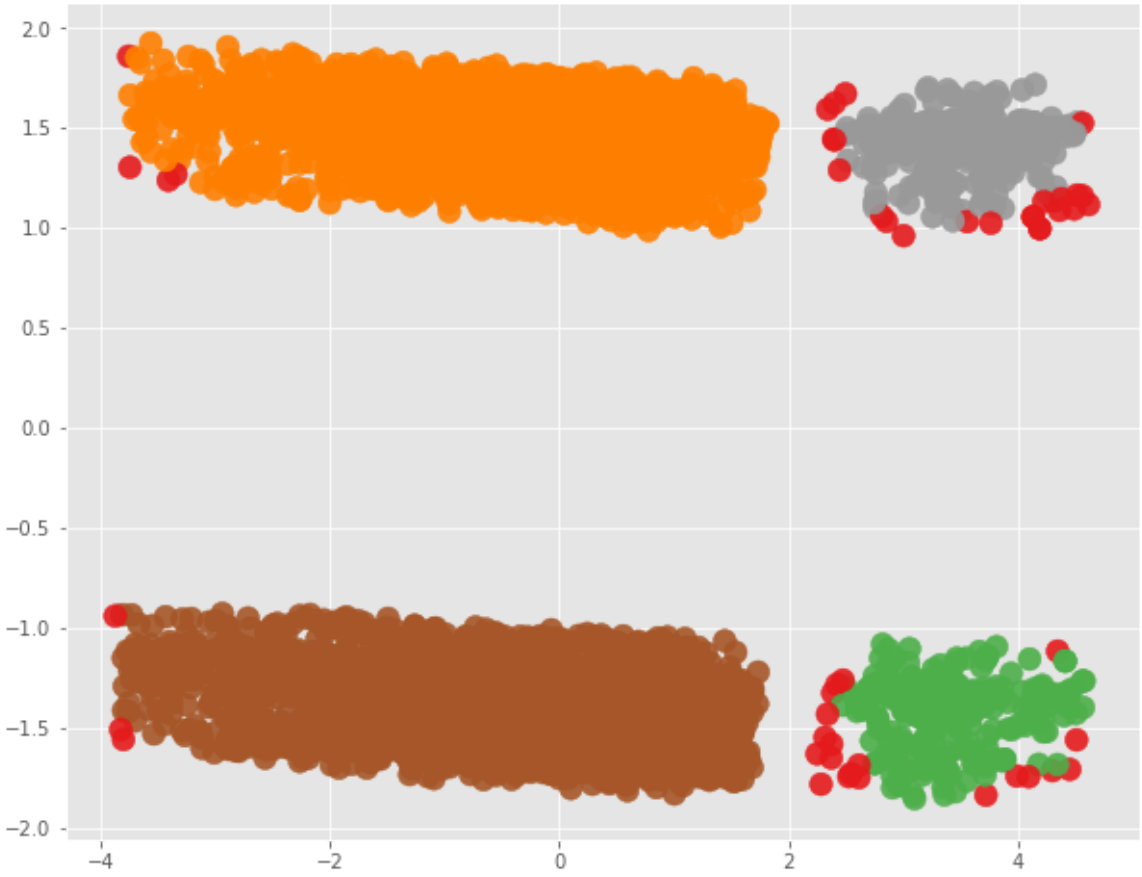
Out[207]:

```
array([   0,    3,    4, ..., 5949, 5950, 5951])
```

In [211]:

```
fig, ax = plt.subplots(figsize=(10,8))
sctr = ax.scatter(principalDf.iloc[:,0],principalDf.iloc[:,1],c=model.labels_, s
=140,alpha=0.9,cmap=plt.cm.Set1)
fig.show()
```

```
/Users/liuhongyang/anaconda3/lib/python3.7/site-packages/ipykernel_l
auncher.py:3: UserWarning: Matplotlib is currently using module://ip
ykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  This is separate from the ipykernel package so we can avoid doing
imports until
```



In [236]:

```
fig = plt.figure(figsize=(60, 60))
fig.subplots_adjust(hspace=.5, wspace=.2)
i = 1
for x in range(10, 1, -1):
    eps = 1/(11-x)
    db = DBSCAN(eps=eps, min_samples=20).fit(principalDf)

    core_samples_mask = np.zeros_like(model.labels_, dtype=bool)

    core_samples_mask[db.core_sample_indices_] = True

    ax = fig.add_subplot(3, 3, i)

    ax.text(2, 3, "eps = {}".format(round(eps, 3)), fontsize=45, ha="center")


    sctr = ax.scatter(principalDf.iloc[:,0],principalDf.iloc[:,1],c=model.labels
_, s=140,alpha=0.9,cmap=plt.cm.Set1)
    i += 1

plt.savefig("multi_eps.png", dpi=300)
```

In [ ]: