PREDICTIVE ANALYTIC REPORT

This analysis is designed to predict customer salary correlation with certain custom attributes and then create models for predicting future salary trends

---

Import Required Libraries

In [36]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Get Data For Analysis

In [38]:
```
data = "ANZ_synthesised_transaction_dataset.xlsx"
customer_data = pd.read_excel(data)
```

In [4]:
```
customer_data.head()
```

Out[4]:

age	merchant_suburb	merchant_state	extraction	amount	tran
26	Ashmore	QLD	2018-08-01T01:01:15.000+0000	16.25	a623070bfead4541a6b0fff
26	Sydney	NSW	2018-08-01T01:13:45.000+0000	14.19	13270a2a902145da9db4c95
38	Sydney	NSW	2018-08-01T01:26:15.000+0000	6.42	feb79e7ecd7048a5a36ec88
40	Buderim	QLD	2018-08-01T01:38:45.000+0000	40.90	2698170da3704fd981b15e6
26	Mermaid Beach	QLD	2018-08-01T01:51:15.000+0000	3.25	329adf79878c4cf0aeb4188

In [5]:
```
customer_data.tail()
```

Out[5]:

chant_state	extraction	amount	transaction_id	country	customer_id
VIC	2018-10-31T23:09:06.000+0000	9.79	f2e3e695c2ee4c50a4c8747f852cbe2e	Australia	CUS-55310383
NSW	2018-10-31T23:21:46.000+0000	63.87	56e147e5485f4683b9076fcaaed76640	Australia	CUS-2688605418
NSW	2018-10-31T23:34:25.000+0000	43.96	2fdd4681827343f6af2e6519644a684a	Australia	CUS-2663907001
VIC	2018-10-31T23:47:05.000+0000	30.77	74aa9cd7e4af4c6d9cd7dbd28e9aedc9	Australia	CUS-1388323263
NSW	2018-10-31T23:59:44.000+0000	22.36	6d5218e04e8040b9996850ce11a19426	Australia	CUS-3129499595

Checking Details of data

In [6]:
```
customer_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12043 entries, 0 to 12042
Data columns (total 23 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   status             12043 non-null  object
 1   card_present_flag  7717 non-null   float64
 2   bpay_biller_code   885 non-null    object
 3   account            12043 non-null  object
 4   currency           12043 non-null  object
 5   long_lat           12043 non-null  object
 6   txn_description    12043 non-null  object
 7   merchant_id        7717 non-null   object
 8   merchant_code      883 non-null    float64
 9   first_name         12043 non-null  object
 10  balance            12043 non-null  float64
 11  date               12043 non-null  datetime64[ns]
 12  gender             12043 non-null  object
 13  age                12043 non-null  int64
 14  merchant_suburb    7717 non-null   object
 15  merchant_state     7717 non-null   object
 16  extraction         12043 non-null  object
 17  amount             12043 non-null  float64
 18  transaction_id     12043 non-null  object
 19  country            12043 non-null  object
 20  customer_id        12043 non-null  object
 21  merchant_long_lat  7717 non-null   object
 22  movement           12043 non-null  object
dtypes: datetime64[ns](1), float64(4), int64(1), object(17)
memory usage: 2.1+ MB
```

In [8]:
```
customer_data.shape
```

Out[8]: (12043, 23)

In [10]:
```
customer_data.describe()
```

Out[10]:

	card_present_flag	merchant_code	balance	age	amount
count	7717.000000	883.0	12043.000000	12043.000000	12043.000000
mean	0.802644	0.0	14704.195553	30.582330	187.933588
std	0.398029	0.0	31503.722652	10.046343	592.599934
min	0.000000	0.0	0.240000	18.000000	0.100000
25%	1.000000	0.0	3158.585000	22.000000	16.000000
50%	1.000000	0.0	6432.010000	28.000000	29.000000
75%	1.000000	0.0	12465.945000	38.000000	53.655000
max	1.000000	0.0	267128.520000	78.000000	8835.980000

Getting the Data Cleaned

In [14]:
```
customer_data.isnull()
```

Out[14]:

ınt_id	merchant_code	first_name	...	age	merchant_suburb	merchant_state	extraction	amount	t
False	True	False	...	False	False	False	False	False	
False	True	False	...	False	False	False	False	False	
False	True	False	...	False	False	False	False	False	
False	True	False	...	False	False	False	False	False	
False	True	False	...	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	
False	True	False	...	False	False	False	False	False	
False	True	False	...	False	False	False	False	False	
False	True	False	...	False	False	False	False	False	
False	True	False	...	False	False	False	False	False	
False	True	False	...	False	False	False	False	False	

In [19]:
```
customer_data['balance'].isnull().value_counts()
```

Out[19]:
```
False    12043
Name: balance, dtype: int64
```

In [21]:
```
customer_data['age'].isnull().value_counts()
```

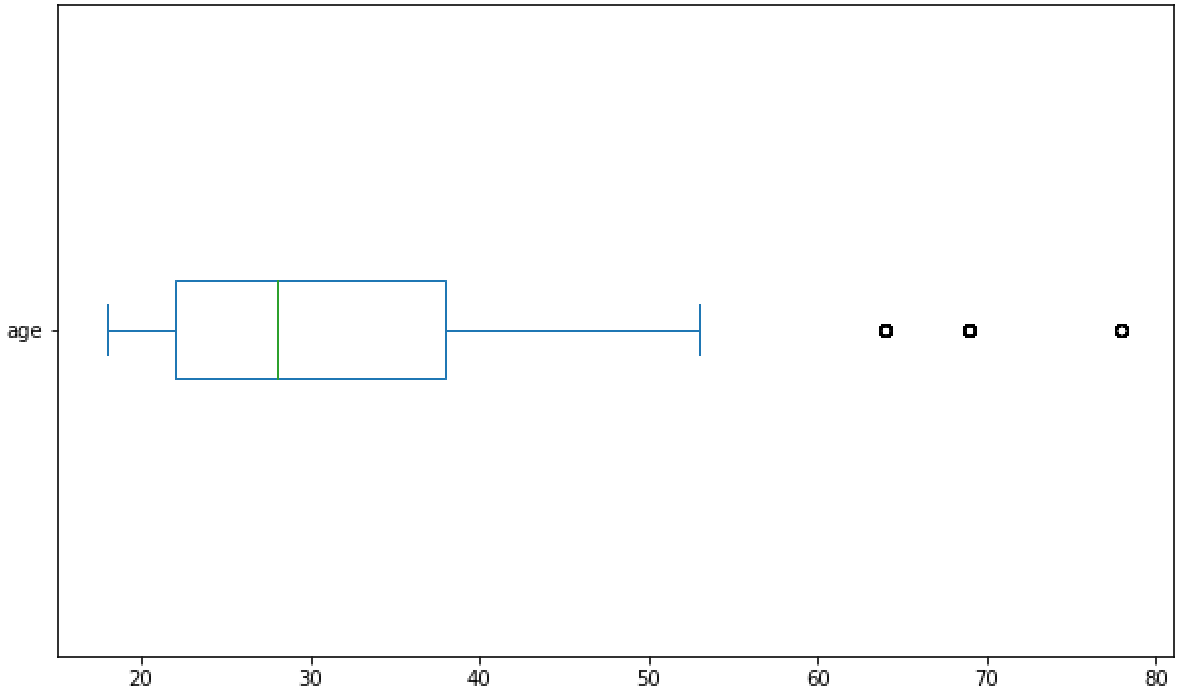
Out[21]:
```
False    12043
Name: age, dtype: int64
```

In [25]:
```
#Checking for outliers
customer_data['age'].describe()
```

Out[25]:
```
count    12043.000000
mean        30.582330
std         10.046343
min         18.000000
25%         22.000000
50%         28.000000
75%         38.000000
max         78.000000
Name: age, dtype: float64
```

In [27]:
```
#Checking for outliers
customer_data['age'].plot(kind='box', vert=False, figsize = (10, 6))
```

Out[27]: <AxesSubplot:>



In [22]:
```
customer_data['gender'].value_counts()
```

Out[22]:
```
M    6285
F    5758
Name: gender, dtype: int64
```

In [24]:
```
customer_data['amount'].isnull().value_counts()
```

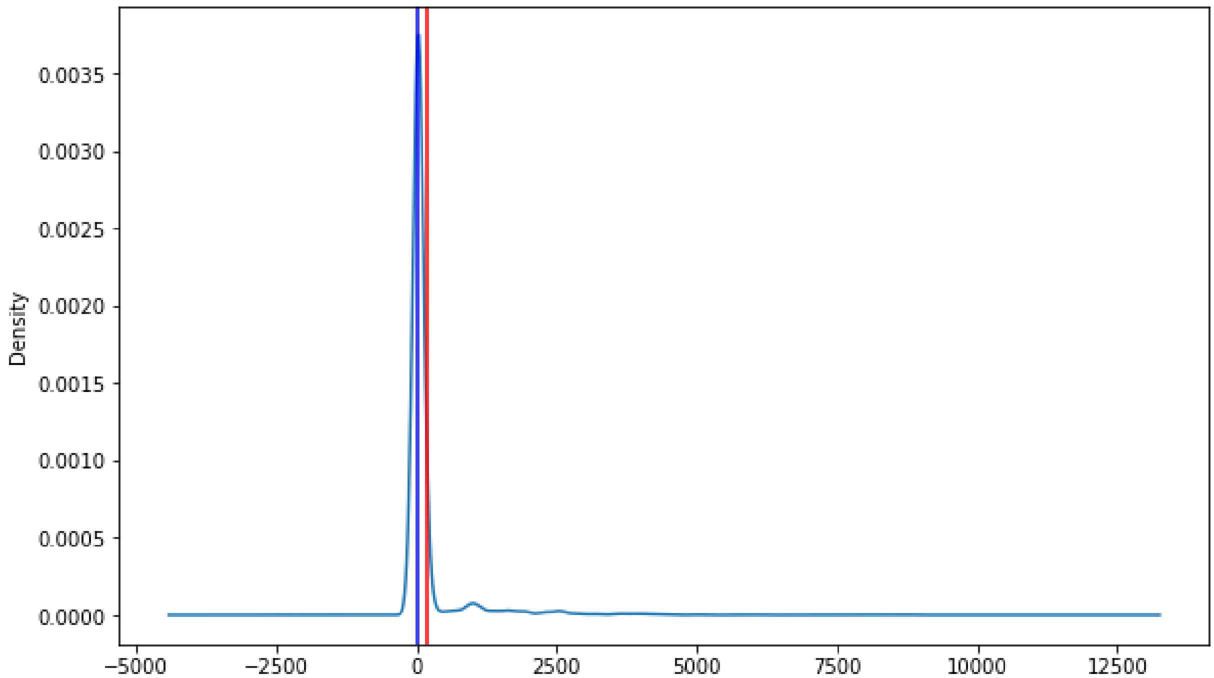
Out[24]:
```
False    12043
Name: amount, dtype: int64
```

In [37]:
```
#Checking for outliers
customer_data['amount'].describe()
```

Out[37]:
```
count    12043.000000
mean       187.933588
std        592.599934
min          0.100000
25%         16.000000
50%         29.000000
75%         53.655000
max       8835.980000
Name: amount, dtype: float64
```

In [35]:
```
dx = customer_data['amount'].plot(kind='density', figsize = (10,6))
dx.axvline(customer_data['amount'].mean(), color="red")
dx.axvline(customer_data['amount'].median(), color='blue')
```

Out[35]: <matplotlib.lines.Line2D at 0x1735be3cb20>

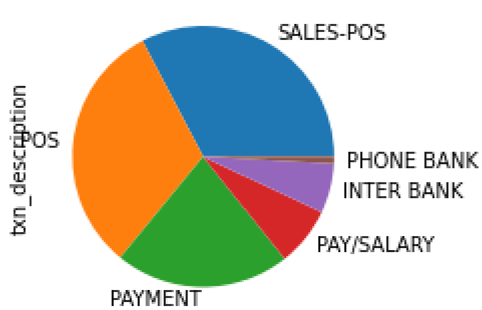


In [39]:
```
customer_data['txn_description'].value_counts()
```

Out[39]:
```
SALES-POS      3934
POS            3783
PAYMENT        2600
PAY/SALARY      883
INTER BANK      742
PHONE BANK      101
Name: txn_description, dtype: int64
```

In [44]:
```
customer_data['txn_description'].value_counts().plot(kind="pie", figsize =(4,3))
```

Out[44]: <AxesSubplot:ylabel='txn_description'>

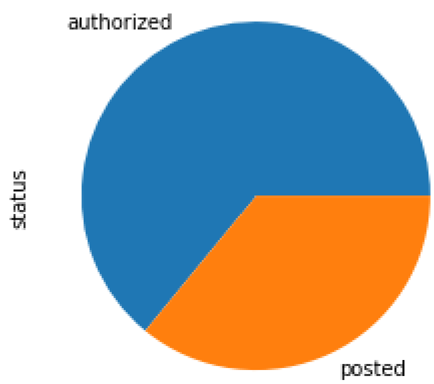


In [41]:
```
customer_data['status'].value_counts()
```

Out[41]:
```
authorized    7717
posted        4326
Name: status, dtype: int64
```

In [43]:
```
customer_data['status'].value_counts().plot(kind="pie", figsize=(4,4))
```

Out[43]: <AxesSubplot:ylabel='status'>

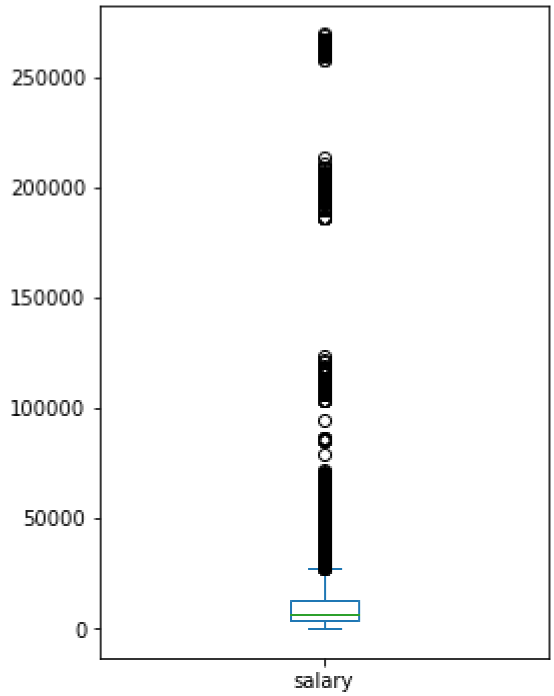


In [39]:
```
customer_data['salary'] = customer_data['amount'] + customer_data['balance']
customer_data['salary'].head()
```

Out[39]:
```
0      51.64
1      35.39
2      12.13
3    2158.12
4      21.20
Name: salary, dtype: float64
```

In [46]:
```
customer_data['salary'].plot(kind="box", figsize=(4,6))
```

Out[46]:
```
<AxesSubplot:>
```



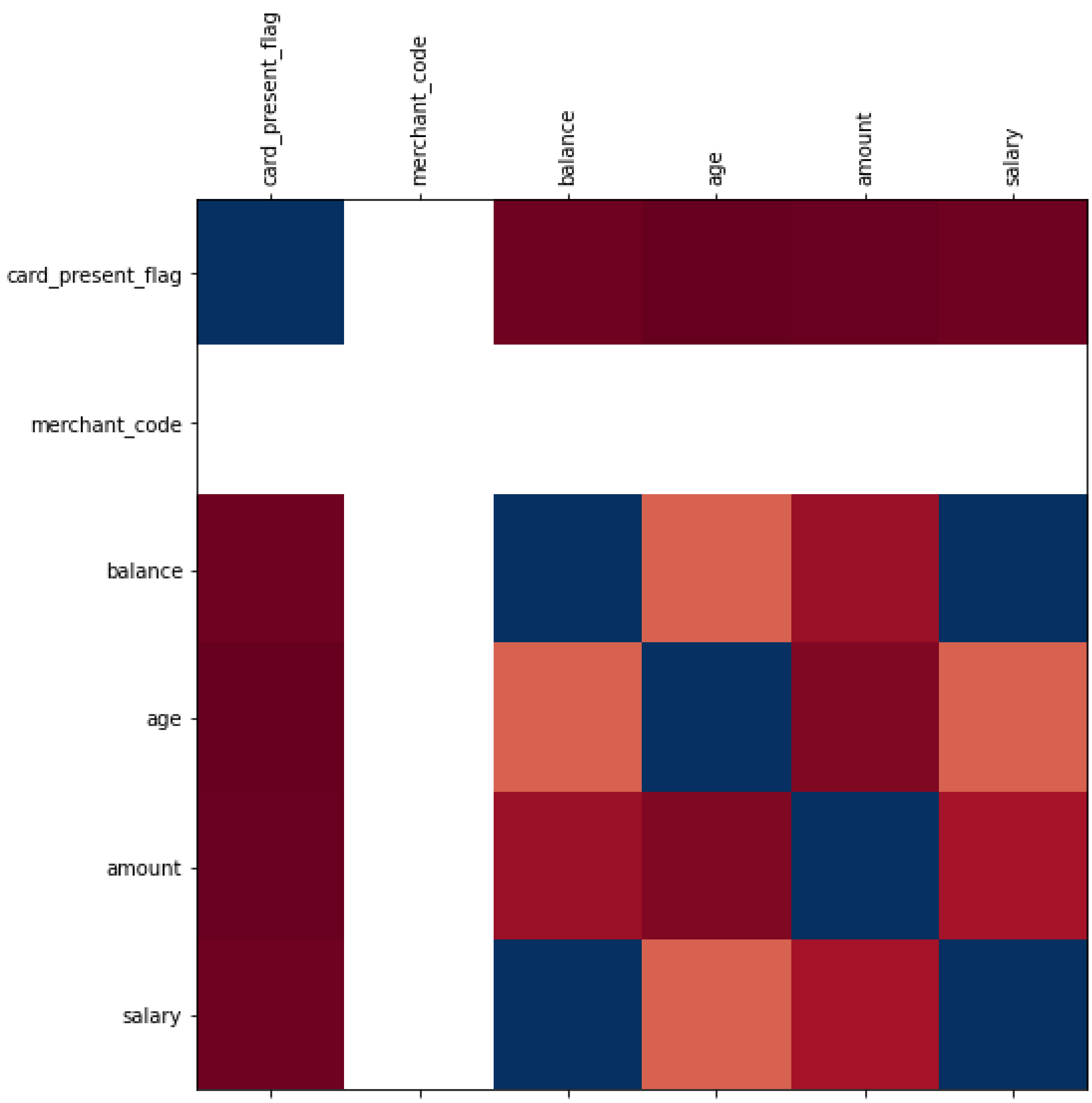
Finding Relationships and Correlations

In [51]:
```
corr = customer_data.corr()
corr
```

Out[51]:

	card_present_flag	merchant_code	balance	age	amount	salary
card_present_flag	1.000000	NaN	0.005925	-0.008405	-0.002074	0.005912
merchant_code	NaN	NaN	NaN	NaN	NaN	NaN
balance	0.005925	NaN	1.000000	0.199329	0.059178	0.999824
age	-0.008405	NaN	0.199329	1.000000	0.029980	0.199636
amount	-0.002074	NaN	0.059178	0.029980	1.000000	0.077888
salary	0.005912	NaN	0.999824	0.199636	0.077888	1.000000

In [53]:
```
fig = plt.figure(figsize=(8,8))
plt.matshow(corr, cmap='RdBu', fignum=fig.number)
plt.xticks(range(len(corr.columns)), corr.columns, rotation='vertical');
plt.yticks(range(len(corr.columns)), corr.columns);
```



It is clear that Merchant code has no impact on the data

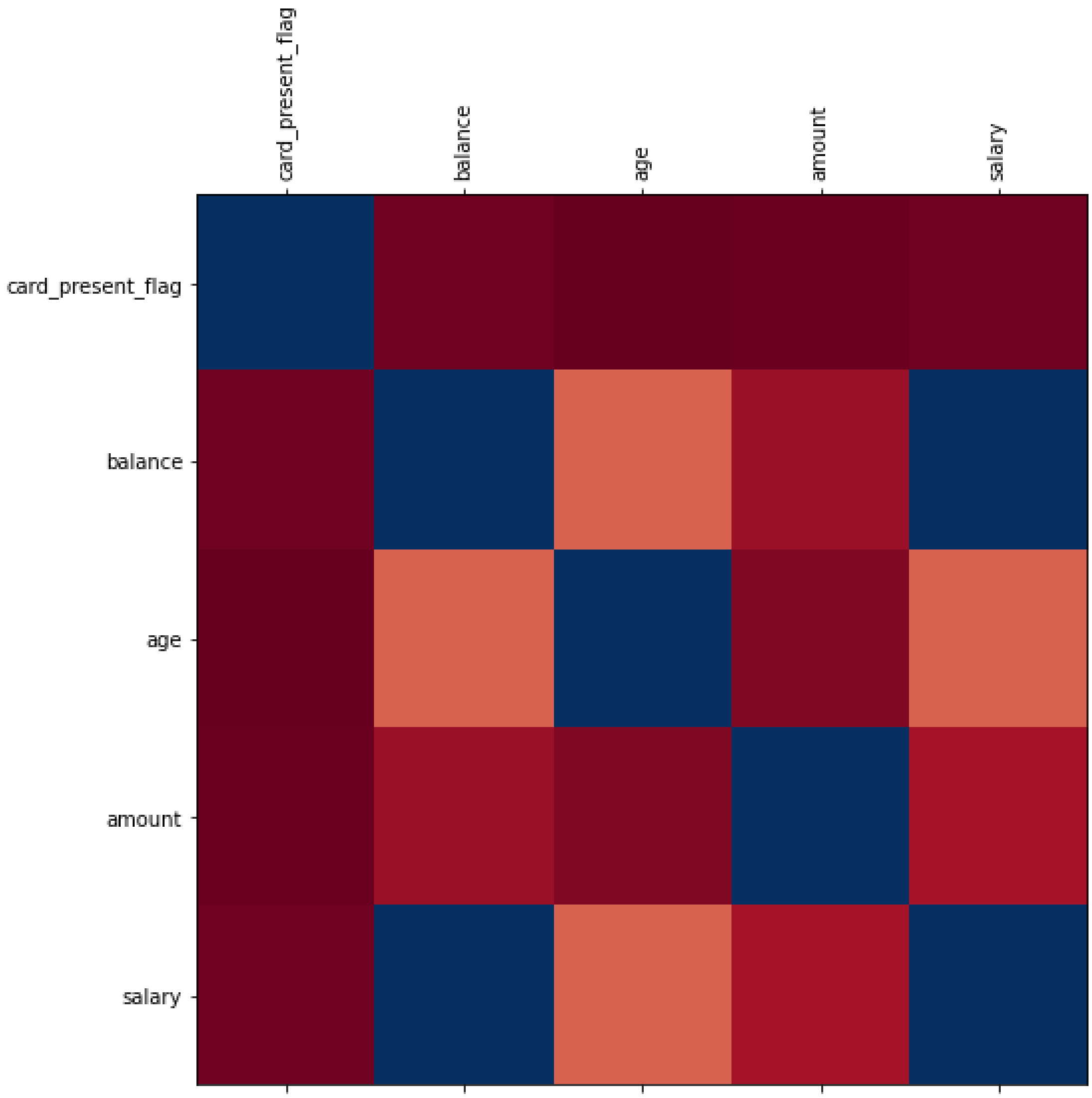
If we decide to drop it the new correlation will look like this

In [57]:
```
corr = customer_data.drop(['merchant_code'], axis = 1).corr()
corr
```

Out[57]:

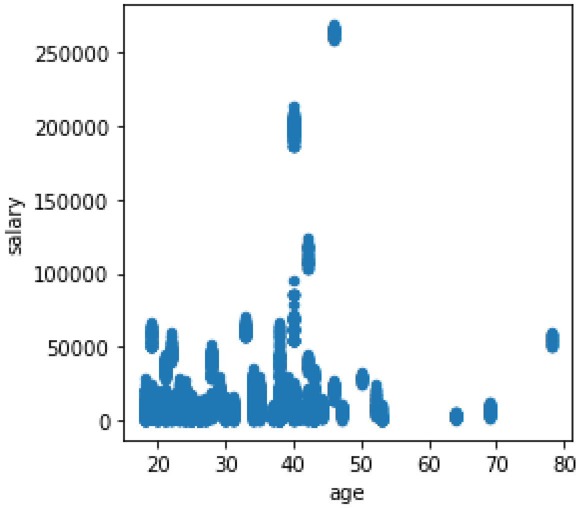
	card_present_flag	balance	age	amount	salary
card_present_flag	1.000000	0.005925	-0.008405	-0.002074	0.005912
balance	0.005925	1.000000	0.199329	0.059178	0.999824
age	-0.008405	0.199329	1.000000	0.029980	0.199636
amount	-0.002074	0.059178	0.029980	1.000000	0.077888
salary	0.005912	0.999824	0.199636	0.077888	1.000000

In [58]:
```
fig = plt.figure(figsize=(8,8))
plt.matshow(corr, cmap='RdBu', fignum=fig.number)
plt.xticks(range(len(corr.columns)), corr.columns, rotation='vertical');
plt.yticks(range(len(corr.columns)), corr.columns);
```



In [67]:
```
customer_data.plot(kind='scatter', x= 'age', y='salary', figsize=(4,4))
```

Out[67]: <AxesSubplot:xlabel='age', ylabel='salary'>

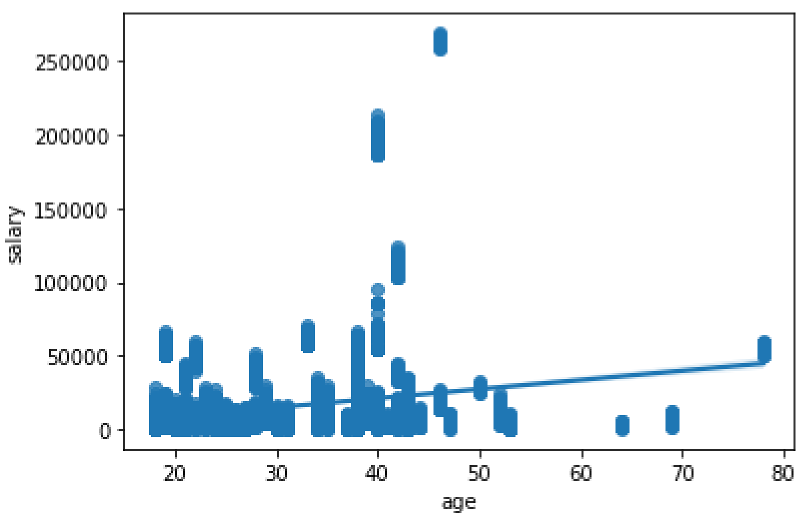


In [ ]:

Creating a regression model

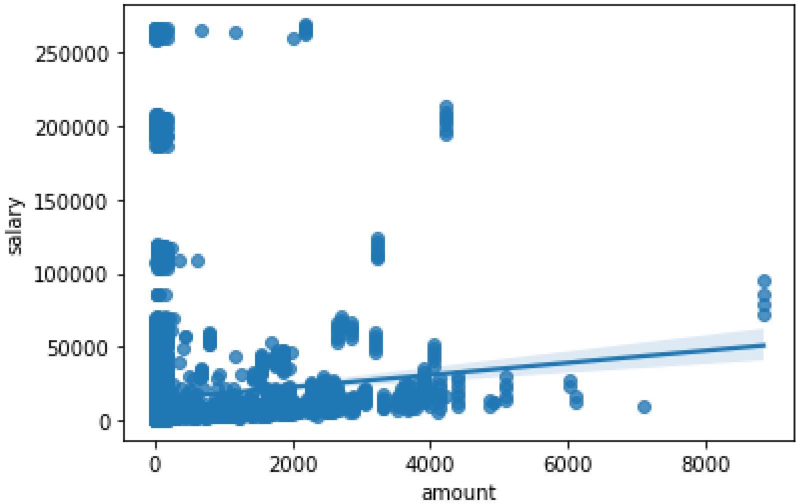
In [8]:
```
#simple linear regression
sns.regplot(x='age', y='salary', data=customer_data)
```

Out[8]: <AxesSubplot:xlabel='age', ylabel='salary'>



In [12]:
```
#Relationship between the amount a customer is willing to spend and his salary
sns.regplot(x='amount', y='salary', data=customer_data)
```

Out[12]: <AxesSubplot:xlabel='amount', ylabel='salary'>



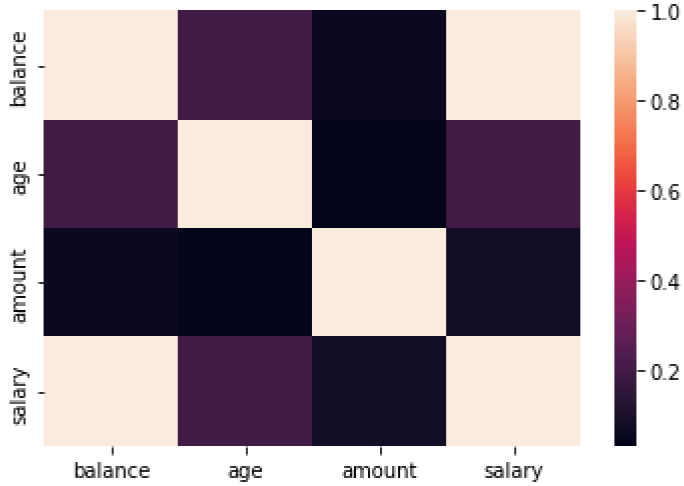
In [60]:
```
#predicting the salary model based on amount spent, age, and account balance
reqdata = customer_data[['balance','age','amount','salary']]
reqdata.head()
```

Out[60]:

	balance	age	amount	salary
0	35.39	26	16.25	51.64
1	21.20	26	14.19	35.39
2	5.71	38	6.42	12.13
3	2117.22	40	40.90	2158.12
4	17.95	26	3.25	21.20

In [61]:
```
#correlation matrix for the new data
sns.heatmap(reqdata.corr())
```

Out[61]: <AxesSubplot:>



In [62]:
```
x = reqdata.iloc[:, :-1].values
y = reqdata.iloc[:, 3].values
```

In [63]:
```
#spliting dataset into training and testing data
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_s
```

In [64]:
```
#fitting mutiple linear regression to the training set
from sklearn.linear_model import LinearRegression
model_fit = LinearRegression()
model_fit.fit(x_train, y_train)
```

Out[64]: LinearRegression()

In [65]:
```
y_predict = model_fit.predict(x_test)
y_predict
```

Out[65]: array([2176.02, 7920.17, 6370.68, ..., 183.91, 360.95, 492.41])

In [66]:
```
from sklearn.metrics import r2_score
r2_score(y_test, y_predict)
```

Out[66]: 1.0

Perfect Model

The Age, Account balance and amount spent can be used to predict the salary of a customer

In [ ]: