Exploratory Data Analysis

June 20, 2021

1 ANZ Virtual Internship Report

Task 1 Exploratory data analysis

```
In [1]: import os
        cwd = os.getcwd()
        cwd

Out[1]: 'C:\\Users\\hp'

In [2]: os.chdir("C:/Users/hp/Desktop/Data Science-Data@ANZ/Module 1")

In [3]: os.listdir('.')

Out[3]: ['ANZ synthesised transaction dataset.xlsx',
         'anz_cleaned_data.xlsx',
         'Data@ANZ.ipynb',
         'Model Answer for Task 1.pdf',
         '~$ANZ synthesised transaction dataset.xlsx']

In [4]: import xlrd
        import numpy as np
        import pandas as pd
        import datetime as dt
        import seaborn as sns
        import matplotlib.pyplot as plt

In [5]: file = "ANZ synthesised transaction dataset.xlsx"
        xl = pd.ExcelFile(file)
        print(xl.sheet_names)

['DSynth_Output_100c_3m_v3']

In [6]: df = pd.read_excel(file, sheet_name = "DSynth_Output_100c_3m_v3")
        df.head()
```

```
Out[6]:       status  card_present_flag bpay_biller_code         account currency  \
0  authorized                1.0              NaN  ACC-1598451071      AUD
1  authorized                0.0              NaN  ACC-1598451071      AUD
2  authorized                1.0              NaN  ACC-1222300524      AUD
3  authorized                1.0              NaN  ACC-1037050564      AUD
4  authorized                1.0              NaN  ACC-1598451071      AUD

       long_lat txn_description                           merchant_id  \
0  153.41 -27.95             POS  81c48296-73be-44a7-befa-d053f48ce7cd
1  153.41 -27.95       SALES-POS  830a451c-316e-4a6a-bf25-e37caedca49e
2  151.23 -33.94             POS  835c231d-8cdf-4e96-859d-e9d571760cf0
3  153.10 -27.66       SALES-POS  48514682-c78a-4a88-b0da-2d6302e64673
4  153.41 -27.95       SALES-POS  b4e02c10-0852-4273-b8fd-7b3395e32eb0

   merchant_code first_name   ...     age merchant_suburb merchant_state  \
0            NaN      Diana   ...      26         Ashmore            QLD
1            NaN      Diana   ...      26          Sydney            NSW
2            NaN    Michael   ...      38          Sydney            NSW
3            NaN     Rhonda   ...      40         Buderim            QLD
4            NaN      Diana   ...      26   Mermaid Beach            QLD

                    extraction amount                    transaction_id  \
0  2018-08-01T01:01:15.000+0000  16.25  a623070bfead4541a6b0fff8a09e706c
1  2018-08-01T01:13:45.000+0000  14.19  13270a2a902145da9db4c951e04b51b9
2  2018-08-01T01:26:15.000+0000   6.42  feb79e7ecd7048a5a36ec889d1a94270
3  2018-08-01T01:38:45.000+0000  40.90  2698170da3704fd981b15e64a006079e
4  2018-08-01T01:51:15.000+0000   3.25  329adf79878c4cf0aeb4188b4691c266

     country     customer_id merchant_long_lat movement
0  Australia  CUS-2487424745     153.38 -27.99    debit
1  Australia  CUS-2487424745     151.21 -33.87    debit
2  Australia  CUS-2142601169     151.21 -33.87    debit
3  Australia  CUS-1614226872     153.05 -26.68    debit
4  Australia  CUS-2487424745     153.44 -28.06    debit

[5 rows x 23 columns]
```

2 Data preparation

The dataset contains 12043 transactions for 100 customers who have one bank account each. Trasactional period is from 01/08/2018 - 31/10/2018 (92 days duration). The data entries are unique and have consistent formats for analysis. For each record/row, information is complete for majority of columns. Some columns contain missing data (blank or NA cells), which is likely due to the nature of transaction. (i.e. merchants are not involved for InterBank transfers or Salary payments) It is also noticed that there is only 91 unique dates in the dataset, suggesting the transaction records for one day are missing (turned out to be 2018-08-16). The range of each feature should also be examined which shows that there is one customer that resides outside Australia

In [7]:
```
df.shape
```

Out[7]:
```
(12043, 23)
```

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12043 entries, 0 to 12042
Data columns (total 23 columns):
status               12043 non-null object
card_present_flag    7717 non-null float64
bpay_biller_code     885 non-null object
account              12043 non-null object
currency             12043 non-null object
long_lat             12043 non-null object
txn_description      12043 non-null object
merchant_id          7717 non-null object
merchant_code        883 non-null float64
first_name           12043 non-null object
balance              12043 non-null float64
date                 12043 non-null datetime64[ns]
gender               12043 non-null object
age                  12043 non-null int64
merchant_suburb      7717 non-null object
merchant_state       7717 non-null object
extraction           12043 non-null object
amount               12043 non-null float64
transaction_id       12043 non-null object
country              12043 non-null object
customer_id          12043 non-null object
merchant_long_lat    7717 non-null object
movement             12043 non-null object
dtypes: datetime64[ns](1), float64(4), int64(1), object(17)
memory usage: 2.1+ MB
```

3 Data Cleaning Stage

In [9]:
```
# checking the unique values in currency
df.currency.value_counts()
```

Out[9]:
```
AUD    12043
Name: currency, dtype: int64
```

In [10]:
```
# droping currency column as of no use in analysis has no unique value.
df.drop(['currency'], axis = 1 , inplace = True)
```

In [11]:
```
# checking for nan in dataset
df.isnull().sum()
```

Out[11]:
```
status                    0
card_present_flag      4326
bpay_biller_code      11158
account                   0
long_lat                  0
txn_description           0
merchant_id            4326
merchant_code         11160
first_name                0
balance                   0
date                      0
gender                    0
age                       0
merchant_suburb        4326
merchant_state         4326
extraction                0
amount                    0
transaction_id            0
country                   0
customer_id               0
merchant_long_lat      4326
movement                  0
dtype: int64
```

In [12]:
```
df.drop(['bpay_biller_code', 'merchant_code'], axis = 1 , inplace = True)
```

In [13]:
```
# counti# these are the unique values in the status column.
df['status'].value_counts()
df.duplicated().sum()
```

Out[13]:
```
0
```

In [14]:
```
# it looks like the NaNs in the dataset are all on commun rows.
df[df.card_present_flag.isnull()][['merchant_id', 'merchant_state', 'merchant_suburb'
```

Out[14]:
```
merchant_id          4326
merchant_state       4326
merchant_suburb      4326
merchant_long_lat    4326
dtype: int64
```

In [15]:
```
df['status'].value_counts()
```

Out[15]:
```
authorized    7717
posted        4326
Name: status, dtype: int64
```

In [16]:
```
# all the transactions happened in Australia.
df['country'].value_counts()
# then we can drop the country column
df.drop(['country'], axis = 1 , inplace = True)
```

In [17]:
```
# the unique movement types
df.movement.value_counts()
```

Out[17]:
```
debit     11160
credit      883
Name: movement, dtype: int64
```

In [18]:
```
# dtype of date column and extraction column to datetime
df['date'] = pd.to_datetime(df['date'])
df['extraction'] = pd.to_datetime(df['extraction'])
```

In [19]:
```
# adding three columns: year, month, and day of the transaction to make the segmentat
df['day']= df['date'].dt.day_name()
df['month'] = df['date'].dt.month_name()
df['year'] = df['date'].dt.year
df['hour']= df.extraction.dt.hour
```

In [20]:
```
# unique values of our new columns
df.year.value_counts()
```

Out[20]:
```
2018    12043
Name: year, dtype: int64
```

In [21]:
```
# because all the transactions are from 2018, I will drop the year columns
df.drop(['year'], axis = 1 , inplace = True)
```

In [22]:
```
# it looks like transactions happened only in August, September, and October
df.month.value_counts()
```

Out[22]:
```
October      4087
September    4013
August       3943
Name: month, dtype: int64
```

In [23]:
```
# unique values of txn_description (mode of transaction)
df.txn_description.value_counts()
```

Out[23]:
```
SALES-POS     3934
POS           3783
PAYMENT       2600
PAY/SALARY     883
INTER BANK     742
PHONE BANK     101
Name: txn_description, dtype: int64
```

In [24]:
```
# Check the distribution of the dataset by gender
df.gender.value_counts()
```

Out[24]:
```
M    6285
F    5758
Name: gender, dtype: int64
```

```
In [25]: # this is a quick summary of our categorical variables showing the number of unique v
         # column with the most frequent value in it and its frequency.
         df.describe(exclude = [np.number]).loc[['unique','top' ,'freq']]
```

```
Out[25]:             status         account       long_lat txn_description  \
         unique           2             100            100               6
         top     authorized  ACC-1598451071  153.41 -27.95        SALES-POS
         freq          7717             578            578            3934

                                         merchant_id first_name                 date  \
         unique                                 5725         80                   91
         top     106e1272-44ab-4dcb-a438-dd98e0071e51    Michael  2018-09-28 00:00:00
         freq                                     14        746                  174

                gender merchant_suburb merchant_state           extraction  \
         unique      2            1609              8                 9442
         top         M       Melbourne            NSW  2018-09-03 11:00:00
         freq     6285             255           2169                   10

                                   transaction_id     customer_id merchant_long_lat  \
         unique                             12043             100              2703
         top     aa78c28d430240d49bfed5aa4a8bb42f  CUS-2487424745     151.21 -33.87
         freq                                   1             578               145

                movement      day    month
         unique        2        7        3
         top       debit   Friday  October
         freq      11160     2073     4087
```

transaction_id has 12043 unique values. it won't be significant in my analysis so I will drop it. there is 100 customers in the dataset. However, there are only 80 unique first names. I decided to drop the first name columns because the customer_id is more accurate. because each customer has a unique account. I will only keep the customer_id column and drop account column. for the location based analysis, I will rely merchat_state. I will drop long_lat, merchant_suburb, and merchant_long_lat. there are 5725 merchants. the highest number of transactions by merchant is 14 only, meaning the merchant_id won't significantly have an impact on the amount of the transaction. I will drop for now.

```
In [26]: df.drop(['transaction_id', 'account', 'first_name', 'long_lat','merchant_id','merchan
```

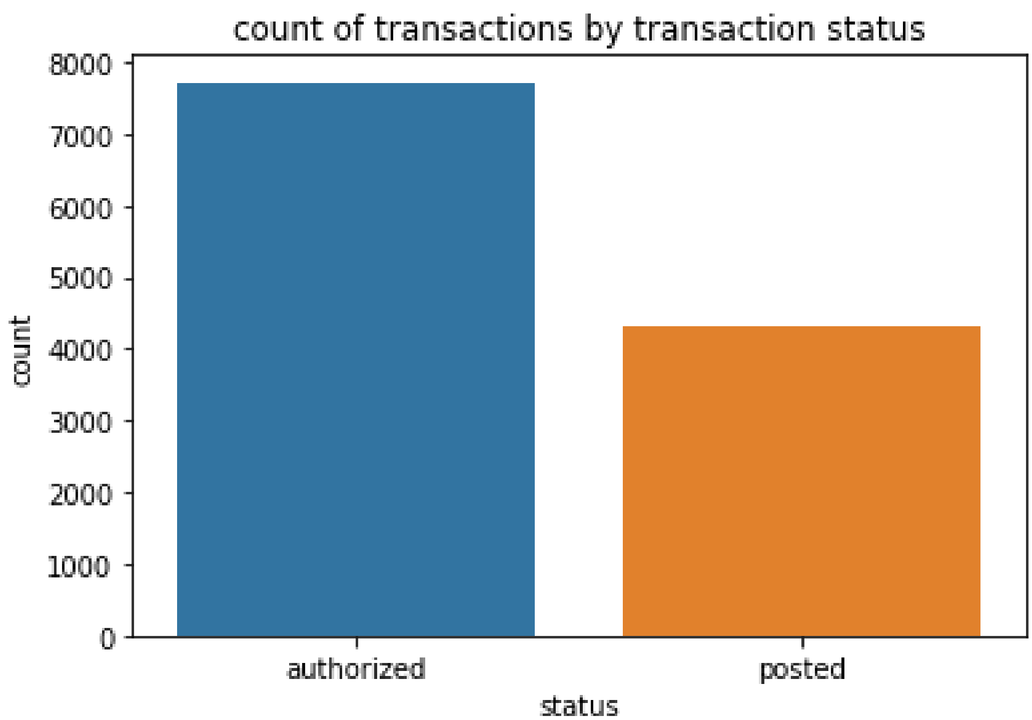
```
In [27]: #final dataset

         df.head()
```

```
Out[27]:        status  card_present_flag txn_description  balance       date gender  \
         0  authorized                1.0             POS    35.39 2018-08-01      F
         1  authorized                0.0       SALES-POS    21.20 2018-08-01      F
         2  authorized                1.0             POS     5.71 2018-08-01      M
         3  authorized                1.0       SALES-POS  2117.22 2018-08-01      F
```

```
4  authorized                1.0       SALES-POS    17.95 2018-08-01      F

   age merchant_state          extraction  amount     customer_id movement  \
0   26            QLD 2018-08-01 01:01:15   16.25  CUS-2487424745    debit
1   26            NSW 2018-08-01 01:13:45   14.19  CUS-2487424745    debit
2   38            NSW 2018-08-01 01:26:15    6.42  CUS-2142601169    debit
3   40            QLD 2018-08-01 01:38:45   40.90  CUS-1614226872    debit
4   26            QLD 2018-08-01 01:51:15    3.25  CUS-2487424745    debit

         day   month  hour
0  Wednesday  August     1
1  Wednesday  August     1
2  Wednesday  August     1
3  Wednesday  August     1
4  Wednesday  August     1
```

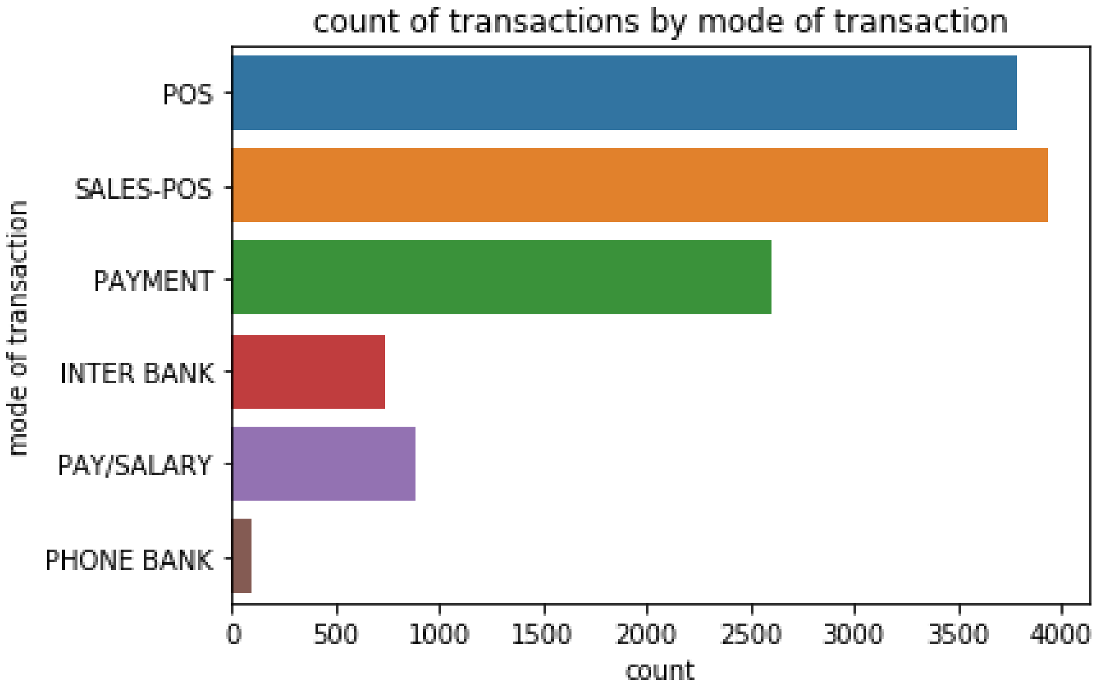
Univariate analysis First, a univariate analysis for my categorical variables.

In [28]:
```
# there are more authorized transactions than posted

sns.countplot(x = 'status', data = df)
plt.title('count of transactions by transaction status');
```

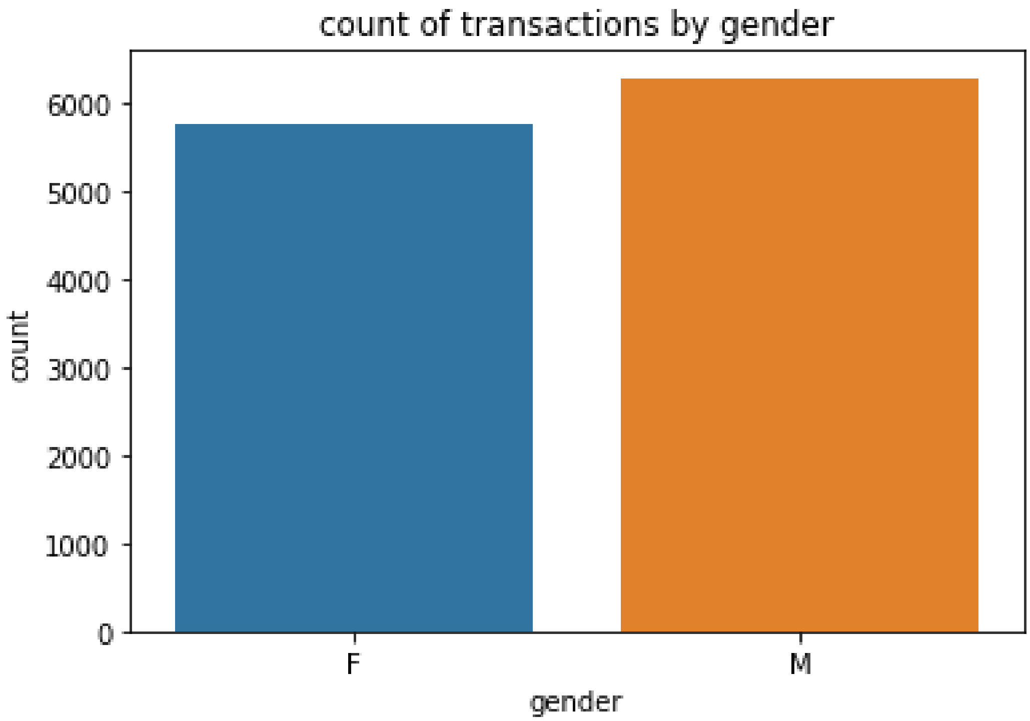


In [29]:
```
# POS and SALES-POS are the most used modes of transaction
```

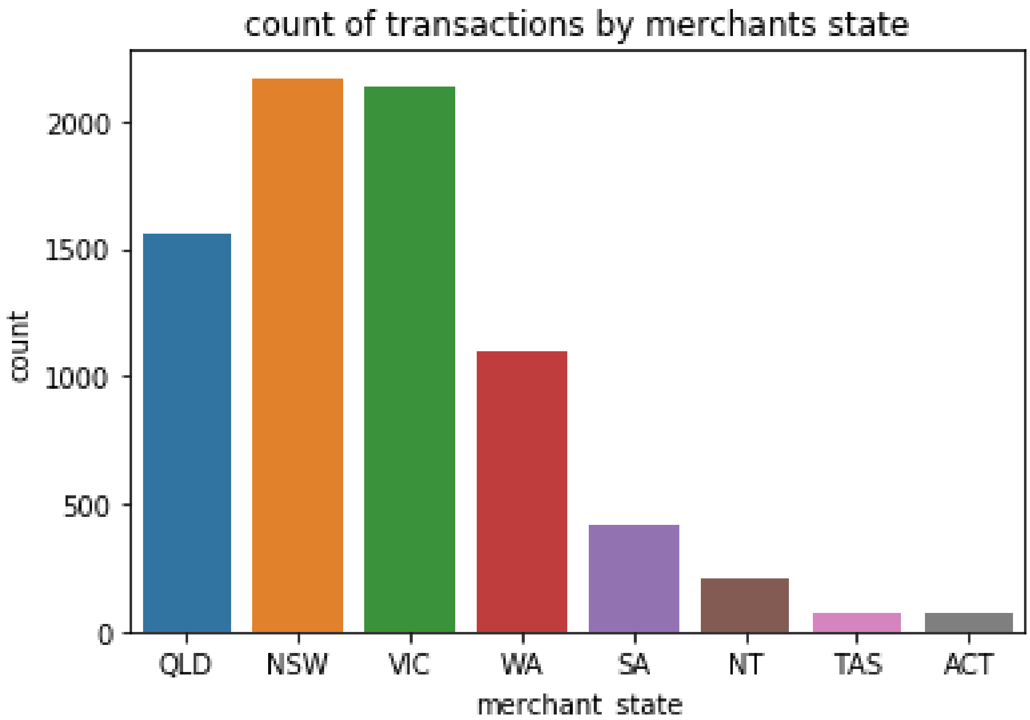
```
sns.countplot(y = 'txn_description', data = df)
plt.title('count of transactions by mode of transaction')
plt.ylabel('mode of transaction');
```



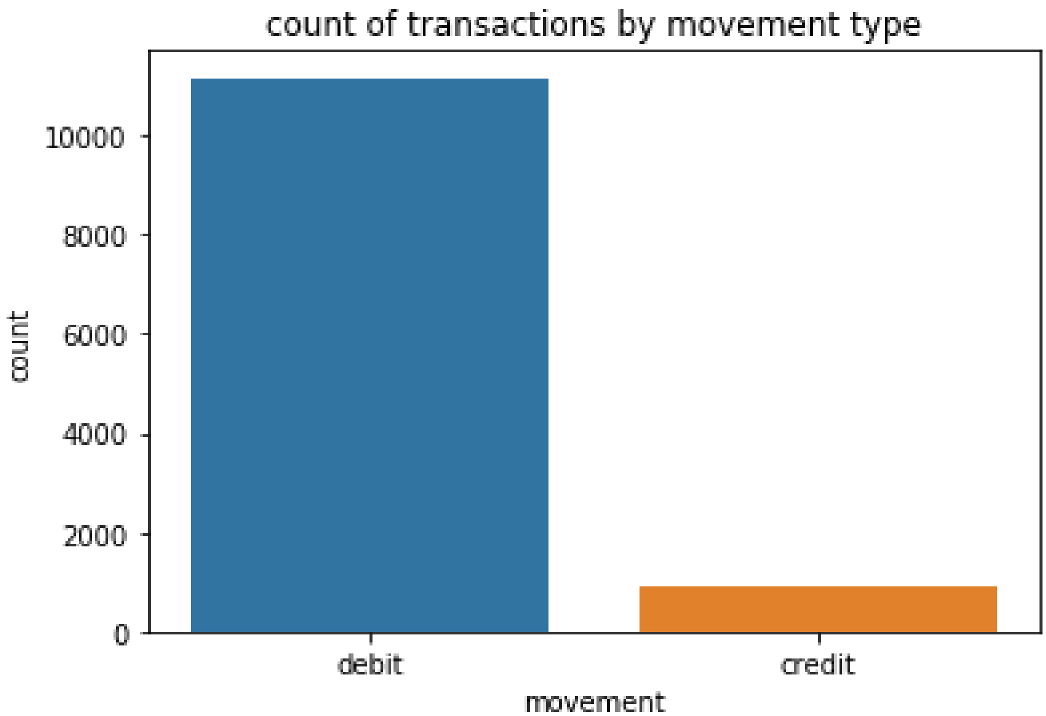

```
In [30]: # there are more males than females

         sns.countplot(x = 'gender', data = df)
         plt.title('count of transactions by gender');
```



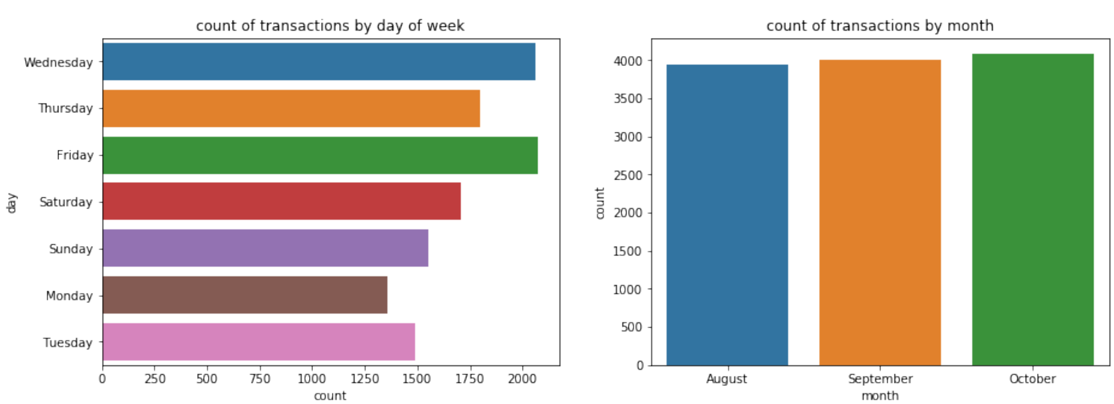
In [31]:
```
# NSW and VIC are the states with the most transactions

sns.countplot(x = 'merchant_state', data = df)
plt.title('count of transactions by merchants state');
```




```
In [32]: # there are much more debit transactions than credit ones.

         sns.countplot(x = 'movement', data = df)
         plt.title('count of transactions by movement type');
```

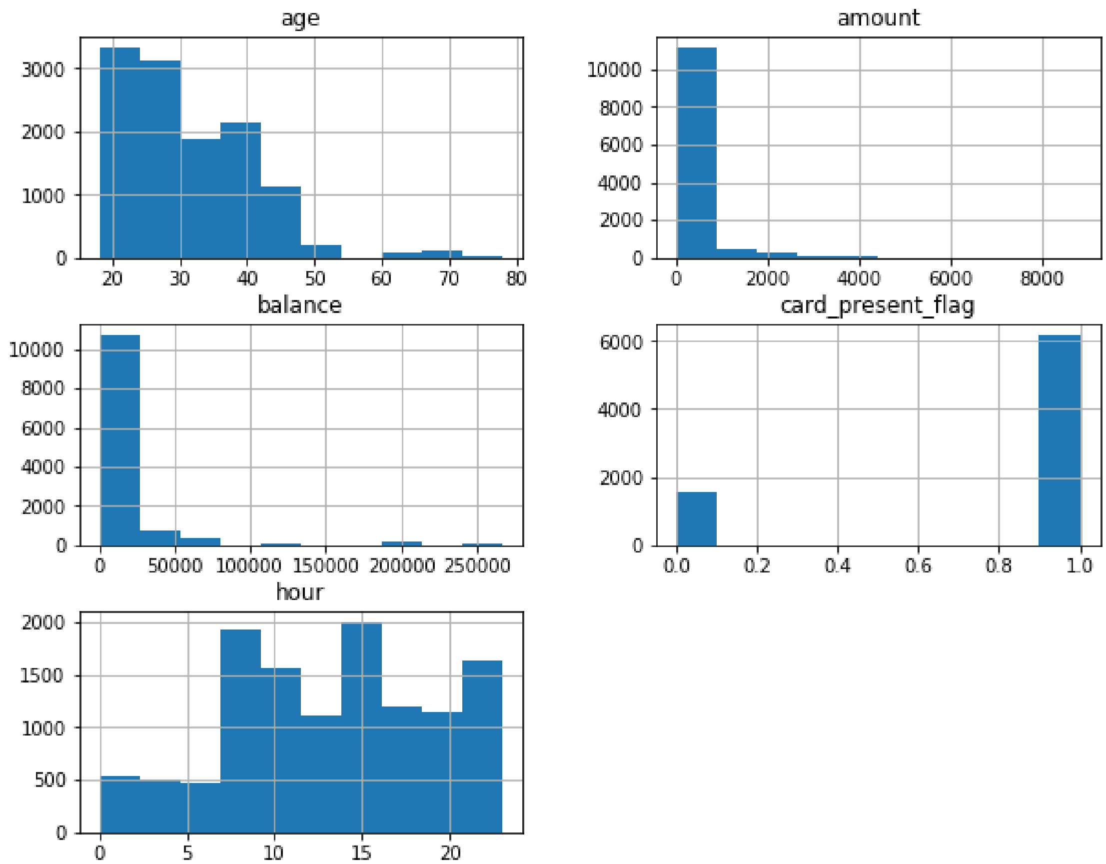


In [33]:
```python
plt.figure(figsize=[15, 5])

# the left plot
plt.subplot(1,2,1)
sns.countplot(y = 'day', data = df)
plt.title('count of transactions by day of week')

# the right plot
plt.subplot(1,2,2)
sns.countplot(x = 'month', data = df)
plt.title('count of transactions by month');
```



Wednesday and Friday are the days with the most number of transactions. Monday scored the lowest number of transactions the difference in the number of transactions between the months is not significants

```
In [34]: # these are histograms of my numeric variables.
         df.hist(figsize=(10,8));
```




```
In [35]: df.describe(percentiles = [0.75, 0.90, 0.95, 0.99])
```

Out[35]:

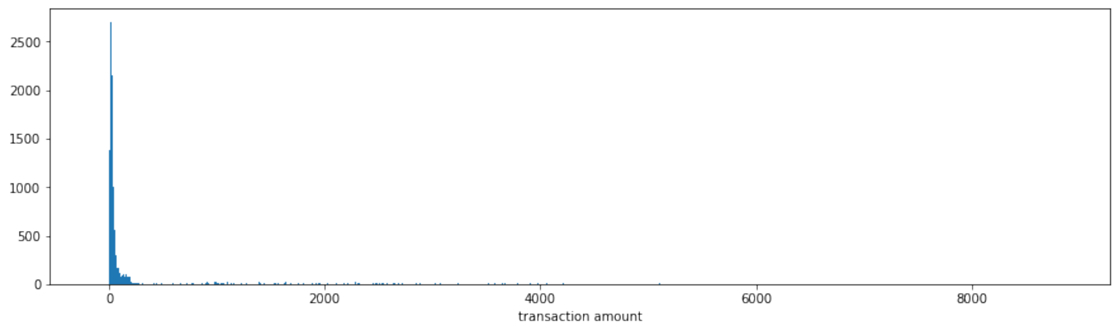
	card_present_flag	balance	age	amount \
count	7717.000000	12043.000000	12043.000000	12043.000000
mean	0.802644	14704.195553	30.582330	187.933588
std	0.398029	31503.722652	10.046343	592.599934
min	0.000000	0.240000	18.000000	0.100000
50%	1.000000	6432.010000	28.000000	29.000000
75%	1.000000	12465.945000	38.000000	53.655000
90%	1.000000	29442.384000	43.000000	186.000000
95%	1.000000	53362.930000	46.000000	1158.150000

```
99%              1.000000   201963.445200     69.000000    3195.010000
max              1.000000   267128.520000     78.000000    8835.980000

               hour
count  12043.000000
mean      13.268621
std        5.777284
min        0.000000
50%       13.000000
75%       18.000000
90%       21.000000
95%       22.000000
99%       23.000000
max       23.000000
```

There seem to be some outliers in amount, age, and balance columns.

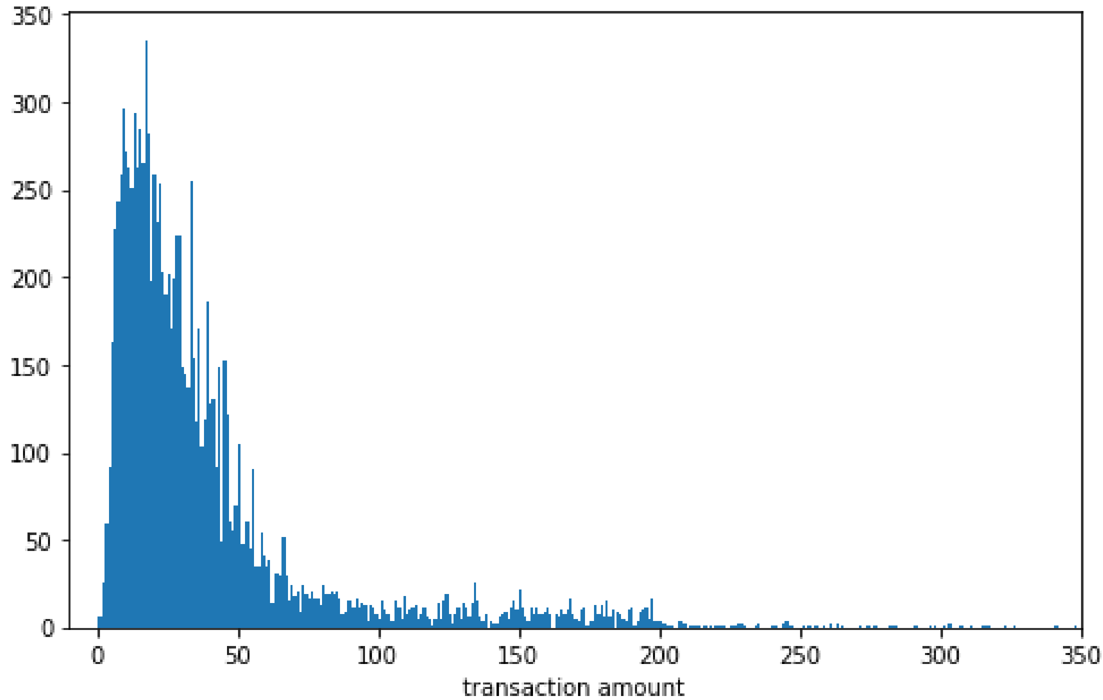
4 Transaction amount analysis

In [36]:
```
plt.figure(figsize=[15,4])
binsize = 10
bins = np.arange(-100, df['amount'].max()+binsize, binsize)
plt.hist(data = df, x = 'amount', bins = bins)
plt.xlabel('transaction amount');
```



Getting rid of the outliers by limiting the x axis

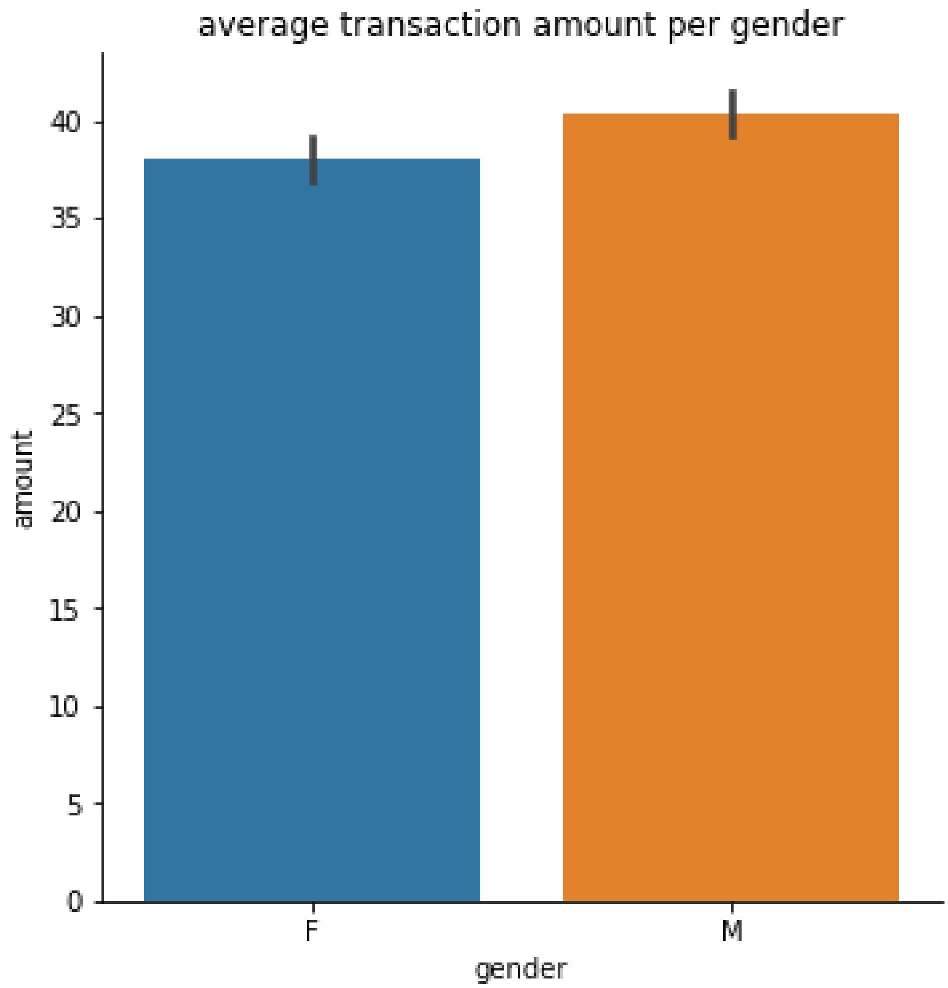
In [37]:
```
plt.figure(figsize=[8, 5])

bin_edges = np.arange(-10, 350+1,1)
plt.hist(data = df, x = 'amount', bins = bin_edges)
plt.xlim(-10, 350)
plt.xlabel('transaction amount');
```



the transaction amount looks normally distributed but long-tailed due to outliers. But with the use of axis limits, we are able to see the normal distribution clearly

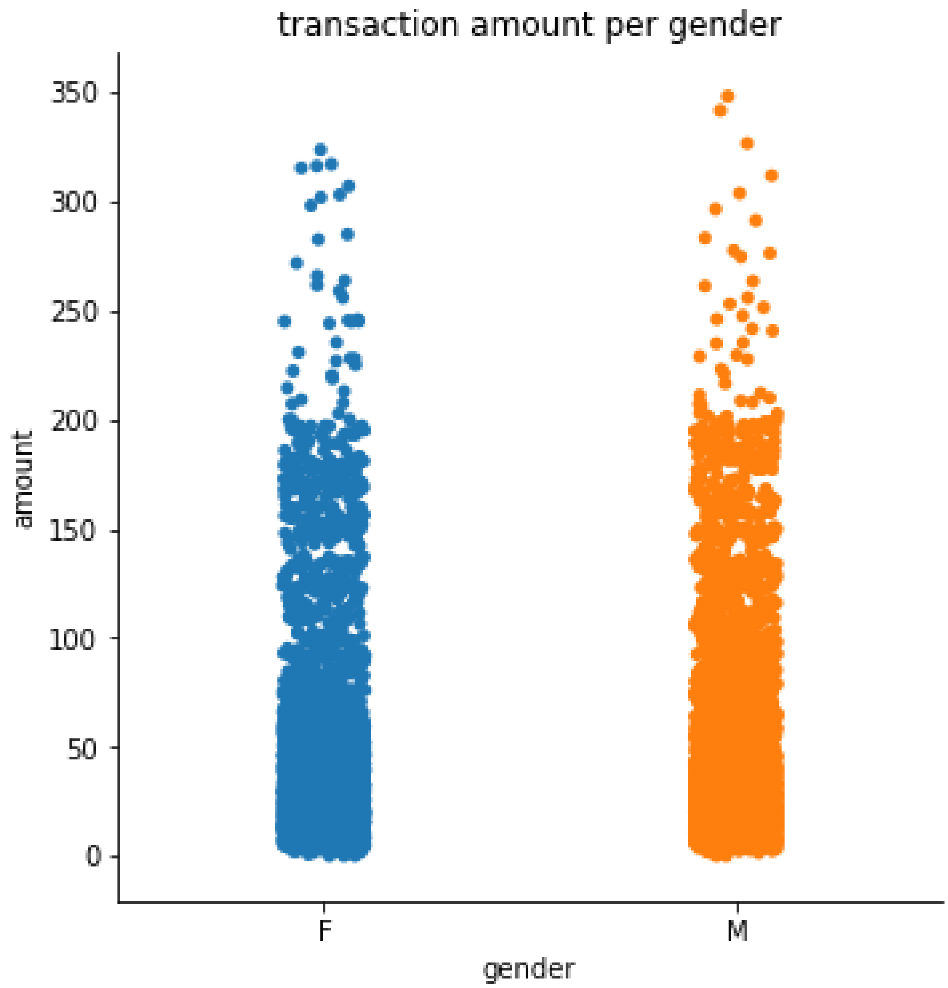
Amount by gender

```
In [38]: sns.catplot(x="gender", y="amount", kind="bar", data=df.query('amount <= 350'))
         plt.title('average transaction amount per gender');
```



this is a barplot showing the average amount of transactions for each gender. with a confidence interval displayed as error bars.

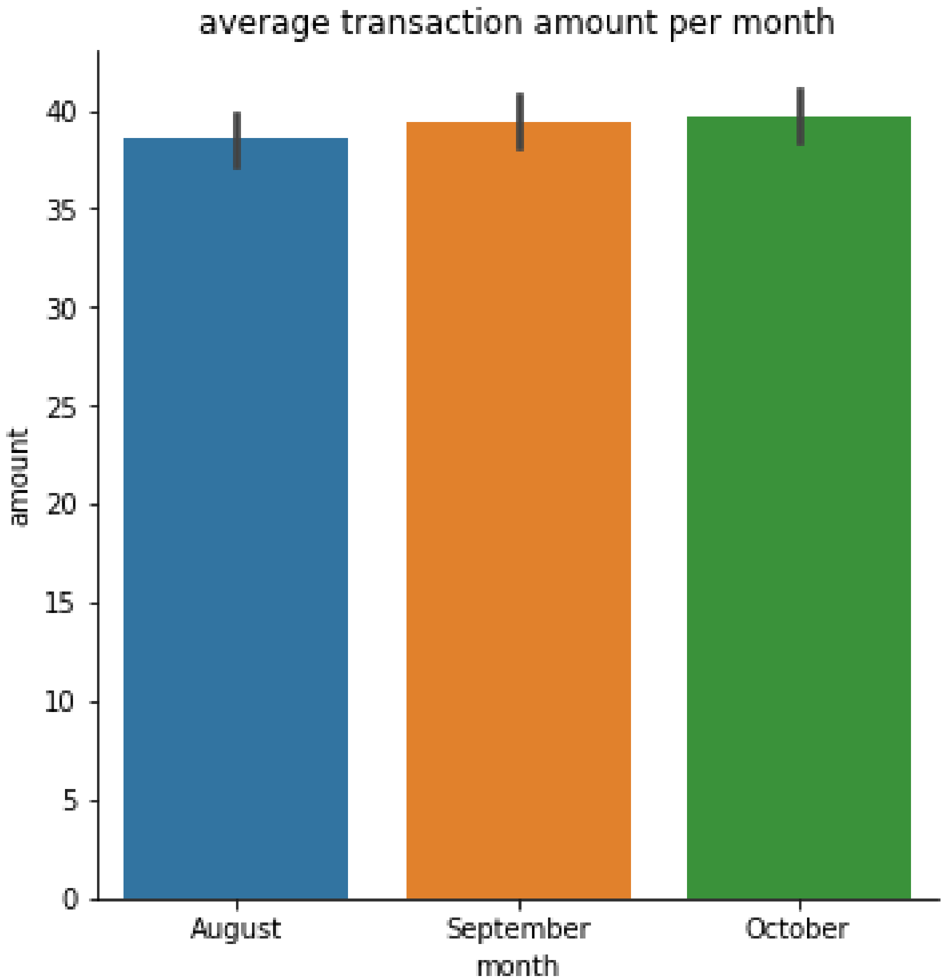
```
In [39]: sns.catplot(x="gender", y = 'amount', data=df.query('amount <= 350'))
         plt.title('transaction amount per gender');
```

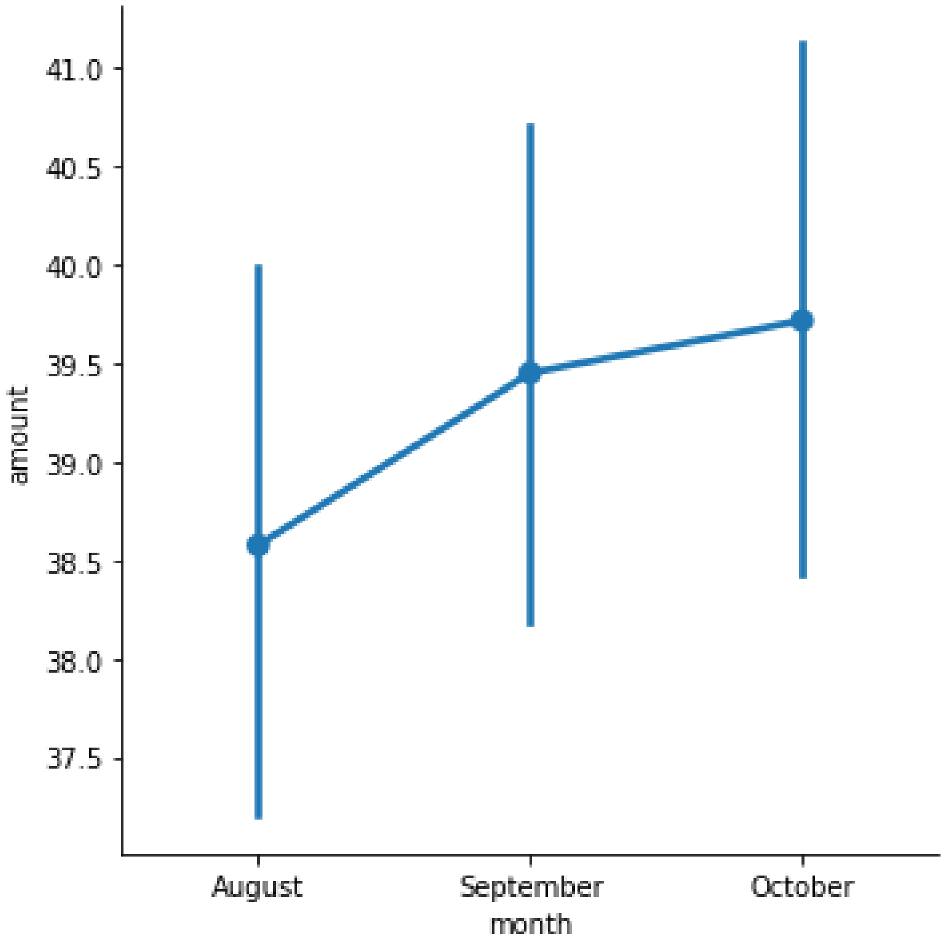


this is a categorical scatterplot showing the transaction amount for each gender.
the highest amounts are made by men the amount of transactions made by men are higher on average.

5 Amount by month

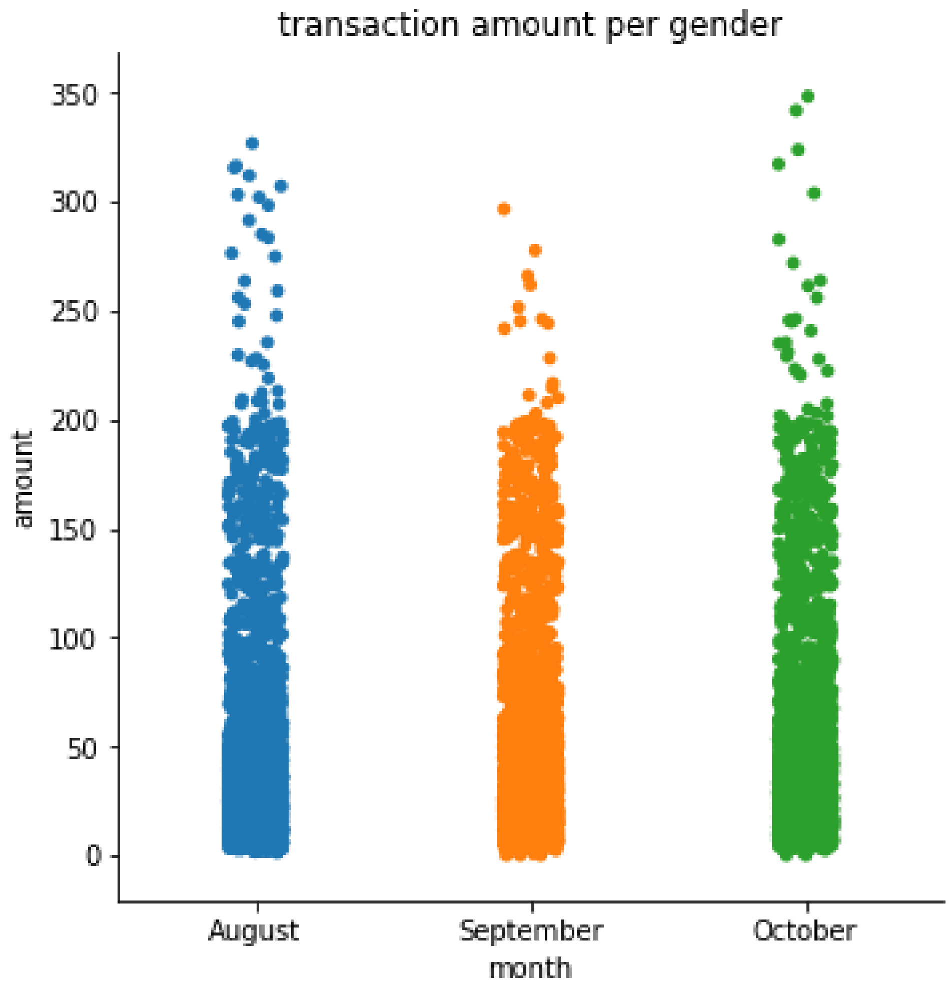
In [40]:
```
sns.catplot(x="month", y="amount", kind="bar", data=df.query('amount <= 350'))
plt.title('average transaction amount per month')
sns.catplot(x="month", y="amount", kind="point", data=df.query('amount <= 350'));
```


average transaction amount per month
amount
40
35
30
25
20
15
10
5
0
August
September
October
month



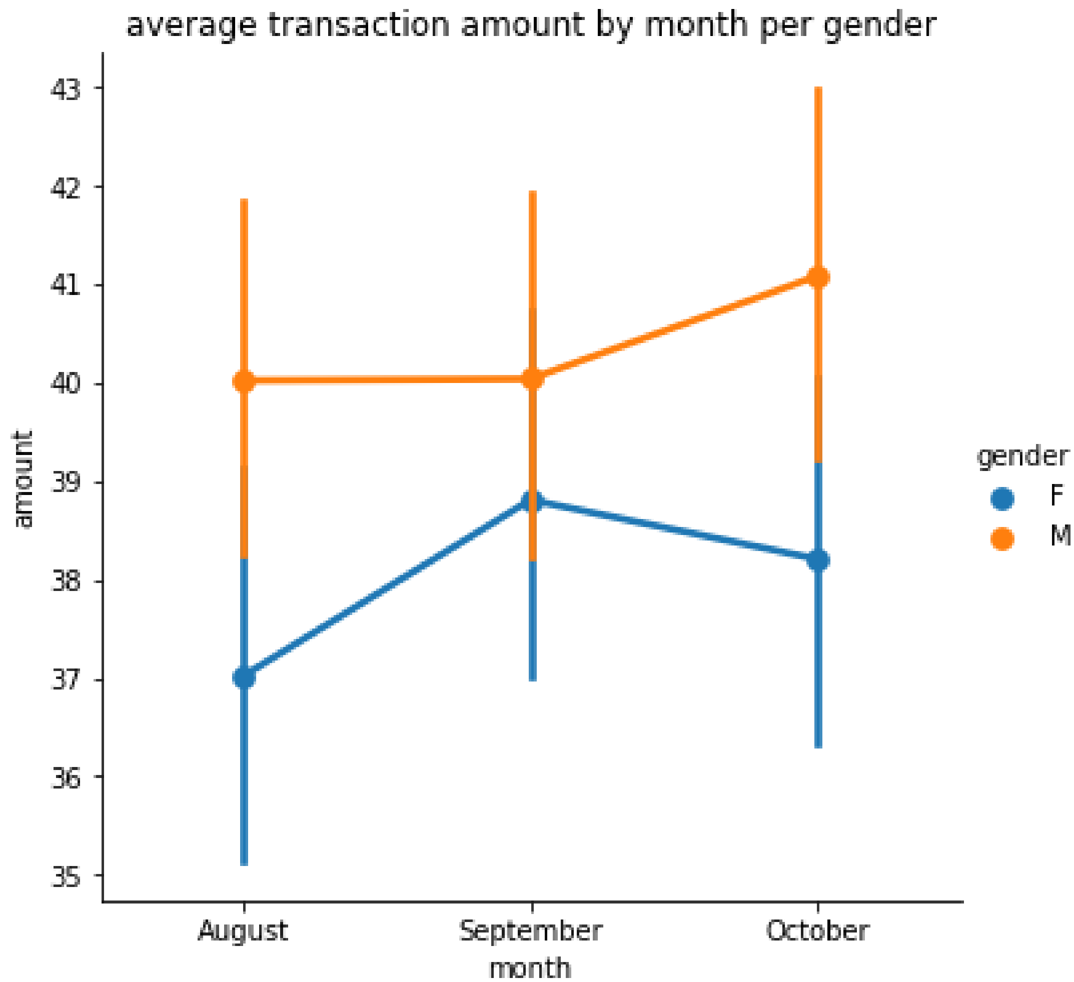
October has higher transaction amounts than Auguest and September.

```
In [41]: sns.catplot(x="month", y = 'amount', data=df.query('amount <= 350'))
         plt.title('transaction amount per gender');
```



outliers in the transaction amount happen in all three months.

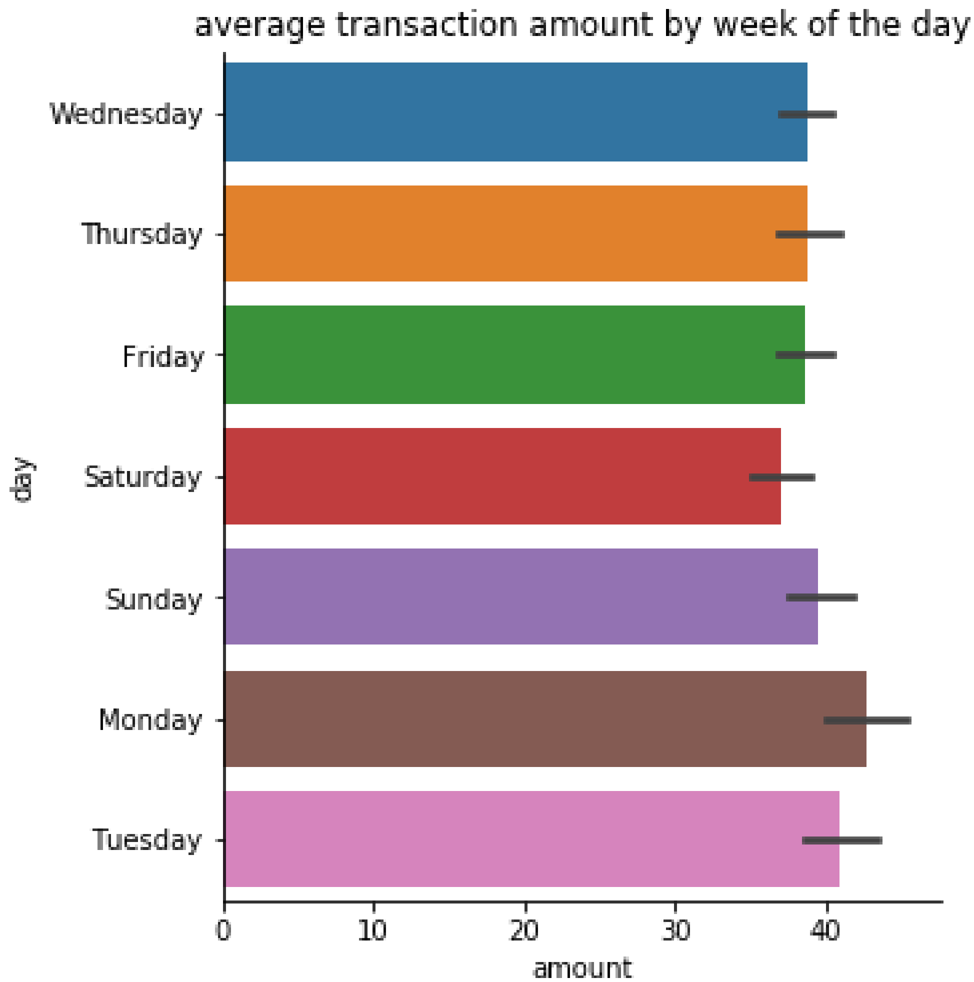
```
In [42]: sns.catplot(x="month", y="amount", hue="gender",kind="point", data=df.query('amount <=
         plt.title('average transaction amount by month per gender');
```

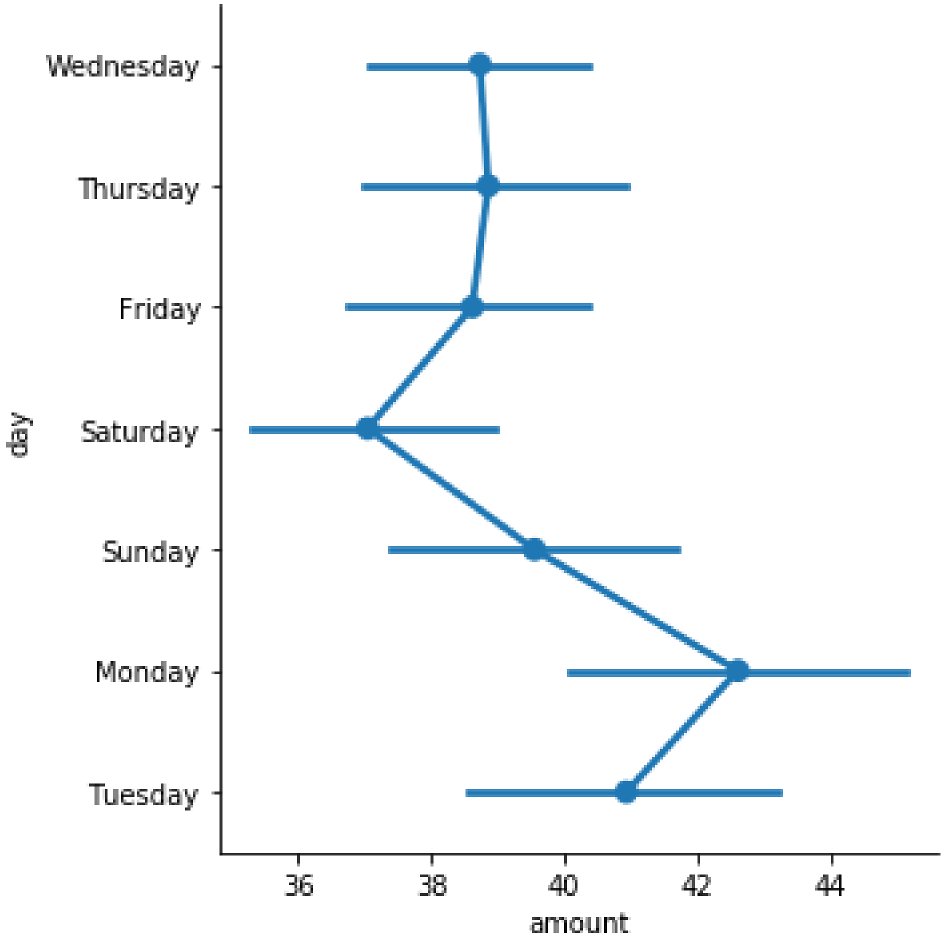


males have higher transaction amounts. October is the month with the highest transaction amounts for both genders.

6 Amount by day

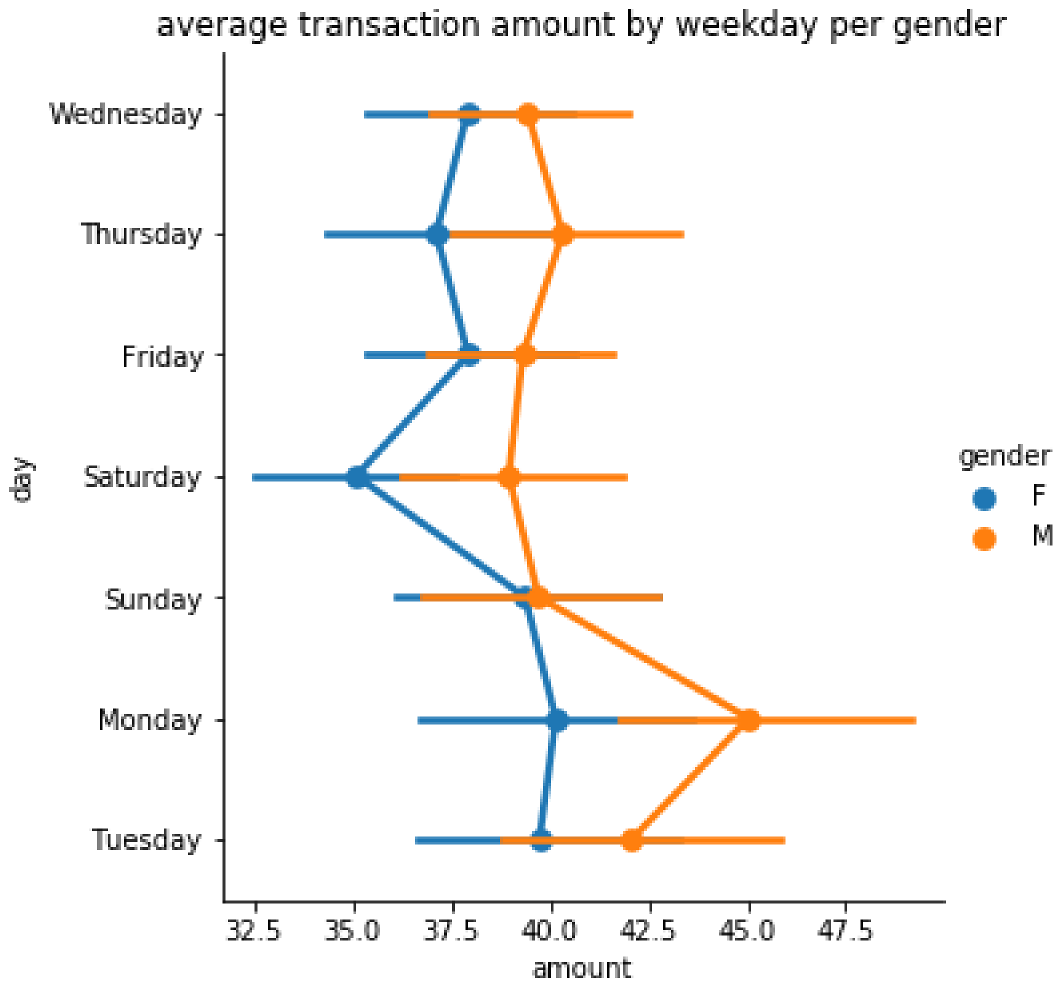
In [43]:
```
sns.catplot(y="day", x="amount", kind="bar", data=df.query('amount <= 350'))
plt.title('average transaction amount by week of the day')
sns.catplot(y="day", x="amount", kind="point", data=df.query('amount <= 350'));
```


average transaction amount by week of the day
Wednesday
Thursday
Friday
Saturday
Sunday
Monday
Tuesday
day
0
10
20
30
40
amount



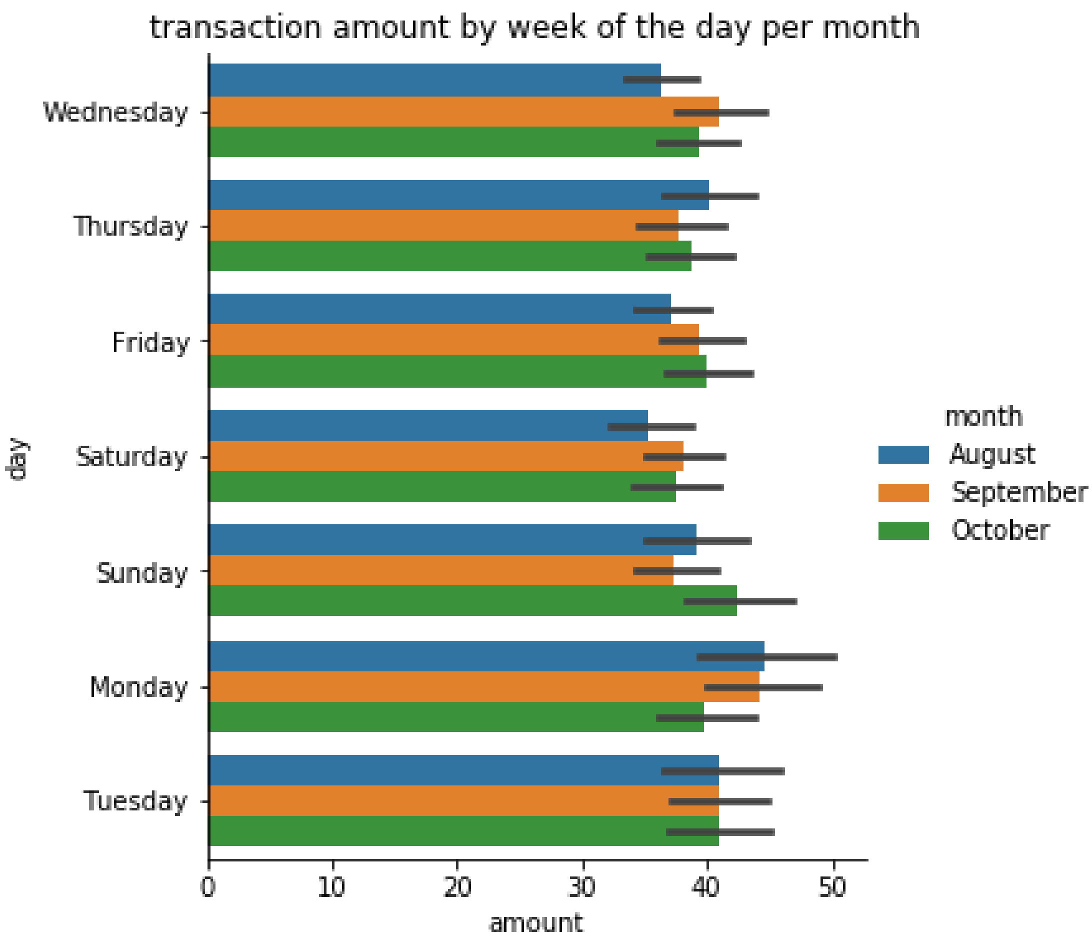
the amount of transactions is the lowest on weekend Saturday. the transactions with the highest amounts happen mostly on Monday.

```
In [44]: sns.catplot(y="day", x="amount", hue="gender",kind="point", data=df.query('amount <= 3
         plt.title('average transaction amount by weekday per gender');
```



only on Sunday do women have almost the same amount of transaction as men.

```
In [45]: sns.catplot(y="day", x="amount",hue='month', kind="bar", data=df.query('amount <= 350
         plt.title('transaction amount by week of the day per month');
```



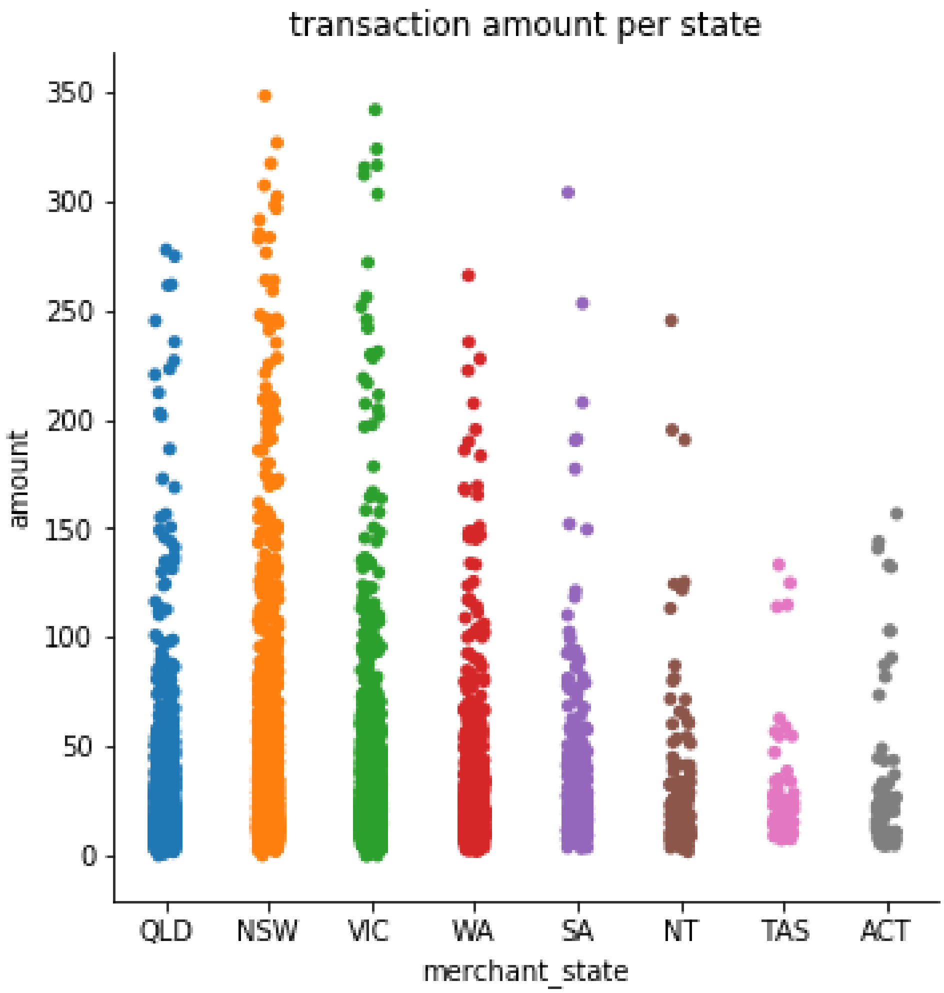
Amount by location

```
In [46]: # min and max amount of transactions in each state
         df.groupby('merchant_state')['amount'].agg(['min' , 'max'])
```

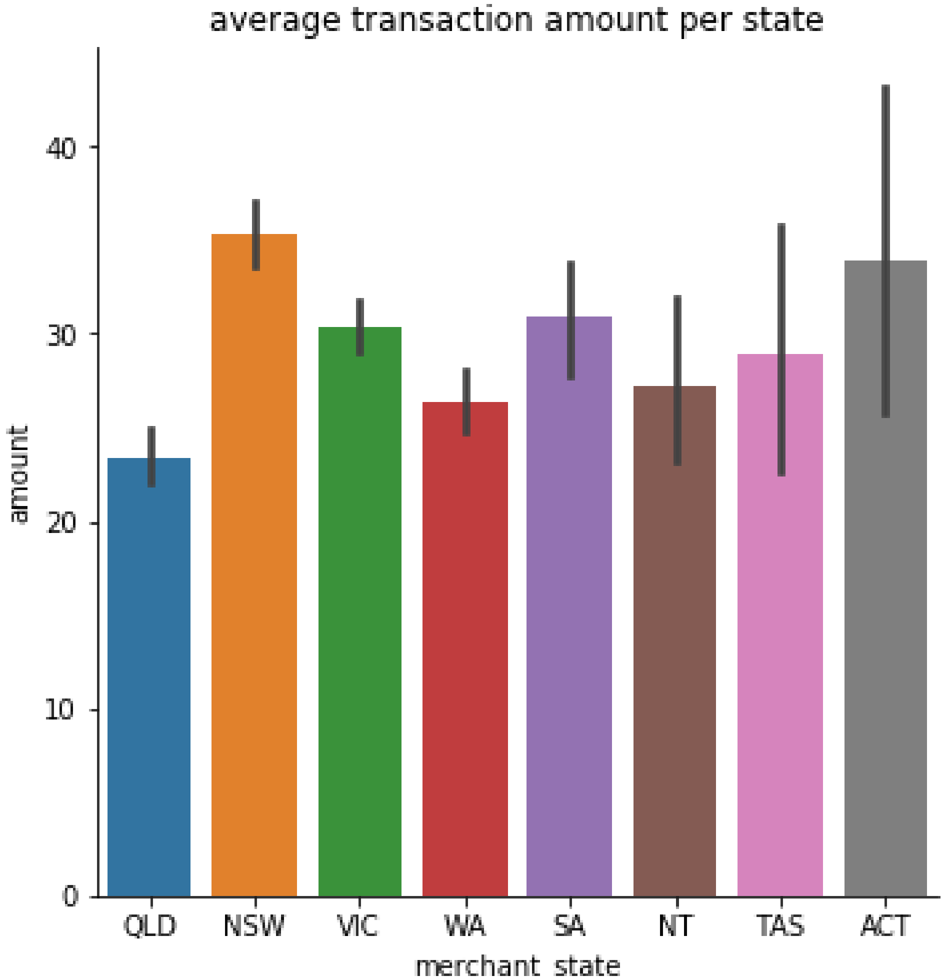
Out[46]:

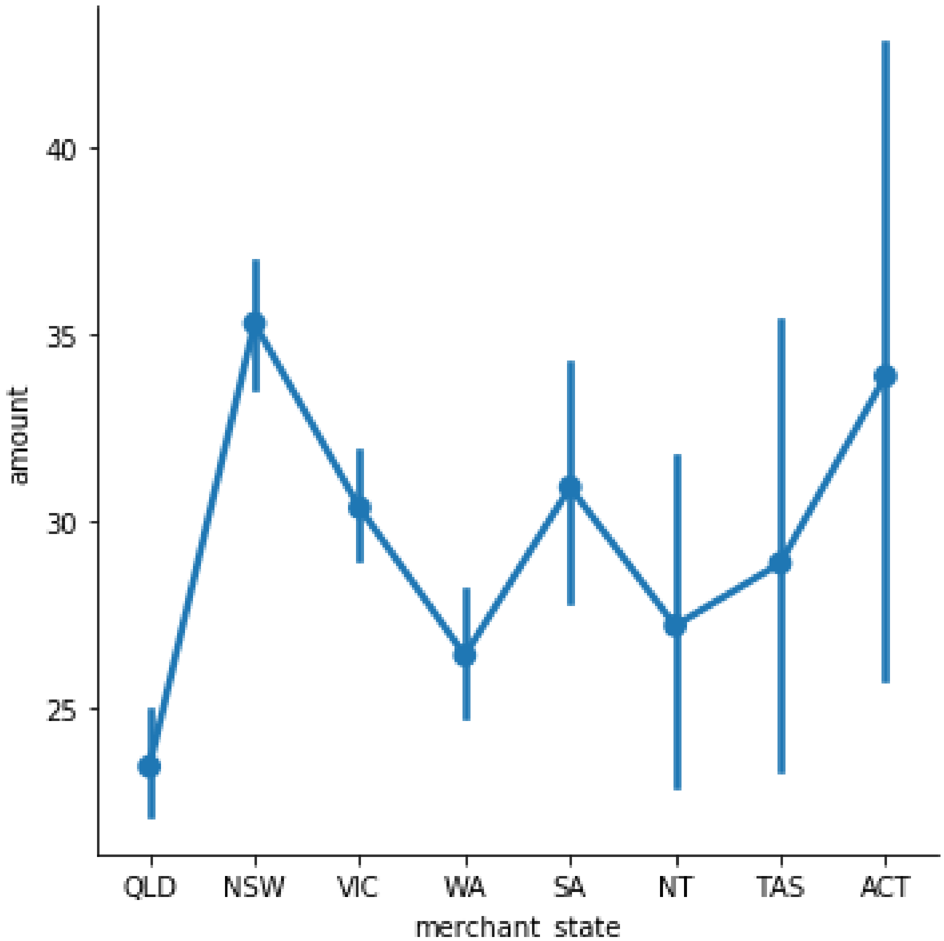
merchant_state	min	max
ACT	4.50	1348.47
NSW	0.10	4233.00
NT	1.71	1285.09
QLD	0.10	7081.09
SA	3.28	909.61
TAS	7.35	133.31
VIC	0.10	3680.71
WA	2.05	1692.56

```
In [47]: sns.catplot(x="merchant_state", y = 'amount', data=df.query('amount <= 350'))
         plt.title('transaction amount per state');
```

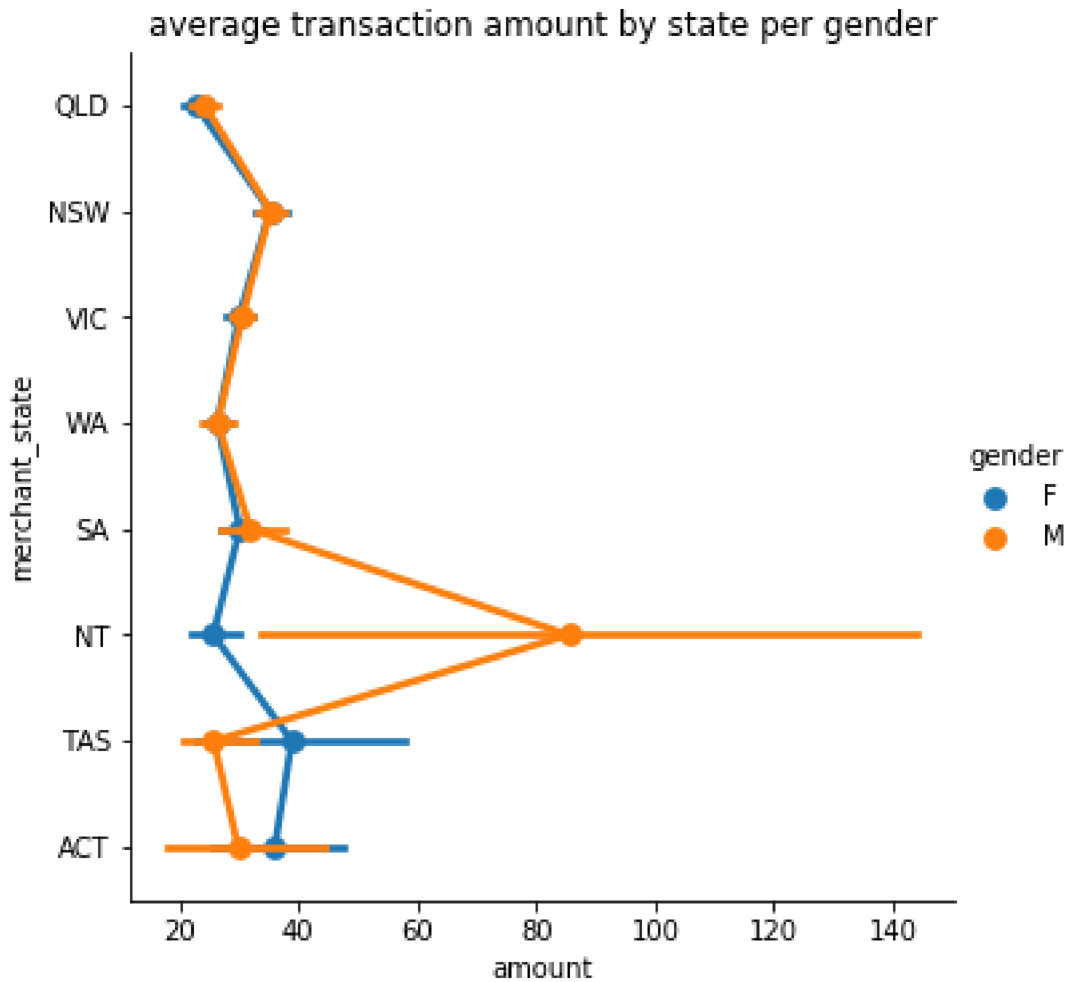


In [48]:
```
sns.catplot(x="merchant_state", y="amount", kind="bar", data=df.query('amount <= 350')
plt.title('average transaction amount per state')
sns.catplot(x="merchant_state", y="amount", kind="point", data=df.query('amount <= 350
```


average transaction amount per state
amount
0
10
20
30
40
QLD
NSW
VIC
WA
SA
NT
TAS
ACT
merchant_state




```
In [49]: sns.catplot(y="merchant_state", x="amount", hue="gender",kind="point", data=df.query(
         plt.title('average transaction amount by state per gender');
```

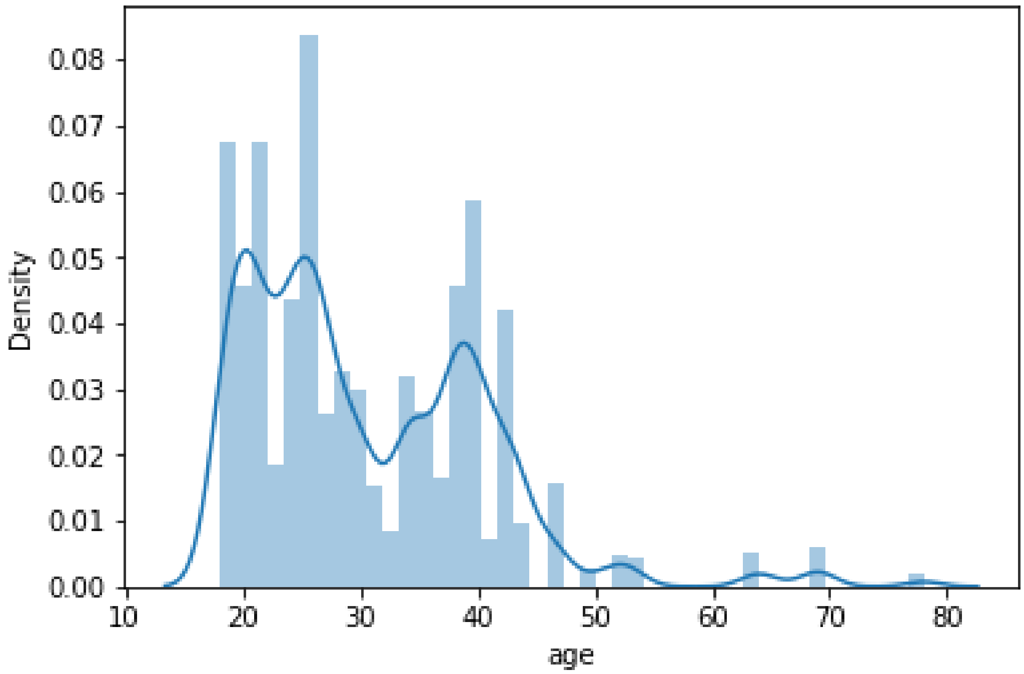


women's transaction amounts are higher than men in the state of TAS and ACT

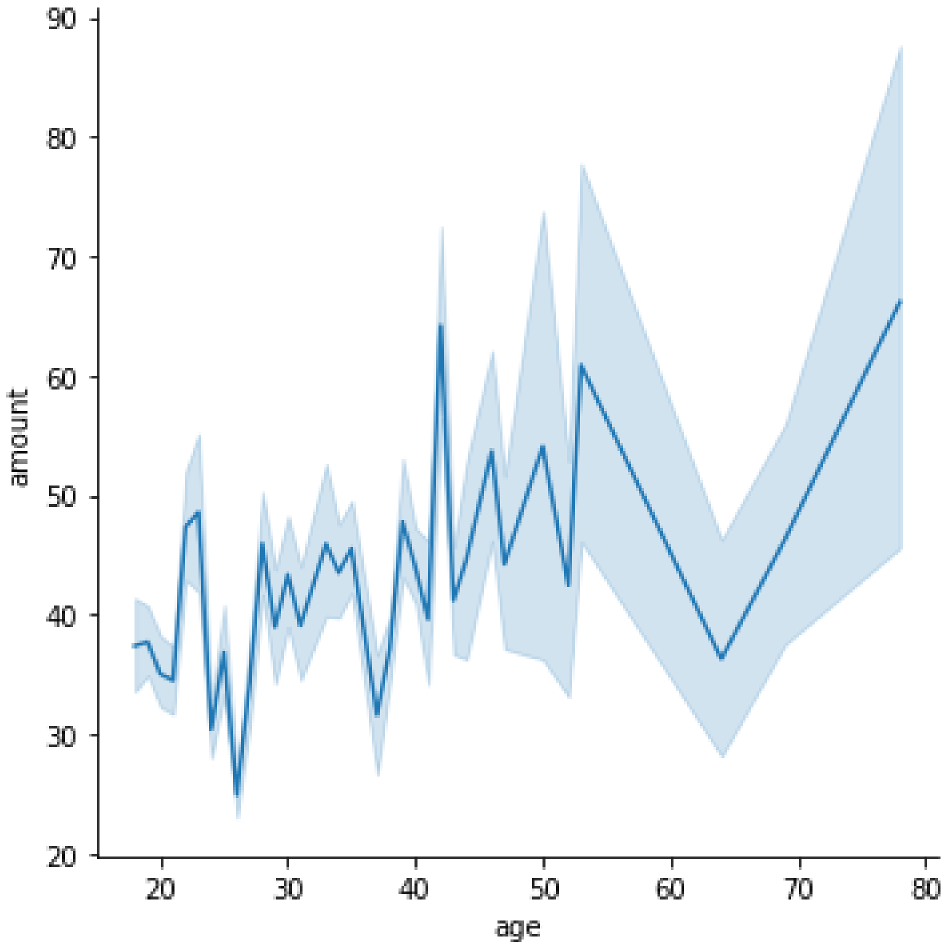
7 Amount by Age

```
In [50]: sns.distplot(df['age']);
```

```
C:\Users\hp\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplo
  warnings.warn(msg, FutureWarning)
```



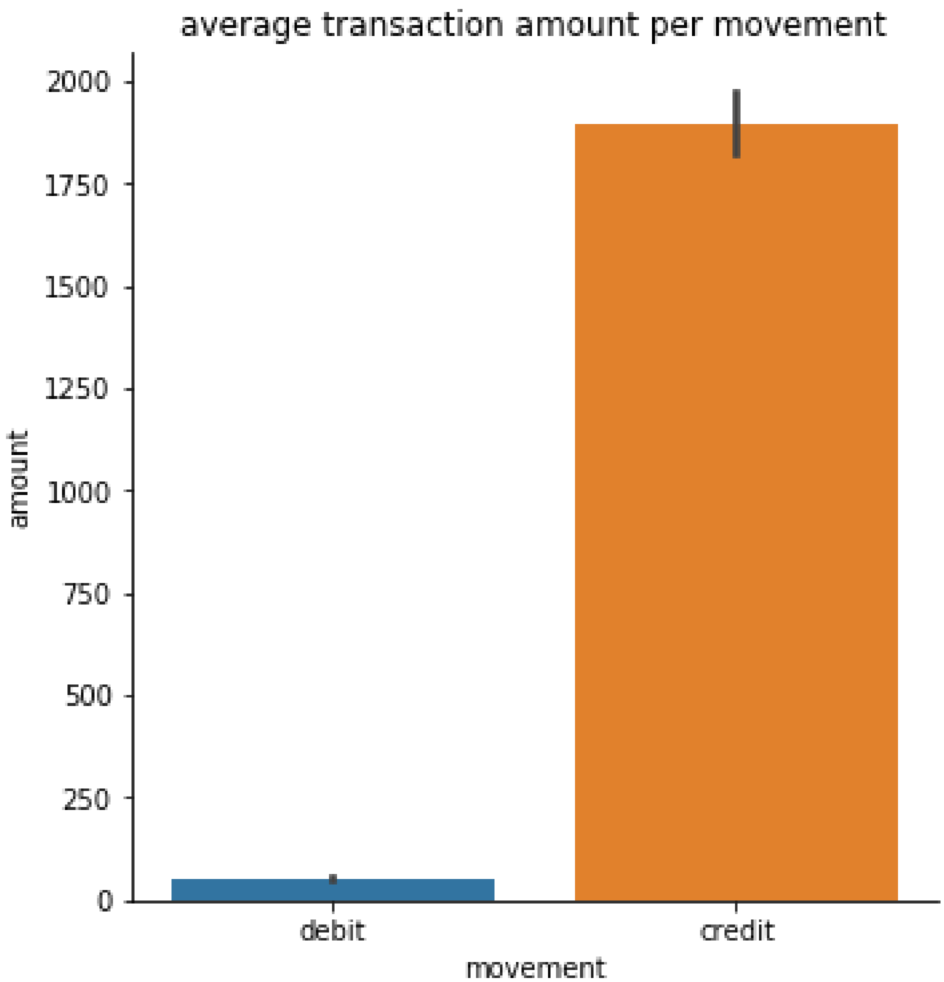

```
In [51]: sns.relplot(x='age' , y='amount', kind="line" , data=df.query('amount <= 350'));
```



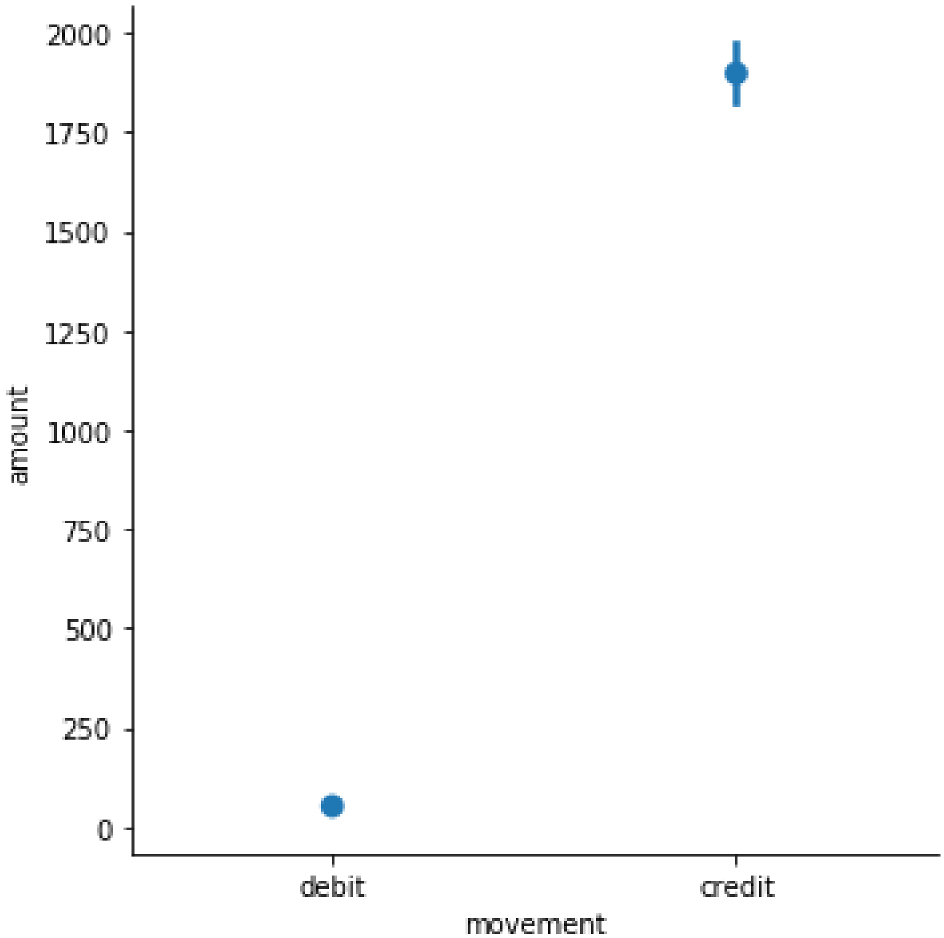
the highest amounts of transactions are made by customers of age between 40 and 45.

8 Amount by transaction type

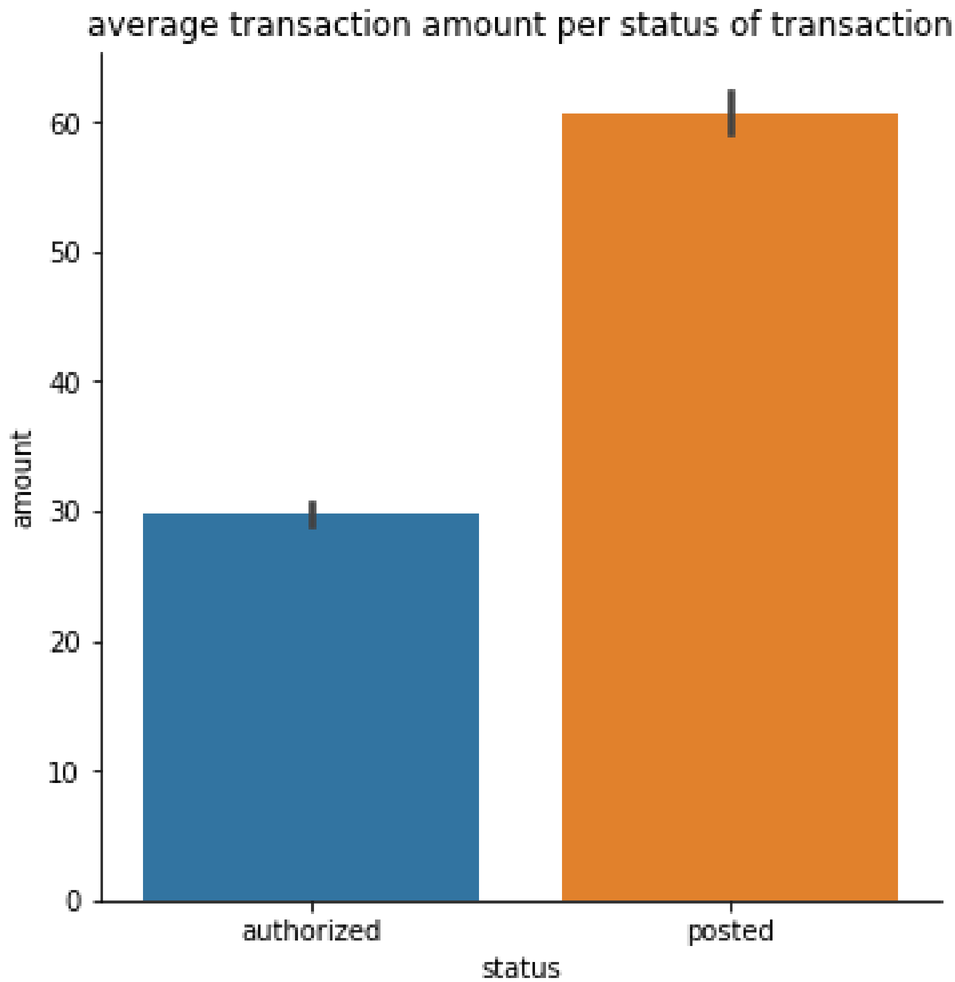
In [52]:
```
sns.catplot(x="movement", y="amount", kind="bar", data=df)
plt.title('average transaction amount per movement')
sns.catplot(x="movement", y="amount", kind="point", data=df, join=False);
```



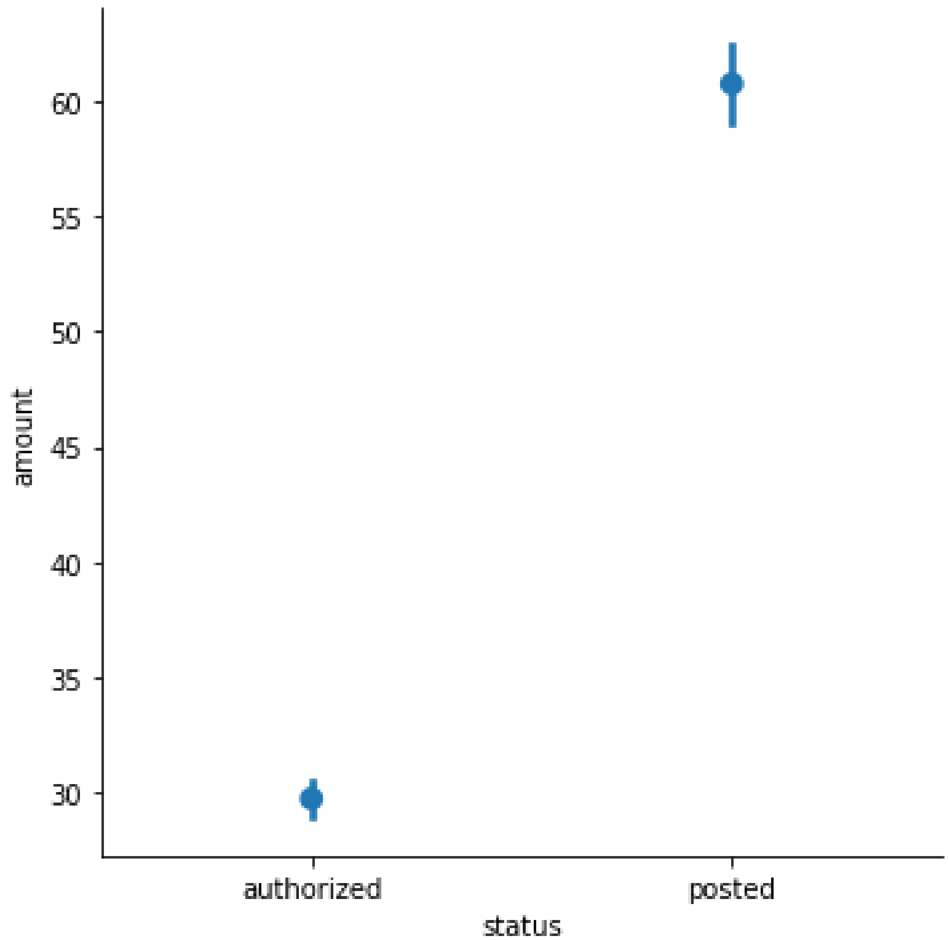
average transaction amount per movement
2000
1750
1500
1250
1000
750
500
250
0
amount
debit
credit
movement




```
In [53]: sns.catplot(x="status", y="amount", kind="bar", data=df.query('amount <= 350') )
         plt.title('average transaction amount per status of transaction')
         sns.catplot(x="status", y="amount", kind="point", data=df.query('amount <= 350'), joi
```

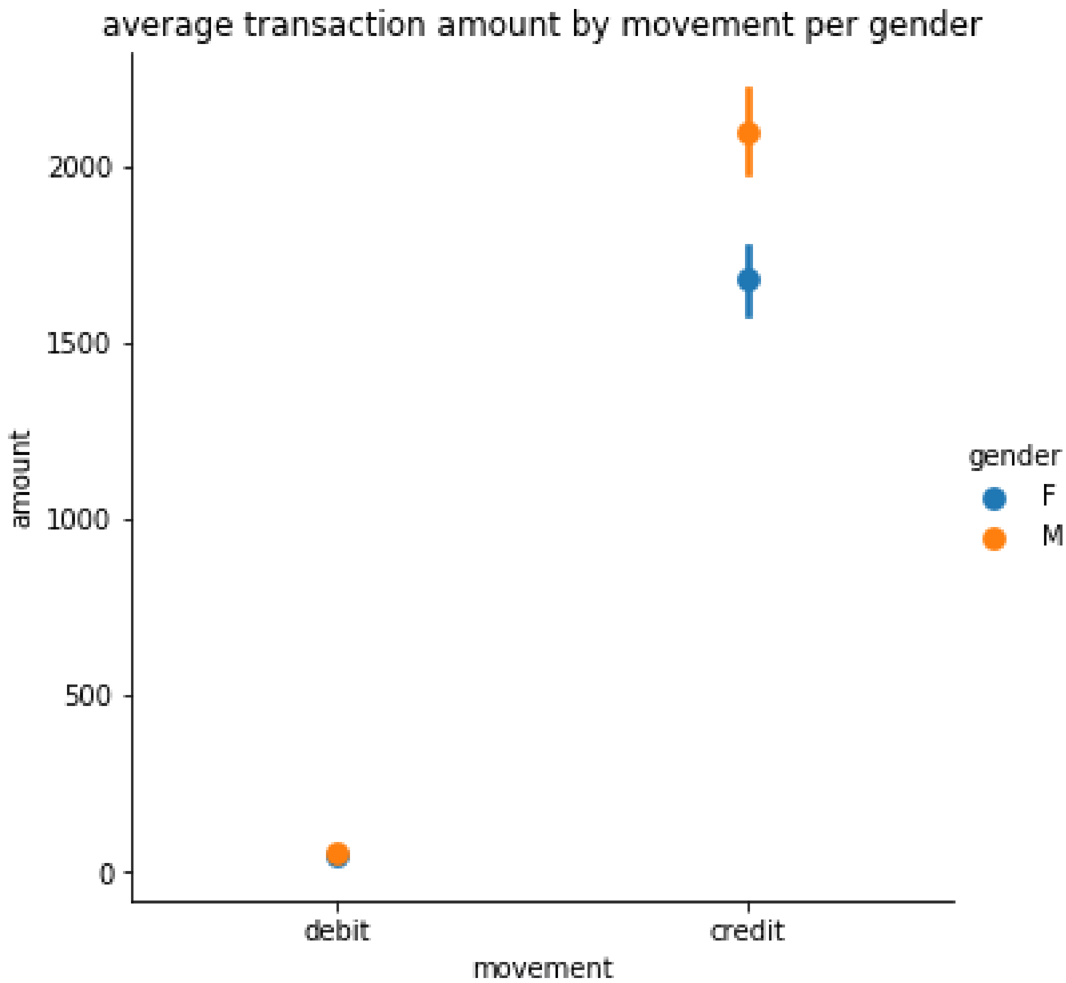


average transaction amount per status of transaction
amount
0
10
20
30
40
50
60
authorized
posted
status

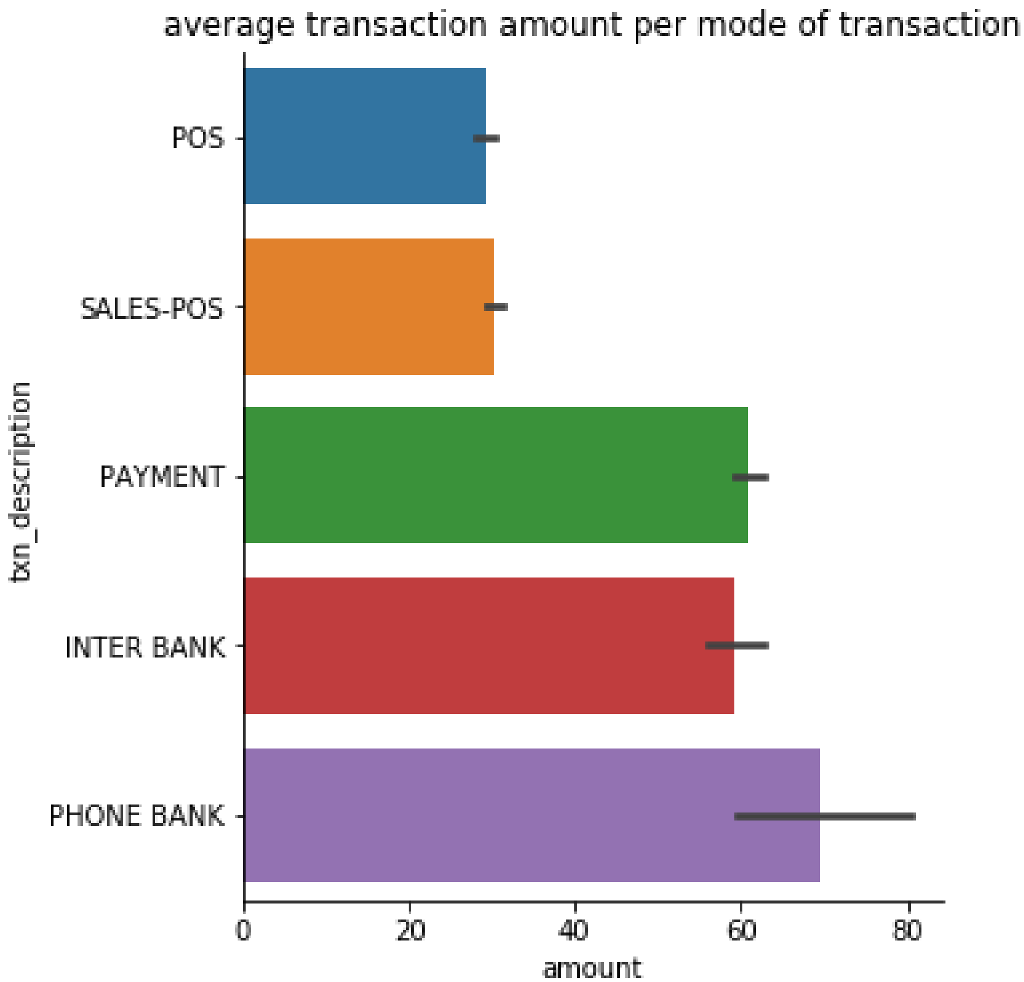


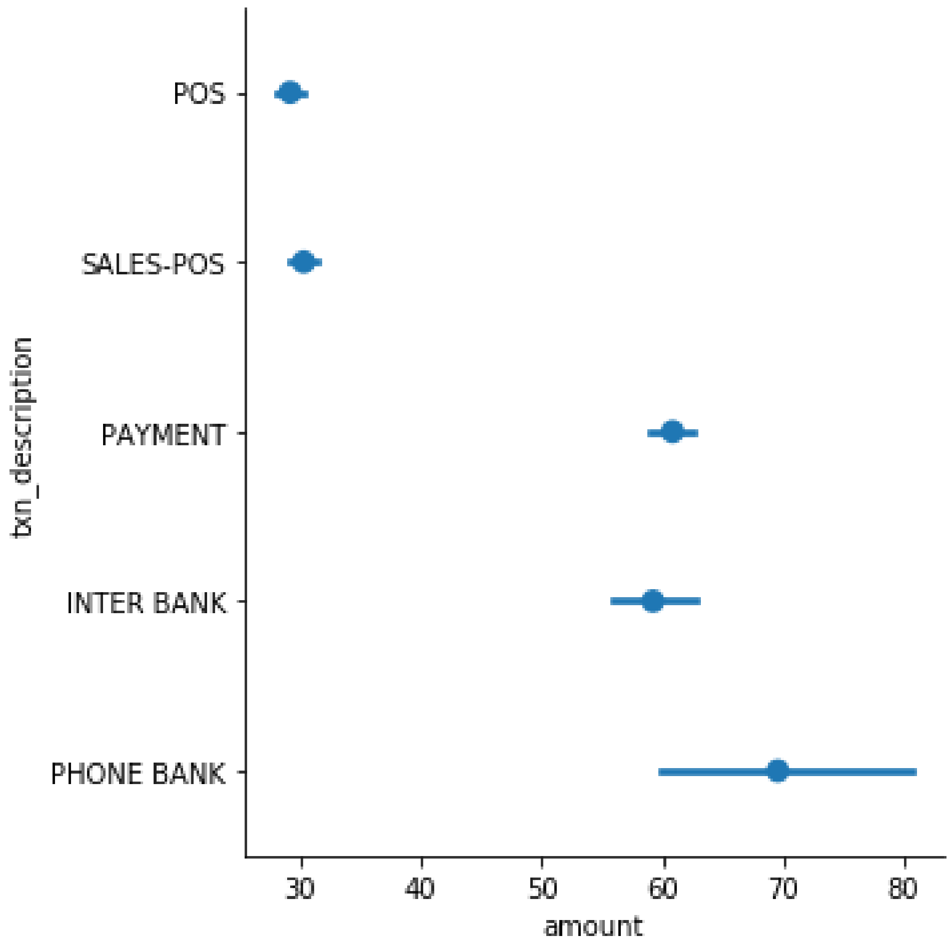
small amount transactions are debit and authorised transactions. high amount transactions are credit and posted transactions.

```
In [54]: sns.catplot(x="movement", y="amount", hue="gender",kind="point", data=df, join=False)
         plt.title('average transaction amount by movement per gender');
```



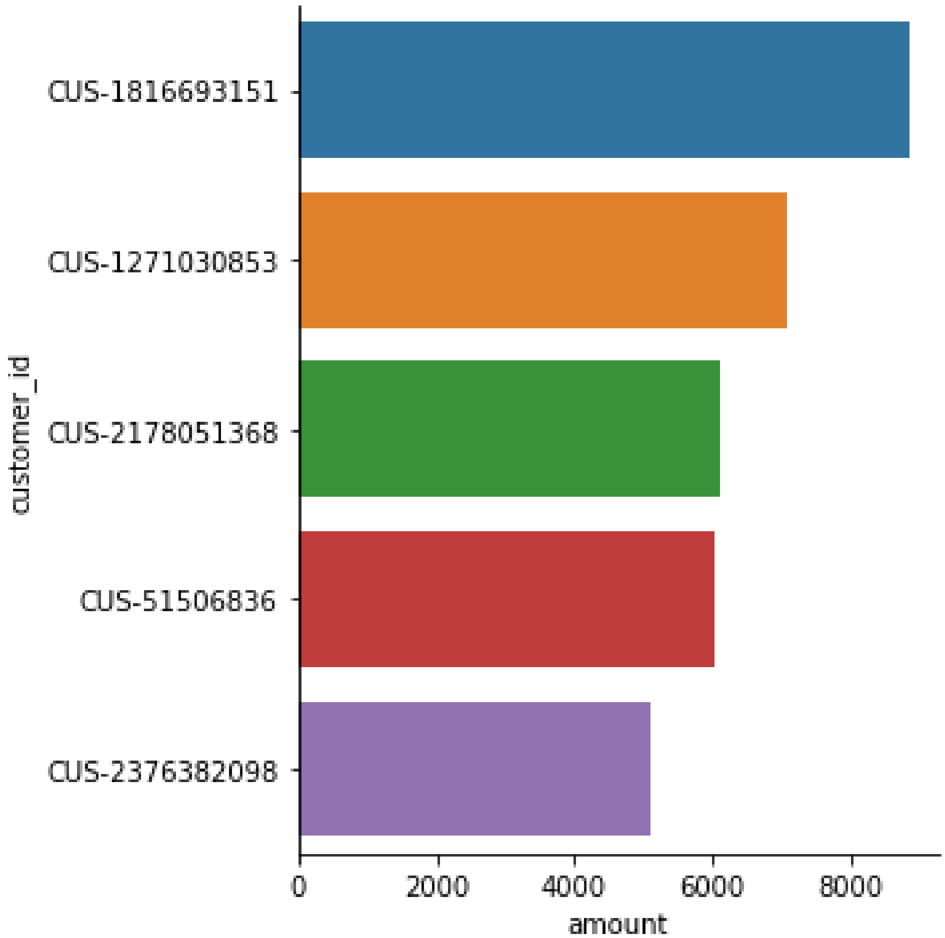

```
In [55]: sns.catplot(y="txn_description", x="amount", kind="bar", data=df.query('amount <= 350
         plt.title('average transaction amount per mode of transaction')
         sns.catplot(y="txn_description", x="amount", kind="point", data=df.query('amount <= 35
```


average transaction amount per mode of transaction
POS
SALES-POS
PAYMENT
INTER BANK
PHONE BANK
txn_description
0
20
40
60
80
amount

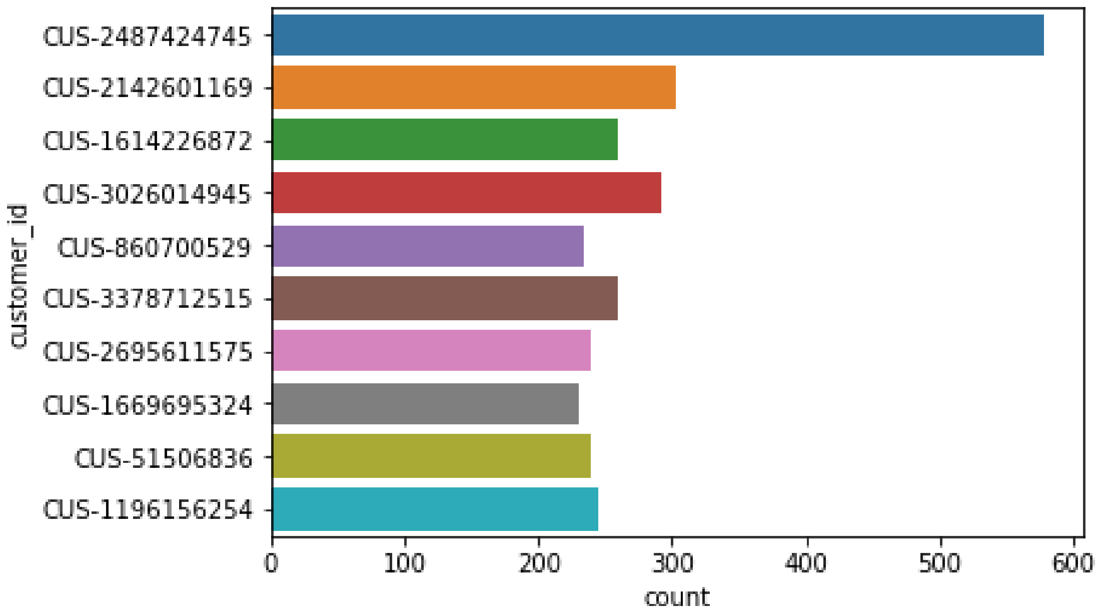


Phone bank transactions are the ones with the highest amount in transactions. # Amount and Customers

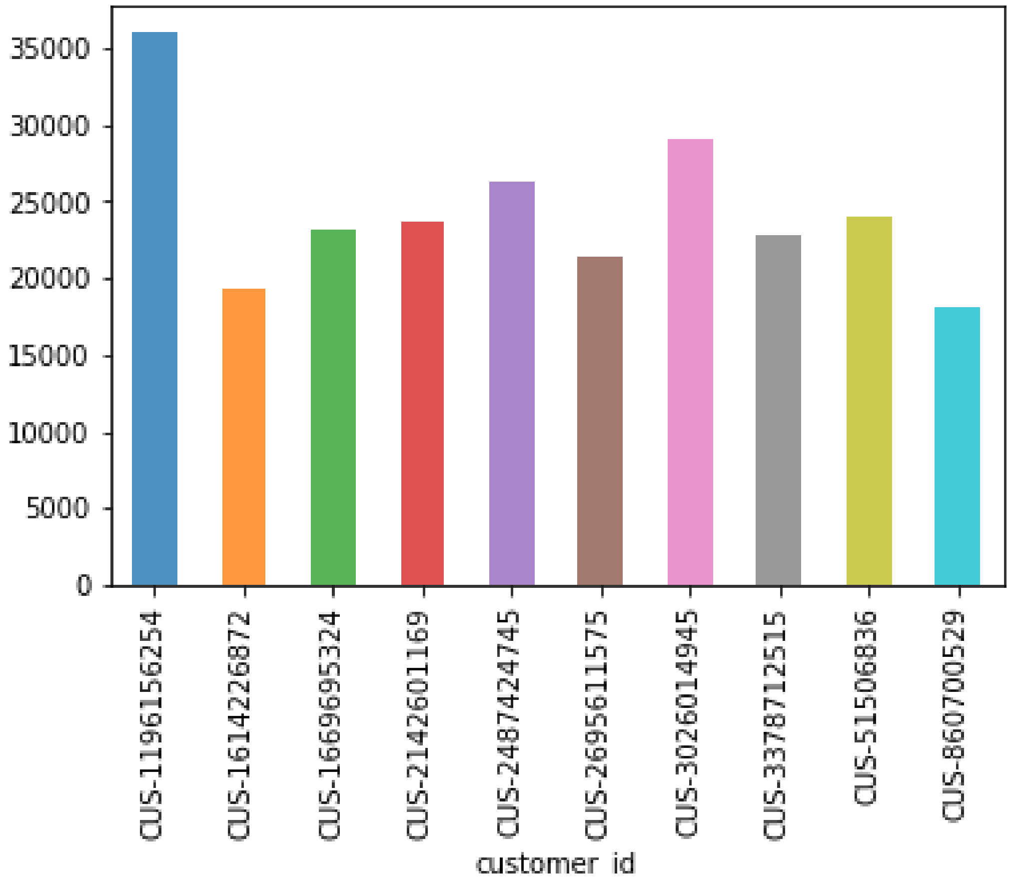
```
In [56]: # customers of the 10 highest transaction amounts
         df1 = df.loc[df.amount.sort_values(ascending =False)[:10].index]
         sns.catplot(y="customer_id", x="amount", kind="bar", data=df1);
```



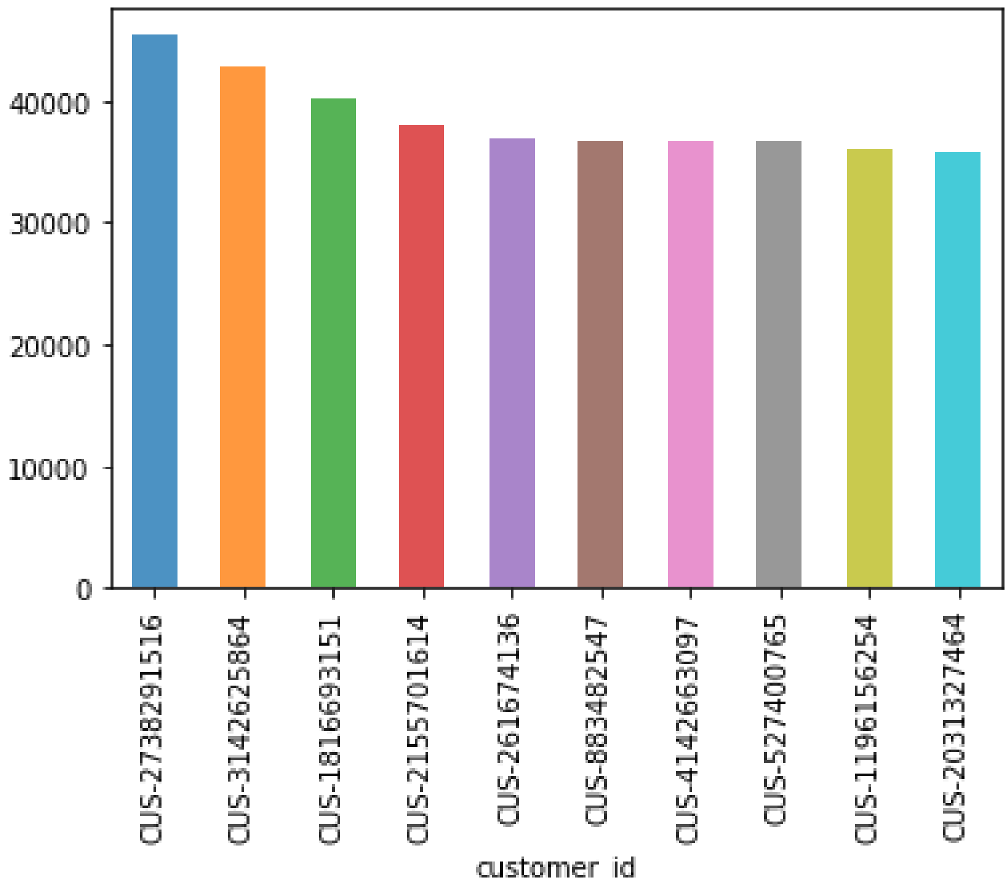

```
In [57]: # cutomers with the highest number of transactions
         df2 = df.loc[df['customer_id'].isin(df.customer_id.value_counts()[:10].index)]
         sns.countplot(y = 'customer_id', data=df2);
```




```
In [58]: # total transactions amounts for the customers with highest number of transactions
         df2.groupby('customer_id')['amount'].sum().plot.bar(alpha = 0.8);
```




```
In [59]: # most spending customers
         df.groupby('customer_id')['amount'].sum().sort_values(ascending=False)[:10].plot.bar(a
```



In [60]:
```
highest_transactions_customers = df1.customer_id.value_counts().index.tolist()
most_dealing_customers = df2.customer_id.value_counts().index.tolist()
most_spending_customers = df.groupby('customer_id')['amount'].sum().sort_values(ascend
```

In [61]:
```
# no customer belongs to top 10 : most spending customers, highest transations made, 
set(highest_transactions_customers).intersection(most_dealing_customers, most_spendin
```

Out[61]: set()

In [62]:
```
# this customer did the 4th highest transaction amount and ranked 9th in the customer
set(highest_transactions_customers).intersection(most_dealing_customers)
```

Out[62]: {'CUS-51506836'}

In [63]:
```
# this customer did the highest transaction amount and is the thirst most spending cu
set(highest_transactions_customers).intersection(most_spending_customers)
```

Out[63]: {'CUS-1816693151'}

In [64]:
```
# this customer did more transactions than all others, and is the 9th most spending c
set(most_dealing_customers).intersection(most_spending_customers)
```

Out[64]: {'CUS-1196156254'}

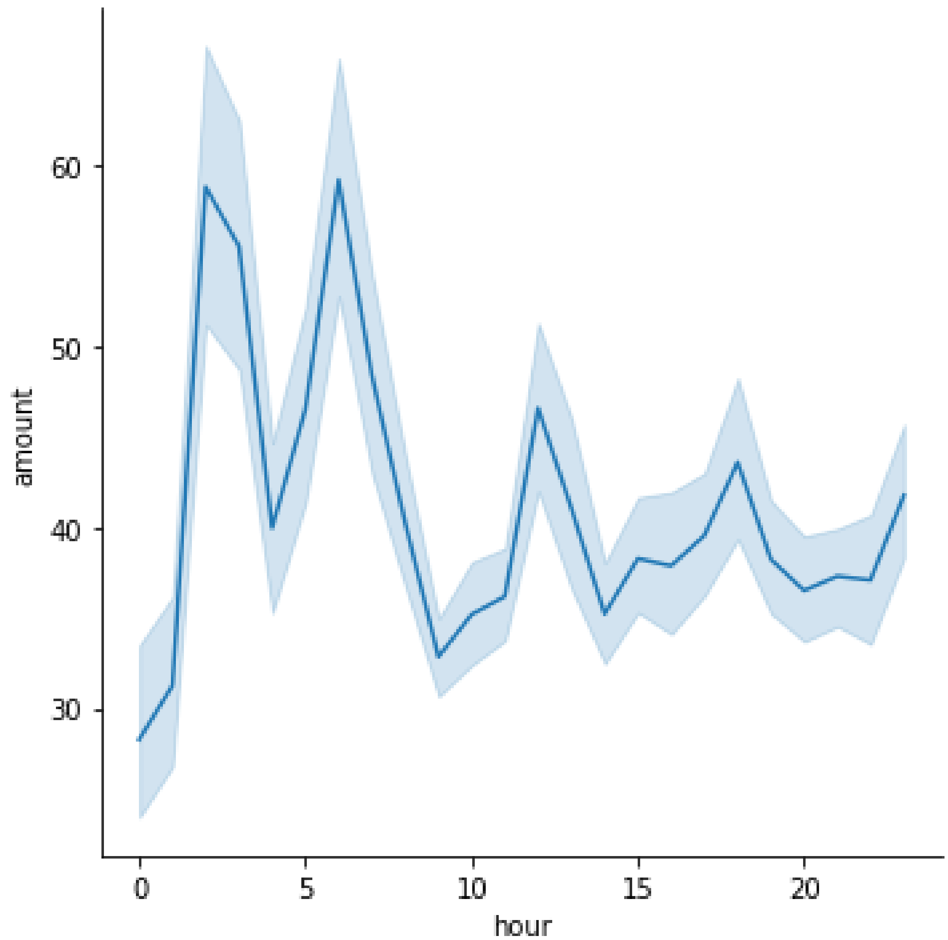
9 Amount and Balance

```
In [65]: sns.heatmap(df[['amount', 'balance']].corr(), annot = True, fmt = '.3f',
                     cmap = 'vlag_r', center = 0)
         plt.show()
```

There is no significant relation between the balance and transaction amount.

10 Transaction by Hours

```
In [66]: sns.relplot(x='hour' , y='amount', kind="line" , data=df.query('amount <= 350'));
```




```
In [67]: writer = pd.ExcelWriter('anz_cleaned_data.xlsx',
                                 engine = 'xlsxwriter')

In [68]: df.to_excel(writer, sheet_name = 'Dataset')

In [69]: writer.save()
         writer.close()
```