
ANZ
Data@ANZ
Virtual Experience Program
EXPLORATORY
DATA
ANALYSIS
Data Segmentation
Data Visualization
Plabanjit Karmakar

About the Dataset

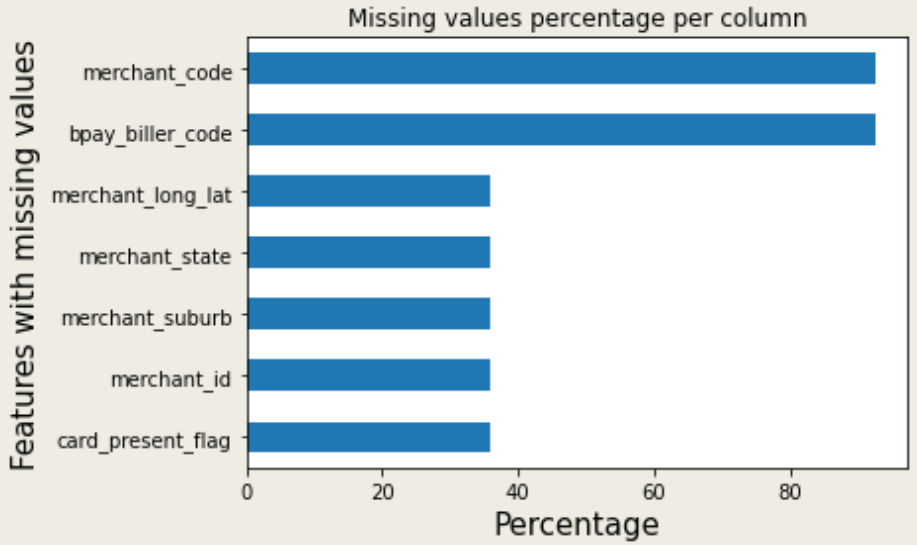
- This dataset is based on a synthesized transaction dataset containing 3 months' (August, September and October) worth of transactions for 100 hypothetical customers. It contains purchases, recurring transactions, and salary transactions.

- It contains 12043 rows of transactions and 23 columns in total.

```
<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
```

Dealing with Missing Values & Data Summary

- In this dataset, we have 5 categorical features and 2 numerical features with missing values.
- As we can see **merchant_code** and **bpay_biller_code** more than 90% missing values, we have dropped these two columns from the dataset.

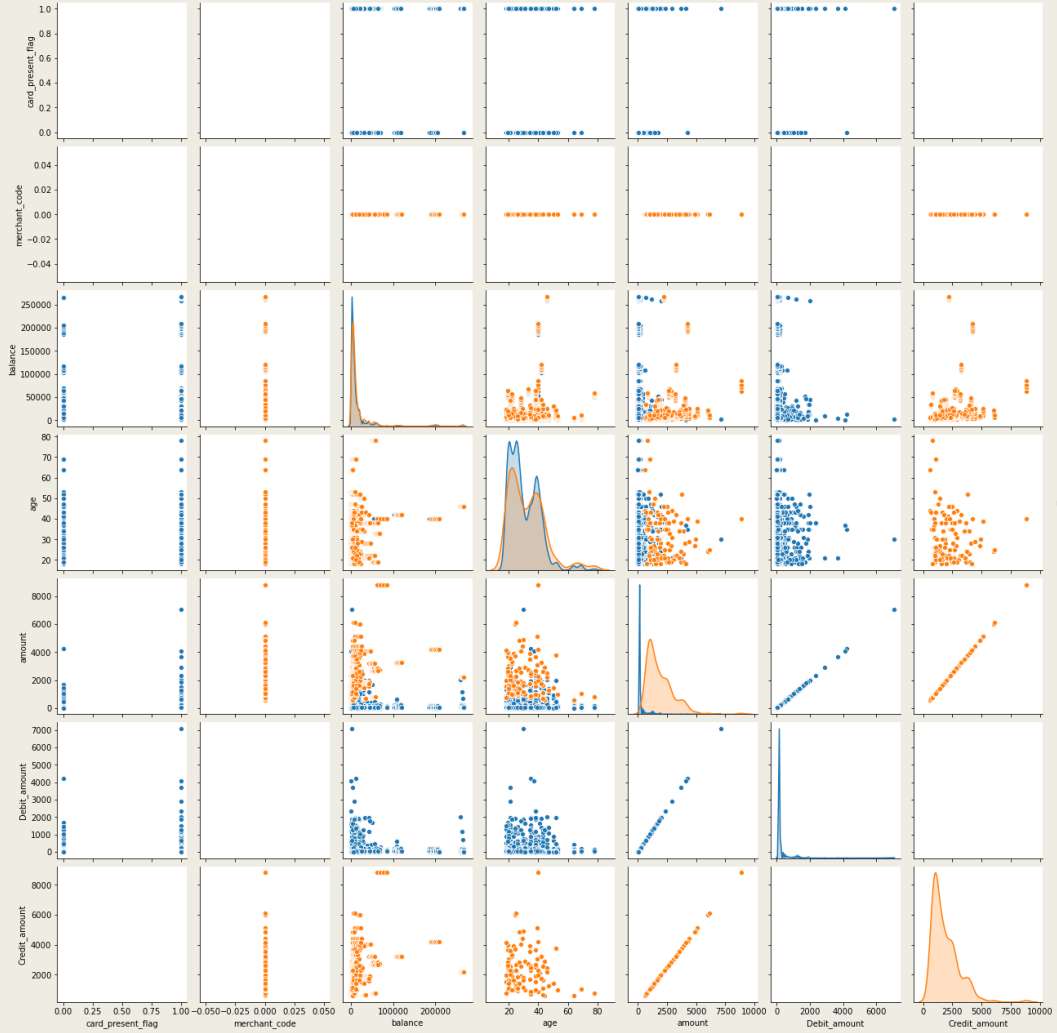


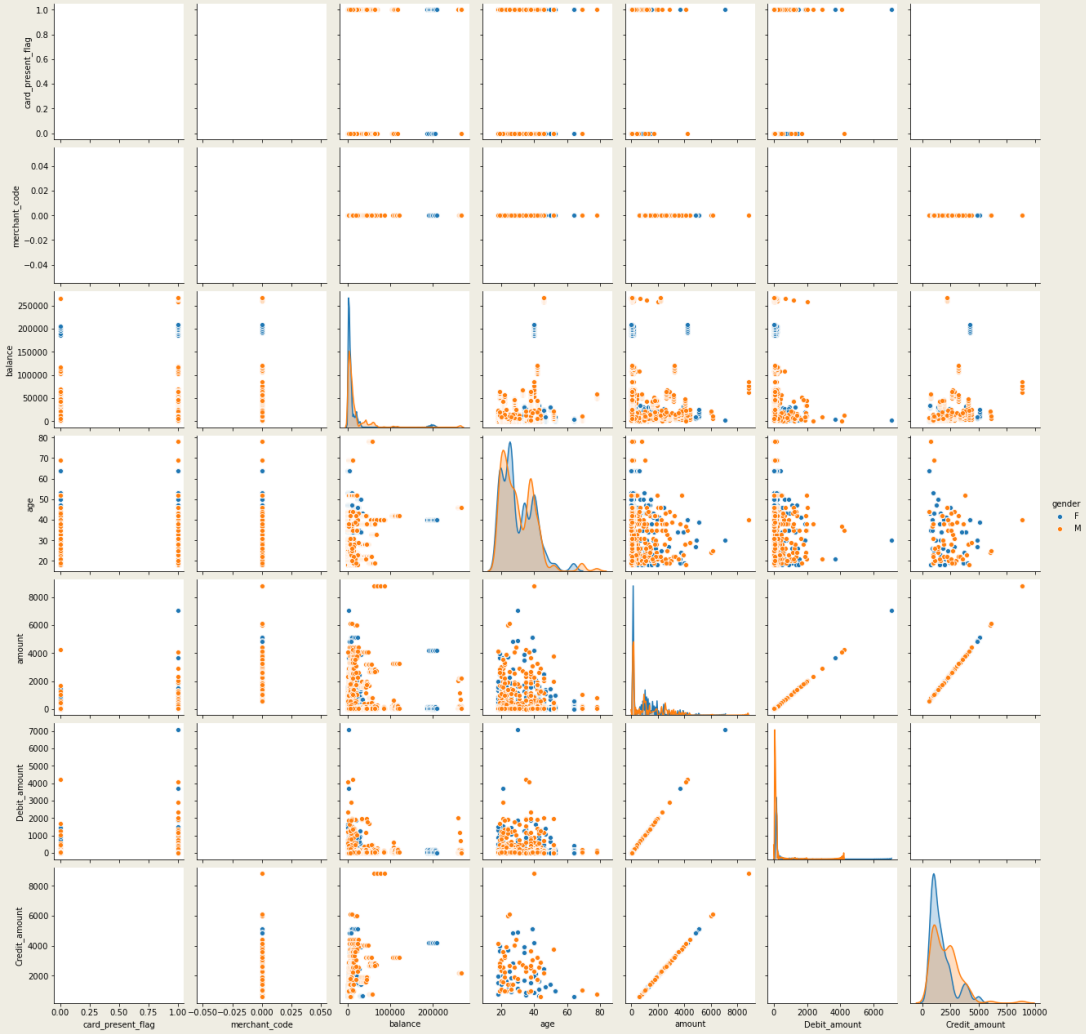
	amount	Debit_amount	Credit_amount
count	12043	11160	883
mean	187.9336	52.57234	1898.728
std	592.5999	156.3541	1150.365
min	0.1	0.1	576
25%	16	15.19	1013.67
50%	29	26.93	1626.48
75%	53.655	45	2538.68
max	8835.98	7081.09	8835.98

As mean is effected by extreme values, in this case we prefer median to mean as a measure of central tendency.

So, the median Transaction amount is AUD 29 whereas, median Debit Transaction amount is AUD 26.93 and median Credit Transaction amount is AUD 1626.48

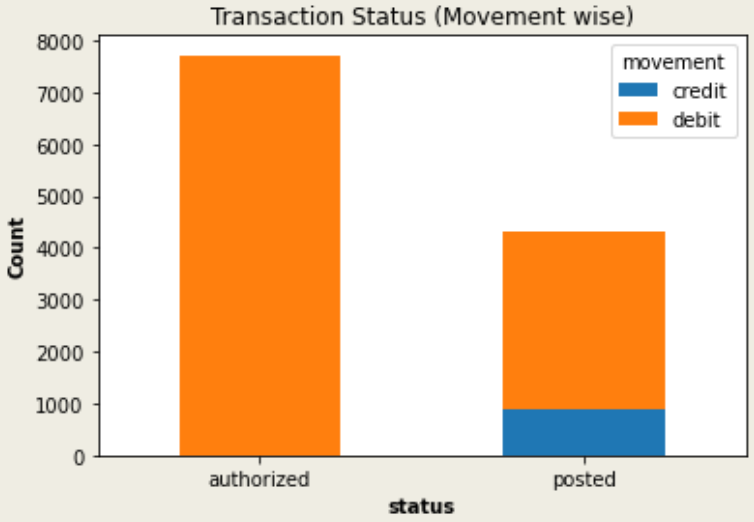
Data Summary

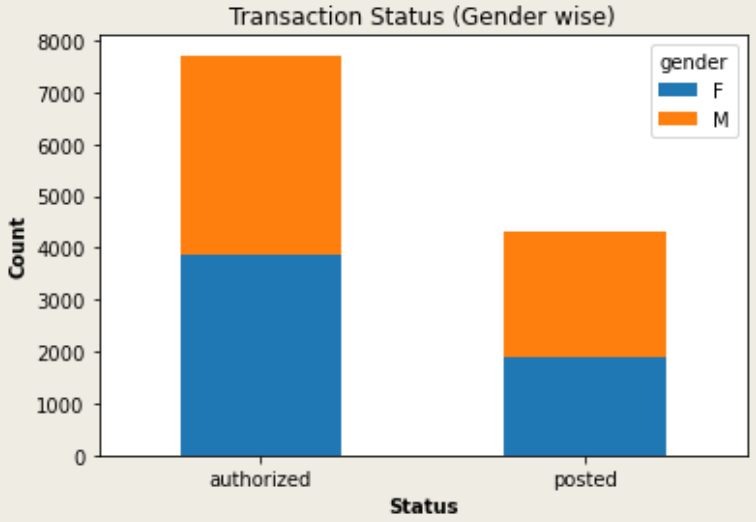


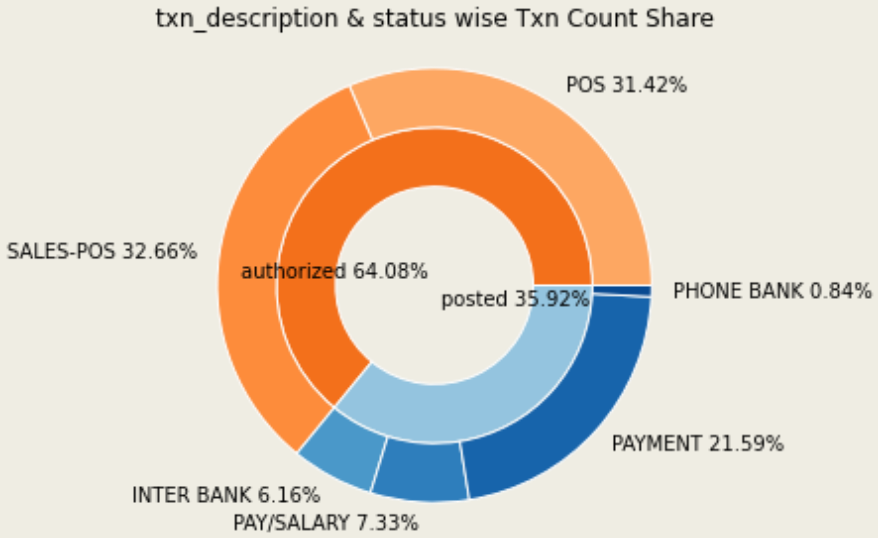


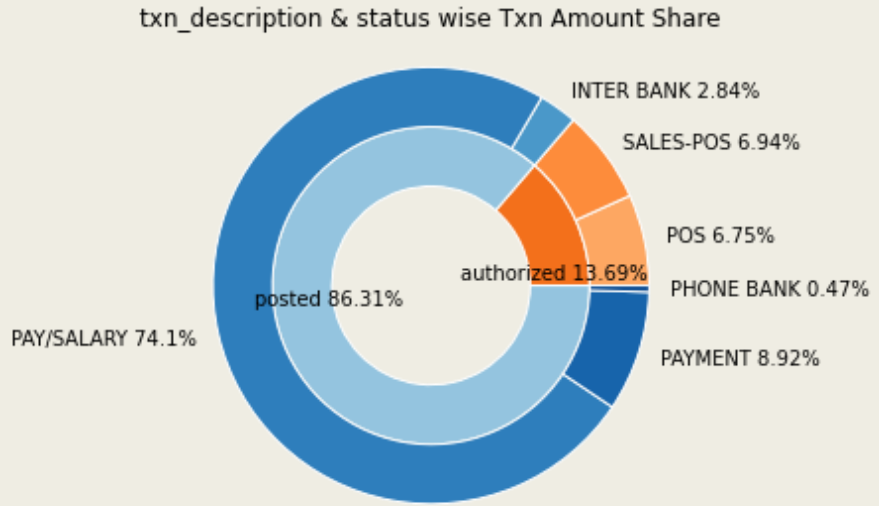
Transaction Status

- Authorized Transactions are those that still need to be settled by a retailer or service provider. Most transactions stay authorized for 3-5 days.
- A posted transaction is a debit or credit that has been fully processed. Once a transaction is posted the account balance on the account is also updated.
- When a retailer or service provider finishes the work on their end, the transaction will move to Posted Transactions.

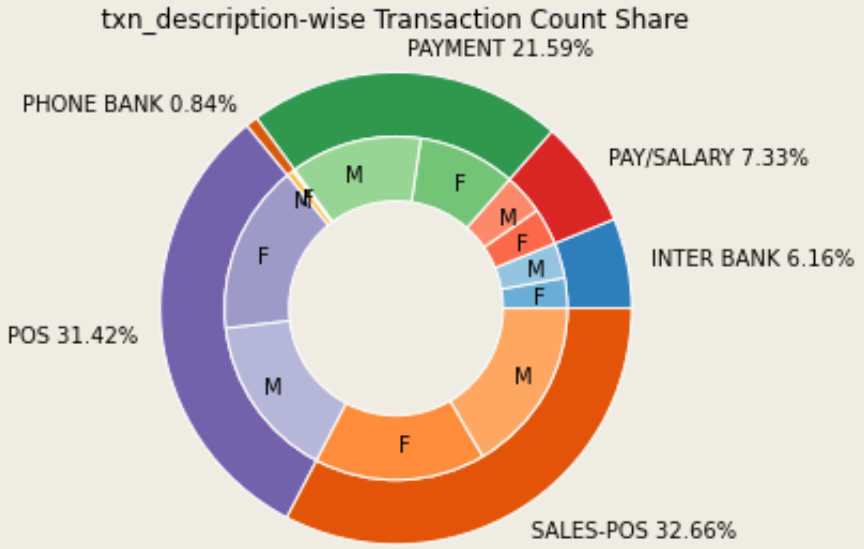


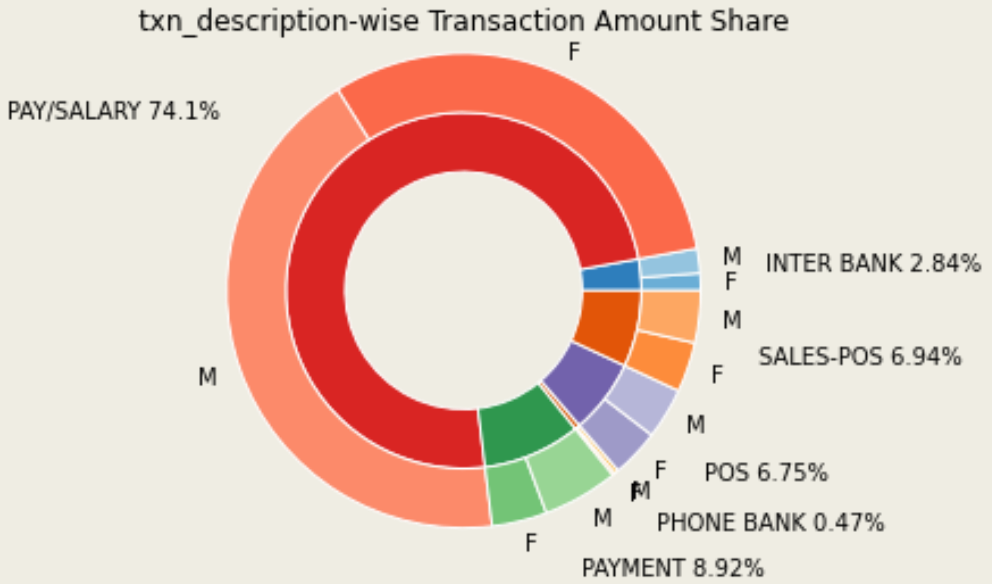


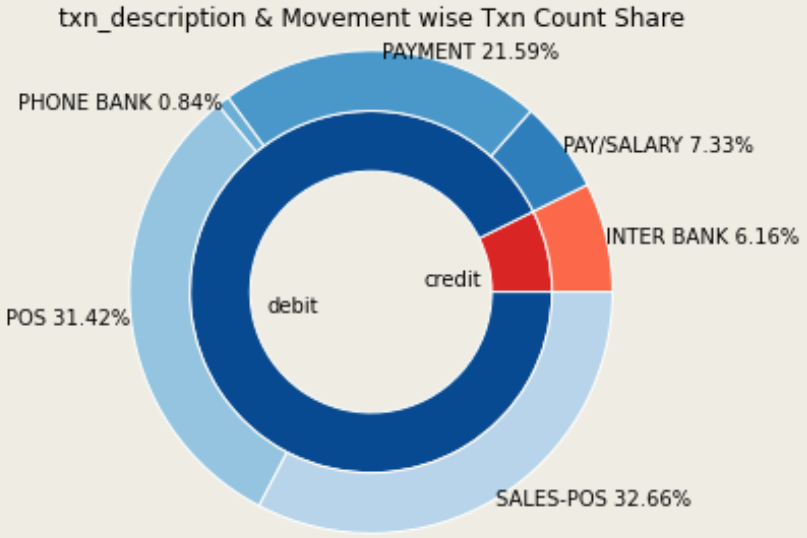


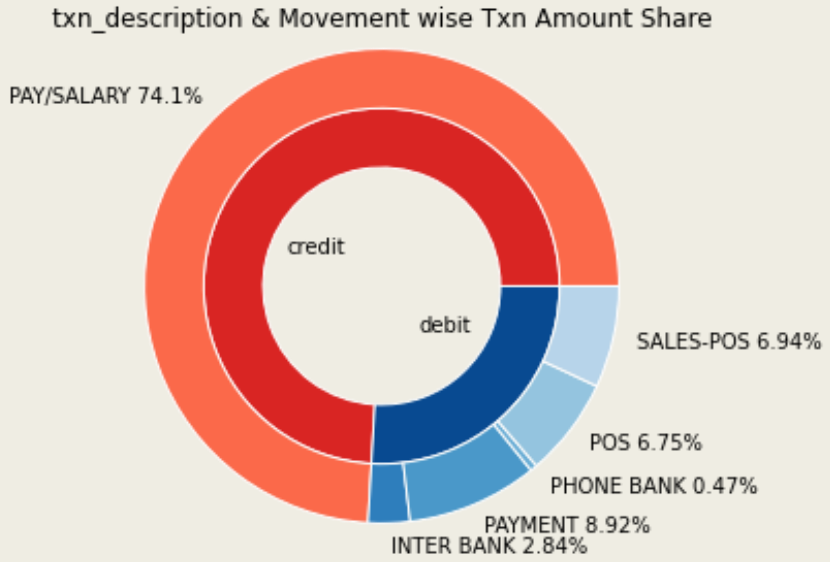


Transaction Description

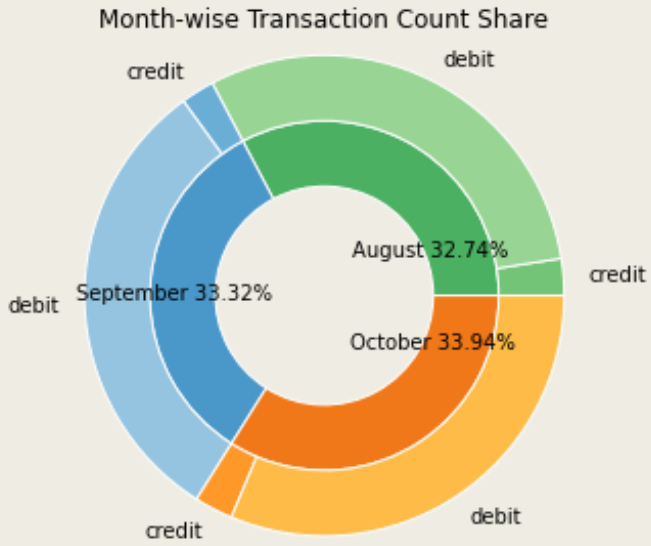


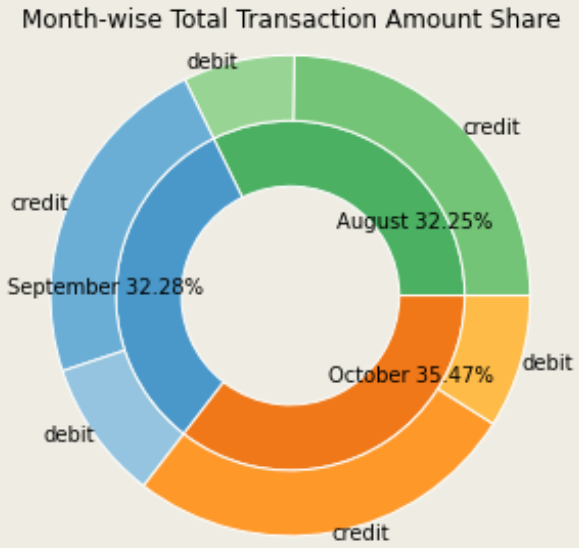






Month Wise Transaction Analysis

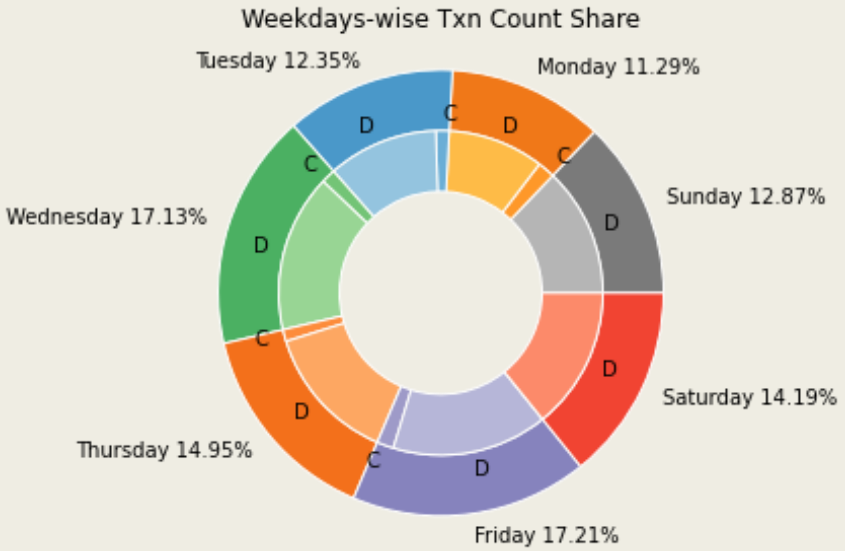


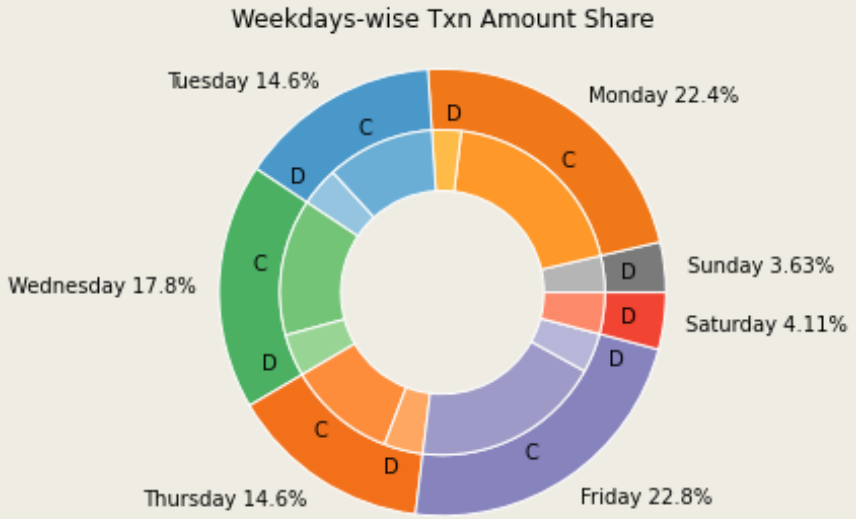


	Txn Count		Txn amount	
movement	credit	debit	credit	debit
month_name				
August	298	3645	559814.3	170121.2
September	272	3741	516595.6	213954.6
October	313	3774	600166.9	202631.6

- Apparently all three months almost same weightage in terms of Txn count and Txn amount.
- But, highest transaction has occurred in the month of October among these three.

Weekdays Wise Transaction Analysis

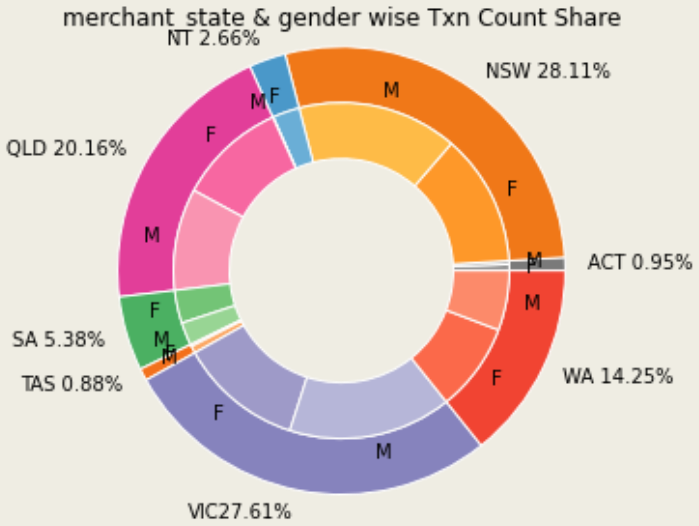


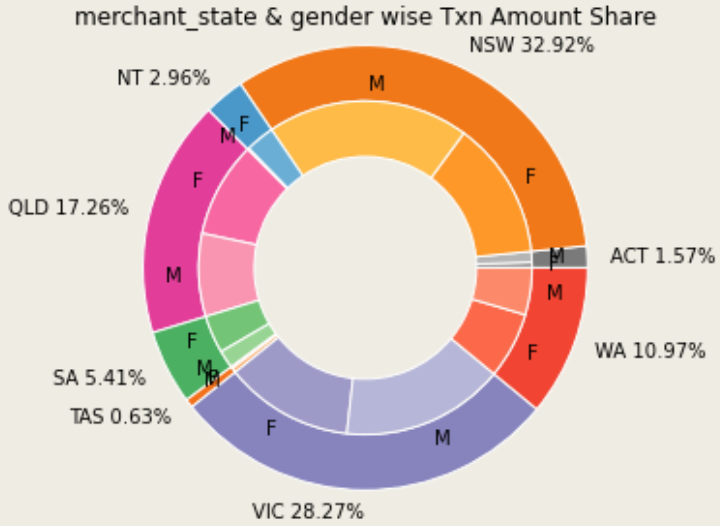


	Txn Count		Txn amount	
day_name	credit	debit	credit	debit
Sunday	0	1550	0	82175
Monday	207	1153	442167.7	65413
Tuesday	160	1327	255243.3	74241
Wednesday	172	1891	307770.3	94959
Thursday	143	1658	243725.4	87677
Friday	201	1872	427670.2	89240
Saturday	0	1709	0	93003

- There is no significant difference in terms of Txn counts throughout the week.
- No credit Txn in Saturdays & Sundays.
- Majority of Credit Txn amount has been reflected on Mondays and Fridays whereas Debit Amount Txn is more or less same throughout the week.

Merchant State Wise Transaction Analysis

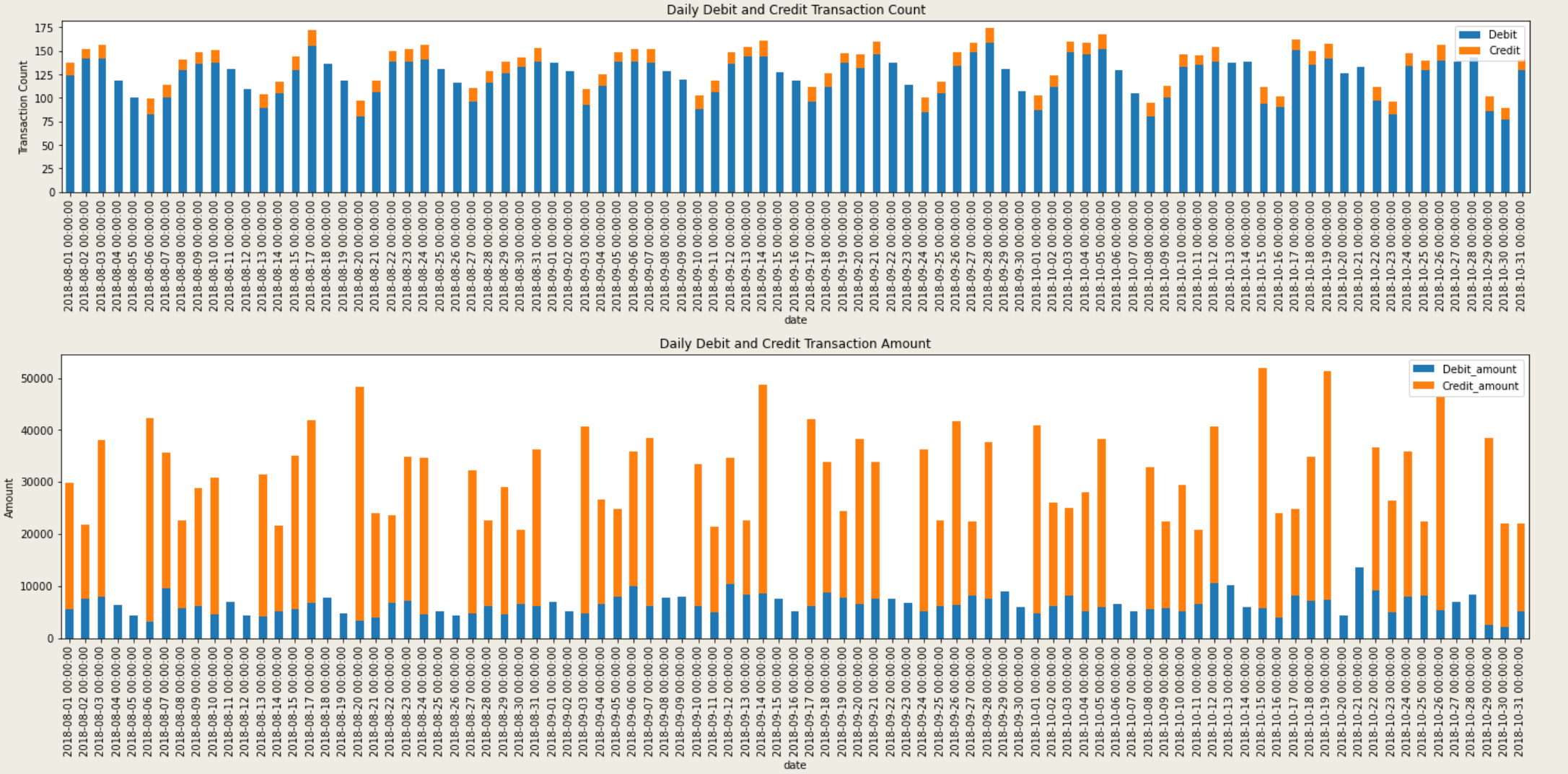




	Txn Count		Txn amount	
Merchant state	F	M	F	M
ACT	46	27	1657.44	3219.24
NSW	980	1189	41430.88	60590.89
NT	200	5	8741.42	427.47
QLD	800	756	28611.05	24872.4
SA	245	170	11349.73	5426.84
TAS	16	52	622.72	1340.21
VIC	918	1213	38626.01	48957.99
WA	657	443	19908.15	14083.91

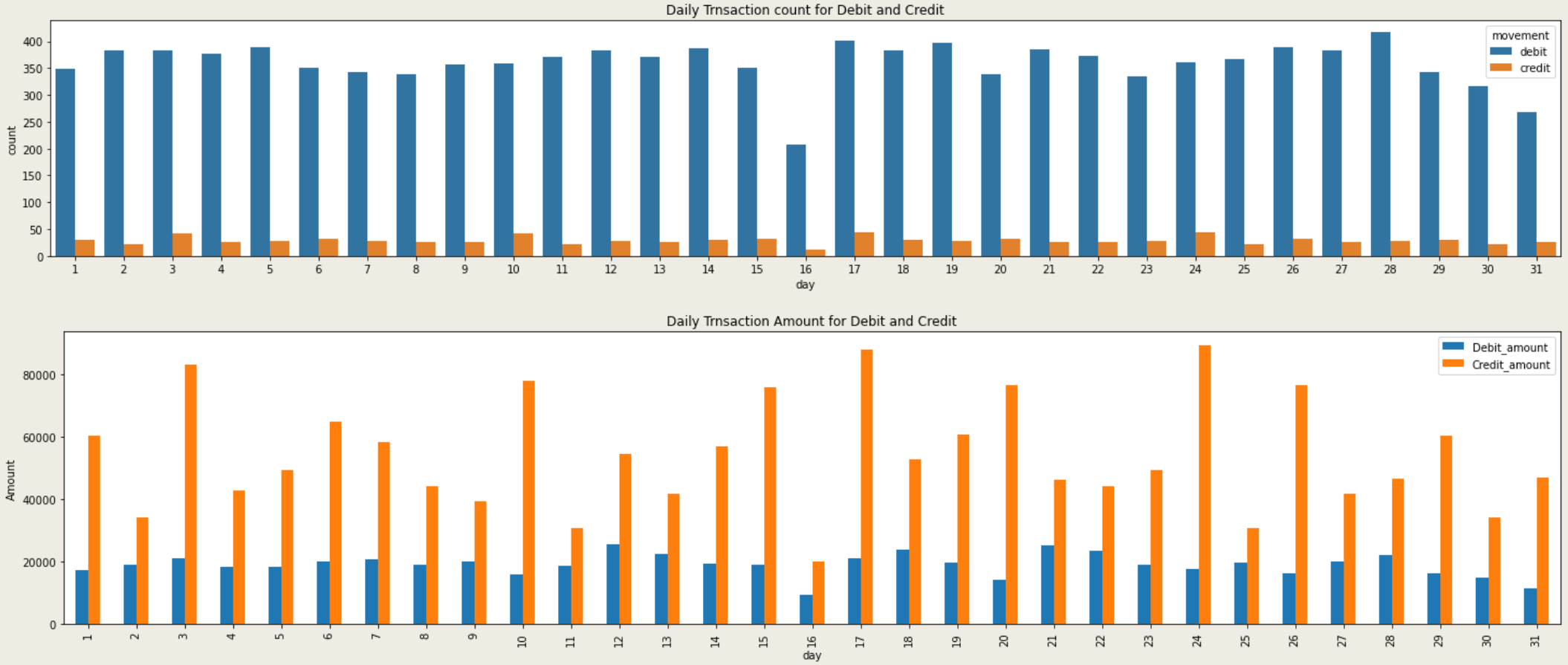
- Almost 60% of Total Debit Txn amount is going to the merchants of VIC and NSW.
- Females have contributed more than the males in QLD, SA WA and NT.
- In total Females have done more Debit Txn than males.

Transaction Trend



- In both cases of daily transaction count and total transaction amount, there is a cyclic trend over the whole time period.
- No. of Credit transactions are very less compared to no. of Debit Transactions where as majority contribution of transaction amount is coming from Credit Transactions. On another note, we see that, there is no credit transactions at frequent intervals, on weekends.

Transactions on various days of a Month



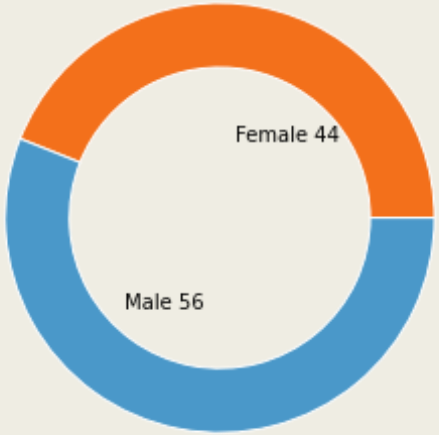
- We can observe a downward trend in no. of Debit Txn counts and Debit amount at the end of the months.
- There are peaks in Credit Amounts at a regular interval on the dates 3, 10, 17 and 24. Other than that, we can see cyclic nature in both Credit and Debit amount over the days.

Customer Analysis

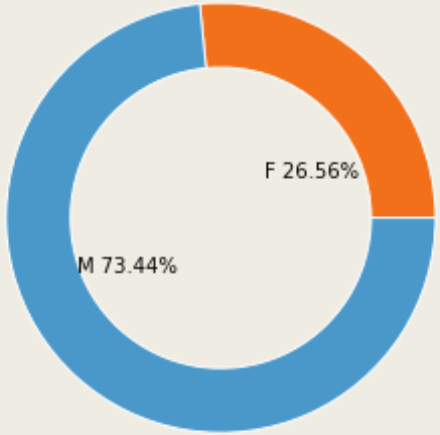
- We have grouped the whole data by Customer Id and collected the data of 100 customers. In this process we have created a few new columns relevant to the study.
- We split the amount column and introduced two new columns, Debit_Amount & Credit_amount in accordance with the movement column.
- Later, we calculated the annual salary of each customer by analyzing the credit transaction of each customer.
- After that, we created this pair plot to take a deep look on the customers. As we can see that all the columns are more or less positively skewed.
- There is strong positive correlation between Total_Transaction_count , total_debit_amount and opening_bal, Closing_bal.

Gender wise Transaction Analysis

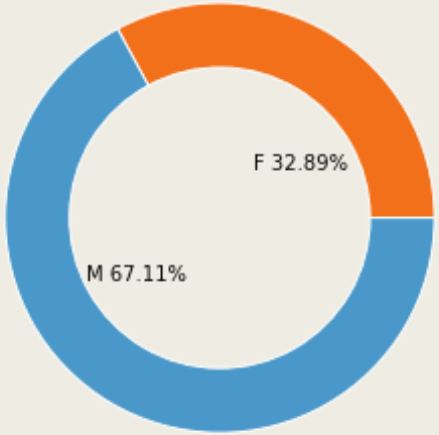
No. of Customers (Gender wise)



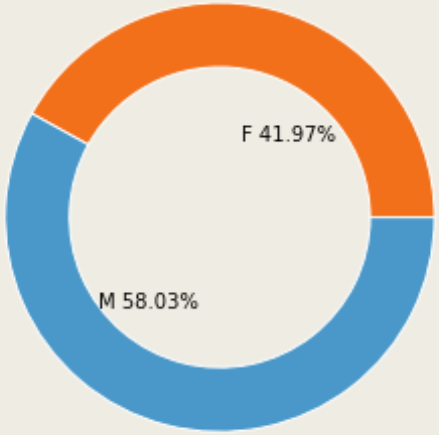
Gender wise Opening_balance share



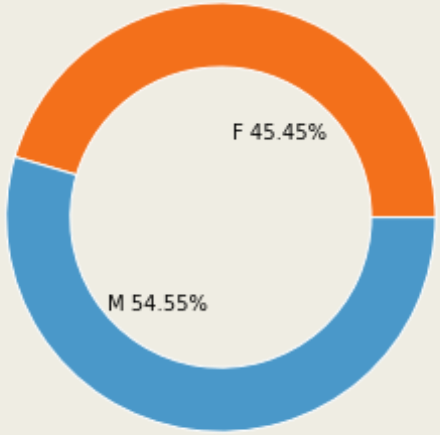
Gender wise Closing_balance share



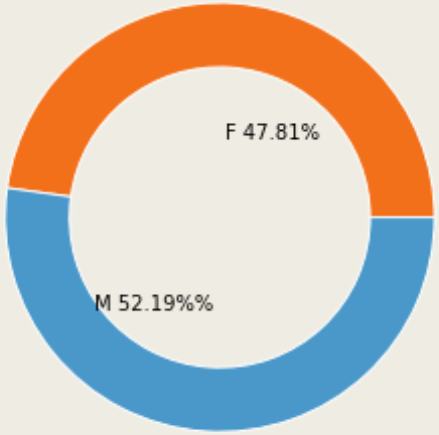
Gender wise total_Credit_amount share



Gender wise total_Debit_amount share



Gender wise total_Txn_count share



ANZ Virtual Internship Report

1. Data Preprocessing and Feature Engineering

In [1]:
```
#importing packages for data analysis
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import datetime
import matplotlib.dates
from matplotlib import rc
#ignoring warnings
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
#importing dataset
pd.options.display.max_columns = None
df=pd.read_excel("ANZ synthesised transaction dataset.xlsx")
df.head()
```

Out[2]:

	status	card_present_flag	bpay_biller_code	account	currency	long_lat	txn_description
0	authorized	1.0	NaN	ACC-1598451071	AUD	153.41 -27.95	POS
1	authorized	0.0	NaN	ACC-1598451071	AUD	153.41 -27.95	SALES-POS
2	authorized	1.0	NaN	ACC-1222300524	AUD	151.23 -33.94	POS
3	authorized	1.0	NaN	ACC-1037050564	AUD	153.10 -27.66	SALES-POS
4	authorized	1.0	NaN	ACC-1598451071	AUD	153.41 -27.95	SALES-POS

In [3]:
```
#Shape of the data
df.shape
```

Out[3]:
```
(12043, 23)
```

In [4]:
```
#data description
df.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 [5]:
```
#No. of unique values in each column.
column_list = df.columns.tolist()
for column in column_list:
    print( column, ':', df[column].nunique())
```

```
status : 2
card_present_flag : 2
bpay_biller_code : 3
account : 100
currency : 1
long_lat : 100
txn_description : 6
merchant_id : 5725
merchant_code : 1
first_name : 80
balance : 12006
date : 91
gender : 2
age : 33
merchant_suburb : 1609
merchant_state : 8
extraction : 9442
amount : 4457
transaction_id : 12043
country : 1
customer_id : 100
merchant_long_lat : 2703
movement : 2
```

The above piece of code shows the number of unique values in each of the columns.

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).

Here we are particularly interested in the 'movement' column and we have found out that there are only two types of transaction took place. Hence we create two columns in the dataframe showing the debit and credit amounts.

Adding Debit & Credit Amount Columns in Dataframe

In [6]:
```
#Creating Duplicate Dataframe
df_new=df.copy(deep=True)
```

In [7]:
```
#Creating two new columns from amount
df_new['Debit_amount'] = np.where(df['movement'] == 'debit', df['amount'
], np.nan)
df_new['Credit_amount'] = np.where(df['movement'] == 'credit', df['amoun
t'], np.nan)
df_new.head()
```

Out[7]:

	status	card_present_flag	bpay_biller_code	account	currency	long_lat	txn_description
0	authorized	1.0	NaN	ACC-1598451071	AUD	153.41 -27.95	POS
1	authorized	0.0	NaN	ACC-1598451071	AUD	153.41 -27.95	SALES-POS
2	authorized	1.0	NaN	ACC-1222300524	AUD	151.23 -33.94	POS
3	authorized	1.0	NaN	ACC-1037050564	AUD	153.10 -27.66	SALES-POS
4	authorized	1.0	NaN	ACC-1598451071	AUD	153.41 -27.95	SALES-POS

Dealing with Missing Values

In [8]:
```
# total null values
df.isnull().sum()
```

Out[8]:
```
status                0
card_present_flag     4326
bpay_biller_code      11158
account               0
currency              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_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
```

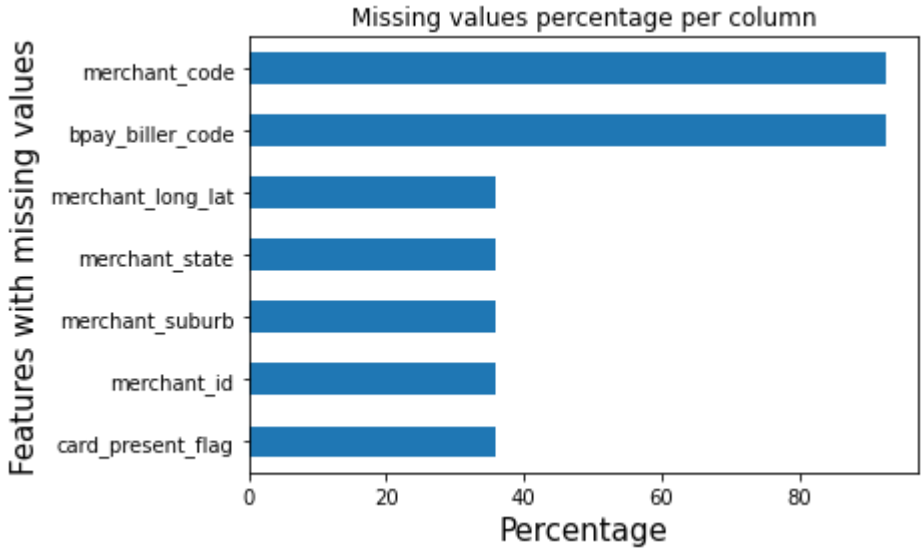
In [9]:
```
# classifying NA as categorical or numerical

NA=df[['card_present_flag','bpay_biller_code','merchant_id','merchant_cod
e','merchant_suburb','merchant_state','merchant_long_lat']]
NAcat=NA.select_dtypes(include='object')
NAnum=NA.select_dtypes(exclude='object')
print(NAcat.shape[1],'categorical features with missing values')
print(NAnum.shape[1],'numerical features with missing values')
```

```
5 categorical features with missing values
2 numerical features with missing values
```

In [10]:
```
# Calculating Percentage of Missing Values in Each Column
df.isna().sum()/len(df) * 100
```

Out[10]:
```
status                0.000000
card_present_flag    35.921282
bpay_biller_code     92.651333
account               0.000000
currency              0.000000
long_lat              0.000000
txn_description       0.000000
merchant_id          35.921282
merchant_code        92.667940
first_name            0.000000
balance               0.000000
date                  0.000000
gender                0.000000
age                   0.000000
merchant_suburb      35.921282
merchant_state       35.921282
extraction            0.000000
amount                0.000000
transaction_id        0.000000
country               0.000000
customer_id           0.000000
merchant_long_lat    35.921282
movement              0.000000
dtype: float64
```

In [11]:
```
# visulaizing missing values percentage

plt.figure()
allna = (df.isnull().sum() / len(df))*100
allna = allna.drop(allna[allna == 0].index).sort_values()
allna.plot.barh()
plt.title('Missing values percentage per column')
plt.xlabel('Percentage', fontsize=15)
plt.ylabel('Features with missing values',fontsize=15)
#plt.yticks(weight='bold')
plt.show()
```



From the above piece of code, we get some information about missing values, which should be dealt carefully in order to do further analysis. Here we gather an idea about dropping columns with highest number of missing values. As we can see merchant_code and bpay_biller_code more than 90% missing values, we should drop these two columns from the dataset.

In [12]:
```
# we should drop merchant_code &bpay_biller_code
# removing columns
df.drop(['bpay_biller_code','merchant_code'],axis=1,inplace=True)
```

Summary of dataset

In [13]:
```
#Summary of df_new
df_new.describe()
```

Out[13]:

	card_present_flag	merchant_code	balance	age	amount	Debit_amount
count	7717.000000	883.0	12043.000000	12043.000000	12043.000000	11160.000000
mean	0.802644	0.0	14704.195553	30.582330	187.933588	52.572343
std	0.398029	0.0	31503.722652	10.046343	592.599934	156.354143
min	0.000000	0.0	0.240000	18.000000	0.100000	0.100000
25%	1.000000	0.0	3158.585000	22.000000	16.000000	15.190000
50%	1.000000	0.0	6432.010000	28.000000	29.000000	26.930000
75%	1.000000	0.0	12465.945000	38.000000	53.655000	45.000000
max	1.000000	0.0	267128.520000	78.000000	8835.980000	7081.090000

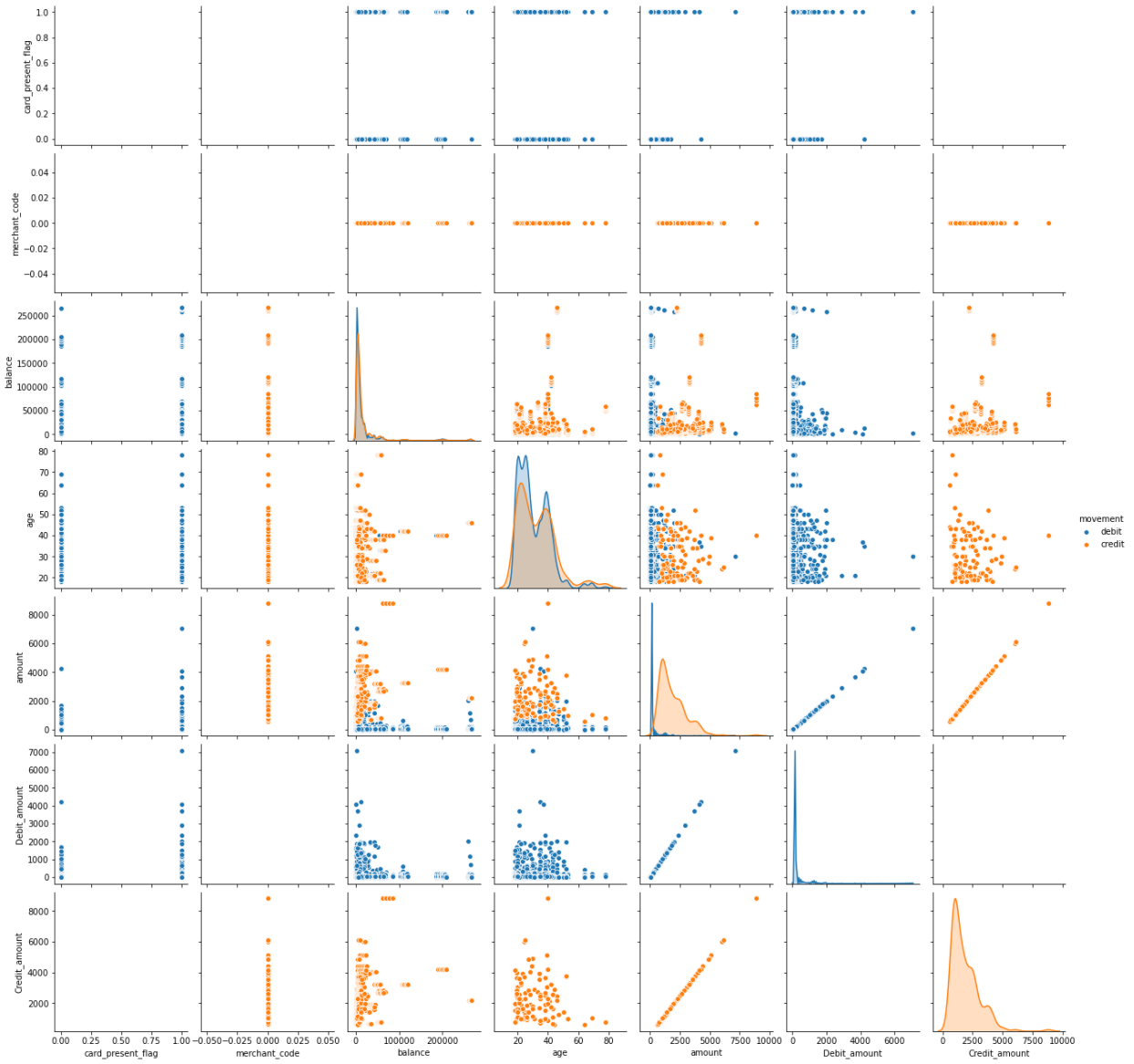
Executing the following piece of code, we get a glimpse of some summery statistics. We can see debit amount on an average in 52.57 AUD, whereas mean credit amount is much higher, which is 1898.73 AUD.

But we also see a significant difference between mean and median, which indicates that the variables are skewed. As mean is effected by extreme values, in this case we prefer median to mean as a measure of central tendency.

So, the median Transaction amount is AUD 29 whereas, median Debit Transaction amount is AUD 26.93 and median Credit Transaction amount is AUD 1626.48

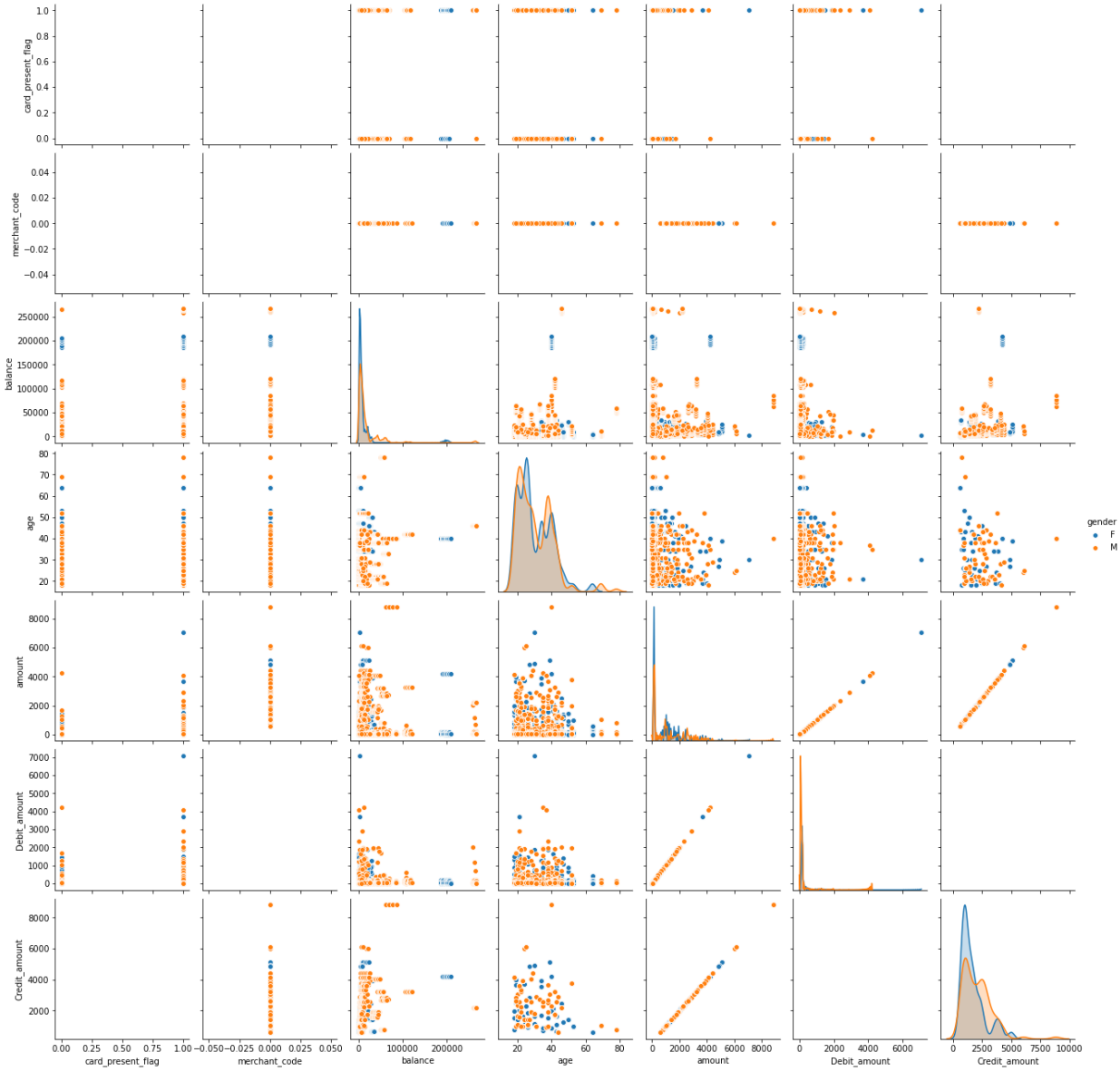
In [14]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.pairplot(df_new, hue='movement')
```

Out[14]:
```
<seaborn.axisgrid.PairGrid at 0xcbec108>
```



In [15]:
```
sns.pairplot(df_new, hue='gender')
```

Out[15]:
```
<seaborn.axisgrid.PairGrid at 0xf30bf48>
```



2. Transaction Status

In [16]:
```
df['status'].value_counts()
```

Out[16]:
```
authorized    7717
posted        4326
Name: status, dtype: int64
```

Authorized Transactions are those that still need to be settled by a retailer or service provider. Most transactions stay authorized for 3-5 days.

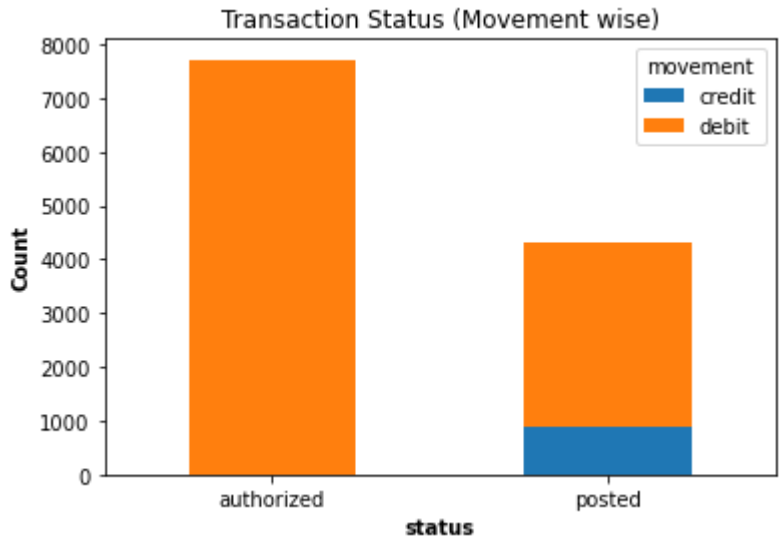
A posted transaction is a debit or credit that has been fully processed. Once a transaction is posted the account balance on the account is also updated.

When a retailer or service provider finishes the work on their end, the transaction will move to Posted Transactions.

Transaction Status (Movement wise)

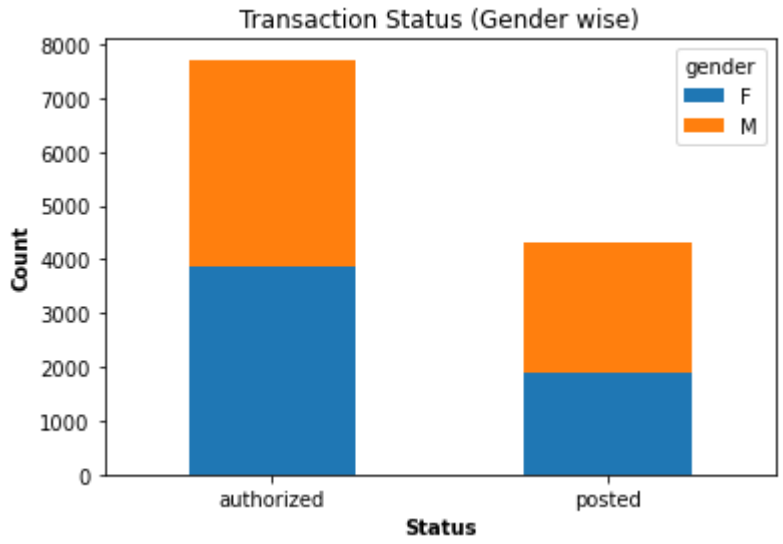
In [17]:
```
#Visualization of Transaction Status (Movement wise)
loc_plt=pd.crosstab(df['status'],df['movement'])
loc_plt.plot(kind='bar',stacked=True);
```

```
plt.title('Transaction Status (Movement wise)',fontsize=12)
plt.ylabel('Count',fontsize=10, fontweight='bold')
plt.xlabel('status',fontsize=10, fontweight='bold')
plt.xticks(fontsize=10, rotation=0)
plt.yticks(fontsize=10);
```



Transaction Status (Gender wise)

In [18]:
```
#Visualization of Transaction Status (Gender wise)
loc_plt=pd.crosstab(df['status'],df['gender'])
loc_plt.plot(kind='bar',stacked=True);
plt.title('Transaction Status (Gender wise)',fontsize=12)
plt.ylabel('Count',fontsize=10, fontweight='bold')
plt.xlabel('Status',fontsize=10, fontweight='bold')
plt.xticks(fontsize=10, rotation=0)
plt.yticks(fontsize=10);
```



Grouping the data by_status_txn_description_count

In [19]:
```
by_status_txn_description_count = df.groupby([
'status', 'txn_description'
]).count()
```

In [20]:
```
by_status_txn_description_count = by_status_txn_description_count.unstack
().fillna(0)
```

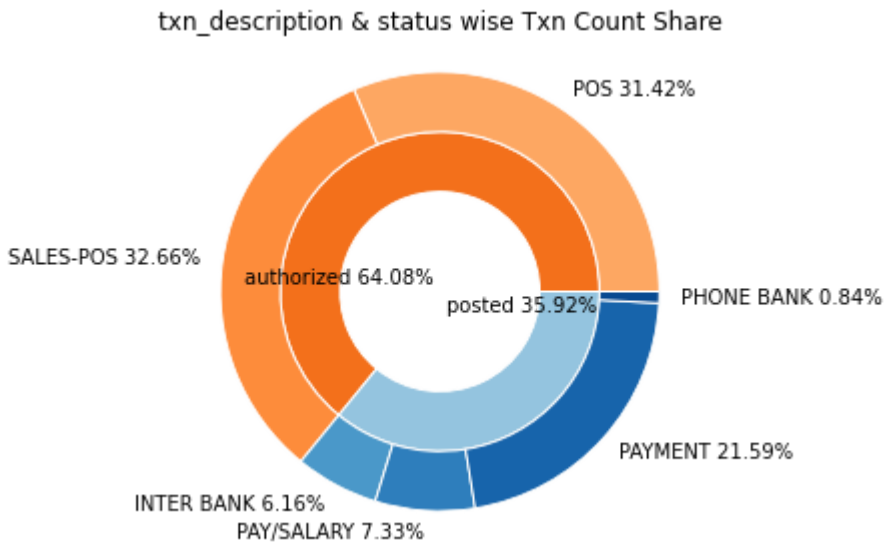
```
by_status_txn_description_count
```

Out[20]:

	card_present_flag					account		
txn_description	INTER BANK	PAY/SALARY	PAYMENT	PHONE BANK	POS	SALES-POS	INTER BANK	PAY/SALARY	PAY
status									
authorized	0.0	0.0	0.0	0.0	3783.0	3934.0	0.0	0.0	
posted	0.0	0.0	0.0	0.0	0.0	0.0	742.0	883.0	

In [21]:

```
#Donut_by_status_txn_description_count
import matplotlib.pyplot as plt

# Make data: I have 2 groups and 6 subgroups
group_names=['authorized 64.08%', 'posted 35.92%']
group_size=[7717.0, 4326.0 ]
subgroup_names=['POS 31.42%', 'SALES-POS 32.66%',  'INTER BANK 6.16%', 'P
AY/SALARY 7.33%',
                'PAYMENT 21.59%', 'PHONE BANK 0.84%']
subgroup_size=[3783.0 , 3934.0, 742.0 , 883.0 , 2600.0, 101.0]

# Create colors
a, b =[plt.cm.Oranges, plt.cm.Blues]

# First Ring (Inside)
fig, ax = plt.subplots()
ax.axis('equal')
mypie, _ = ax.pie(group_size, radius=1.1-0.3, labels=group_names, labeldi
stance=0.1, colors=[a(0.6), b(0.4)])
plt.setp( mypie, width=0.3, edgecolor='white')

# Second Ring (outside)
mypie2, _ = ax.pie(subgroup_size, radius=1.1, labels=subgroup_names,
                   colors=[a(0.4), a(0.5), b(0.6), b(0.7), b(0.8), b(0.9
)])

plt.setp( mypie2, width=0.3, edgecolor='white')
plt.title('txn_description & status wise Txn Count Share')
plt.margins(0,0)
plt.tight_layout()
# show it
plt.show()
```



Grouping the data by_status_txn_description_sum

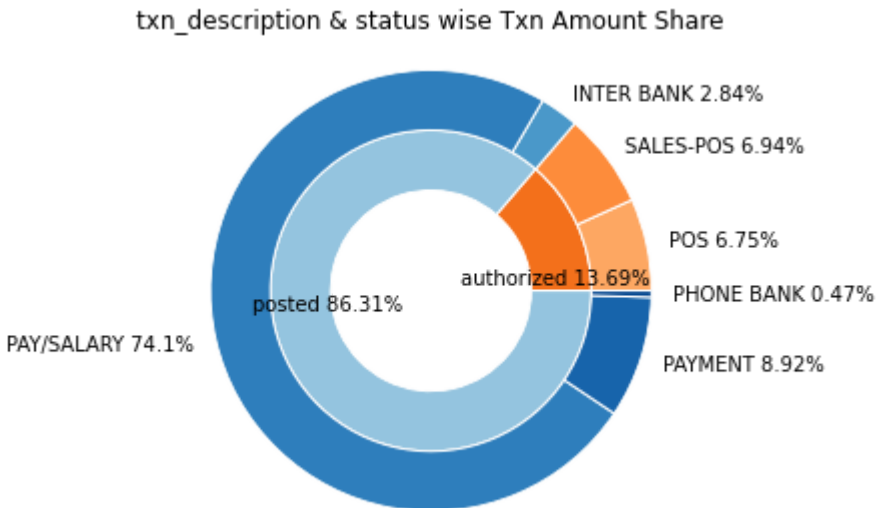
In [22]:
```
by_status_txn_description_sum = df.groupby([
'status', 'txn_description'
]).sum()
```

In [23]:
```
by_status_txn_description_sum = by_status_txn_description_sum.unstack().f
illna(0)
by_status_txn_description_sum
```

Out[23]:

	card_present_flag						balance	
txn_description	INTER BANK	PAY/SALARY	PAYMENT	PHONE BANK	POS	SALES-POS	INTER BANK	PAY/SALARY
status								
authorized	0.0	0.0	0.0	0.0	3025.0	3169.0	0.00	0.0
posted	0.0	0.0	0.0	0.0	0.0	0.0	17676922.73	14342444.5

In [24]:
```
#Donut_by_status_txn_description_sum
import matplotlib.pyplot as plt
# Make data: I have 2 groups and 6 subgroups
group_names=['authorized 13.69%', 'posted 86.31%']
group_size=[309866.35, 1953417.85 ]
subgroup_names=['POS 6.75%', 'SALES-POS 6.94%', 'INTER BANK 2.84%', 'PAY/
SALARY 74.1%',  'PAYMENT 8.92%', 'PHONE BANK 0.47%']
subgroup_size=[152861.24 , 157005.11, 64331.00 , 1676576.85 , 201794.00 ,
10716.00]

# Create colors
a, b =[plt.cm.Oranges, plt.cm.Blues]

# First Ring (Inside)
fig, ax = plt.subplots()
ax.axis('equal')
mypie, _ = ax.pie(group_size, radius=1.1-0.3, labels=group_names, labeldi
stance=0.2, colors=[a(0.6), b(0.4)])
plt.setp( mypie, width=0.3, edgecolor='white')

# Second Ring (outside)
mypie2, _ = ax.pie(subgroup_size, radius=1.1, labels=subgroup_names,
                   colors=[a(0.4), a(0.5), b(0.6), b(0.7), b(0.8), b(0.9
)])

plt.setp( mypie2, width=0.3, edgecolor='white')
plt.title('txn_description & status wise Txn Amount Share')
plt.margins(0,0)
plt.tight_layout()
# show it
plt.show()
```



3. Transaction Description

In [25]:
```
df['txn_description'].value_counts()
```

Out[25]:
```
SALES-POS     3934
POS           3783
PAYMENT       2600
PAY/SALARY     883
INTER BANK     742
PHONE BANK     101
Name: txn_description, dtype: int64
```

grouping the data by_txn_description_gender_count

In [26]:
```
by_txn_description_gender_count = df.groupby([
'txn_description', 'gender'
]).count()
```

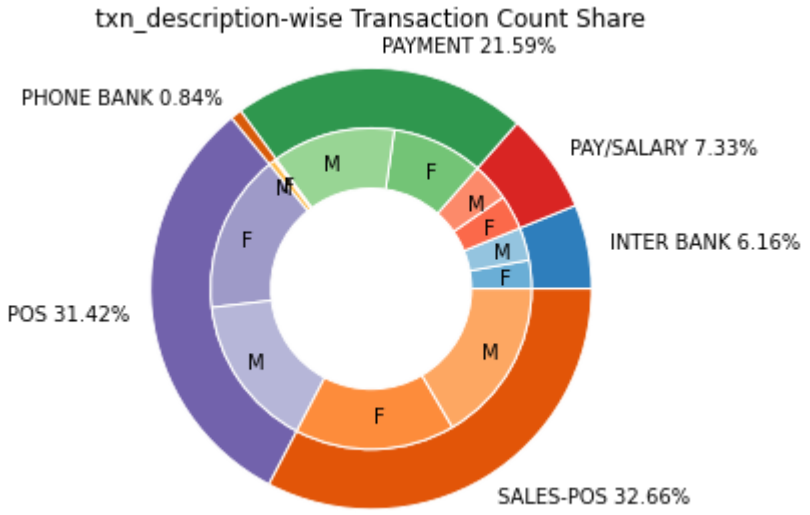
In [27]:
```
by_txn_description_gender_count = by_txn_description_gender_count.unstack
().fillna(0)
by_txn_description_gender_count
```

Out[27]:

	status		card_present_flag		account		currency		long_lat		merchant_id		fi
gender	F	M	F	M	F	M	F	M	F	M	F	M	F
txn_description													
INTER BANK	344	398	0	0	344	398	344	398	344	398	0	0	
PAY/SALARY	419	464	0	0	419	464	419	464	419	464	0	0	
PAYMENT	1107	1493	0	0	1107	1493	1107	1493	1107	1493	0	0	1
PHONE BANK	26	75	0	0	26	75	26	75	26	75	0	0	
POS	1921	1862	1921	1862	1921	1862	1921	1862	1921	1862	1921	1862	1
SALES-POS	1941	1993	1941	1993	1941	1993	1941	1993	1941	1993	1941	1993	1

In [28]:
```
#Donut_by_txn_description_gender_count
import matplotlib.pyplot as plt

# Make data: I have 6 groups and 12 subgroups
group_names=['INTER BANK 6.16%', 'PAY/SALARY 7.33%', 'PAYMENT 21.59%', 'P
HONE BANK 0.84%', 'POS 31.42%', 'SALES-POS 32.66%']
group_size=[742, 883,2600,101,3783,3934]
subgroup_names=['F', 'M', 'F', 'M', 'F', 'M', 'F', 'M', 'F', 'M', 'F',
'M']
subgroup_size=[344,398,419,464,1107,1493, 26,75,1921,1862,1941,1993]

# Create colors
a, b, c, d, e, f =[plt.cm.Blues, plt.cm.Reds, plt.cm.Greens,  plt.cm.YlOr
Br,  plt.cm.Purples, plt.cm.Oranges ]

# First Ring (outside)
fig, ax = plt.subplots()
ax.axis('equal')
mypie, _ = ax.pie(group_size, radius=1.1, labels=group_names, colors=[a(
0.7), b(0.7), c(0.7), d(0.7), e(0.7), f(0.7)])
plt.setp( mypie, width=0.3, edgecolor='white')
```

```
# Second Ring (Inside)
mypie2, _ = ax.pie(subgroup_size, radius=1.1-0.3, labels=subgroup_names,
labeldistance=0.8 ,
colors=[a(0.5), a(0.4), b(0.5), b(0.4), c(0.5), c(0.4), d(0.5), d(0.4), e
(0.5), e(0.4), f(0.5), f(0.4)])
plt.setp( mypie2, width=0.3, edgecolor='white')
plt.title('txn_description-wise Transaction Count Share')
plt.margins(0,0)

plt.tight_layout()
# show it
plt.show()
```



grouping the data by_txn_description_gender_sum

In [29]:
```
by_txn_description_gender_sum = df.groupby([
'txn_description', 'gender'
]).sum()
```

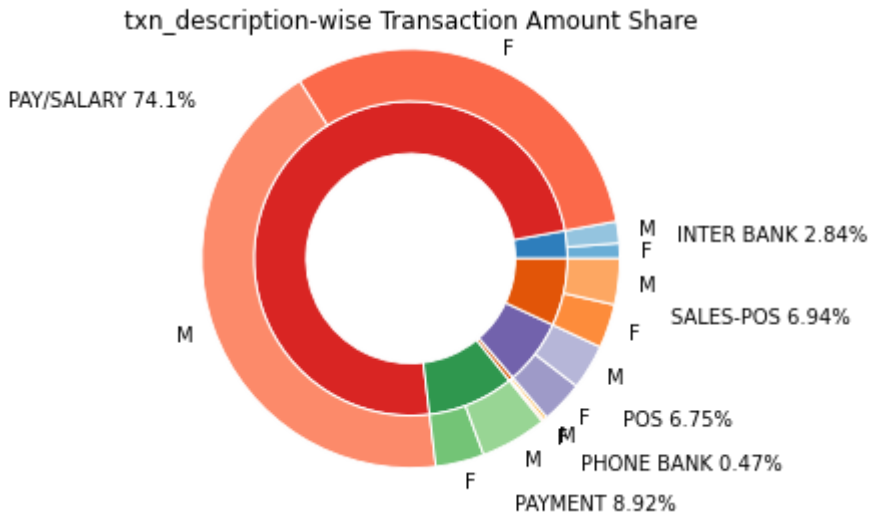
In [30]:
```
by_txn_description_gender_sum = by_txn_description_gender_sum.unstack().f
illna(0)
by_txn_description_gender_sum
```

Out[30]:

	card_present_flag		balance		age		amount	
gender	F	M	F	M	F	M	F	M
txn_description								
INTER BANK	0.0	0.0	5324032.46	12352890.27	10030	12966	26449.00	37882.00
PAY/SALARY	0.0	0.0	4913956.70	9428487.84	13502	15312	703656.23	972920.62
PAYMENT	0.0	0.0	16552464.60	33090435.75	34545	46744	85033.00	116761.00
PHONE BANK	0.0	0.0	2506722.85	1304692.91	978	2314	4237.00	6479.00
POS	1531.0	1494.0	20098842.85	24809052.35	57735	56074	74126.26	78734.98
SALES-POS	1571.0	1598.0	20052718.10	26648330.37	58603	59500	76821.14	80183.97

In [31]:
```
#Donut_by_txn_description_gender_sum
import matplotlib.pyplot as plt

# Make data: I have 6 groups and 12 subgroups
group_names=['INTER BANK 2.84%', 'PAY/SALARY 74.1%', 'PAYMENT 8.92%', 'PH
ONE BANK 0.47%', 'POS 6.75%', 'SALES-POS 6.94%']
group_size=[64331, 1676576.85, 201794, 10716, 152861.24, 157005.11]
subgroup_names=['F', 'M', 'F', 'M', 'F', 'M', 'F', 'M', 'F', 'M', 'F',
```

```
'M']
subgroup_size=[26449, 37882, 703656.23, 972920.62, 85033, 116761, 4237, 6
479, 74126.26, 78734.98, 76821.14, 80183.97]

# Create colors
a, b, c, d, e, f =[plt.cm.Blues, plt.cm.Reds, plt.cm.Greens,  plt.cm.YlOr
Br,  plt.cm.Purples, plt.cm.Oranges ]
# Second Ring (Inside)

fig, ax = plt.subplots()
ax.axis('equal')
mypie, _ = ax.pie(group_size, radius=1.2-0.3, labels=group_names, labeldi
stance=1.7,
                  colors=[a(0.7), b(0.7), c(0.7), d(0.7), e(0.7), f(0.7
)])
plt.setp( mypie, width=0.3, edgecolor='white')

# First Ring (outside)
mypie2, _ = ax.pie(subgroup_size, radius=1.2 , labels=subgroup_names,
colors=[a(0.5), a(0.4), b(0.5), b(0.4), c(0.5), c(0.4), d(0.5), d(0.4), e
(0.5), e(0.4), f(0.5), f(0.4)])
plt.setp( mypie2, width=0.3, edgecolor='white')
plt.title('txn_description-wise Transaction Amount Share')
plt.margins(0,0)

# show it
plt.show()
```



grouping the data by_movement_txn_description_count

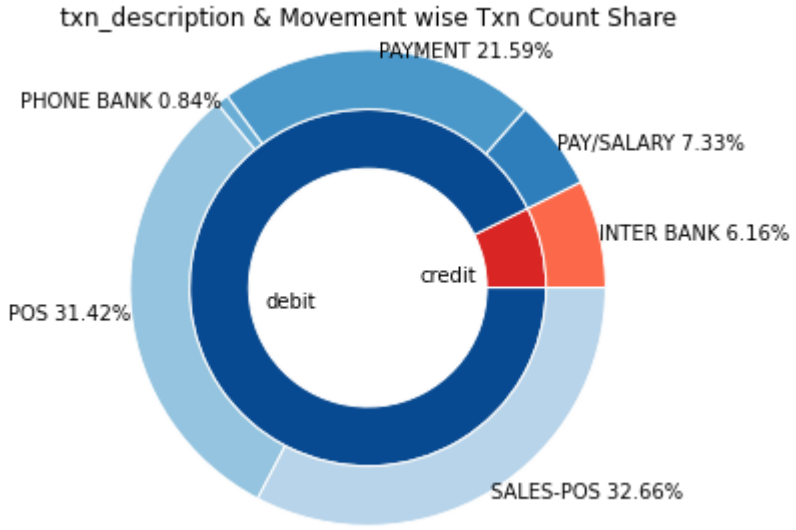
In [32]:
```
by_movement_txn_description_count = df.groupby([
'movement', 'txn_description'
]).count()
```

In [33]:
```
by_movement_txn_description_count = by_movement_txn_description_count.uns
tack().fillna(0)
by_movement_txn_description_count
```

Out[33]:

	status						card_present_flag		
txn_description	INTER BANK	PAY/SALARY	PAYMENT	PHONE BANK	POS	SALES-POS	INTER BANK	PAY/SALARY	PAY
movement									
credit	0.0	883.0	0.0	0.0	0.0	0.0	0.0	0.0	
debit	742.0	0.0	2600.0	101.0	3783.0	3934.0	0.0	0.0	

In [34]:
```
#Donut_by_movement_txn_description_count
import matplotlib.pyplot as plt

# Make data: I have 2 groups and 6 subgroups
group_names=['credit', 'debit']
group_size=[883.0, 11160.0 ]
subgroup_names=['INTER BANK 6.16%', 'PAY/SALARY 7.33%', 'PAYMENT 21.59%',
'PHONE BANK 0.84%', 'POS 31.42%', 'SALES-POS 32.66%']
subgroup_size=[883.0, 742.0, 2600.0, 101.0, 3783.0, 3934.0 ]

# Create colors
a, b =[plt.cm.Reds, plt.cm.Blues]

# First Ring (outside)
fig, ax = plt.subplots()
ax.axis('equal')
mypie, _ = ax.pie(group_size, radius=1.2-0.3, labels=group_names, labeldi
stance=0.3, colors=[a(0.7), b(0.9)])
plt.setp( mypie, width=0.3, edgecolor='white')

# Second Ring (Inside)
mypie2, _ = ax.pie(subgroup_size, radius=1.2, labels=subgroup_names, labe
ldistance=1,
                   colors=[a(0.5), b(0.7), b(0.6), b(0.5), b(0.4), b(0.3
)])

plt.setp( mypie2, width=0.3, edgecolor='white')
plt.title('txn_description & Movement wise Txn Count Share')
plt.margins(0,0)
plt.tight_layout()
# show it
plt.show()
```



grouping the data by_movement_txn_description_sum

In [35]:
```
by_movement_txn_description_sum = df.groupby([
'movement', 'txn_description'
```

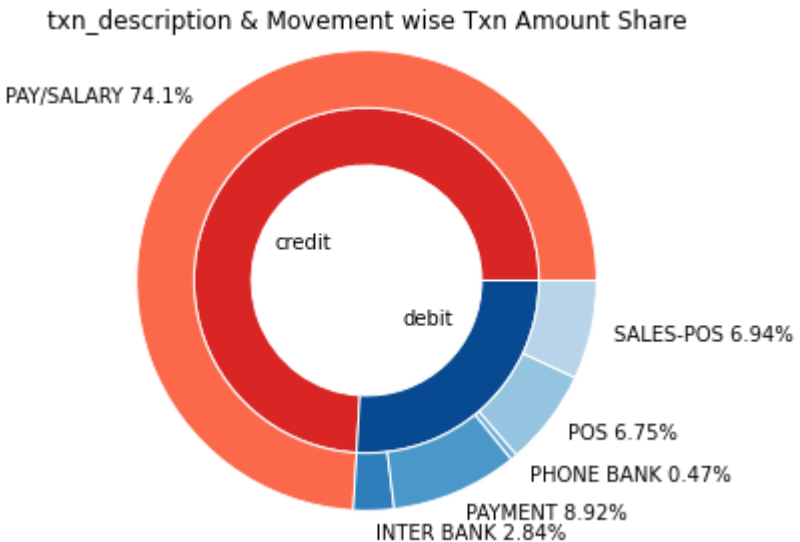
```
]).sum()
```

In [36]:
```
by_movement_txn_description_sum = by_movement_txn_description_sum.unstack
().fillna(0)
by_movement_txn_description_sum
```

Out[36]:

	card_present_flag						balance	
txn_description	INTER BANK	PAY/SALARY	PAYMENT	PHONE BANK	POS	SALES-POS	INTER BANK	PAY/SALARY
movement								
credit	0.0	0.0	0.0	0.0	0.0	0.0	0.00	14342444.54
debit	0.0	0.0	0.0	0.0	3025.0	3169.0	17676922.73	0.00

In [37]:
```
#Donut_by_movement_txn_description_sum
import matplotlib.pyplot as plt

# Make data: I have 2 groups and 6 subgroups
group_names=['credit', 'debit']
group_size=[1676576.85, 586707.35 ]
subgroup_names=['PAY/SALARY 74.1%', 'INTER BANK 2.84%', 'PAYMENT 8.92%',
'PHONE BANK 0.47%', 'POS 6.75%', 'SALES-POS 6.94%']
subgroup_size=[1676576.85, 64331.00, 201794.0, 10716.00, 152861.24, 15700
5.11 ]

# Create colors
a, b =[plt.cm.Reds, plt.cm.Blues]

# First Ring (outside)
fig, ax = plt.subplots()
ax.axis('equal')
mypie, _ = ax.pie(group_size, radius=1.2-0.3, labels=group_names, labeldi
stance=0.3, colors=[a(0.7), b(0.9)])
plt.setp( mypie, width=0.3, edgecolor='white')

# Second Ring (Inside)
mypie2, _ = ax.pie(subgroup_size, radius=1.2, labels=subgroup_names,
                   colors=[a(0.5), b(0.7), b(0.6), b(0.5), b(0.4), b(0.3
)])

plt.setp( mypie2, width=0.3, edgecolor='white')
plt.title('txn_description & Movement wise Txn Amount Share')
plt.margins(0,0)
plt.tight_layout()
# show it
plt.show()
```



4. Month Wise Transaction Analysis

day and month extraction from date column

In [38]:
```
# converting the date column to pandas Timestap
df_new['date'] = pd.to_datetime(df_new['date'])
```

In [39]:
```
# extracting day name
df_new['day_name'] = df_new['date'].dt.day_name()
df_new['day_name'].head()
```

Out[39]:
```
0    Wednesday
1    Wednesday
2    Wednesday
3    Wednesday
4    Wednesday
Name: day_name, dtype: object
```

In [40]:
```
# extracting month name
df_new['month_name'] = df_new['date'].dt.month_name()
df_new['month_name'].head()
```

Out[40]:
```
0    August
1    August
2    August
3    August
4    August
Name: month_name, dtype: object
```

In [41]:
```
# months generated
df_new['month_name'].value_counts()
```

Out[41]:
```
October      4087
September    4013
August       3943
Name: month_name, dtype: int64
```

In [42]:
```
df_new.head()
```

Out[42]:

	status	card_present_flag	bpay_biller_code	account	currency	long_lat	txn_description
0	authorized	1.0	NaN	ACC-1598451071	AUD	153.41 -27.95	POS
1	authorized	0.0	NaN	ACC-1598451071	AUD	153.41 -27.95	SALES-POS
2	authorized	1.0	NaN	ACC-1222300524	AUD	151.23 -33.94	POS
3	authorized	1.0	NaN	ACC-1037050564	AUD	153.10 -27.66	SALES-POS
4	authorized	1.0	NaN	ACC-1598451071	AUD	153.41 -27.95	SALES-POS

grouping the data by_month_name_movement_count

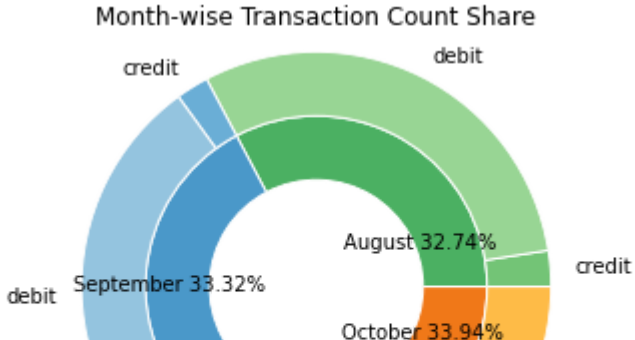
In [43]:
```
by_month_name_movement_count = df_new.groupby([
'month_name', 'movement'
]).count()
```

In [44]:
```
by_month_name_movement_count = by_month_name_movement_count.unstack().fil
lna(0)
by_month_name_movement_count
```

Out[44]:

	status		card_present_flag		bpay_biller_code		account		currency		long_lat	
movement	credit	debit	credit	debit	credit	debit	credit	debit	credit	debit	credit	c
month_name												
August	298	3645	0	2535	298	1	298	3645	298	3645	298	
October	313	3774	0	2581	313	0	313	3774	313	3774	313	
September	272	3741	0	2601	272	1	272	3741	272	3741	272	

In [45]:
```
#Donut_by_month_name_movement_count
import matplotlib.pyplot as plt

# Make data: I have 3 groups and 6 subgroups
group_names=['August 32.74%', 'September 33.32%', 'October 33.94%']
group_size=[3943, 4013, 4087 ]
subgroup_names=['credit', 'debit', 'credit', 'debit', 'credit', 'debit']
subgroup_size=[298 , 3645 , 272, 3741,313, 3774 ]

# Create colors
a, b, c =[plt.cm.Greens, plt.cm.Blues, plt.cm.YlOrBr ]

# First Ring (Inside)
fig, ax = plt.subplots()
ax.axis('equal')
mypie, _ = ax.pie(group_size, radius=1.1-0.3, labels=group_names, labeldi
stance=0.3, colors=[a(0.6), b(0.6), c(0.6)])
plt.setp( mypie, width=0.3, edgecolor='white')

# Second Ring (outside)
mypie2, _ = ax.pie(subgroup_size, radius=1.1, labels=subgroup_names,
                   colors=[a(0.5), a(0.4), b(0.5), b(0.4), c(0.5), c(0.4
)])

plt.setp( mypie2, width=0.3, edgecolor='white')
plt.title('Month-wise Transaction Count Share')
plt.margins(0,0)
plt.tight_layout()
# show it
plt.show()
```



grouping the data by_month_name_movement_sum

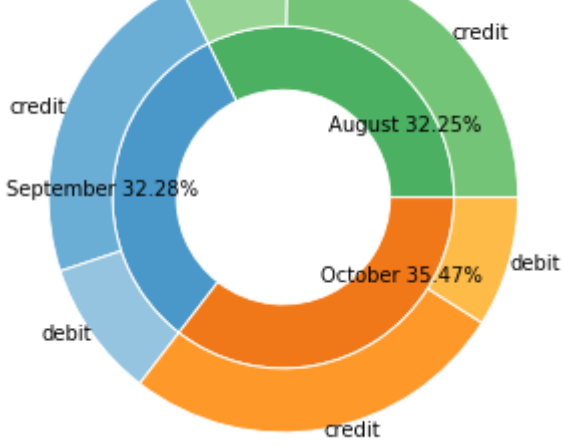
In [46]:
```
by_month_name_movement_sum = df_new.groupby([
'month_name', 'movement'
]).sum()
```

In [47]:
```
by_month_name_movement_sum = by_month_name_movement_sum.unstack().fillna(
0)
by_month_name_movement_sum
```

Out[47]:

	card_present_flag		merchant_code		balance		age		amount
movement	credit	debit	credit	debit	credit	debit	credit	debit	credit
month_name									
August	0.0	2041.0	0.0	0.0	3712788.11	38848539.90	9748	110446	559814.33
October	0.0	2074.0	0.0	0.0	6210112.27	69199090.26	10172	115406	600166.91
September	0.0	2079.0	0.0	0.0	4419544.16	54692552.35	8894	113637	516595.61

In [48]:
```
#Donut_by_month_name_movement_sum
import matplotlib.pyplot as plt

# Make data: I have 3 groups and 6 subgroups
group_names=['August 32.25%', 'September 32.28%', 'October 35.47%']
group_size=[729935.52, 730550.21, 802798.47 ]
subgroup_names=['credit', 'debit', 'credit', 'debit', 'credit', 'debit']
subgroup_size=[559814.33 , 170121.19 , 516595.61, 213954.60,600166.91, 20
2631.56 ]

# Create colors
a, b, c =[plt.cm.Greens, plt.cm.Blues, plt.cm.YlOrBr ]

# First Ring (Inside)
fig, ax = plt.subplots()
ax.axis('equal')
mypie, _ = ax.pie(group_size, radius=1.1-0.3, labels=group_names, labeldi
stance=0.5, colors=[a(0.6), b(0.6), c(0.6)])
plt.setp( mypie, width=0.3, edgecolor='white')

# Second Ring (outside)
mypie2, _ = ax.pie(subgroup_size, radius=1.1, labels=subgroup_names, labe
ldistance=1,
                   colors=[a(0.5), a(0.4), b(0.5), b(0.4), c(0.5), c(0.4
)])

plt.setp( mypie2, width=0.3, edgecolor='white')
plt.title('Month-wise Total Transaction Amount Share')
plt.margins(0,0)
plt.tight_layout()
# show it
plt.show()
```



Apparently all three months almost same weightage in terms of Txn count and Txn amount.

But, highest transaction has occurred in the month of October among these three.

5. Weekdays Wise Transaction Analysis

grouping the data by_day_name_movement_count

In [49]:
```
by_day_name_movement_count = df_new.groupby([
'day_name', 'movement'
]).count()
```

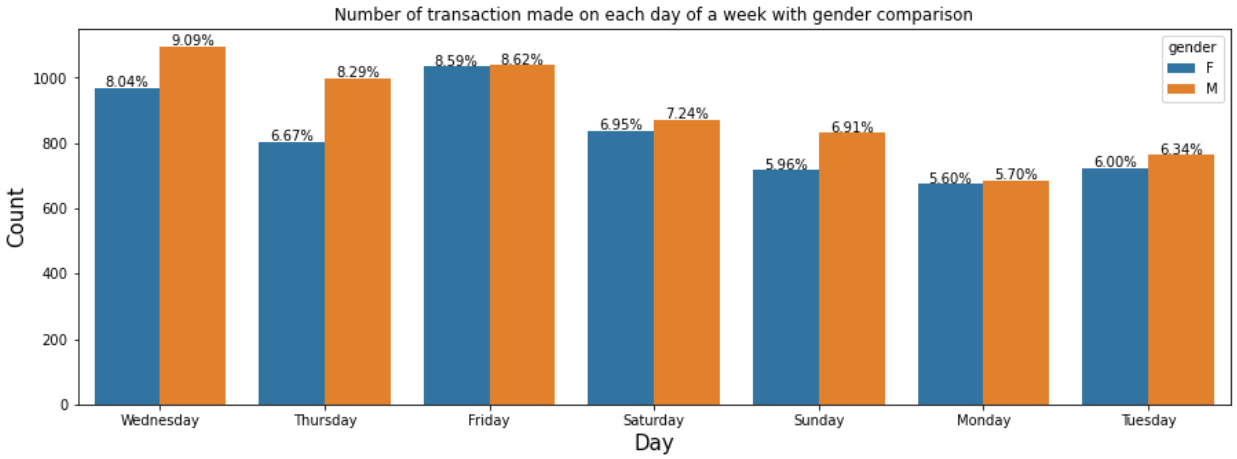
In [50]:
```
by_day_name_movement_count = by_day_name_movement_count.unstack().fillna(
0)
by_day_name_movement_count
```

Out[50]:

	status		card_present_flag		bpay_biller_code		account		currency		long_
movement	credit	debit	credit	debit	credit	debit	credit	debit	credit	debit	credit
day_name											
Friday	201.0	1872.0	0.0	1317.0	201.0	0.0	201.0	1872.0	201.0	1872.0	201.0
Monday	207.0	1153.0	0.0	742.0	207.0	0.0	207.0	1153.0	207.0	1153.0	207.0
Saturday	0.0	1709.0	0.0	1337.0	0.0	1.0	0.0	1709.0	0.0	1709.0	0.0
Sunday	0.0	1550.0	0.0	1126.0	0.0	0.0	0.0	1550.0	0.0	1550.0	0.0
Thursday	143.0	1658.0	0.0	1103.0	143.0	0.0	143.0	1658.0	143.0	1658.0	143.0
Tuesday	160.0	1327.0	0.0	841.0	160.0	1.0	160.0	1327.0	160.0	1327.0	160.0
Wednesday	172.0	1891.0	0.0	1251.0	172.0	0.0	172.0	1891.0	172.0	1891.0	172.0

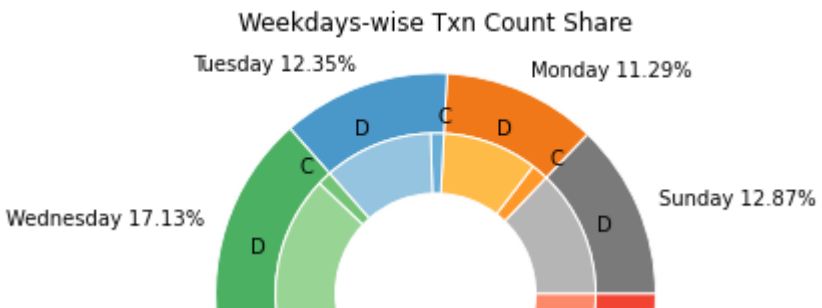
In [51]:
```
# visualize day wise gender transaction count
plt.figure(figsize=(15,5))
ax = sns.countplot(x="day_name", hue="gender", data=df_new) # for Seaborn
version 0.7 and more
total = float(len(df))
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2.,
            height + 3,
            '{:.2%}'.format(height/total),
            ha="center")
plt.ylabel("Count",fontsize=15)
plt.xlabel("Day",fontsize=15)
plt.title('Number of transaction made on each day of a week with gender c
```

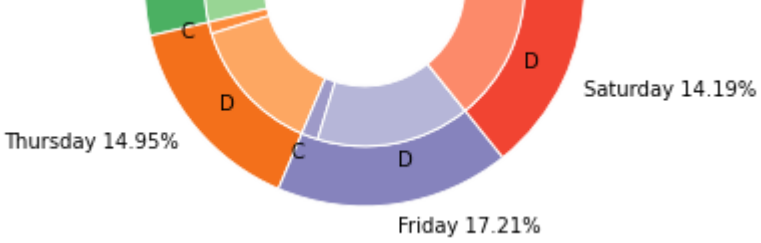
```
omparison')
plt.show()
```



In [52]:

```
#Donut_by_day_name_movement_count
import matplotlib.pyplot as plt

# Make data: I have 7 groups and 12 subgroups
group_names=['Sunday 12.87%', 'Monday 11.29%', 'Tuesday 12.35%', 'Wednesd
ay 17.13%', 'Thursday 14.95%', 'Friday 17.21%',
             'Saturday 14.19%']
group_size=[1550, 1360.0, 1487.0, 2063.0, 1801.0, 2073.0, 1709.0]
subgroup_names=[ '', 'D', 'C', 'D','C', 'D', 'C', 'D', 'C', 'D', 'C', 'D'
,'', 'D' ]
subgroup_size=[0, 1550, 207.0, 1153.0, 160.0, 1327.0, 172.0, 1891.0 , 14
3.0, 1658.0, 201.0, 1872.0, 0, 1709.0 ]

# Create colors
a, b, c, d, e, f, g =[plt.cm.Greys, plt.cm.YlOrBr, plt.cm.Blues, plt.cm.G
reens, plt.cm.Oranges, plt.cm.Purples, plt.cm.Reds ]

# First Ring (outside)
fig, ax = plt.subplots()
ax.axis('equal')
mypie, _ = ax.pie(group_size, radius=1.1, labels=group_names, colors=[a(
0.6), b(0.6), c(0.6)
                                                                      ,d(
0.6), e(0.6), f(0.6), g(0.6)])
plt.setp( mypie, width=0.3, edgecolor='white')

# Second Ring (Inside)
mypie2, _ = ax.pie(subgroup_size, radius=1.1-0.3, labels=subgroup_names,
                   colors=[a(0.5), a(0.4), b(0.5), b(0.4), c(0.5), c(0.4
), d(0.5), d(0.4), e(0.5), e(0.4), f(0.5), f(0.4),
                           g(0.5), g(0.4)])

plt.setp( mypie2, width=0.3, edgecolor='white')
plt.title('Weekdays-wise Txn Count Share')
plt.margins(0,0)
plt.tight_layout()
# show it
plt.show()
```





grouping the data by_day_name_movement_sum

In [53]:
```
by_day_name_movement_sum = df_new.groupby([
'day_name', 'movement'
]).sum()
```

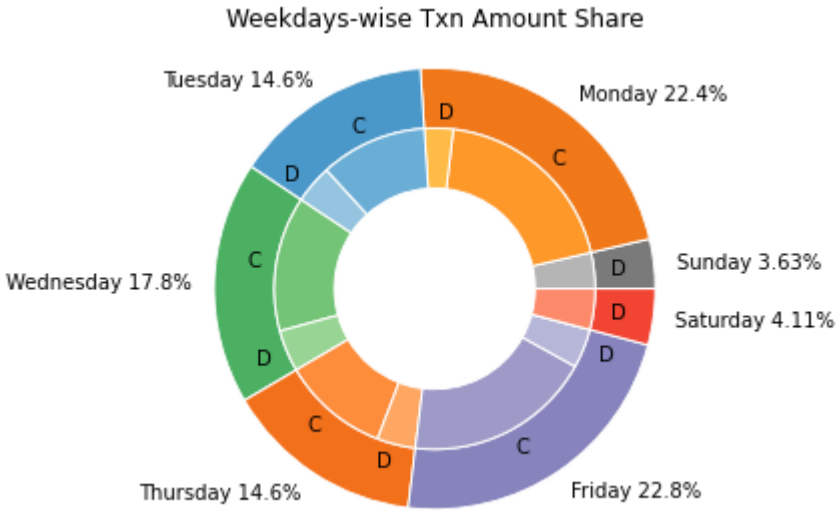
In [54]:
```
by_day_name_movement_sum = by_day_name_movement_sum.unstack().fillna(0)
by_day_name_movement_sum
```

Out[54]:

	card_present_flag		merchant_code		balance		age		amount
movement	credit	debit	credit	debit	credit	debit	credit	debit	credit
day_name									
Friday	0.0	1058.0	0.0	0.0	4268622.08	24067712.78	6806.0	57210.0	427670.18
Monday	0.0	588.0	0.0	0.0	4751544.97	16212141.07	6514.0	34184.0	442167.65
Saturday	0.0	1059.0	0.0	0.0	0.00	26263096.84	0.0	52644.0	0.00
Sunday	0.0	892.0	0.0	0.0	0.00	21139592.40	0.0	45963.0	0.00
Thursday	0.0	901.0	0.0	0.0	1845416.91	28554212.37	5050.0	51987.0	243725.41
Tuesday	0.0	679.0	0.0	0.0	1758344.41	20199826.76	5184.0	40897.0	255243.33
Wednesday	0.0	1017.0	0.0	0.0	1718516.17	26303600.29	5260.0	56604.0	307770.28

In [55]:
```
#Donut_by_day_name_movement_sum
import matplotlib.pyplot as plt

# Make data: I have 7 groups and 12 subgroups
group_names=['Sunday 3.63%', 'Monday 22.4%', 'Tuesday 14.6%', 'Wednesday
 17.8%', 'Thursday 14.6%', 'Friday 22.8%',
             'Saturday 4.11%']
group_size=[82174.56, 507580.56000000006, 331402.77, 402729.72000000003,
331402.77, 516909.83999999997, 93002.56]
subgroup_names=[ '', 'D', 'C', 'D','C', 'D', 'C', 'D', 'C', 'D', 'C', 'D'
,'', 'D' ]
subgroup_size=[0, 82174.56, 442167.65, 65412.91, 243725.41, 87677.36,3077
70.28, 94959.44, 243725.41, 87677.36,
               427670.18, 89239.66, 0, 93002.56]

# Create colors
a, b, c, d, e, f, g =[plt.cm.Greys, plt.cm.YlOrBr, plt.cm.Blues, plt.cm.G
reens, plt.cm.Oranges, plt.cm.Purples, plt.cm.Reds ]

# First Ring (outside)
fig, ax = plt.subplots()
ax.axis('equal')
mypie, _ = ax.pie(group_size, radius=1.1, labels=group_names, colors=[a(
0.6), b(0.6), c(0.6)
                                                                      ,d(
0.6), e(0.6), f(0.6), g(0.6)])
plt.setp( mypie, width=0.3, edgecolor='white')

# Second Ring (Inside)
```

```
mypie2, _ = ax.pie(subgroup_size, radius=1.1-0.3, labels=subgroup_names,
                   colors=[a(0.5), a(0.4), b(0.5), b(0.4), c(0.5), c(0.4
), d(0.5), d(0.4), e(0.5), e(0.4), f(0.5), f(0.4),
                           g(0.5), g(0.4)])

plt.setp( mypie2, width=0.3, edgecolor='white')
plt.title('Weekdays-wise Txn Amount Share')
plt.margins(0,0)
plt.tight_layout()
# show it
plt.show()
```



There is no significant difference in terms of Txn counts throughout the week.

No credit Txn in Saturdays & Sundays.

Majority of Credit Txn amount has been reflected on Mondays and Fridays whereas Debit Amount Txn is more or less same throughout the week.

6. Merchant State Wise Transaction Analysis

grouping the data by_merchant_state_gender_count

In [56]:
```
by_merchant_state_gender_count = df_new.groupby([
'merchant_state','gender'
]).count()
```

In [57]:
```
by_merchant_state_gender_count = by_merchant_state_gender_count.unstack()
.fillna(0)
by_merchant_state_gender_count
```

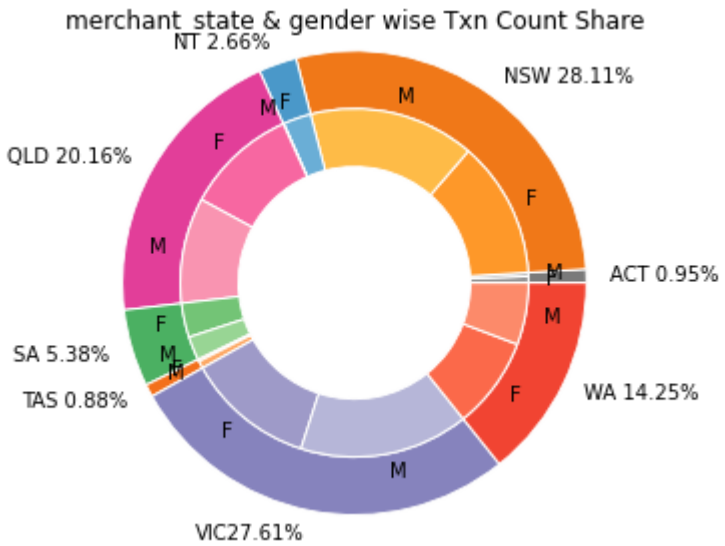
Out[57]:

	status		card_present_flag		bpay_biller_code		account		currency		long_lat		t
gender	F	M	F	M	F	M	F	M	F	M	F	M	F
merchant_state													
ACT	46	27	46	27	0	0	46	27	46	27	46	27	
NSW	980	1189	980	1189	0	1	980	1189	980	1189	980	1189	
NT	200	5	200	5	0	0	200	5	200	5	200	5	
QLD	800	756	800	756	0	0	800	756	800	756	800	756	
SA	245	170	245	170	0	0	245	170	245	170	245	170	

TAS	16	52	16	52	0	0	16	52	16	52	16	52
VIC	918	1213	918	1213	0	1	918	1213	918	1213	918	1213
WA	657	443	657	443	0	0	657	443	657	443	657	443

In [58]:

```
#Donut_by_merchant_state_gender_count
import matplotlib.pyplot as plt

# Make data: I have 8 groups and 16 subgroups
group_names=['ACT 0.95%', 'NSW 28.11%', 'NT 2.66%', 'QLD 20.16%', 'SA 5.38%', 'TAS 0.88%', 'VIC27.61%', 'WA 14.25%' ]
group_size=[73 ,2169 ,205,1556,415,68 ,2131,1100]
subgroup_names=[ 'F', 'M', 'F', 'M', 'F', 'M', 'F', 'M', 'F', 'M', 'F', 'M', 'F', 'M', 'F', 'M']
subgroup_size=[46,27, 980,1189, 200,5,800,756,245,170, 16,52,918,1213,657,443]

# Create colors
a, b, c, d, e, f, g, h =[plt.cm.Greys, plt.cm.YlOrBr, plt.cm.Blues, plt.cm.RdPu, plt.cm.Greens, plt.cm.Oranges,
                         plt.cm.Purples, plt.cm.Reds ]

# First Ring (outside)
fig, ax = plt.subplots()
ax.axis('equal')
mypie, _ = ax.pie(group_size, radius=1.2, labels=group_names, colors=[a(0.6), b(0.6), c(0.6)
                                                                      ,d(0.6), e(0.6), f(0.6), g(0.6), h(0.6)])
plt.setp( mypie, width=0.3, edgecolor='white')

# Second Ring (Inside)
mypie2, _ = ax.pie(subgroup_size, radius=1.2-0.3, labels=subgroup_names,
                   colors=[a(0.5), a(0.4), b(0.5), b(0.4), c(0.5), c(0.4), d(0.5), d(0.4), e(0.5), e(0.4), f(0.5), f(0.4),
                           g(0.5), g(0.4), h(0.5), h(0.4)])

plt.setp( mypie2, width=0.3, edgecolor='white')
plt.title('merchant_state & gender wise Txn Count Share')
plt.margins(0,0)
plt.tight_layout()

# show it
plt.show()
```



grouping the data by_merchant_state_gender_sum

In [59]:
```
by_merchant_state_gender_sum = df_new.groupby([
'merchant_state','gender'
]).sum()
```

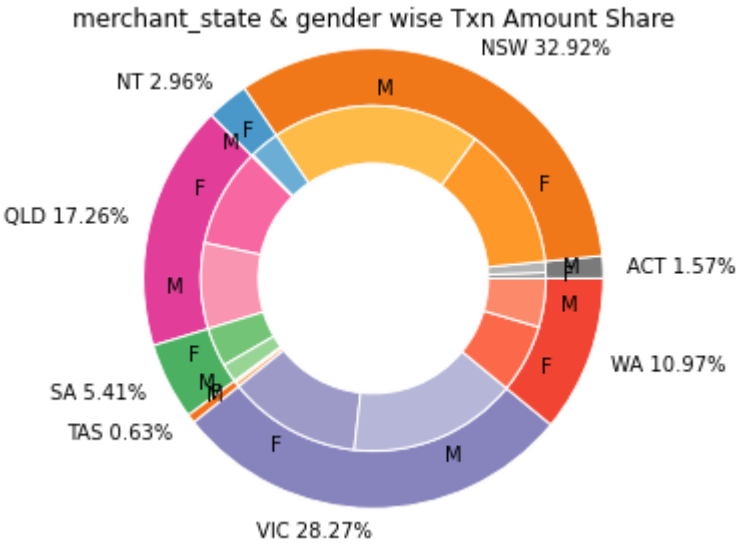
In [60]:
```
by_merchant_state_gender_sum = by_merchant_state_gender_sum.unstack().fil
lna(0)
by_merchant_state_gender_sum
```

Out[60]:

	card_present_flag		merchant_code		balance		age		amount
gender	F	M	F	M	F	M	F	M	F
merchant_state									
ACT	39.0	25.0	0.0	0.0	663951.76	247767.46	1433	794	1657.4
NSW	784.0	928.0	0.0	0.0	8678729.83	11814886.13	28423	36303	41430.8
NT	163.0	4.0	0.0	0.0	946134.77	190059.32	5346	209	8741.4
QLD	645.0	621.0	0.0	0.0	2395555.00	8639602.51	24248	22298	28611.0
SA	202.0	129.0	0.0	0.0	2770261.48	1857125.51	7598	5222	11349.7
TAS	15.0	45.0	0.0	0.0	43436.35	733876.61	954	1372	622.7
VIC	727.0	992.0	0.0	0.0	20539592.58	21341479.14	31013	37271	38626.0
WA	527.0	348.0	0.0	0.0	4113899.18	6632586.04	17323	12105	19908.1

In [61]:
```
#Donut_by_merchant_state_gender_sum
import matplotlib.pyplot as plt

# Make data: I have 8 groups and 16 subgroups
group_names=['ACT 1.57%', 'NSW 32.92%', 'NT 2.96%', 'QLD 17.26%', 'SA 5.4
1%', 'TAS 0.63%', 'VIC 28.27%', 'WA 10.97%' ]
group_size=[4876.68 ,102021.77,9168.89 ,53483.45,16776.57,1962.93 ,87584.
00 ,33992.06]
subgroup_names=[ 'F', 'M', 'F', 'M', 'F', 'M', 'F', 'M', 'F', 'M', 'F',
'M', 'F', 'M', 'F', 'M']
subgroup_size=[1657.44 ,3219.24,41430.88 ,60590.89 , 8741.42 ,427.47, 286
11.05 , 24872.40,11349.73 ,5426.84,
               622.72 ,1340.21, 38626.01 ,48957.99 ,19908.15 ,14083.91]

# Create colors
a, b, c, d, e, f, g, h =[plt.cm.Greys, plt.cm.YlOrBr, plt.cm.Blues, plt.c
m.RdPu, plt.cm.Greens, plt.cm.Oranges,
                         plt.cm.Purples, plt.cm.Reds ]

# First Ring (outside)
fig, ax = plt.subplots()
ax.axis('equal')
mypie, _ = ax.pie(group_size, radius=1.2, labels=group_names, colors=[a(
0.6), b(0.6), c(0.6)
                                                                      ,d(
0.6), e(0.6), f(0.6), g(0.6), h(0.6)])
plt.setp( mypie, width=0.3, edgecolor='white')

# Second Ring (Inside)
mypie2, _ = ax.pie(subgroup_size, radius=1.2-0.3, labels=subgroup_names,
                   colors=[a(0.5), a(0.4), b(0.5), b(0.4), c(0.5), c(0.4
), d(0.5), d(0.4), e(0.5), e(0.4), f(0.5), f(0.4),
                           g(0.5), g(0.4), h(0.5), h(0.4)])

plt.setp( mypie2, width=0.3, edgecolor='white')
plt.title('merchant_state & gender wise Txn Amount Share')
plt.margins(0,0)
plt.tight_layout()
```

```
# show it
plt.show()
```



Almost 60% of Total Debit Txn amount is going to the merchants of VIC and NSW.

Females have contributed more than the males in QLD, SA WA and NT.

In total Females have done more Debit Txn than males.

7. Transaction Trend

grouping the data by_date_count

In [65]:
```
by_date_count = df_new.groupby([
'date',
]).count()
```

In [66]:
```
drop_features2 = ['card_present_flag', 'merchant_code', 'balance', 'age',
'amount','status','bpay_biller_code', 'account',
                  'currency', 'long_lat', 'txn_description', 'merchant_id'
, 'merchant_code', 'first_name', 'gender',
                   'merchant_suburb', 'merchant_state', 'extraction', 'tra
nsaction_id', 'transaction_id', 'country',
                  'customer_id', 'merchant_long_lat', 'movement', 'day_nam
e', 'month_name']
by_date_count.drop(drop_features2, axis = 1, inplace = True )
```

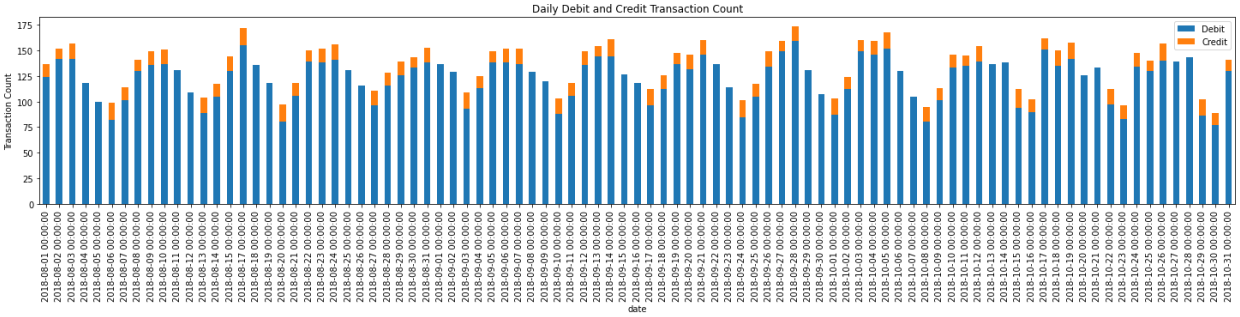
In [73]:
```
by_date_count = by_date_count.unstack()
by_date_count.head()
```

Out[73]:

	Debit_amount	Credit_amount
date		
2018-08-01	124	13
2018-08-02	142	10
2018-08-03	142	15
2018-08-04	118	0
2018-08-05	100	0

In [74]:
```
by_date_count.rename(columns = {'Debit_amount':'Debit', 'Credit_amount':
'Credit'}, inplace = True)
```

In [75]:
```
# Visualize this data in bar plot
ax = (by_date_count).plot(
kind='bar',
figsize=(25, 4),
stacked=True
)

ax.set_ylabel('Transaction Count')
plt.title('Daily Debit and Credit Transaction Count')
plt.show()
```



grouping the data by_date_sum

In [76]:
```
by_date_sum = df_new.groupby([
'date',
]).sum()
```

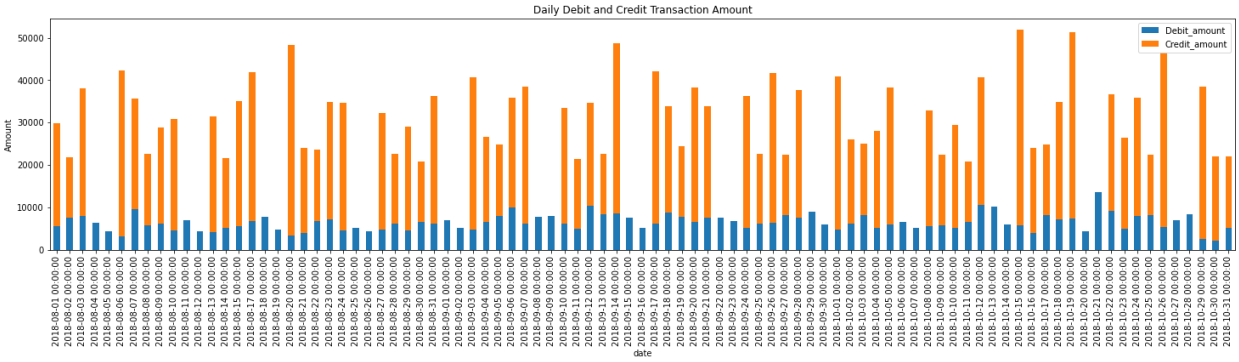
In [81]:
```
by_date_sum = by_date_sum.unstack()
by_date_sum.head()
```

Out[81]:

date	card_present_flag	merchant_code	balance	age	amount	Debit_amount	Credit_amoun
2018-08-01	63.0	0.0	1360954.62	4142.0	29867.94	5546.44	24321.5
2018-08-02	85.0	0.0	2122469.92	4787.0	21786.32	7558.09	14228.2
2018-08-03	79.0	0.0	1599482.51	4985.0	38096.58	8035.90	30060.6
2018-08-04	74.0	0.0	968403.51	3662.0	6296.05	6296.05	0.0
2018-08-05	54.0	0.0	1329752.54	2991.0	4426.50	4426.50	0.0

In [82]:
```
drop_features = ['card_present_flag', 'merchant_code', 'balance', 'age',
'amount']
by_date_sum.drop(drop_features, axis = 1, inplace = True )
```

In [83]:
```
# Visualize this data in bar plot
ax = (by_date_sum).plot(
kind='bar',
figsize=(25, 5),
stacked=True
)
ax.set_ylabel('Amount')
plt.title('Daily Debit and Credit Transaction Amount')
plt.show()
```



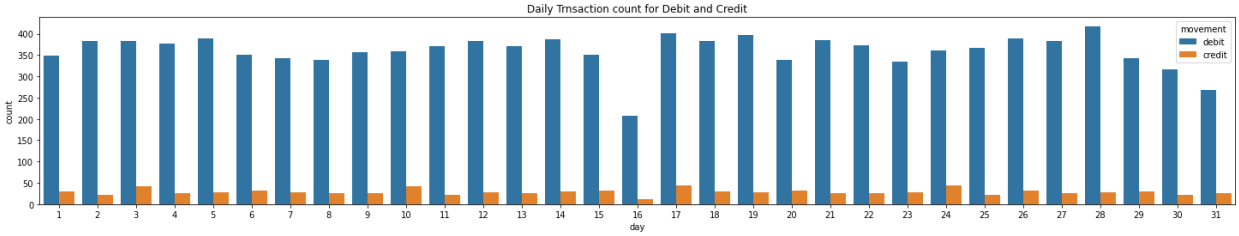
In both cases of daily transaction count and total transaction amount, there is a cyclic trend over the whole time period.

No. of Credit transactions are very less compared to no. of Debit Transactions where as majority contribution of transaction amount is coming from Credit Transactions. On another note, we see that, there is no credit transactions at frequent intervals, on weekends.

8. Transactions on various days of a Month

In [84]:
```
df_new['day'] = pd.DatetimeIndex(df_new['date']).day
plt.figure(figsize=(25,4))
plt.title('Daily Trnsaction count for Debit and Credit')
sns.countplot(x=df_new['day'],hue='movement',data=df_new)
```

Out[84]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x13b70d88>
```



In [85]:
```
# grouping the data by_day_movement_sum
by_day_movement_sum = df_new.groupby([
'day',
]).sum()
```

In [90]:
```
by_day_movement_sum = by_day_movement_sum.unstack().fillna(0)
by_day_movement_sum.head()
```

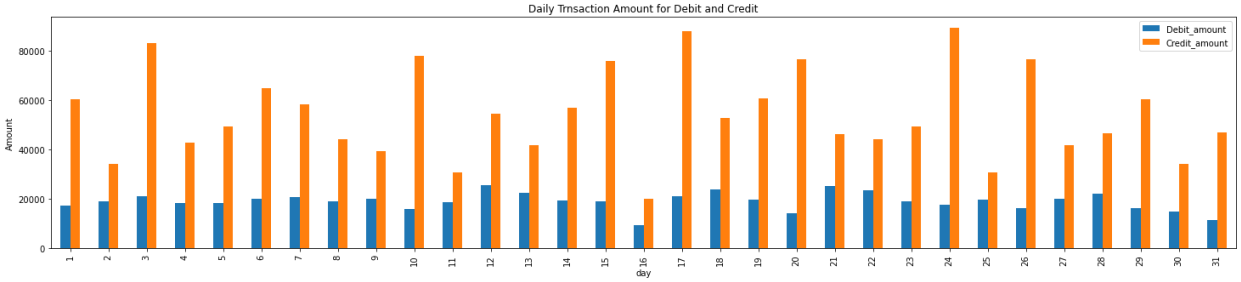
Out[90]:

day	card_present_flag	merchant_code	balance	age	amount	Debit_amount	Credit_amour
1	199.0	0.0	5032438.74	11144.0	77551.86	17200.40	60351.4
2	212.0	0.0	5748222.34	12223.0	53073.61	18872.36	34201.2
3	208.0	0.0	6228701.26	12782.0	103795.05	20885.43	82909.6

4	212.0	0.0	4677514.33	12718.0	60825.68	18088.96	42736.7
5	216.0	0.0	6275648.32	12641.0	67536.56	18396.24	49140.3

In [91]:
```
drop_features = ['card_present_flag', 'merchant_code', 'balance', 'age',
'amount']
by_day_movement_sum.drop(drop_features, axis = 1, inplace = True )
```

In [92]:
```
# Visualize this data in bar plot
ax = (by_day_movement_sum).plot(
kind='bar',
figsize=(25,5),

)
ax.set_ylabel('Amount')
plt.title('Daily Trnsaction Amount for Debit and Credit')
plt.show()
```



We can observe a downward trend in no. of Debit Txn counts and Debit amount at the end of the months.

There are peaks in Credit Amounts at a regular interval on the dates 3, 10, 17 and 24. Other than that, we can see cyclic nature in both Credit and Debit amount over the days.

9. Customer Analysis

In [1]:
```
#importing packages for data analysis
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import datetime
import matplotlib.dates
import warnings
warnings.filterwarnings("ignore")
pd.options.display.max_columns = None
%matplotlib inline
```

In [2]:
```
#importing dataset
df_original = pd.read_excel('ANZ synthesised transaction dataset.xlsx')
```

In [3]:
```
df_original.head()
```

Out[3]:

	status	card_present_flag	bpay_biller_code	account	currency	long_lat	txn_description
0	authorized	1.0	NaN	ACC-1598451071	AUD	153.41 -27.95	POS
1	authorized	0.0	NaN	ACC-1598451071	AUD	153.41 -27.95	SALES-POS
2	authorized	1.0	NaN	ACC-1222300524	AUD	151.23 -33.94	POS
3	authorized	1.0	NaN	ACC-1037050564	AUD	153.10 -27.66	SALES-POS
4	authorized	1.0	NaN	ACC-1598451071	AUD	153.41 -27.95	SALES-POS

In [4]:
```
df_original.columns
```

Out[4]:
```
Index(['status', 'card_present_flag', 'bpay_biller_code', 'account',
       'currency', 'long_lat', 'txn_description', 'merchant_id',
       'merchant_code', 'first_name', 'balance', 'date', 'gender', 'age',
       'merchant_suburb', 'merchant_state', 'extraction', 'amount',
       'transaction_id', 'country', 'customer_id', 'merchant_long_lat',
       'movement'],
      dtype='object')
```

In [5]:
```
df_original.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 [6]:
```
column_list = df_original.columns.tolist()
for column in column_list:
    print( column, ':', df_original[column].nunique())
```

```
status : 2
card_present_flag : 2
bpay_biller_code : 3
account : 100
currency : 1
long_lat : 100
txn_description : 6
merchant_id : 5725
merchant_code : 1
first_name : 80
balance : 12006
date : 91
gender : 2
age : 33
merchant_suburb : 1609
merchant_state : 8
extraction : 9442
amount : 4457
transaction_id : 12043
country : 1
customer_id : 100
merchant_long_lat : 2703
movement : 2
```

In [7]:
```
df_original['movement'].value_counts()
```

Out[7]:
```
debit     11160
credit      883
Name: movement, dtype: int64
```

Adding Debit & Credit Amount Columns in Dataframe

In [8]:
```
df = df_original
```

```
df['Debit_amount'] = np.where(df_original['movement'] == 'debit', df_orig
inal['amount'], np.nan)
df['Credit_amount'] = np.where(df_original['movement'] == 'credit', df_or
iginal['amount'], np.nan)
df
```

Out[8]:

	status	card_present_flag	bpay_biller_code	account	currency	long_lat	txn_descripti
0	authorized	1.0	NaN	ACC-1598451071	AUD	153.41 -27.95	PC
1	authorized	0.0	NaN	ACC-1598451071	AUD	153.41 -27.95	SALES-PC
2	authorized	1.0	NaN	ACC-1222300524	AUD	151.23 -33.94	PC
3	authorized	1.0	NaN	ACC-1037050564	AUD	153.10 -27.66	SALES-PC
4	authorized	1.0	NaN	ACC-1598451071	AUD	153.41 -27.95	SALES-PC
...	...	...	...	...	...	...	
12038	authorized	0.0	NaN	ACC-3021093232	AUD	149.83 -29.47	PC
12039	authorized	1.0	NaN	ACC-1608363396	AUD	151.22 -33.87	SALES-PC
12040	authorized	1.0	NaN	ACC-3827517394	AUD	151.12 -33.89	PC
12041	authorized	1.0	NaN	ACC-2920611728	AUD	144.96 -37.76	SALES-PC
12042	authorized	1.0	NaN	ACC-1443681913	AUD	150.92 -33.77	SALES-PC

12043 rows × 25 columns

In [9]:
```
df.describe()
```

Out[9]:

	card_present_flag	merchant_code	balance	age	amount	Debit_amount
count	7717.000000	883.0	12043.000000	12043.000000	12043.000000	11160.000000
mean	0.802644	0.0	14704.195553	30.582330	187.933588	52.572343
std	0.398029	0.0	31503.722652	10.046343	592.599934	156.354143
min	0.000000	0.0	0.240000	18.000000	0.100000	0.100000
25%	1.000000	0.0	3158.585000	22.000000	16.000000	15.190000

50%	1.000000	0.0	6432.010000	28.000000	29.000000	26.930000
75%	1.000000	0.0	12465.945000	38.000000	53.655000	45.000000
max	1.000000	0.0	267128.520000	78.000000	8835.980000	7081.090000

Calculating Percentage of Missing Values in Each Column

In [10]:
```
df.isna().sum()/len(df) * 100
```

Out[10]:
```
status                0.000000
card_present_flag    35.921282
bpay_biller_code     92.651333
account               0.000000
currency              0.000000
long_lat              0.000000
txn_description       0.000000
merchant_id          35.921282
merchant_code        92.667940
first_name            0.000000
balance               0.000000
date                  0.000000
gender                0.000000
age                   0.000000
merchant_suburb      35.921282
merchant_state       35.921282
extraction            0.000000
amount                0.000000
transaction_id        0.000000
country               0.000000
customer_id           0.000000
merchant_long_lat    35.921282
movement              0.000000
Debit_amount          7.332060
Credit_amount        92.667940
dtype: float64
```

Extracting Customer Details

In [11]:
```
customer_details = df.groupby(['customer_id'])
customer_details
```

Out[11]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000000000C4119C8>
```

In [12]:
```
df1 = customer_details.apply(lambda x: x['transaction_id'].count())
df2 = customer_details.apply(lambda x: x['Debit_amount'].count())
df3 = customer_details.apply(lambda x: x['Debit_amount'].sum())
df4 = customer_details.apply(lambda x: x['Credit_amount'].count())
df5 = customer_details.apply(lambda x: x['Credit_amount'].sum())
```

In [13]:
```
frames = [df1,df2,df3,df4,df5]
col_names = ['total_tr_count','Debit_tr_count','total_Debit_amount','credit_transaction_count',
             'total_Credit_amount']
customer_df = pd.concat(frames, axis = 1, join = 'outer', keys = col_names)
customer_df
```

Out[13]:

customer_id	total_tr_count	Debit_tr_count	total_Debit_amount	credit_transaction_count	total_Credit

CUS-1005756958	73	60	3652.86	13
CUS-1117979751	100	93	8933.82	7
CUS-1140341822	80	74	5511.54	6
CUS-1147642491	118	105	6732.75	13
CUS-1196156254	245	238	8724.61	7
...	...	...	...	...
CUS-72755508	58	46	2734.53	12
CUS-809013380	124	111	5328.18	13
CUS-860700529	233	227	7248.16	6
CUS-880898248	78	72	2858.57	6
CUS-883482547	178	171	8797.19	7

100 rows × 5 columns

In [14]:
```
df6 = customer_details.first()
df7 = customer_details.last()
#There are some meaningless columns in df6 and df7 which will be dropped
 later.
```

In [15]:
```
df6['Opening_bal'] = np.where(df6['movement']=='debit', df6['balance']+df
6['amount'],
                              df6['balance']-df6['amount'])
```

In [16]:
```
df6['Closing_bal']= df7['balance']
```

In [17]:
```
final_customer_df = pd.DataFrame()
final_customer_df = pd.merge(df6,customer_df, on = 'customer_id')
drop_features = ['status','bpay_biller_code','currency','long_lat',
                 'card_present_flag','amount', 'txn_description', 'mercha
nt_id',
                 'merchant_code', 'balance','date','merchant_suburb','mer
chant_state',
                 'extraction','transaction_id', 'country','merchant_long_
lat','movement',
                'Debit_amount', 'Credit_amount']
final_customer_df.drop(drop_features, axis = 1, inplace = True )
```

In [18]:
```
final_customer_df.head()
```

Out[18]:

customer_id	account	first_name	gender	age	Opening_bal	Closing_bal	total_tr_count	Debit_t
CUS-1005756958	ACC-2828321672	Stephanie	F	53	470.44	9310.03	73	
CUS-1117979751	ACC-4065652575	Lucas	M	21	2390.35	18387.41	100	
CUS-1140341822	ACC-80388494	Dustin	M	28	832.74	6820.26	80	

CUS-1147642491	ACC-3233697971	Robin	F	34	14.89	15387.21	118
CUS-1196156254	ACC-3485804958	Jessica	F	34	12563.48	30899.53	245

In [19]:
```
final_customer_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 100 entries, CUS-1005756958 to CUS-883482547
Data columns (total 11 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   account                   100 non-null    object 
 1   first_name                100 non-null    object 
 2   gender                    100 non-null    object 
 3   age                       100 non-null    int64  
 4   Opening_bal               100 non-null    float64
 5   Closing_bal               100 non-null    float64
 6   total_tr_count            100 non-null    int64  
 7   Debit_tr_count            100 non-null    int64  
 8   total_Debit_amount        100 non-null    float64
 9   credit_transaction_count  100 non-null    int64  
 10  total_Credit_amount       100 non-null    float64
dtypes: float64(4), int64(4), object(3)
memory usage: 9.4+ KB
```

Identification of Annual Salary

In [20]:
```
def credit_details(df,customer_id,customer_detail):
    list_of_customers = customer_detail.index.tolist()
    list_of_indices = [0,1,2]
    for x in list_of_indices:
        try:
            each_customer_details = df[df[customer_id] == list_of_customers[x]]
            customer_credit_details = each_customer_details[each_customer_details['Credit_amount'].notnull()]
            credit_related_details = customer_credit_details[['account','txn_description','customer_id','date',
                                                              'Credit_amount','transaction_id']]
            print(credit_related_details.set_index('customer_id'))
        except Exception:
            continue
```

In [21]:
```
ctn_count = final_customer_df['credit_transaction_count'].value_counts()
ctn_count
```

Out[21]:
```
6     28
13    27
7     24
14     8
12     5
2      4
4      2
5      1
3      1
Name: credit_transaction_count, dtype: int64
```

In [22]:
```
suspected = ctn_count.index.tolist()[5:]
suspected
```

Out[22]: [2, 4, 5, 3]

In [23]:
```
for count in suspected:
    suspected_customer_details = final_customer_df[final_customer_df['cre
dit_transaction_count']
                                                   == count]
    print('Number of Credit Transactions : ', count)
    with pd.option_context('expand_frame_repr', False):
              credit_details(df,'customer_id', suspected_customer_details
)
```

```
Number of Credit Transactions :  2
                        account txn_description       date  Credit_amount
transaction_id
customer_id
CUS-1739931018  ACC-1217063613      PAY/SALARY 2018-09-26         4863.62
8659baa692924427aefbf4077c5a9d67
CUS-1739931018  ACC-1217063613      PAY/SALARY 2018-10-26         4863.62
e6d8f31d269d4e8388e115719a59dd98
                        account txn_description       date  Credit_amount
transaction_id
customer_id
CUS-2178051368  ACC-3100725361      PAY/SALARY 2018-09-26         6107.23
7eb1fdb7aefb40d1a8ada1d27f556542
CUS-2178051368  ACC-3100725361      PAY/SALARY 2018-10-26         6107.23
56e1a0f55f354624a3c713c37415d216
                       account txn_description       date  Credit_amount
transaction_id
customer_id
CUS-497688347  ACC-211792489      PAY/SALARY 2018-08-23         4910.9  2
b4ecc820d834f6ea016b40abbfbfde5
CUS-497688347  ACC-211792489      PAY/SALARY 2018-10-23         4910.9  1
c410d75288f4fbb832f88d7f612693b
Number of Credit Transactions :  4
                        account txn_description       date  Credit_amount
transaction_id
customer_id
CUS-1816693151  ACC-1523339231      PAY/SALARY 2018-08-20         8835.98
b608ce5142664a79af4fa071a886c8f7
CUS-1816693151  ACC-1523339231      PAY/SALARY 2018-09-20         8835.98
854ded55d0034ac8b9e91e16334768ca
CUS-1816693151  ACC-1523339231      PAY/SALARY 2018-10-19         8835.98
873a3f11d03d41a99c55a5b1a3850e1a
CUS-1816693151  ACC-1523339231      PAY/SALARY 2018-10-19         8835.98
d996300131a641c8bf25f86e1aef9bc6
                        account txn_description       date  Credit_amount
transaction_id
customer_id
CUS-2110742437  ACC-2270192619      PAY/SALARY 2018-08-06         3026.95
c1d51fe6ac554d37b93d57dba64d6674
CUS-2110742437  ACC-2270192619      PAY/SALARY 2018-09-06         3026.95
11372fe1819f48a8b0260bf3c046e4e0
CUS-2110742437  ACC-2270192619      PAY/SALARY 2018-10-05         3026.95
812d58434d704041aa033b5ae904efe9
CUS-2110742437  ACC-2270192619      PAY/SALARY 2018-10-05         3026.95
efbc0cb742af4879a122c99ae728354b
Number of Credit Transactions :  5
                       account txn_description       date  Credit_amount
transaction_id
customer_id
CUS-2376382098  ACC-354106658      PAY/SALARY 2018-08-15         5103.51
41ce67e6c8a4474385fd1646963b6758
CUS-2376382098  ACC-354106658      PAY/SALARY 2018-09-14         5103.51
188e04b6e00f44f3a43afbf232a6f5c3
CUS-2376382098  ACC-354106658      PAY/SALARY 2018-09-14         5103.51
```

```
CUS-2376382098  ACC-354106658      PAY/SALARY 2018-10-15         5103.51
dfff2531d8434969b7f385d364772534
CUS-2376382098  ACC-354106658      PAY/SALARY 2018-10-15         5103.51
e4e7f0dd7c504c45990277dad7a8a86c
Number of Credit Transactions :  3
                      account txn_description       date  Credit_amount
transaction_id
customer_id

CUS-423725039  ACC-2153562714      PAY/SALARY 2018-08-24         3712.56
13d1b673d560462aa9b879f6b3730e39
CUS-423725039  ACC-2153562714      PAY/SALARY 2018-09-24         3712.56
8d7ddef22c7c4404b63137f3ebd1a6ff
CUS-423725039  ACC-2153562714      PAY/SALARY 2018-10-24         3712.56
fe1b1a6bdd9b43f7985acc4af7b0a101
```

Calculating Annual Salary of Each customer

In [24]:
```
#calculating wages
final_customer_df['wage'] = final_customer_df['total_Credit_amount'] / fi
nal_customer_df['credit_transaction_count']

final_customer_df['Annual_Salary'] = pd.Series()
#calculating annual salary
weekly = [12,13,14]
bi_weekly = [6,7]
monthly = [2,3,4,5]
for i in range(len(final_customer_df['credit_transaction_count'])):
    final_customer_df['Annual_Salary'][i] = np.where(final_customer_df['c
redit_transaction_count'][i]
                                                     in weekly,final_customer_df
['wage'][i] * 52,
                                                     np.where(final_customer_df[
'credit_transaction_count'][i]
                                                     in bi_weekly, final_custome
r_df['wage'][i] * 26,
                                                              final_customer_df[
'wage'][i]*12))
```

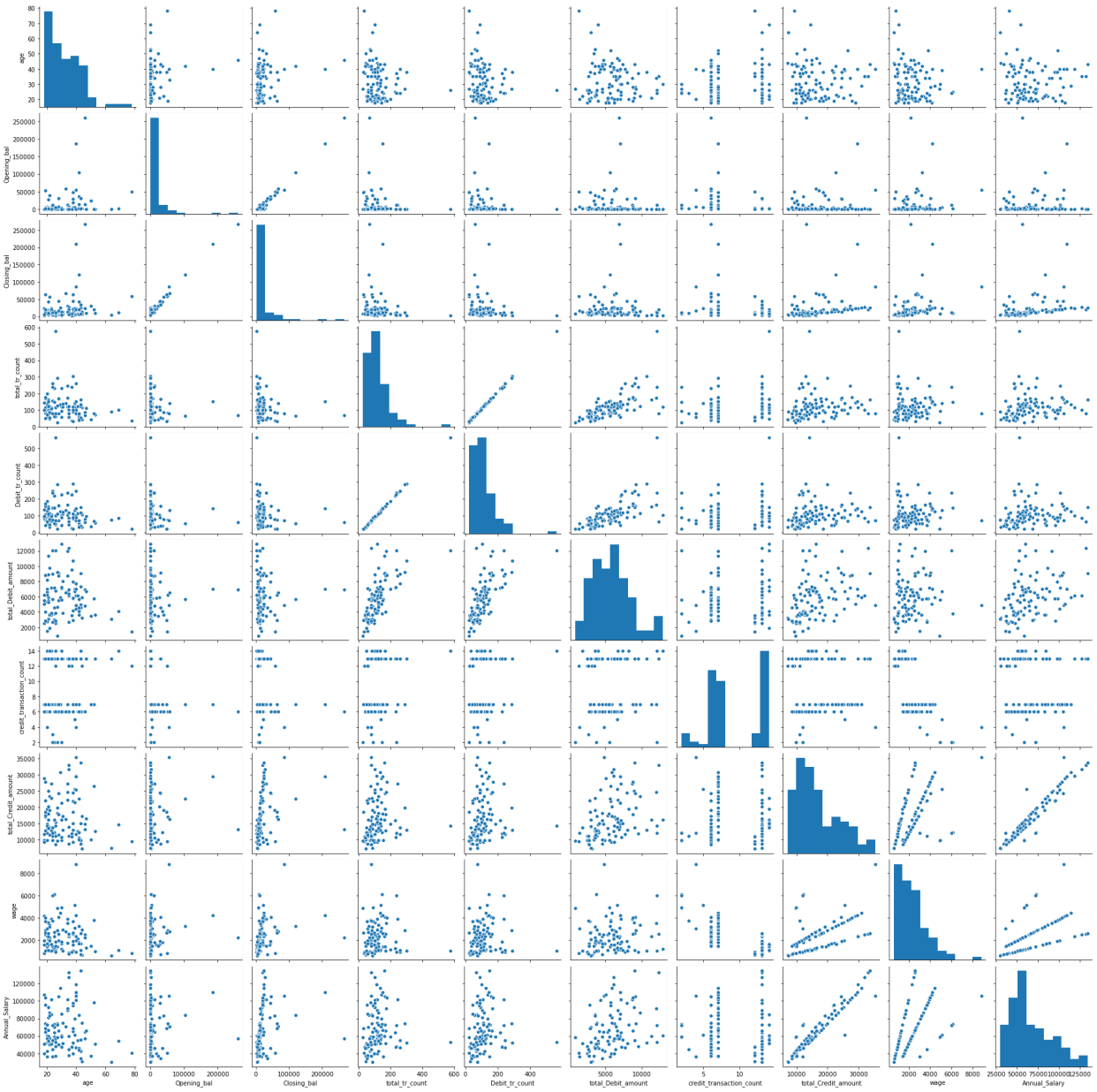
In [25]:
```
final_customer_df.head()
```

Out[25]:

customer_id	account	first_name	gender	age	Opening_bal	Closing_bal	total_tr_count	Debit_t
CUS-1005756958	ACC-2828321672	Stephanie	F	53	470.44	9310.03	73	
CUS-1117979751	ACC-4065652575	Lucas	M	21	2390.35	18387.41	100	
CUS-1140341822	ACC-80388494	Dustin	M	28	832.74	6820.26	80	
CUS-1147642491	ACC-3233697971	Robin	F	34	14.89	15387.21	118	
CUS-1196156254	ACC-3485804958	Jessica	F	34	12563.48	30899.53	245	

In [26]:
```
#Gender wise pairplot
import seaborn as sns
import matplotlib.pyplot as plt
sns.pairplot(final_customer_df)
```

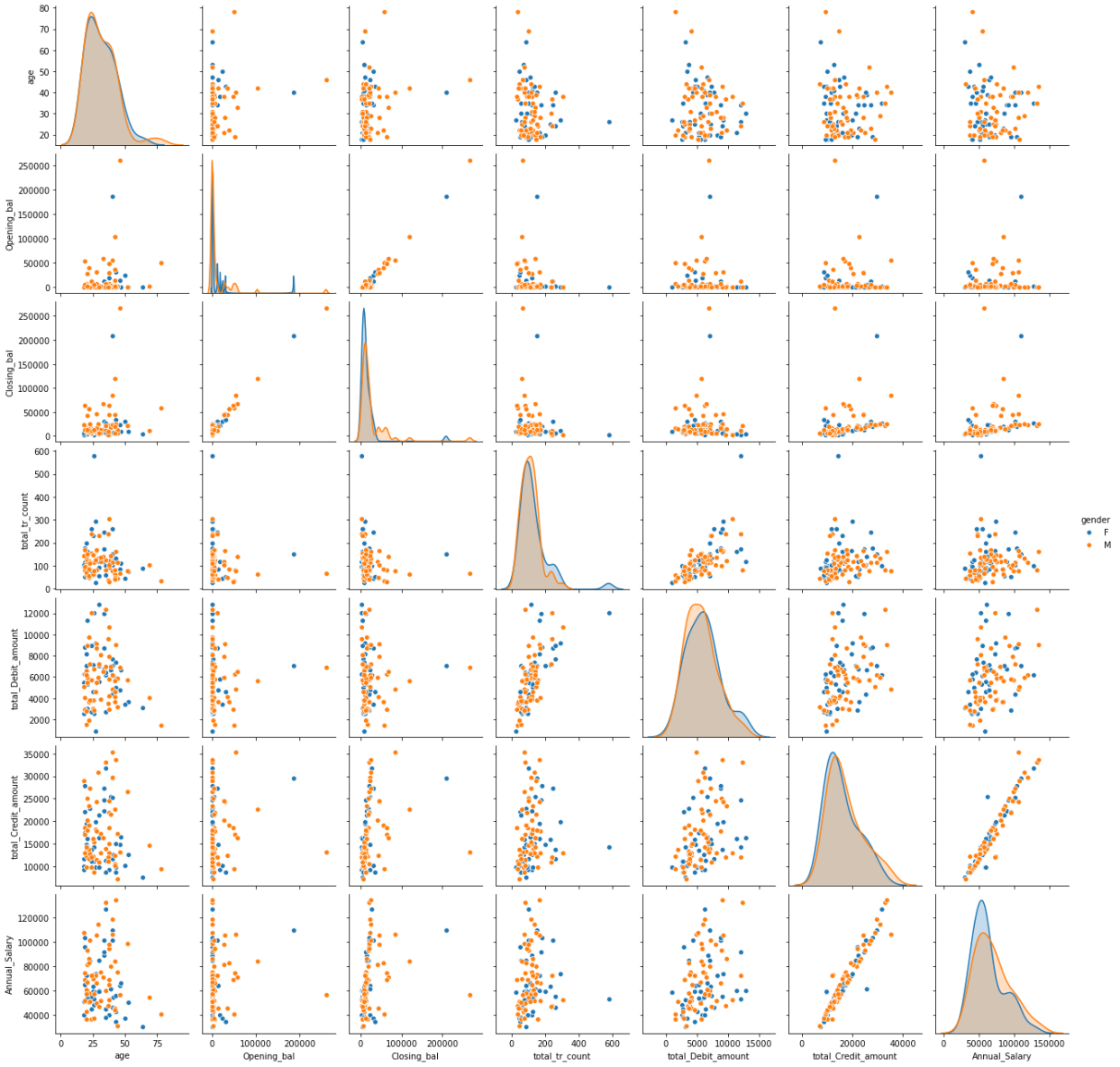
Out[26]:
```
<seaborn.axisgrid.PairGrid at 0xc7abc48>
```



In [27]:
```
drop_features2 = ['wage', 'credit_transaction_count', 'Debit_tr_count',
                  'account','first_name']
final_customer_df1 = final_customer_df.drop(drop_features2, axis = 1, inp
lace = True )
```

In [28]:
```
#Gender wise pairplot
import seaborn as sns
import matplotlib.pyplot as plt
sns.pairplot(final_customer_df, hue='gender')
```

Out[28]:
```
<seaborn.axisgrid.PairGrid at 0xc7a6b48>
```



We have grouped the whole data by Customer Id and collected the data of 100 customers. In this process we have created a few new columns relevant to the study.

We split the amount column and introduced two new columns, Debit_Amount & Credit_amount in accordance with the movement column.

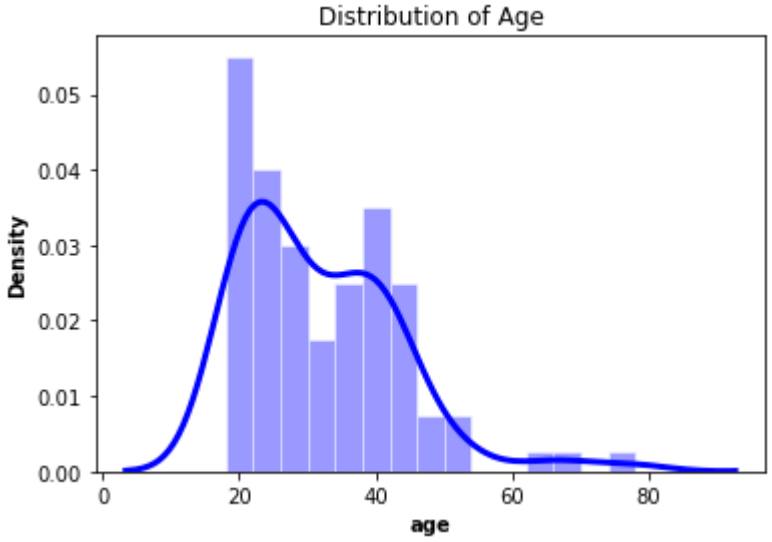
Later, we calculated the annual salary of each customer by analyzing the credit transaction of each customer.

After that, we created this pair plot to take a deep look on the customers. As we can see that all the columns are more or less positively skewed.

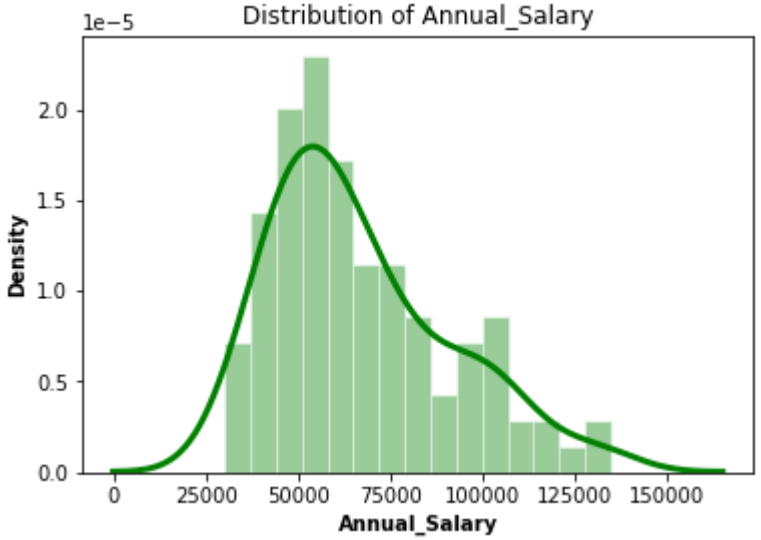
There is strong positive correlation between Total_Transaction_count , total_debit_amount and opening_bal, Closing_bal.

In [29]:
```
# Density Plot and Histogram of age
sns.distplot(final_customer_df['age'], hist=True, kde=True,
             bins=15, color = 'blue',
             hist_kws={'edgecolor':'white'},
             kde_kws={'linewidth': 3})
plt.ylabel('Density',fontsize=10, fontweight='bold')
```

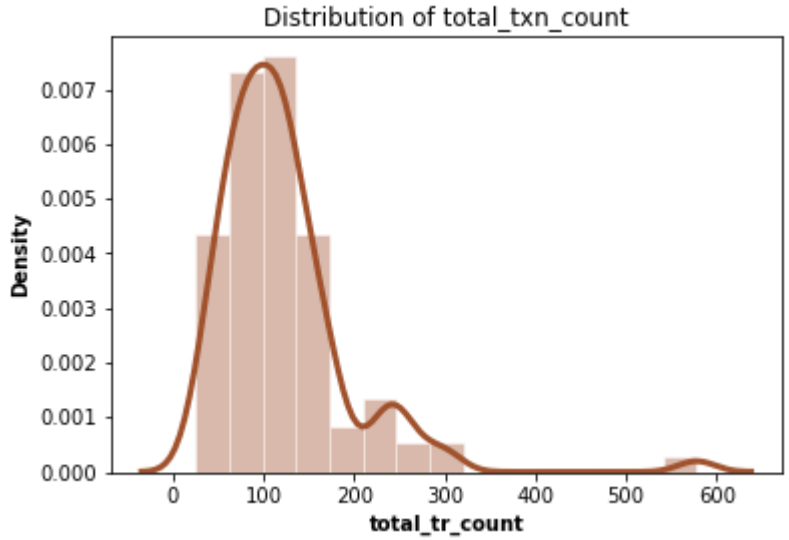
```
plt.xlabel('age',fontsize=10, fontweight='bold')
plt.title('Distribution of Age')
plt.show()
```



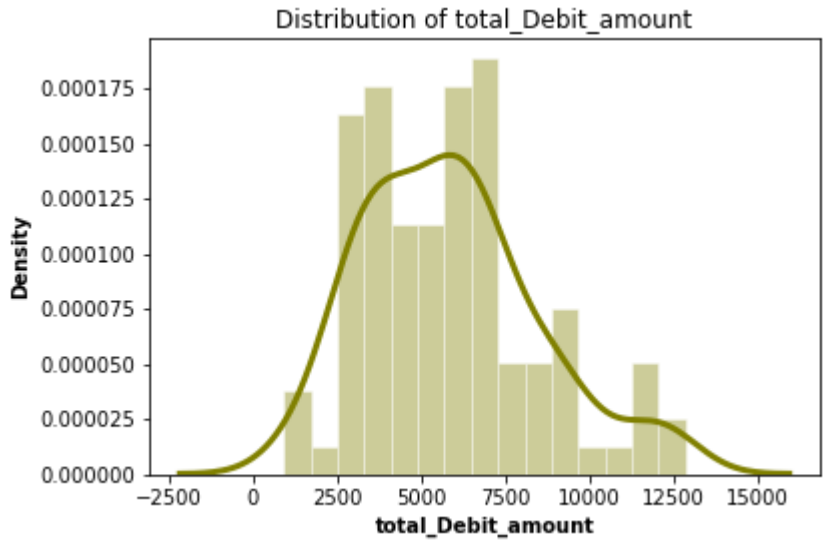
In [30]:
```
# Density Plot and Histogram of annual salary
sns.distplot(final_customer_df['Annual_Salary'], hist=True, kde=True,
             bins=15, color = 'green',
             hist_kws={'edgecolor':'white'},
             kde_kws={'linewidth': 3})
plt.ylabel('Density',fontsize=10, fontweight='bold')
plt.xlabel('Annual_Salary',fontsize=10, fontweight='bold')
plt.title('Distribution of Annual_Salary')
plt.show()
```



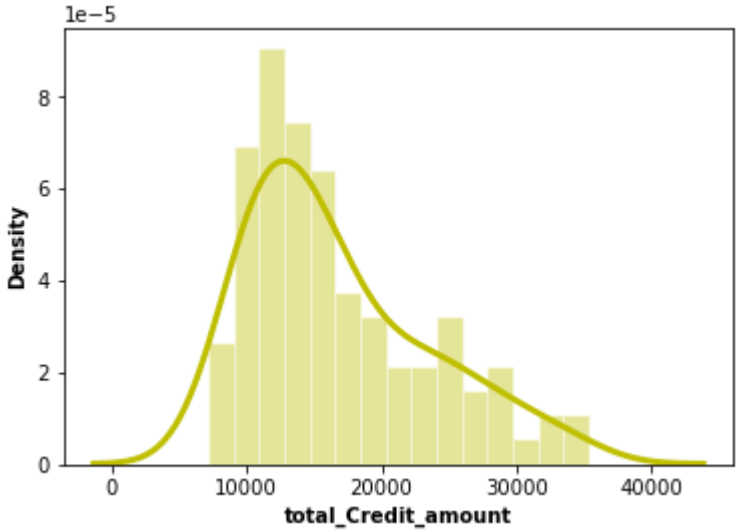
In [31]:
```
# Density Plot and Histogram of annual salary
sns.distplot(final_customer_df['total_tr_count'], hist=True, kde=True,
             bins=15, color = 'sienna',
             hist_kws={'edgecolor':'white'},
             kde_kws={'linewidth': 3})
plt.ylabel('Density',fontsize=10, fontweight='bold')
plt.xlabel('total_tr_count',fontsize=10, fontweight='bold')
plt.title('Distribution of total_txn_count')
plt.show()
```



In [32]:
```
# Density Plot and Histogram of annual salary
sns.distplot(final_customer_df['total_Debit_amount'], hist=True, kde=True
,
             bins=15, color = 'olive',
             hist_kws={'edgecolor':'white'},
             kde_kws={'linewidth': 3})
plt.ylabel('Density',fontsize=10, fontweight='bold')
plt.xlabel('total_Debit_amount',fontsize=10, fontweight='bold')
plt.title('Distribution of total_Debit_amount')
plt.show()
```



In [33]:
```
# Density Plot and Histogram of annual salary
sns.distplot(final_customer_df['total_Credit_amount'], hist=True, kde=Tru
e,
              bins= 15,color = 'y',
             hist_kws={'edgecolor':'white'},
             kde_kws={'linewidth': 3})
plt.ylabel('Density',fontsize=10, fontweight='bold')
plt.xlabel('total_Credit_amount',fontsize=10, fontweight='bold')
plt.show()
```



10. Gender wise Transaction Analysis

In [34]:
```
#sumplot
by_gender_df = final_customer_df.groupby([
'gender' # grouping the data by these two columns
]).count()
```

In [35]:
```
by_gender_df = by_gender_df.unstack().fillna(0)
by_gender_df
```

Out[35]:

	gender	
age	F	44
	M	56
Opening_bal	F	44
	M	56
Closing_bal	F	44
	M	56
total_tr_count	F	44
	M	56
total_Debit_amount	F	44
	M	56
total_Credit_amount	F	44
	M	56
Annual_Salary	F	44
	M	56

dtype: int64

In [36]:
```
#sumplot
by_gender_df1 = final_customer_df.groupby([
'gender' # grouping the data by these two columns
]).sum()
```

In [37]:
```
by_gender_df1 = by_gender_df1.unstack().fillna(0)
by_gender_df1
```

Out[37]:

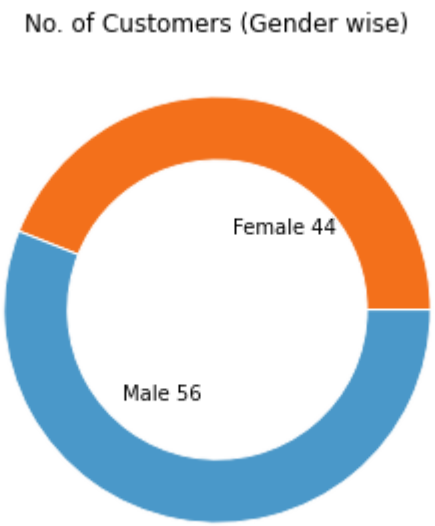
	gender

```
age                   F            1391.00
                      M            1786.00

Opening_bal           F          321151.76
                      M          888151.24
Closing_bal           F          757431.46
                      M         1545545.12
total_tr_count        F            5758.00
                      M            6285.00
total_Debit_amount    F          266666.40
                      M          320040.95
total_Credit_amount   F          703656.23
                      M          972920.62
Annual_Salary         F         2769958.68
                      M         3900695.08
dtype: float64
```

In [38]:

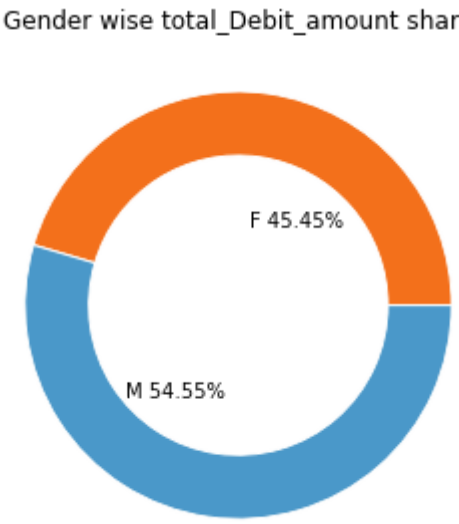
```
#Donut_No. of Customers (Gender wise)
# Make data: I have 2 groups.
group_names=['Female 44', 'Male 56']
group_size=[44 , 56]
#subgroup_names=['M', 'F', 'M', 'F']
#subgroup_size=[464.00 , 419.00, 5821.00 , 5339.00]

# Create colors
a, b =[plt.cm.Oranges, plt.cm.Blues]

# First Ring (outside)
fig, ax1 = plt.subplots()
ax1.axis('equal')
mypie, _ = ax1.pie(group_size, radius=1, labels=group_names, labeldistanc
e=0.4, colors=[a(0.6), b(0.6)])
plt.setp( mypie, width=0.3, edgecolor='white')

# Second Ring (Inside)
#mypie2, _ = ax.pie(subgroup_size, radius=1.3-0.3, labels=subgroup_names,
 #                  colors=[a(0.4), a(0.5), b(0.6), b(0.7), b(0.8), b(0.
9)])

#plt.setp( mypie2, width=0.3, edgecolor='white')
plt.margins(0,0)
plt.tight_layout()
plt.title('No. of Customers (Gender wise)')
# show it
#plt.show()
```

Out[38]: Text(0.5, 1.0, 'No. of Customers (Gender wise)')



In [39]:
```
#Donut_Gender wise total_Debit_amount share
import matplotlib.pyplot as plt

# Make data: I have 2 groups and 7 subgroups
group_names=['F 45.45%', 'M 54.55%']
group_size=[266666.40 , 320040.95 ]
#subgroup_names=['M', 'F', 'M', 'F']
#subgroup_size=[464.00 , 419.00, 5821.00 , 5339.00]

# Create colors
a, b =[plt.cm.Oranges, plt.cm.Blues]

# First Ring (outside)
fig, ax2 = plt.subplots()
ax2.axis('equal')
mypie, _ = ax2.pie(group_size, radius=1, labels=group_names, labeldistanc
e=0.4, colors=[a(0.6), b(0.6)])
plt.setp( mypie, width=0.3, edgecolor='white')

# Second Ring (Inside)
#mypie2, _ = ax.pie(subgroup_size, radius=1.3-0.3, labels=subgroup_names,
 #                  colors=[a(0.4), a(0.5), b(0.6), b(0.7), b(0.8), b(0.
9)])

#plt.setp( mypie2, width=0.3, edgecolor='white')
plt.margins(0,0)
plt.tight_layout()
plt.title('Gender wise total_Debit_amount share')
# show it
#plt.show()
```

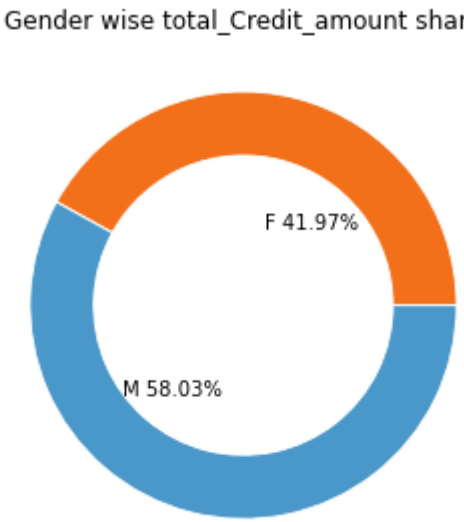
Out[39]:
```
Text(0.5, 1.0, 'Gender wise total_Debit_amount share')
```



In [40]:
```
#Donut_Gender wise total_Credit_amount share
import matplotlib.pyplot as plt

# Make data: I have 3 groups and 7 subgroups
group_names=['F 41.97%', 'M 58.03%']
group_size=[703656.23 , 972920.62 ]
#subgroup_names=['M', 'F', 'M', 'F']
#subgroup_size=[464.00 , 419.00, 5821.00 , 5339.00]

# Create colors
a, b =[plt.cm.Oranges, plt.cm.Blues]

# First Ring (outside)
fig, ax3 = plt.subplots()
```

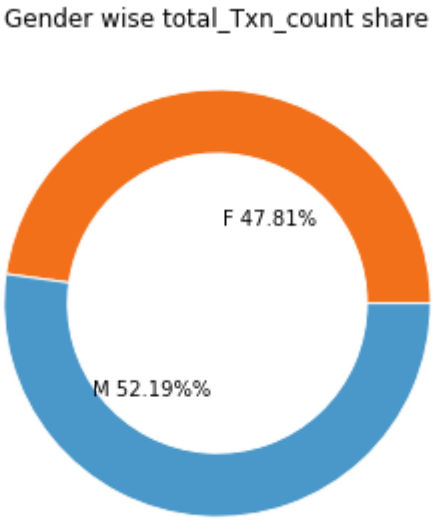
```
ax3.axis('equal')
mypie, _ = ax3.pie(group_size, radius=1, labels=group_names, labeldistanc
e=0.4, colors=[a(0.6), b(0.6)])
plt.setp( mypie, width=0.3, edgecolor='white')

# Second Ring (Inside)
#mypie2, _ = ax.pie(subgroup_size, radius=1.3-0.3, labels=subgroup_names,
 #                  colors=[a(0.4), a(0.5), b(0.6), b(0.7), b(0.8), b(0.
9)])

#plt.setp( mypie2, width=0.3, edgecolor='white')
plt.margins(0,0)
plt.tight_layout()
plt.title('Gender wise total_Credit_amount share')
# show it
#plt.show()
```

Out[40]: Text(0.5, 1.0, 'Gender wise total_Credit_amount share')



In [41]:
```
import matplotlib.pyplot as plt

# Make data: I have 3 groups and 7 subgroups
group_names=['F 47.81%', 'M 52.19%%']
group_size=[5758.0 , 6285.0 ]
#subgroup_names=['M', 'F', 'M', 'F']
#subgroup_size=[464.00 , 419.00, 5821.00 , 5339.00]

# Create colors
a, b =[plt.cm.Oranges, plt.cm.Blues]

# First Ring (outside)
fig, ax = plt.subplots()
ax.axis('equal')
```

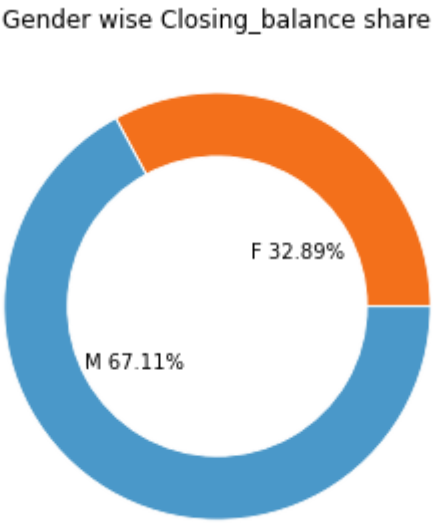
```
mypie, _ = ax.pie(group_size, radius=1, labels=group_names, labeldistance
=0.4, colors=[a(0.6), b(0.6)])
plt.setp( mypie, width=0.3, edgecolor='white')

# Second Ring (Inside)
#mypie2, _ = ax.pie(subgroup_size, radius=1.3-0.3, labels=subgroup_names,
 #                  colors=[a(0.4), a(0.5), b(0.6), b(0.7), b(0.8), b(0.
9)])

#plt.setp( mypie2, width=0.3, edgecolor='white')
plt.margins(0,0)
plt.tight_layout()
plt.title('Gender wise total_Txn_count share')
# show it
plt.show()
```



In [42]:

```
import matplotlib.pyplot as plt

# Make data: I have 3 groups and 7 subgroups
group_names=['F 32.89%', 'M 67.11%']
group_size=[757431.46 , 1545545.12 ]
#subgroup_names=['M', 'F', 'M', 'F']
#subgroup_size=[464.00 , 419.00, 5821.00 , 5339.00]

# Create colors
a, b =[plt.cm.Oranges, plt.cm.Blues]

# First Ring (outside)
fig, ax = plt.subplots()
ax.axis('equal')
mypie, _ = ax.pie(group_size, radius=1, labels=group_names, labeldistance
=0.3, colors=[a(0.6), b(0.6)])
```

```
plt.setp( mypie, width=0.3, edgecolor='white')

# Second Ring (Inside)
#mypie2, _ = ax.pie(subgroup_size, radius=1.3-0.3, labels=subgroup_names,
 #                  colors=[a(0.4), a(0.5), b(0.6), b(0.7), b(0.8), b(0.
9)])

#plt.setp( mypie2, width=0.3, edgecolor='white')
plt.margins(0,0)
plt.tight_layout()
plt.title('Gender wise Closing_balance share')
# show it
plt.show()
```



In [43]:

```
import matplotlib.pyplot as plt

# Make data: I have 3 groups and 7 subgroups
group_names=['F 26.56%', 'M 73.44%']
group_size=[321151.76 , 888151.24 ]
#subgroup_names=['M', 'F', 'M', 'F']
#subgroup_size=[464.00 , 419.00, 5821.00 , 5339.00]

# Create colors
a, b =[plt.cm.Oranges, plt.cm.Blues]

# First Ring (outside)
fig, ax = plt.subplots()
ax.axis('equal')
mypie, _ = ax.pie(group_size, radius=1, labels=group_names, labeldistance
=0.3, colors=[a(0.6), b(0.6)])
```

```
plt.setp( mypie, width=0.3, edgecolor='white')

# Second Ring (Inside)
#mypie2, _ = ax.pie(subgroup_size, radius=1.3-0.3, labels=subgroup_names,
 #                  colors=[a(0.4), a(0.5), b(0.6), b(0.7), b(0.8), b(0.
9)])

#plt.setp( mypie2, width=0.3, edgecolor='white')
plt.margins(0,0)
plt.tight_layout()
plt.title('Gender wise Opening_balance share')
# show it
plt.show()
```