Project Title-"Digital Payment Analysis and UPI Transaction System"

- Domain- FinTech(Banking & Finance sector)
- Project Description:-
- A Python-based project that simulates secure UPI payments, tracks transaction history, and provides basic analysis of digital payment patterns. It demonstrates the core concepts of FinTech and digital payment systems related to all public and private sector banks.

In [ ]:

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import scipy
import warnings
warnings.filterwarnings("ignore")
```

In [ ]:

Business Problem Understanding

- To analyze the behaviour of Transactions of all private and Public sector Bank
- for better payment service of UPI Transactions

Objectives :-

- Which age group citizen make more transcations and less.
- Which bank receives and sends max. UPI Transactions and which minimun.
- In which time transactions is more or less.

In [ ]:

Load the dataset

In [2]:
```
df=pd.read_excel("UPI transactions.xlsx")
df.head()
```

Out[2]:

	Date (dd/mm/yyyy)	Time	Transaction (thousands)	Sent From Bank	City	Age	Gender	Bank Received	Balance (thousands)	Customer ID	Age Groups
0	2023-06-30	1:22:34	643.714122	Axis Bank	Mumbai	71	Male	Bank of India	484.504798	11	Senior Adult
1	2023-08-10	10:37:27	198.788436	Axis Bank	Hyderabad	83	Female	Bandhan Bank	981.841670	12	Senior Adult
2	2022-01-21	9:59:33	369.966567	Axis Bank	Mumbai	69	Male	ICICI	681.097801	22	Senior Adult
3	2023-09-11	2:47:45	288.951851	Axis Bank	Bangalore	78	Male	Bank of India	236.090819	10	Senior Adult
4	2023-07-14	6:43:18	698.746405	Axis Bank	Delhi	69	Male	Bandhan Bank	765.327698	17	Senior Adult

In [ ]:

Data Understanding And Data Exploration

In [ ]:

To the First five rows of the dataset

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

Out[3]:

	Date (dd/mm/yyyy)	Time	Transaction (thousands)	Sent From Bank	City	Age	Gender	Bank Received	Balance (thousands)	Customer ID	Age Groups
0	2023-06-30	1:22:34	643.714122	Axis Bank	Mumbai	71	Male	Bank of India	484.504798	11	Senior Adult
1	2023-08-10	10:37:27	198.788436	Axis Bank	Hyderabad	83	Female	Bandhan Bank	981.841670	12	Senior Adult
2	2022-01-21	9:59:33	369.966567	Axis Bank	Mumbai	69	Male	ICICI	681.097801	22	Senior Adult
3	2023-09-11	2:47:45	288.951851	Axis Bank	Bangalore	78	Male	Bank of India	236.090819	10	Senior Adult
4	2023-07-14	6:43:18	698.746405	Axis Bank	Delhi	69	Male	Bandhan Bank	765.327698	17	Senior Adult

In [ ]:

To the Last five rows of the dataset

In [4]:
```
df.tail()
```

Out[4]:

	Date (dd/mm/yyyy)	Time	Transaction (thousands)	Sent From Bank	City	Age	Gender	Bank Received	Balance (thousands)	Customer ID	Age Groups
2995	2022-05-09	6:9:57	267.617823	Union Bank	Delhi	17	Female	ICICI	896.352820	4	Teen
2996	2022-12-08	1:43:49	587.781140	KreditBee	Bangalore	17	Male	Bank of Baroda	590.202647	2	Teen
2997	2023-06-01	3:3:46	858.627031	KreditBee	Delhi	17	Female	ICICI	116.655068	20	Teen
2998	2023-01-17	1:27:33	666.802048	Union Bank	Mumbai	17	Male	Bank of Baroda	603.403235	2	Teen
2999	2023-09-05	11:51:3	329.419126	ICICI	Chennai	17	Female	Bandhan Bank	646.865141	13	Teen

In [ ]:

To know the number of rows and columns in the dataset

In [5]:
```
df.shape
```

Out[5]:
```
(3000, 11)
```

Observation :-There are 3000 Rows amd 11 columns in tis dataset

In [ ]:
```

```

To know the names of columns in the dataset

In [6]:
```
df.columns
```

Out[6]:
```
Index(['Date (dd/mm/yyyy)', 'Time', 'Transaction (thousands)',
       'Sent From Bank', 'City', 'Age', 'Gender', 'Bank Received ',
       'Balance (thousands)', 'Customer ID', 'Age Groups'],
      dtype='object')
```

In [7]:
```
df.columns.tolist()
```

Out[7]:
```
['Date (dd/mm/yyyy)',
 'Time',
 'Transaction (thousands)',
 'Sent From Bank',
 'City',
 'Age',
 'Gender',
 'Bank Received ',
 'Balance (thousands)',
 'Customer ID',
 'Age Groups']
```

In [ ]:
```

```

we understand each columns(Brief descriptions of columns)

1.Date (dd/mm/yyyy)-The day the transaction happened,Example: 2024-03-01.

2.Time-The exact time the transaction was made, Example: 14:30:05.

3.Transaction (thousands)- How much money was sent in the transaction (in Indian Rupees ₹).

4.Sent From Bank-The name of the bank account from which the money was sent.

5.City-The city where the customer is located

6.Age-Age of the customer making the transaction.

7.Gender-Customer — usually "Male" or "Female".

8.Bank Received-The name of the bank account that received the money.

9.Balance (thousands)-The remaining bank balance of the customer after the transaction.

10.Customer ID-It showsmultiple transactions. A unique identifier for each customer. Essential for tracking individual customer behaviour and preventing double-counting if a customer has

11.Age Groups-It shows the 'Senior Adult', 'Adult', 'Middle Age Adult', 'Teen'

In [ ]:

To know the Column Names,Count of Null Values,Data Types & Memory uses

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 11 columns):
 #   Column                   Non-Null Count  Dtype         
---  ------                   --------------  -----         
 0   Date (dd/mm/yyyy)        3000 non-null   datetime64[ns]
 1   Time                     3000 non-null   object        
 2   Transaction (thousands)  3000 non-null   float64       
 3   Sent From Bank           3000 non-null   object        
 4   City                     3000 non-null   object        
 5   Age                      3000 non-null   int64         
 6   Gender                   3000 non-null   object        
 7   Bank Received            3000 non-null   object        
 8   Balance (thousands)      3000 non-null   float64       
 9   Customer ID              3000 non-null   int64         
 10  Age Groups               3000 non-null   object        
dtypes: datetime64[ns](1), float64(2), int64(2), object(6)
memory usage: 257.9+ KB
```

Observation :-There is no missing values in any columns

In [ ]:

To find missing values or null values in the dataset

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
Date (dd/mm/yyyy)          0
Time                       0
Transaction (thousands)    0
Sent From Bank             0
City                       0
Age                        0
Gender                     0
Bank Received              0
Balance (thousands)        0
Customer ID                0
Age Groups                 0
dtype: int64
```

Observation :-There is no missing values in any columns

In [ ]:

To find data types in the dataset

In [10]:
```
df.dtypes
```

Out[10]:
```
Date (dd/mm/yyyy)          datetime64[ns]
Time                               object
Transaction (thousands)           float64
Sent From Bank                     object
City                               object
Age                                 int64
Gender                             object
Bank Received                      object
Balance (thousands)               float64
Customer ID                         int64
Age Groups                         object
dtype: object
```

Observation :-

- There is only one wrong datatype in time column.
- convert to part of datetime in month & year for better analyzing.

In [ ]:

To find the duplicate values in the dataset

In [11]:
```
df.duplicated().sum()
```

Out[11]:
```
0
```

Observation :-There is no duplicate values in this dataset

In [ ]:

In [12]:
```
df.columns.tolist()
```

Out[12]:
```
['Date (dd/mm/yyyy)',
 'Time',
 'Transaction (thousands)',
 'Sent From Bank',
 'City',
 'Age',
 'Gender',
 'Bank Received ',
 'Balance (thousands)',
 'Customer ID',
 'Age Groups']
```

In [ ]:

In [ ]:

Analysing the each coulmns separately

In [ ]:

1.Date (dd/mm/yyyy)

To know the all distinct values & datatypes from a specific column.

In [13]:
```
df['Date (dd/mm/yyyy)'].unique()
```

Out[13]:
```
<DatetimeArray>
['2023-06-30 00:00:00', '2023-08-10 00:00:00', '2022-01-21 00:00:00',
 '2023-09-11 00:00:00', '2023-07-14 00:00:00', '2022-10-03 00:00:00',
 '2024-01-28 00:00:00', '2024-01-13 00:00:00', '2023-11-24 00:00:00',
 '2023-11-28 00:00:00',
 ...
 '2023-03-06 00:00:00', '2023-10-07 00:00:00', '2023-11-07 00:00:00',
 '2023-02-25 00:00:00', '2023-02-28 00:00:00', '2023-03-15 00:00:00',
 '2022-11-13 00:00:00', '2024-01-03 00:00:00', '2023-06-09 00:00:00',
 '2022-12-08 00:00:00']
Length: 749, dtype: datetime64[ns]
```

Obervations:-

- It shows the date and time of transactions
- Data Format is wrong so we will have to change into correct Date Time Format.
- we have to create separate column for year,month and date.

In [ ]:

In [ ]:

To know the number of distinct/unique values from a specific column.

In [14]:
```
df['Date (dd/mm/yyyy)'].nunique()    # count of the unique values
```

Out[14]:
```
749
```

In [ ]:

2.Time

In [ ]:

To know the all distinct values & datatypes from a specific column.

In [15]:
```
df['Time'].unique()
```

Out[15]:
```
array(['1:22:34', '10:37:27', '9:59:33', ..., '3:3:46', '1:27:33',
       '11:51:3'], dtype=object)
```

Obervations:-

- It shows the time of transactions
- Data Types is wrong so we will have to change into correct Data Type i.e; datetime.
- convert to part of datetime for better understanding.

To know the No. of distinct values from a specific column.

In [16]:
```
df['Time'].nunique()
```

Out[16]:
```
2897
```

In [ ]:
```

```

3.Transaction (thousands)

In [ ]:
```

```

To know the all distinct values & datatypes from a specific column.

In [17]:
```
df['Transaction (thousands)'].unique()
```

Out[17]:
```
array([643.71412243, 198.78843565, 369.9665675 , ..., 858.62703052,
       666.80204755, 329.41912551])
```

Obervations:-

- It shows the transactions in indian repees

In [ ]:
```

```

To know the No. of distinct values from a specific column.

In [18]:
```
df['Transaction (thousands)'].nunique()
```

Out[18]:
```
3000
```

4.Sent From Bank

In [ ]:
```

```

To know the all distinct values & datatypes from a specific column.

In [19]:
```
df['Sent From Bank'].unique()
```

Out[19]:
```
array(['Axis Bank', 'KreditBee', 'HDFC', 'Union Bank', 'ICICI'],
      dtype=object)
```

Obervations:-

- It shows five bank name by which transactions has been sent to other bank

In [ ]:
```

```

To know the No. of distinct values from a specific column.

In [20]:
```
df['Sent From Bank'].nunique()
```

Out[20]:
```
5
```

In [ ]:
```

```

To count the occurrences of the Unique values

In [21]:
```
df['Sent From Bank'].value_counts()
```

Out[21]:
```
Sent From Bank
ICICI         624
Union Bank    613
Axis Bank     599
HDFC          586
KreditBee     578
Name: count, dtype: int64
```

Observation:- ICICI bank has Maximun Transactions Sent

In [ ]:
```

```

In [22]:
```
round(df['Sent From Bank'].value_counts()/len(df['Sent From Bank']),4)*100
```

Out[22]:
```
Sent From Bank
ICICI         20.80
Union Bank    20.43
Axis Bank     19.97
HDFC          19.53
KreditBee     19.27
Name: count, dtype: float64
```

Observation:- ICICI bank has done maximun Transactions but Union bank has almost similar Tr. to ICICI

In [ ]:

5.City

In [ ]:

To know the No. of distinct values from a specific column.

In [23]:
```
df['City'].nunique()
```

Out[23]:
```
5
```

In [ ]:

To know the all distinct values & datatypes from a specific column.

In [24]:
```
df['City'].unique()
```

Out[24]:
```
array(['Mumbai', 'Hyderabad', 'Bangalore', 'Delhi', 'Chennai'],
      dtype=object)
```

Obervations:-

- It shows five city names by which transactions has been done

In [ ]:

To count the occurrences of the Unique values

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

Out[25]:
```
City
Bangalore    616
Hyderabad    615
Delhi        600
Chennai      597
Mumbai       572
Name: count, dtype: int64
```

Observation:-

- Bangalore and Hyderabad has maximum Transactions
- Mumbai has mnimum Transactions

In [ ]:

6.Age

In [ ]:

To know the all distinct values & datatypes from a specific column.

In [26]:
```
df['Age'].unique()
```

Out[26]:
```
array([71, 83, 69, 78, 64, 62, 82, 72, 76, 84, 65, 61, 68, 77, 79, 74, 73,
       81, 66, 80, 75, 67, 60, 63, 70, 21, 22, 19, 29, 36, 34, 28, 18, 38,
       27, 39, 30, 32, 25, 31, 23, 20, 37, 24, 35, 26, 33, 52, 58, 54, 44,
       46, 41, 59, 42, 45, 55, 40, 51, 49, 47, 48, 53, 56, 57, 43, 50, 17],
      dtype=int64)
```

Observation:-

- It shows the unique age group persons

In [27]:
```
df['Age'].mean()   # Average. Age
```

Out[27]:
```
50.344
```

In [28]:
```
df['Age'].max()   # Max. Age
```

Out[28]:
```
84
```

In [29]:
```
df['Age'].min()   # Min. Age
```

Out[29]: 17

Observation:-

- It shows the maximum age people=84 years
- It shows the minimum age people=17 years
- It shows the average age of the people=50 years

In [ ]:

6.Gender

In [ ]:

To know the all distinct values & datatypes from a specific column.

In [30]:
```
df['Gender'].unique()
```

Out[30]:
```
array(['Male', 'Female'], dtype=object)
```

Observation:- Male and Female

In [ ]:

To count the occurrences of the Unique values

In [31]:
```
df['Gender'].value_counts()
```

Out[31]:
```
Gender
Male      1514
Female    1486
Name: count, dtype: int64
```

Observation:-

- Maximun No. of Male does the transactions

In [ ]:

7.Bank Received

In [32]:
```
df.columns
```

Out[32]:
```
Index(['Date (dd/mm/yyyy)', 'Time', 'Transaction (thousands)',
       'Sent From Bank', 'City', 'Age', 'Gender', 'Bank Received ',
       'Balance (thousands)', 'Customer ID', 'Age Groups'],
      dtype='object')
```

In [33]:
```
# Here we need to replace 'Bank Received '(space) name with 'Bank Received'.

df.rename(columns={'Bank Received ':'Bank Received'},inplace=True)
```

In [ ]:
```

```

To know the No. of distinct values from a specific column.

In [34]:
```
df['Bank Received'].nunique()
```

Out[34]:
```
8
```

In [ ]:
```

```

To know the all distinct values & datatypes from a specific column.

In [35]:
```
df['Bank Received'].unique()
```

Out[35]:
```
array(['Bank of India', 'Bandhan Bank', 'ICICI', 'Axis Bank', 'HDFC',
       'Union Bank', 'KreditBee', 'Bank of Baroda'], dtype=object)
```

Observation:- There are 8 Types of bank that receive UPI transactions

In [ ]:
```

```

To count the occurrences of the Unique values

In [36]:
```
df['Bank Received'].value_counts()
```

Out[36]:
```
Bank Received
HDFC              433
Axis Bank         400
Bandhan Bank      386
Bank of India     383
Union Bank        360
Bank of Baroda    349
ICICI             348
KreditBee         341
Name: count, dtype: int64
```

Observation:-

- There are 8 Types of bank that receive UPI transactions.
- In which HDFC receives more no. of Transactions that is approx similar to Axis Bank.

In [ ]:

8.Balance (thousands)

In [ ]:

To know the all distinct values & datatypes from a specific column.

In [37]:
```
df['Balance (thousands)'].unique()
```

Out[37]:
```
array([484.50479759, 981.84167013, 681.09780098, ..., 116.65506848,
       603.40323485, 646.86514058])
```

Observation:-

- It will show the all UPI Transactions Balance in Indian Rupees

In [ ]:

To know the No. of distinct values from a specific column.

In [38]:
```
df['Balance (thousands)'].nunique()
```

Out[38]:
```
3000
```

In [ ]:

9.Customer ID

In [ ]:

To know the No. of distinct values from a specific column.

In [39]:
```
df['Customer ID'].nunique()
```

Out[39]:
```
22
```

In [ ]:

To know the all distinct values & datatypes from a specific column.

In [40]:
```
df['Customer ID'].unique()
```

Out[40]:
```
array([11, 12, 22, 10, 17,  4,  6,  7, 21,  8, 20, 18, 15,  9, 19, 16, 13,
       14,  1,  3,  5,  2], dtype=int64)
```

Observation:-

- Here 22 Types of Unique Customer ID
- so only 22 types of Customers have done the UPI Transactions

In [ ]:

To count the occurrences of the Unique values

In [41]:
```
df['Customer ID'].value_counts()
```

Out[41]:
```
Customer ID
2     163
16    160
17    150
8     146
15    146
20    145
7     143
5     142
18    142
19    138
1     138
21    136
12    135
3     131
9     130
22    130
14    128
10    126
4     122
11    119
6     118
13    112
Name: count, dtype: int64
```

Observation:-

- Here 22 Types of Unique Customer ID
- In which Customer ID has done max. UPI Transactions

In [ ]:

In [ ]:

10.Age Groups

In [ ]:

To know the No. of distinct values from a specific column.

In [42]:
```
df['Age Groups'].nunique()
```

Out[42]:
```
4
```

In [ ]:

To know the all distinct values & datatypes from a specific column.

In [43]:
```
df['Age Groups'].unique()
```

Out[43]:
```
array(['Senior Adult', 'Adult', 'Middle Age Adult', 'Teen'], dtype=object)
```

Observation:-

- Here 4 Types of Unique Age Groups(Citizens) have done the UPI Transactions

In [ ]:

To count the occurrences of the Unique values

In [44]:
```
df['Age Groups'].value_counts()
```

Out[44]:
```
Age Groups
Senior Adult        1109
Adult               1007
Middle Age Adult     844
Teen                  40
Name: count, dtype: int64
```

In [ ]:

Observation:-

- Here 4 Types of Unique Age Groups(Citizens) have done the UPI Transactions.
- In which Senior Adults have done Max. UPI Transactions.
- Teen has done Minimum Min. UPI Transactions.

In [ ]:

Conclusion:-

- Time column-Change the wrong datatype(object) from write datatypes(datatime).
- continuous_variable=['Transaction (thousands)','Age','Balance (thousands)']
- timeseries_variable=['Date (dd/mm/yyyy)','Time']
- discreate_variable=['Sent From Bank','City','Gender','Bank Received','Customer ID','Age Groups']

In [45]:
```
timeseries_variable=['Date (dd/mm/yyyy)','Time']
```

In [46]:
```
continuous_variable=['Transaction (thousands)','Age','Balance (thousands)']
```

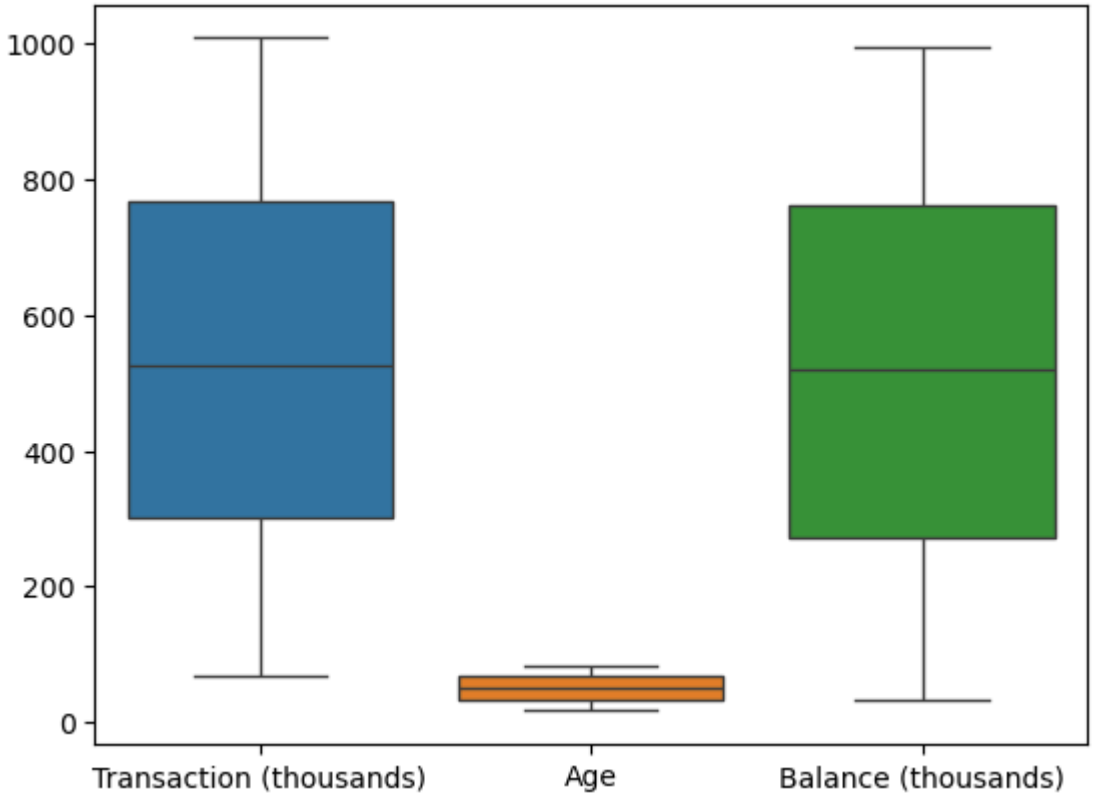
In [47]:
```
discreate_variable=['Sent From Bank','City','Gender','Bank Received','Customer ID','Age Groups']
```

In [ ]:

In [ ]:

In [48]:
```
# to check the outliers in df[continuous_variable]
```

In [49]:
```
sns.boxplot(df[continuous_variable])
plt.show()
```



No outliers -'Transaction (thousands)' , 'Age' , & 'Balance (thousands)' so no need to treat them

In [ ]:

To know the all descriptive statistical measures of continuous variable

In [50]:
```
df[continuous_variable].describe()
```

Out[50]:

	Transaction (thousands)	Age	Balance (thousands)
count	3000.000000	3000.000000	3000.000000
mean	533.791762	50.344000	515.370087
std	270.417288	19.725686	280.392138
min	68.032821	17.000000	34.135495
25%	300.962858	33.000000	271.387805
50%	525.320922	50.000000	519.442642
75%	768.333817	68.000000	760.813343
max	1008.807429	84.000000	996.780463

Observations:-

The average transaction amount is about 533 (in thousands)

The smallest transaction is about 68 (in thousands).

The largest transaction is about 1008 (in thousands).

The average age of customers is about 50 years.

The youngest customer is 17 years old.

The oldest customer is 84 years old.

The average balance is about 515 (in thousands)

The smallest balance is about 34 (in thousands).

The largest balance is about 996 (in thousands).

In [ ]:

Brief descriptions about all discreate variable

In [51]:
```
df[discreate_variable].describe(include="object")
```

Out[51]:

	Sent From Bank	City	Gender	Bank Received	Age Groups
count	3000	3000	3000	3000	3000
unique	5	5	2	8	4
top	ICICI	Bangalore	Male	HDFC	Senior Adult
freq	624	616	1514	433	1109

Observations

- In Sent From Bank, ICICI Bank has done max no. of transactions out of 5.
- In Bank Received, HDFC Bank has done max no. of transactions out of 8.
- In City, Bangalore City has done max no. of transactions out of 5.
- In Gender, Male has done max no. of transactions out of 2.
- In Age Groups, Senior Adult has done max no. of transactions out of 4.

In [ ]:

Data processing

In [ ]:

Here,we have created a new column for better understanding and analysing.

In [ ]:

To create a new column as T year for Year in original dataset

In [52]:
```
df['T Year']=df['Date (dd/mm/yyyy)'].dt.year
```

In [ ]:

To create a new column as T Date for Day in original dataset

In [53]:
```
df['T Date']=df['Date (dd/mm/yyyy)'].dt.day
```

In [ ]:
```

```

To create a new column as T Month for Month in original dataset

In [54]:
```
df['T Month']=df['Date (dd/mm/yyyy)'].dt.month
```

In [ ]:
```

```

To create a new column as T Day for Day of the week by date in original dataset

In [55]:
```
df['T day']=df['Date (dd/mm/yyyy)'].dt.dayofweek
```

In [ ]:
```

```

To convert 0 with Mon,1 with Tue and so on - for better analysing & understanding

In [56]:
```
df['Day Name']=df['T day'].replace({0:'Mon',1:'Tue',2:'Wed',3:'Thu',4:'Fri',5:'Sat',6:'Sun',})
```

In [ ]:
```

```

In [ ]:
```

```

To convert 1 with Jan ,2 with Feb ,3 with Mar and so on - for better analysing & understanding

In [57]:
```
df['Month Name'] = df['T Month'].replace({
    1: 'Jan', 2: 'Feb', 3: 'Mar', 4: 'Apr',
    5: 'May', 6: 'Jun', 7: 'Jul', 8: 'Aug',
    9: 'Sep', 10: 'Oct', 11: 'Nov', 12: 'Dec'
})
```

In [58]:
```
df.shape
```

Out[58]:
```
(3000, 17)
```

In [59]:
```
df.columns
```

Out[59]:
```
Index(['Date (dd/mm/yyyy)', 'Time', 'Transaction (thousands)',
       'Sent From Bank', 'City', 'Age', 'Gender', 'Bank Received',
       'Balance (thousands)', 'Customer ID', 'Age Groups', 'T Year', 'T Date',
       'T Month', 'T day', 'Day Name', 'Month Name'],
      dtype='object')
```

In [ ]:

To know the all distinct values & datatypes from a specific column.

In [60]:
```
df['T day'].unique()
```

Out[60]:
```
array([4, 3, 0, 6, 5, 1, 2])
```

Observations-It shows all days of the week as 0:Mon,1:Tue,2:Wed and so on

In [ ]:

In [61]:
```
df['Day Name'].unique()
```

Out[61]:
```
array(['Fri', 'Thu', 'Mon', 'Sun', 'Sat', 'Tue', 'Wed'], dtype=object)
```

Observations-It shows all days of the week

In [62]:
```
df['Day Name'].value_counts()
```

Out[62]:
```
Day Name
Fri    461
Tue    444
Sun    432
Thu    426
Mon    423
Sat    413
Wed    401
Name: count, dtype: int64
```

Most Transactions has been done on Friday

In [ ]:

In [63]:
```
df['T Month'].unique()
```

Out[63]:
```
array([ 6,  8,  1,  9,  7, 10, 11,  2,  5,  3,  4, 12])
```

Observations-It shows all Months as 1:Jan,2:Feb,3:Mar and so on

In [ ]:

In [64]:
```
df['Month Name'].unique()
```

Out[64]:
```
array(['Jun', 'Aug', 'Jan', 'Sep', 'Jul', 'Oct', 'Nov', 'Feb', 'May',
       'Mar', 'Apr', 'Dec'], dtype=object)
```

In [65]:
```
df['Month Name'].value_counts()
```

Out[65]:
```
Month Name
Jan    370
Aug    275
Jul    270
May    258
Dec    255
Sep    238
Apr    234
Jun    232
Mar    232
Oct    218
Nov    209
Feb    209
Name: count, dtype: int64
```

Most Transactions has been done in January Month

Less Transactions has been done in February & november Month

In [ ]:

In [66]:
```
df['T Year'].unique()
```

Out[66]:
```
array([2023, 2022, 2024])
```

Transactions Years

In [67]:
```
df['T Year'].value_counts()
```

Out[67]:
```
T Year
2022    1454
2023    1423
2024     123
Name: count, dtype: int64
```

Most Transactions has been done in year 2022

Less Transactions has been done in Year 2024

Trends of UPI Transactions is decreasing as year is increasing so it is an alarming conditions of Digital Payments

In [ ]:

In [ ]:

Data Analyzing and Visualization

In [ ]:

Univariate analysis and visualizations

In [ ]:

Sent From Bank

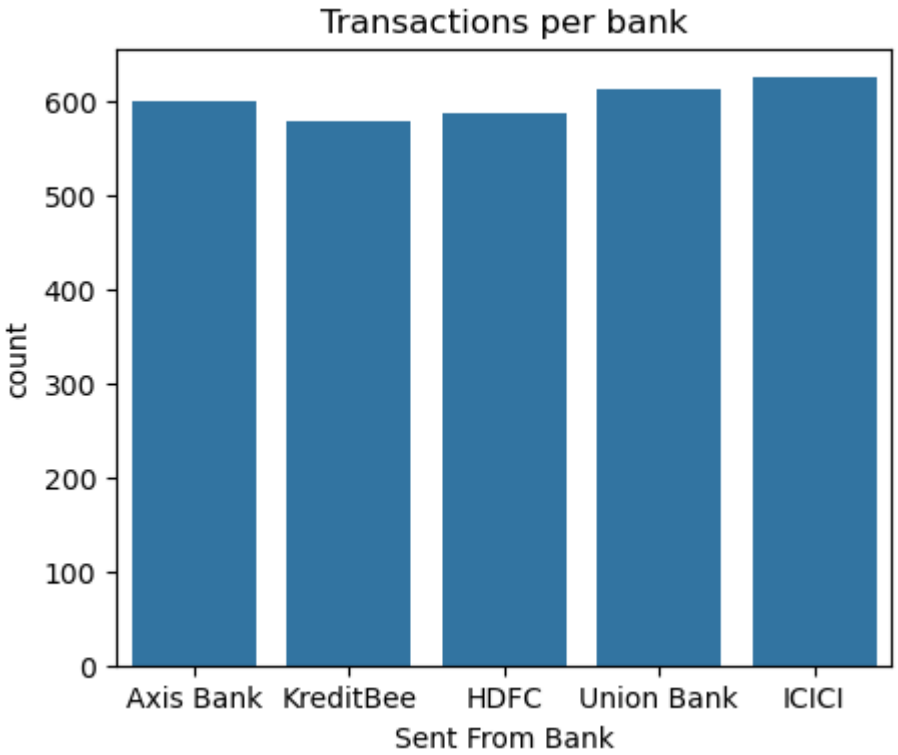
In [68]:
```
# which bank has done max. times and min. times transaction ?

df['Sent From Bank'].value_counts()
```

Out[68]:
```
Sent From Bank
ICICI         624
Union Bank    613
Axis Bank     599
HDFC          586
KreditBee     578
Name: count, dtype: int64
```

In [ ]:

In [69]:
```
# which bank has done max. times and min. times transaction ?

plt.figure(figsize=(5,4))
```

```
sns.countplot(x=df['Sent From Bank'])
plt.title("Transactions per bank")
plt.savefig("Sent From Bank")
plt.show()
```



Obesrvations:-

- ICICI bank has more Transactions and KreditBee has less Transactions

In [ ]:

City

In [ ]:

In [70]:
```
# which city has done max. or min. Transactions?

df['City'].value_counts()
```

Out[70]:
```
City
Bangalore    616
Hyderabad    615
Delhi        600
Chennai      597
Mumbai       572
Name: count, dtype: int64
```

Obesrvations:-

- Bangalore and Hyderabad have almost same Transactions which is max.
- Mumbai has less Transactions

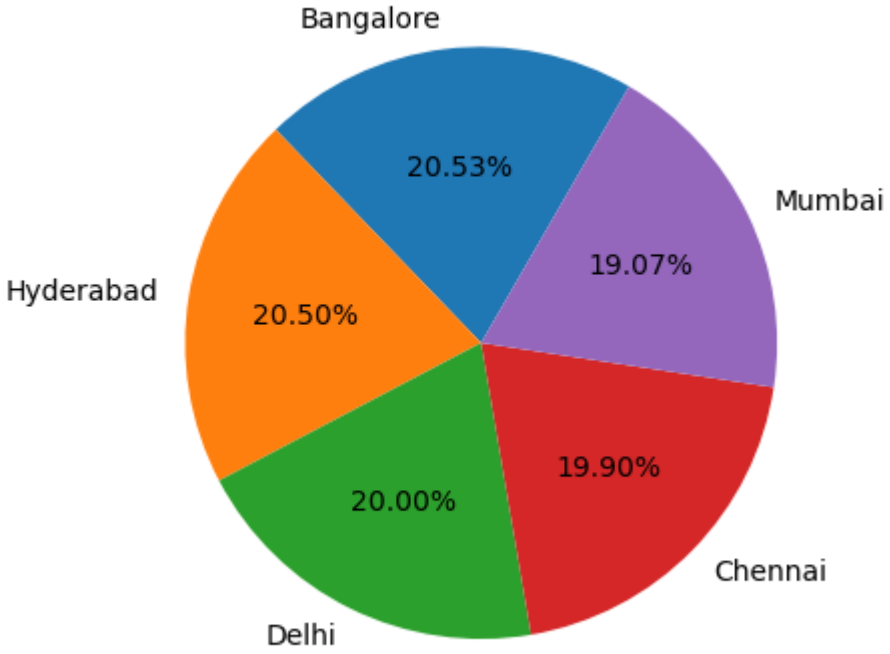
In [71]:
```
round(df['City'].value_counts()/len(df['City']),4)*100
```

Out[71]:
```
City
Bangalore    20.53
Hyderabad    20.50
Delhi        20.00
Chennai      19.90
Mumbai       19.07
Name: count, dtype: float64
```

Obesrvations:-

- Bangalore and Hyderabad have almost same Transactions which is max.
- Mumbai has less Transactions

In [72]:
```
a=df['City'].value_counts().index
b=df['City'].value_counts().values

plt.pie(b,labels=a,autopct="%0.2f%%",startangle=60)
plt.show()
```



Obesrvations:-

- Bangalore and Hyderabad have almost same Transactions which is max.
- Mumbai has less Transactions

In [ ]:

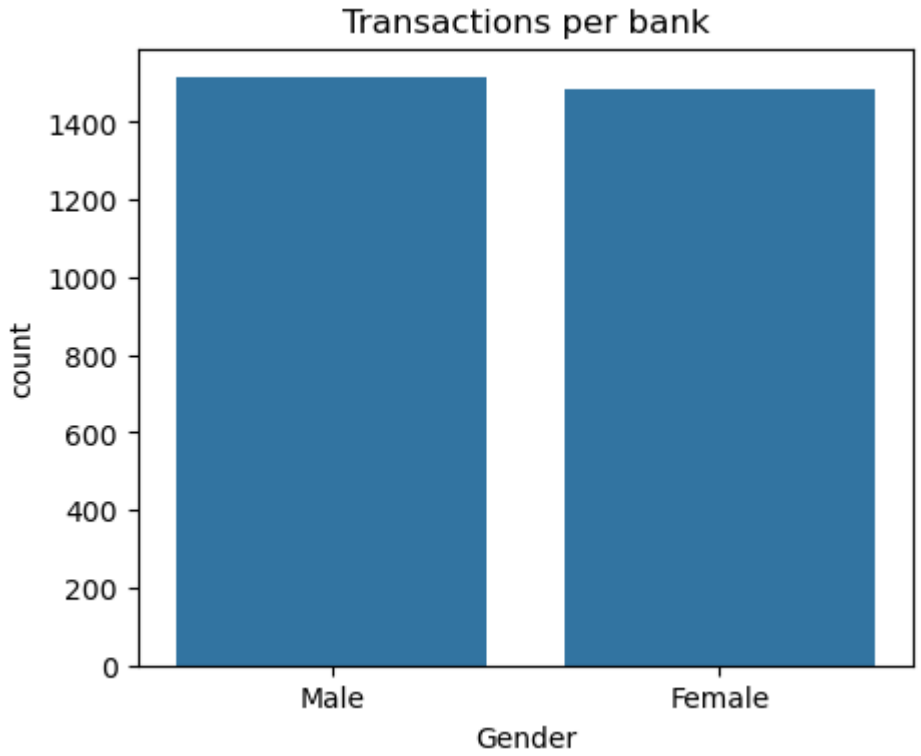
Gender

In [ ]:

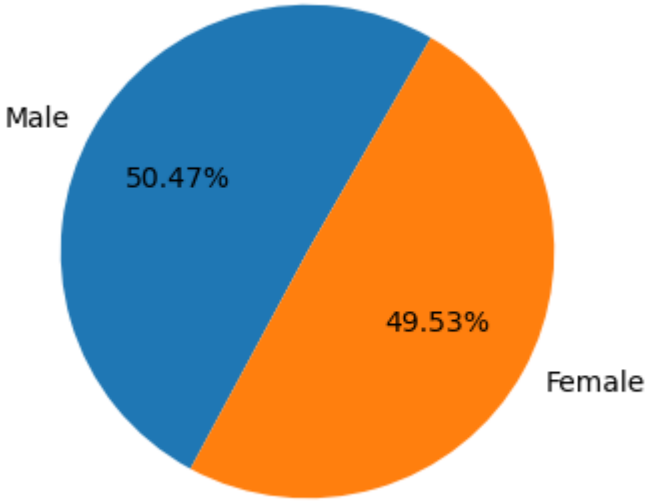
In [73]:
```
# who does the max. or min. Transactions either male or female?

df['Gender'].value_counts()
```

Out[73]:
```
Gender
Male      1514
Female    1486
Name: count, dtype: int64
```

In [74]:
```
plt.figure(figsize=(5,4))
sns.countplot(x=df['Gender'])
plt.title("Transactions per bank")
plt.savefig('Gender')
plt.show()
```



In [75]:
```
a=df['Gender'].value_counts().index
b=df['Gender'].value_counts().values

plt.figure(figsize=(5,4))
plt.pie(b,labels=a,autopct="%0.2f%%",startangle=60)
plt.savefig("Gender")
plt.show()
```



Observation:-Male does slightly more transactions than Female so no meaning

In [ ]:

Bank Received

In [ ]:

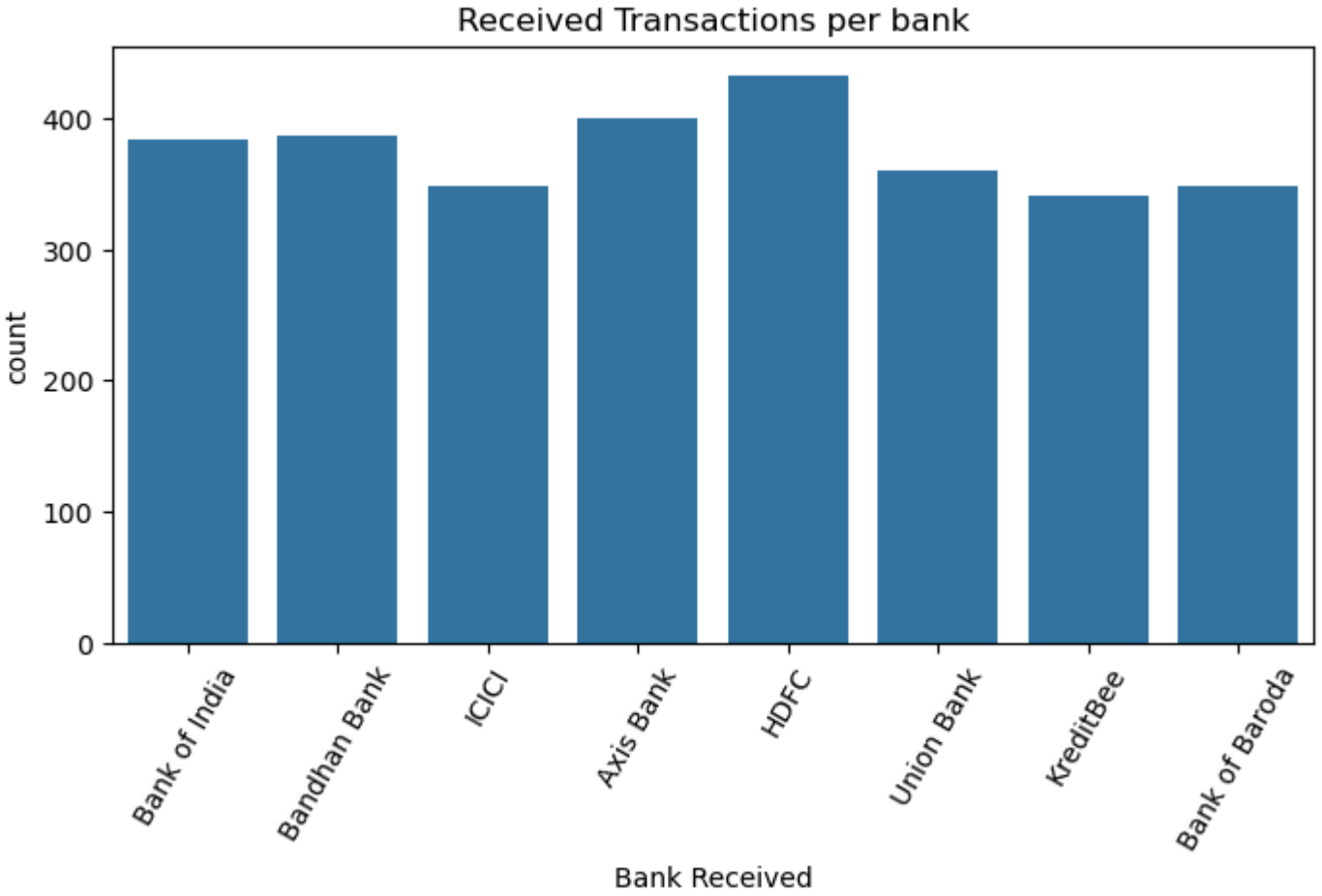
In [76]:
```
# which banks have revceived the max. or min. Transactions?

df['Bank Received'].value_counts()
```

Out[76]:
```
Bank Received
HDFC              433
Axis Bank         400
Bandhan Bank      386
Bank of India     383
Union Bank        360
Bank of Baroda    349
ICICI             348
KreditBee         341
Name: count, dtype: int64
```

Observation:-

- HDFC Bank receives maximum transactions.
- KreditBee receives minimum transactions.

In [77]:
```
plt.figure(figsize=(8,4))
sns.countplot(x=df['Bank Received'])
plt.title("Received Transactions per bank")
plt.xticks(rotation=60)
plt.savefig("Bank Received")
plt.show()
```



Observation:-

- HDFC Bank receives maximum transactions.
- KreditBee receives minimum transactions.

In [ ]:

Customer ID

In [ ]:

In [78]:
```
df['Customer ID'].nunique()
```

Out[78]:
```
22
```

In [79]:
```
df['Customer ID'].value_counts()
```

Out[79]:
```
Customer ID
2     163
16    160
17    150
8     146
15    146
20    145
7     143
5     142
18    142
19    138
1     138
21    136
12    135
3     131
9     130
22    130
14    128
10    126
4     122
11    119
6     118
13    112
Name: count, dtype: int64
```

Observation:-

- Maximum transactions has been done by customer id 2.
- Minimum transactions has been done by customer id 13.

In [ ]:

T Year

In [ ]:

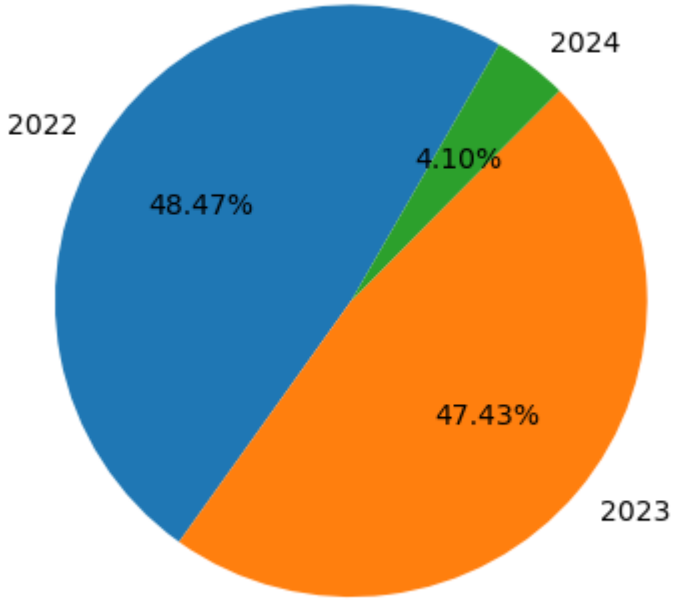
In [80]:
```
df['T Year'].unique()
```

Out[80]:
```
array([2023, 2022, 2024])
```

In [81]:
```
# In which year max.or min. UPI Transactions
# What is the trands of Transactions

df['T Year'].value_counts()
```

Out[81]:
```
T Year
2022    1454
2023    1423
2024     123
Name: count, dtype: int64
```

In [82]:
```
a=df['T Year'].value_counts().index
b=df['T Year'].value_counts().values

plt.pie(b,labels=a,autopct="%0.2f%%",startangle=60)
plt.savefig("T Year")
plt.show()
```



Observations:-

Max. transactions sent In year 2022

Min. transactions sent In year 2024

Year wise Transactions is decreasing

In [ ]:

Age Groups

In [ ]:

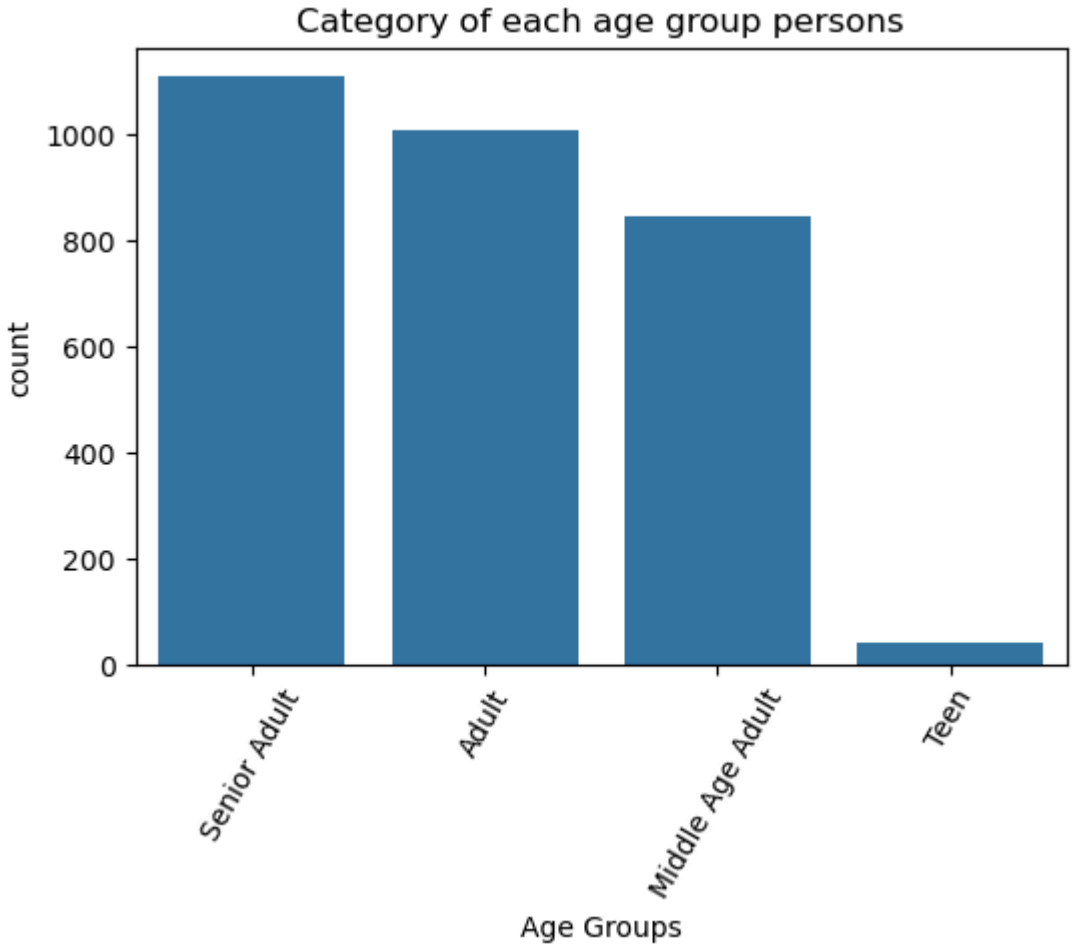
In [83]:
```
# Which age-group person is max. or min.
# Which age-group person did max. or min Transactions.


df['Age Groups'].value_counts()
```

Out[83]:
```
Age Groups
Senior Adult        1109
Adult               1007
Middle Age Adult     844
Teen                  40
Name: count, dtype: int64
```

Observation:-

- Senior Adult age group person is Maximum.
- Teen age group person is Minimum.
- Maximum UPI Transactions have been done by Senior Adult, Adult and Middle Age Adult.
- Whereas Minimun UPI Transactions have been done by Teen.

In [84]:
```
plt.figure(figsize=(6,4))
sns.countplot(x=df['Age Groups'])
plt.title("Category of each age group persons")
plt.xticks(rotation=60)
plt.savefig('Age Groups')
plt.show()
```



Observation:-

- Senior Adult age group person is Maximum.
- Teen age group person is Minimum.
- Maximum UPI Transactions have been done by Senior Adult, Adult and Middle Age Adult.
- Whereas Minimun UPI Transactions have been done by Teen.

In [ ]:

Bivariate Analysis and Visualizations

In [ ]:

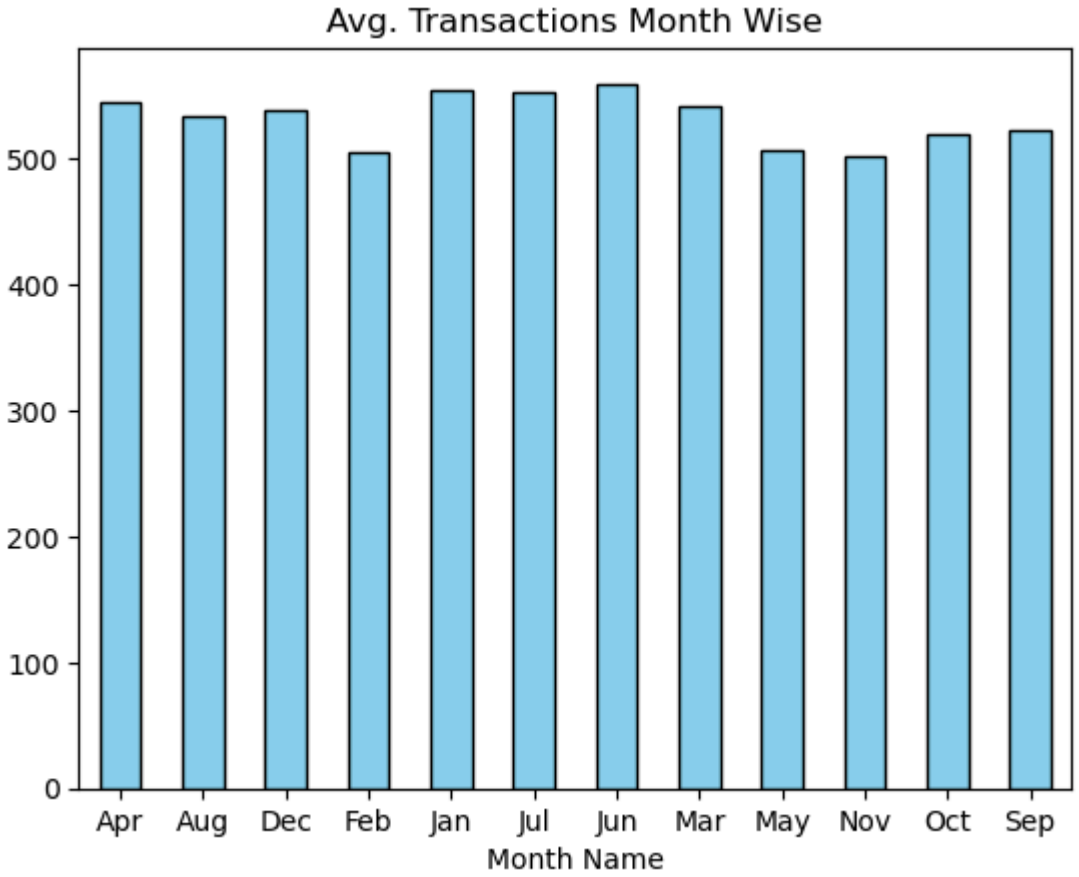
In [85]:
```
# Which month avg. Transaction is max. or min  ?
# Monthly avg. Transaction ?

a=df.groupby('Month Name')['Transaction (thousands)'].mean()
print("Avg. Monthly trans. sent -",a)

b=df.groupby('Month Name')['Transaction (thousands)'].mean().min()
print("Minimum Monthly Trans. sent -",b)

e=df.groupby('Month Name')['Transaction (thousands)'].mean().max()
print("Maximum Monthly Trans. sent -",e)
```

```
Avg. Monthly trans. sent - Month Name
Apr    545.103650
Aug    534.660805
Dec    539.512509
Feb    505.042411
Jan    554.606815
Jul    552.954488
Jun    559.997464
Mar    541.855674
May    506.625340
Nov    502.825099
Oct    519.838738
Sep    522.701538
Name: Transaction (thousands), dtype: float64
Minimum Monthly Trans. sent - 502.8250992796405
Maximum Monthly Trans. sent - 559.997464174406
```

Minimum Avg. Monthly Trans. sent - 503 in November month

Maximum Avg. Monthly Trans. sent - 560 in June month

In [86]:
```
a.plot(kind="bar",color="skyblue",edgecolor="black")
plt.title("Avg. Transactions Month Wise")
plt.xticks(rotation=0)
plt.savefig('Average Transactions Month Wise')
plt.show()
```



Minimum Avg. Monthly Trans. sent - 503 in November month

Maximum Avg. Monthly Trans. sent - 560 in June month

In [ ]:

In [ ]:

In [87]:
```
# what is Day wise avg. Transactions?
#which Day avg.min. Transactions?
#which Day max. avg. Transactions?

b=df.groupby('Day Name')['Transaction (thousands)'].mean()
print("Day wise avg. Transactions",b)

c=df.groupby('Day Name')['Transaction (thousands)'].mean().min()
print("Day wise avg.min. Transactions",c)
```

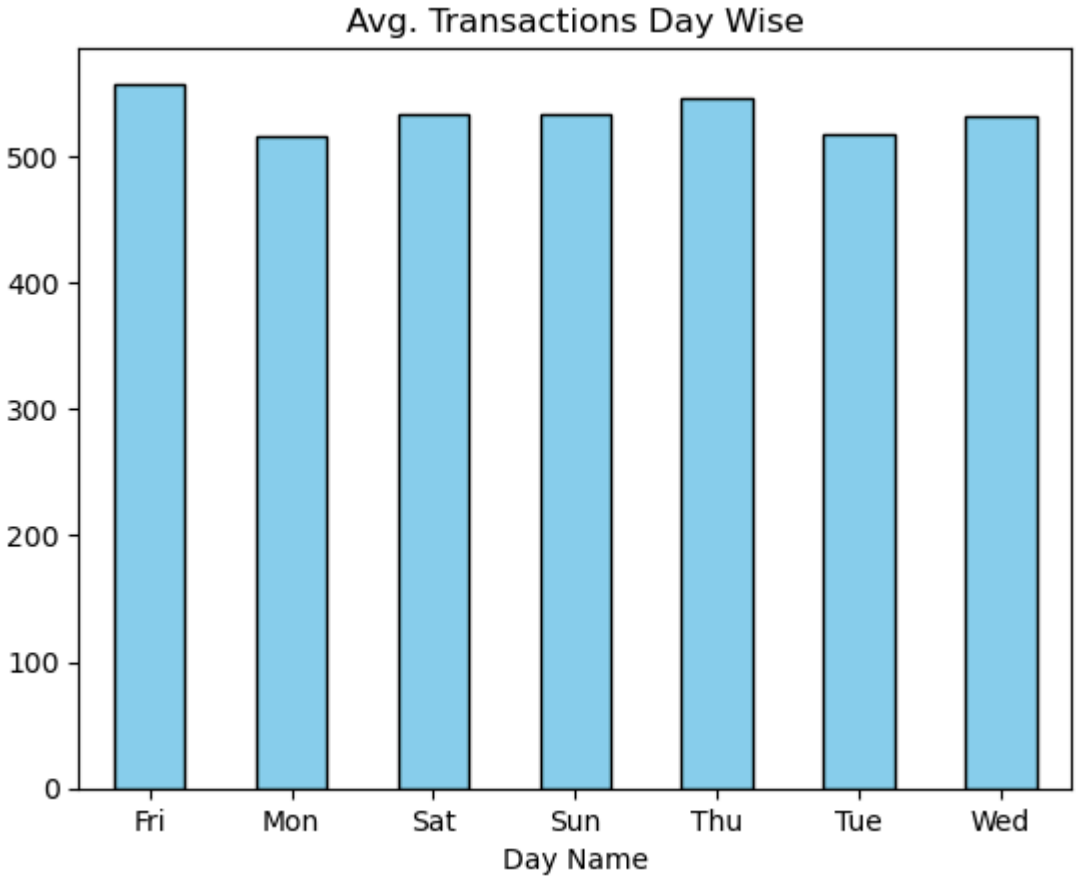
```
d=df.groupby('Day Name')['Transaction (thousands)'].mean().max()
print("Day wise max. avg. Transactions",d)
```

```
Day wise avg. Transactions Day Name
Fri    557.209567
Mon    516.249609
Sat    533.707618
Sun    533.381940
Thu    545.822420
Tue    516.974212
Wed    531.743005
Name: Transaction (thousands), dtype: float64
Day wise avg.min. Transactions 516.249608624639
Day wise max. avg. Transactions 557.2095667588576
```

Day wise avg.min. Transactions 516 on Thu.

Day wise max. avg. Transactions 557 on Fri.

In [88]:
```
b.plot(kind="bar",color="skyblue",edgecolor="black")
plt.title("Avg. Transactions Day Wise")
plt.xticks(rotation=0)
plt.savefig('Avg Transactions Day Wise')
plt.show()
```



Day wise avg.min. Transactions 516 on Thu.

Day wise max. avg. Transactions 557 on Fri.

In [ ]:

In [89]:
```
#Avg. Transactions of each year?
#Min. Avg. Transactions in which year?
# Max. Avg. Transactions in which year?



c=df.groupby('T Year')['Transaction (thousands)'].mean()
print('Avg. Transactions of each year',c)

e=df.groupby('T Year')['Transaction (thousands)'].mean().min()
print('Min. Avg. Transactions',e)
```

```
f=df.groupby('T Year')['Transaction (thousands)'].mean().max()
print('Max. Avg. Transactions',f)
```

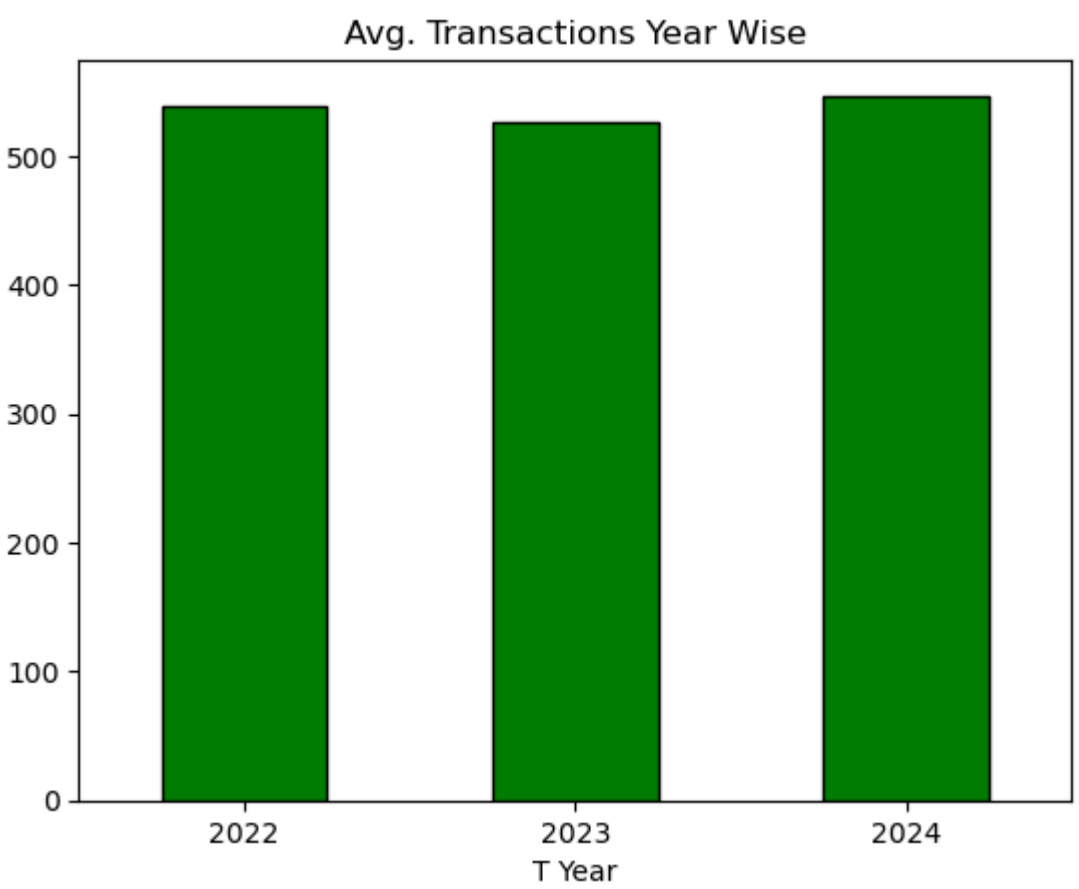
```
Avg. Transactions of each year T Year
2022    539.626080
2023    526.680281
2024    547.096941
Name: Transaction (thousands), dtype: float64
Min. Avg. Transactions 526.6802811270184
Max. Avg. Transactions 547.096941244142
```

Min. Avg. Transactions 526 in year 2023

Max. Avg. Transactions 547 in year 2024

In [ ]:

In [90]:
```
c.plot(kind="bar",color="green",edgecolor="black")
plt.title("Avg. Transactions Year Wise")
plt.xticks(rotation=0)
plt.savefig("Avg Transactions Year Wise")
plt.show()
```



Min. Avg. Transactions 526 in year 2023

Max. Avg. Transactions 547 in year 2024

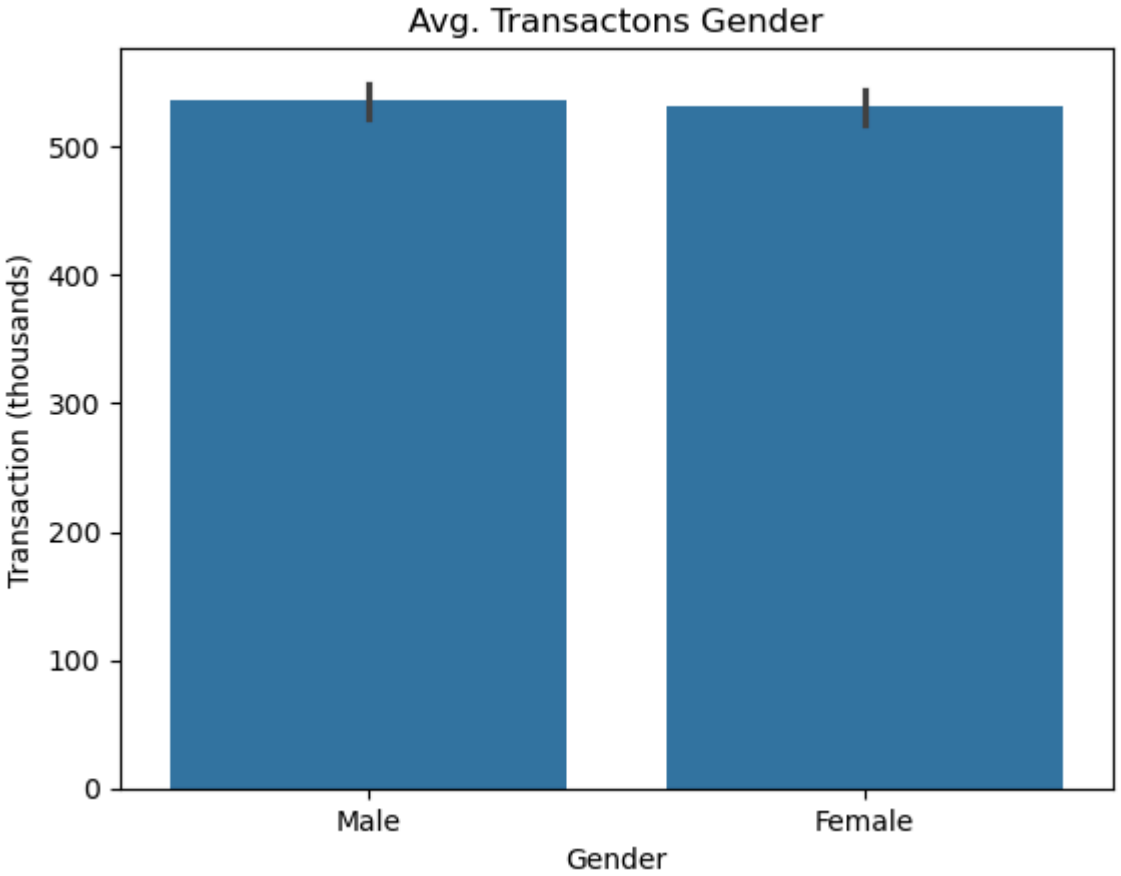
In [ ]:

In [91]:
```
#which gender done max. or min. avg. transactons?


df.groupby("Gender")['Transaction (thousands)'].mean()
```

Out[91]:
```
Gender
Female    531.472044
Male      536.068578
Name: Transaction (thousands), dtype: float64
```

In [92]:
```
sns.barplot(x="Gender",y="Transaction (thousands)",data=df)
plt.title("Avg. Transactons Gender")
```

```
plt.show()
```



Male does slightly more transaction by Female so it doesnot affect the transction

In [ ]:

In [ ]:

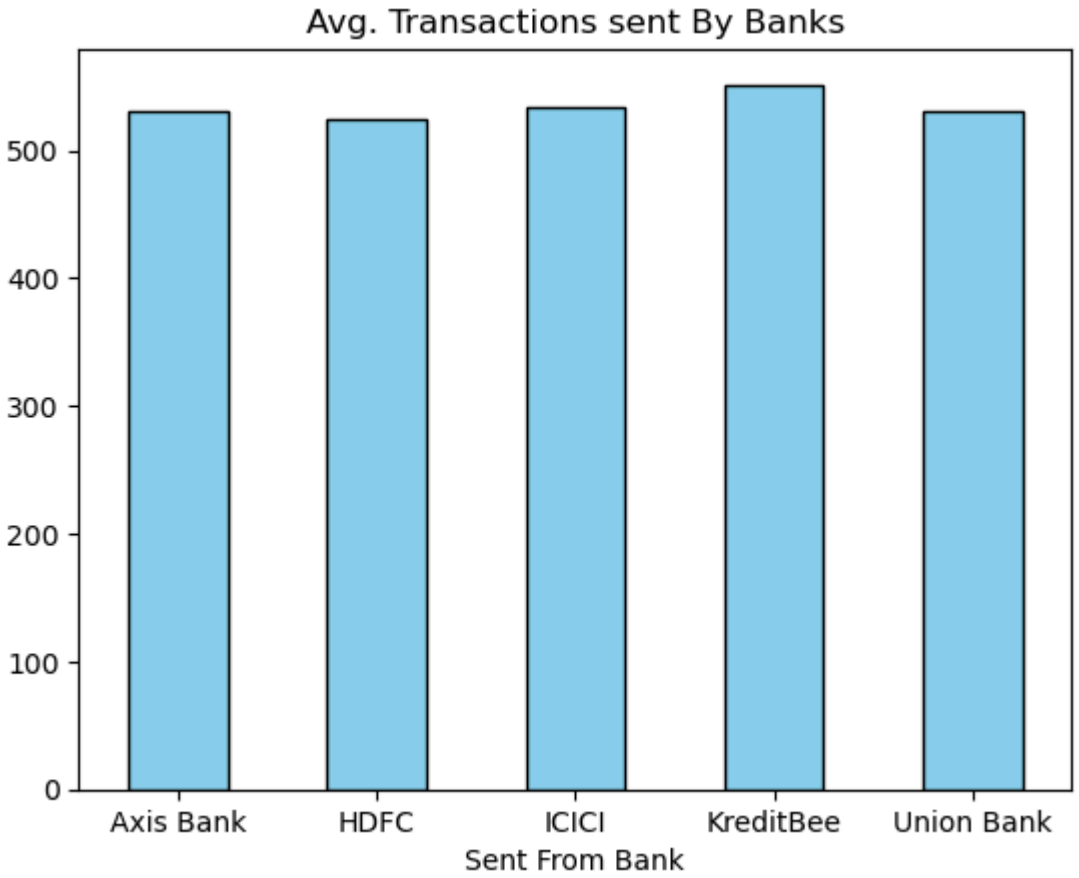
In [93]:
```
#which bank sent max. or min. avg. transactons?


x=df.groupby("Sent From Bank")['Transaction (thousands)'].mean()
x
```

Out[93]:
```
Sent From Bank
Axis Bank     530.369697
HDFC          524.275412
ICICI         533.282720
KreditBee     551.565300
Union Bank    529.992306
Name: Transaction (thousands), dtype: float64
```

Max. Avg. Transactions sent by KreditBee

Min. Avg. Transactions sent by HDFC

In [94]:
```
x.plot(kind="bar",color="skyblue",edgecolor="black")
plt.title("Avg. Transactions sent By Banks")
plt.xticks(rotation=0)
plt.show()
```



Max. Avg. Transactions sent by KreditBee

Min. Avg. Transactions sent by HDFC

In [ ]:

In [ ]:

In [95]:
```
#Avg. Transactions received by each bank?
#Min. Avg. Transactions received?
#Max.Avg. Transactions received?
s=df.groupby("Bank Received")['Transaction (thousands)'].mean()
s
```

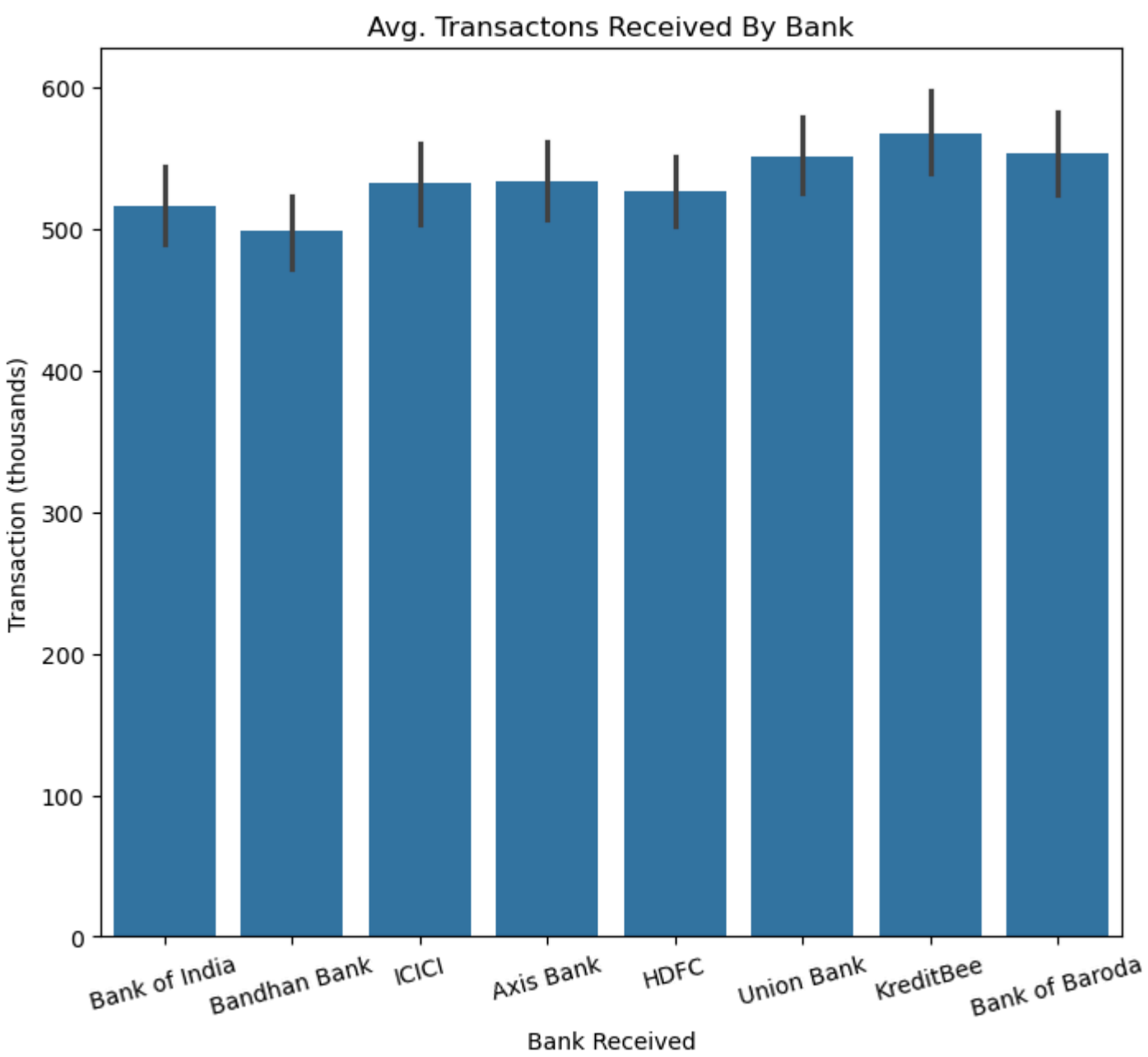
Out[95]:
```
Bank Received
Axis Bank          533.793879
Bandhan Bank       497.995993
Bank of Baroda     552.974673
Bank of India      515.910187
HDFC               526.443505
ICICI              532.384000
KreditBee          567.552512
Union Bank         550.817892
Name: Transaction (thousands), dtype: float64
```

Min. Avg. Transactions received by Bandhan Bank

Max.Avg. Transactions received by KreditBee Bank

In [ ]:

In [96]:
```
plt.figure(figsize=(8,7))
sns.barplot(x="Bank Received",y="Transaction (thousands)",data=df)
plt.title("Avg. Transactons Received By Bank")
plt.xticks(rotation=15)
plt.show()
```



Min. Avg. Transactions received by Bandhan Bank

Max.Avg. Transactions received by KreditBee Bank

In [ ]:

In [ ]:

In [97]:
```
#Max. money transformed in which city by which bank?

pd.crosstab(df['City'],df['Sent From Bank'],margins=1)
```

Out[97]:

Sent From Bank / City	Axis Bank	HDFC	ICICI	KreditBee	Union Bank	All
Bangalore	121	117	127	115	136	616
Chennai	120	121	130	105	121	597
Delhi	130	108	121	134	107	600
Hyderabad	128	126	118	113	130	615
Mumbai	100	114	128	111	119	572
All	599	586	624	578	613	3000

In Mumbai max. trans. done by ICICI Bank.

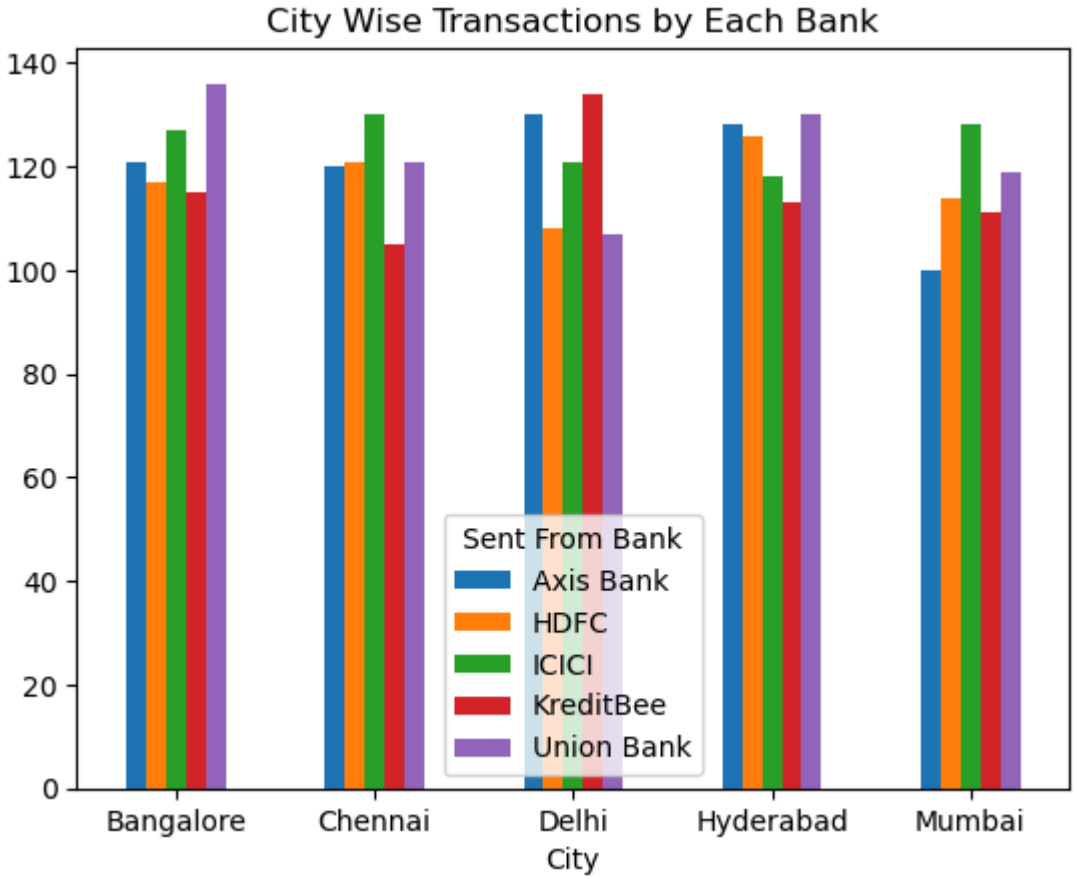
In Hyderabad max. trans. done by Union Bank.

In Chennai max. trans. done by ICICI Bank.

In Delhi max. trans. done by KreditBee Bank.

In Bangalore max. trans. done by Union Bank.

In [98]:
```
# plt.figure(figsize=(20,10))
pd.crosstab(df['City'],df['Sent From Bank']).plot(kind="bar")
plt.title("City Wise Transactions by Each Bank")
plt.xticks(rotation=0)
plt.show()
```



In Mumbai max. trans. done by ICICI Bank.

In Hyderabad max. trans. done by Union Bank.

In Chennai max. trans. done by ICICI Bank.

In Delhi max. trans. done by KreditBee Bank.

In Bangalore max. trans. done by Union Bank.

In [ ]:

In [ ]:

In [99]:
```
#Max. money received in which city by which bank?

pd.crosstab(df['City'],df['Bank Received'],margins=1)
```

Out[99]:

Bank Received City	Axis Bank	Bandhan Bank	Bank of Baroda	Bank of India	HDFC	ICICI	KreditBee	Union Bank	All
Bangalore	78	66	75	76	93	85	66	77	616
Chennai	82	83	66	82	81	64	66	73	597
Delhi	82	82	77	82	81	60	74	62	600
Hyderabad	75	89	65	70	87	76	67	86	615
Mumbai	83	66	66	73	91	63	68	62	572
All	400	386	349	383	433	348	341	360	3000

In [ ]:

In Bangalore max. trans. received by HDFC Bank.

In Chennai max. trans. received by Bandhan Bank.

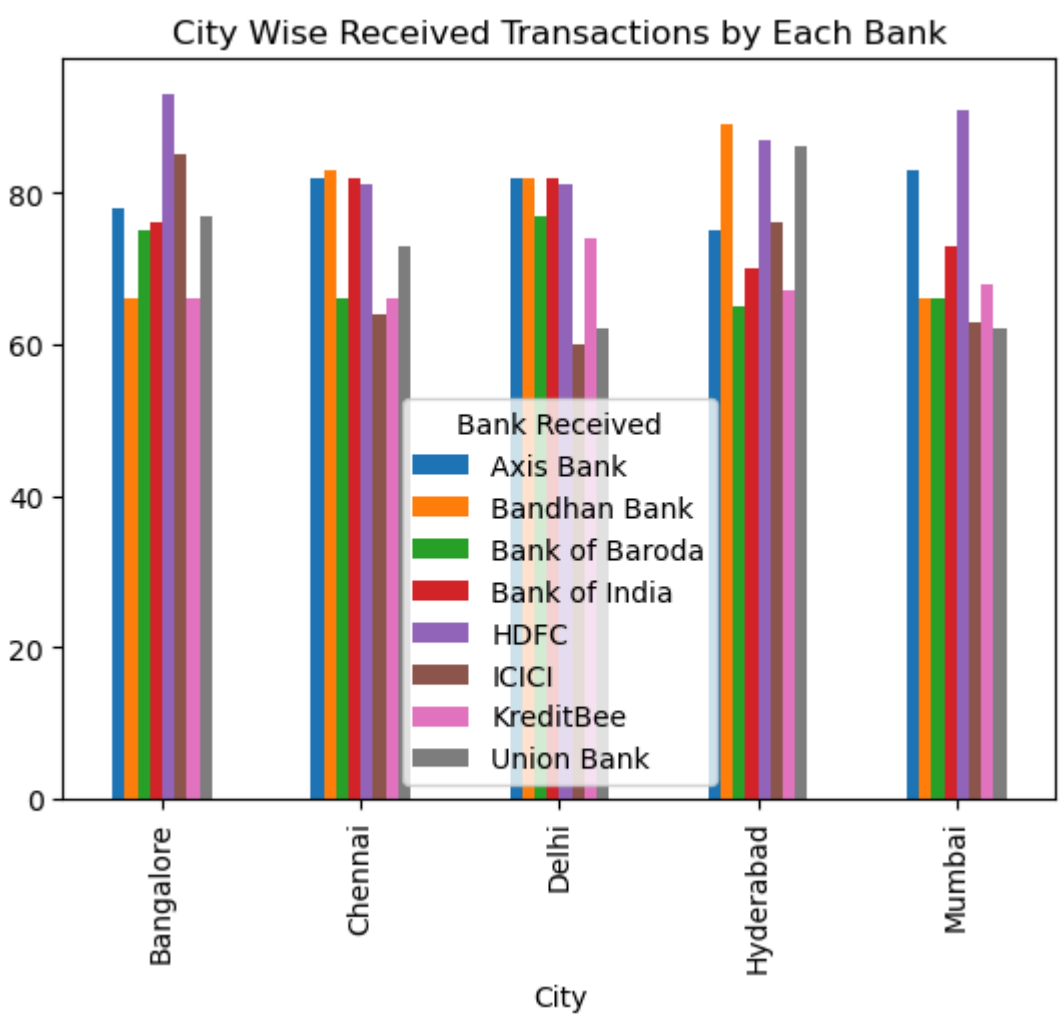
In Delhi max. trans. received by Axis Bank,Bandhan Bank & Bank of India.

In Hyderabad max. trans. received by Bandhan Bank.

In Mumbai max. trans. received by HDFC Bank.

In [100…

```
pd.crosstab(df['City'],df['Bank Received']).plot(kind="bar")
plt.title("City Wise Received Transactions by Each Bank")
# plt.xticks(rotation=0)
plt.show()
```



In Bangalore max. trans. received by HDFC Bank.

In Chennai max. trans. received by Bandhan Bank.

In Delhi max. trans. received by Axis Bank,Bandhan Bank & Bank of India.

In Hyderabad max. trans. received by Bandhan Bank.

In Mumbai max. trans. received by HDFC Bank.

In [ ]:

In [101…

```
#which age group persons are using which bank max. or min.?
```

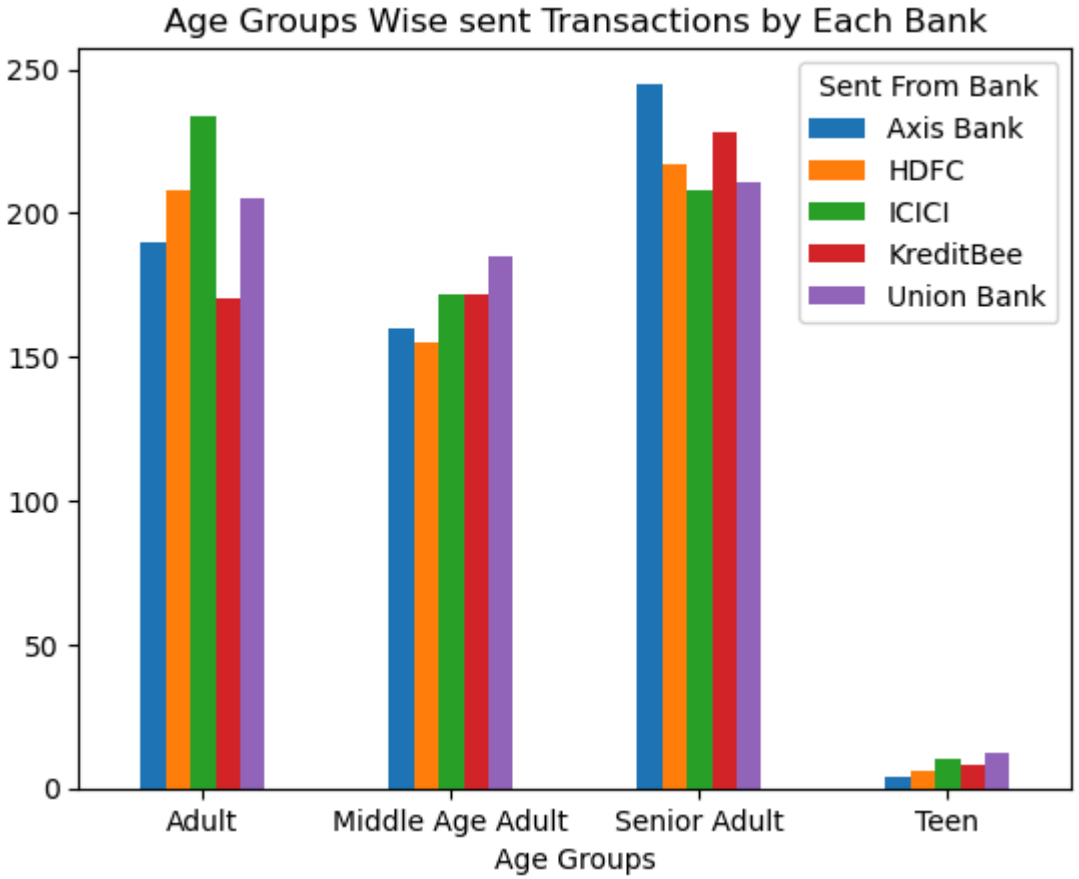
```
pd.crosstab(df['Age Groups'],df['Sent From Bank'],margins=1)
```

Out[101…

Sent From Bank	Axis Bank	HDFC	ICICI	KreditBee	Union Bank	All
Age Groups						
Adult	190	208	234	170	205	1007
Middle Age Adult	160	155	172	172	185	844
Senior Adult	245	217	208	228	211	1109
Teen	4	6	10	8	12	40
All	599	586	624	578	613	3000

In [102…

```
pd.crosstab(df['Age Groups'],df['Sent From Bank']).plot(kind="bar")
plt.title("Age Groups Wise sent Transactions by Each Bank")
plt.xticks(rotation=0)
plt.show()
```



Adults are using ICICI Bank the most and least KreditBee

Middle Age Adults are using Union Bank the most and least HDFC

Senior Adults are using Axis Bank the most and least ICICI

Teens are using Union Bank the most and least Axis bank

In [ ]:

In [103…

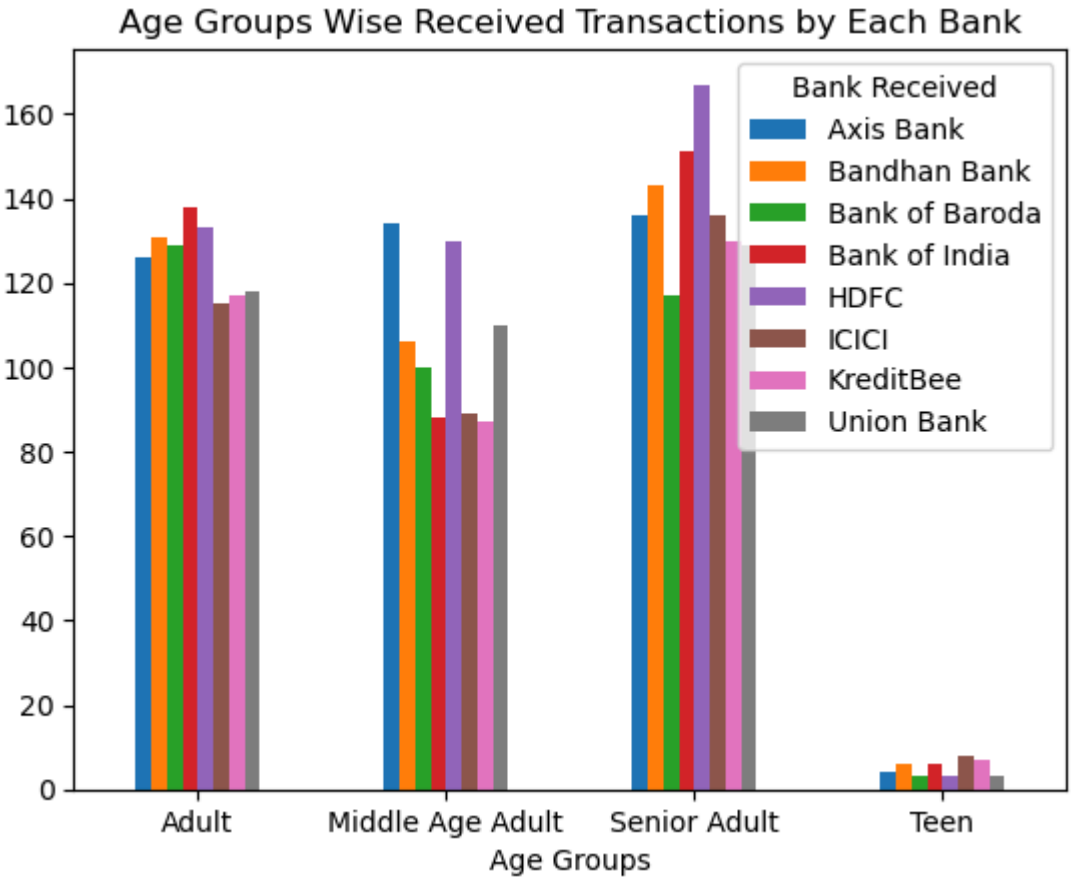
```
#which age group persons are using which bank max. or min.?

pd.crosstab(df['Age Groups'],df['Bank Received'],margins=1)
```

Out[103…

Bank Received	Axis Bank	Bandhan Bank	Bank of Baroda	Bank of India	HDFC	ICICI	KreditBee	Union Bank	All
Age Groups									
Adult	126	131	129	138	133	115	117	118	1007
Middle Age Adult	134	106	100	88	130	89	87	110	844
Senior Adult	136	143	117	151	167	136	130	129	1109
Teen	4	6	3	6	3	8	7	3	40
All	400	386	349	383	433	348	341	360	3000

In [104…

```
pd.crosstab(df['Age Groups'],df['Bank Received']).plot(kind="bar")
plt.title("Age Groups Wise Received Transactions by Each Bank")
plt.xticks(rotation=0)
plt.show()
```



Adults are using Bank of India the most and least ICICI

Middle Age Adults are using Axis Bank the most and least KreditBee

Senior Adults are using HDFC Bank the most and least Bank of Baroda

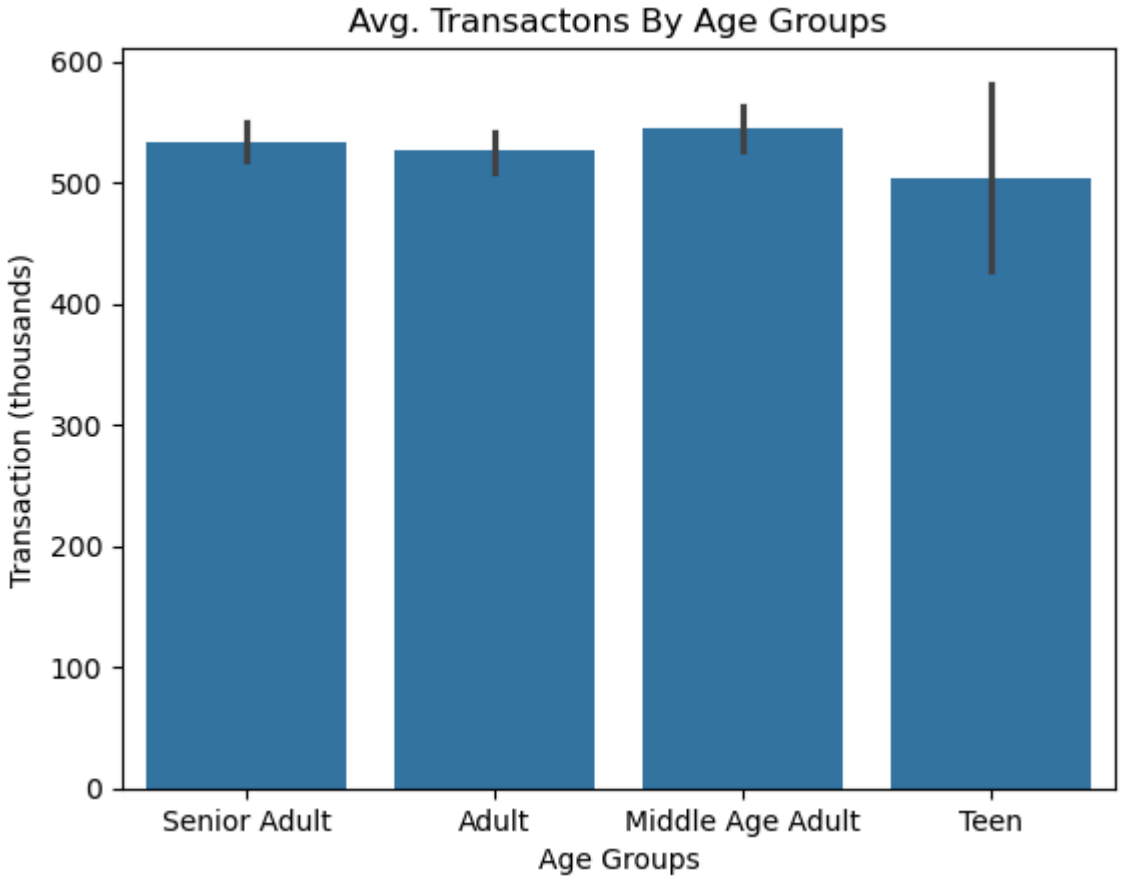
Teens are using ICICI Bank the most and least Union bank

In [ ]:

In [105…
```
#Min. Avg. Transactions by which age group?
#Max.Avg. Transactions by which age group?
df.groupby("Age Groups")['Transaction (thousands)'].mean()
```

Out[105…
```
Age Groups
Adult                525.862410
Middle Age Adult     545.222166
Senior Adult         533.366047
Teen                 504.034601
Name: Transaction (thousands), dtype: float64
```

In [106…
```
# plt.figure(figsize=(8,7))
sns.barplot(x="Age Groups",y="Transaction (thousands)",data=df)
plt.title("Avg. Transactons By Age Groups")
# plt.xticks(rotation=15)
plt.show()
```



Max. Transactions done by Middle Age Adult

Min. Transactions done by Teen Age Adult

In [ ]:

Multivariate Analysis and Visualizations

In [ ]:

In [107…

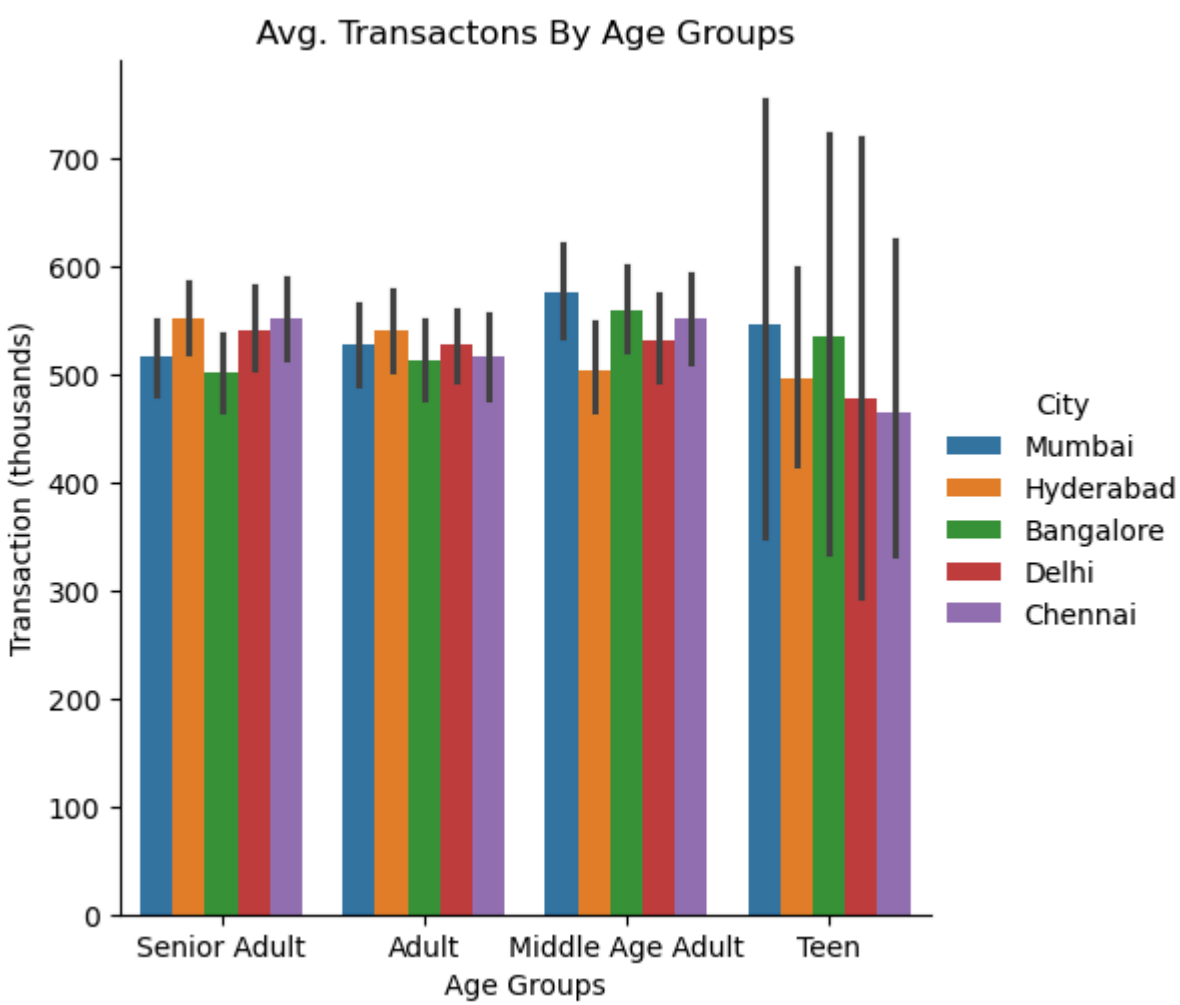
```
#which age groups and city do highest or lowest avg. Transactions?

a=df.groupby(['Age Groups','City'])['Transaction (thousands)'].mean()
a
```

Out[107…

```
Age Groups         City
Adult              Bangalore    513.942037
                   Chennai      516.561417
                   Delhi        528.199853
                   Hyderabad    542.155816
                   Mumbai       528.244573
Middle Age Adult   Bangalore    560.240202
                   Chennai      551.698653
                   Delhi        532.175992
                   Hyderabad    504.903847
                   Mumbai       576.141944
Senior Adult       Bangalore    502.323111
                   Chennai      552.084574
                   Delhi        541.628651
                   Hyderabad    552.313538
                   Mumbai       517.658950
Teen               Bangalore    535.835062
                   Chennai      465.474412
                   Delhi        478.689969
                   Hyderabad    496.813794
                   Mumbai       547.277192
Name: Transaction (thousands), dtype: float64
```

In [108…

```
sns.catplot(y="Transaction (thousands)",x="Age Groups",hue="City",data=df,kind="bar")
plt.title("Avg. Transactons By Age Groups")
plt.show()
```



Senior Adult people do the max. trans. in Hyderabad and Min. in Bangalore

Adult people do the max. trans. in Hyderabad and Min. in Bangalore

Middle Age Adult people do the max. trans. in Mumbai and Min. in Chennai

Teen Age Adult people do the max. trans. in Mumbai and Min. in Hyderabad

In [ ]:

In [ ]:

In [109…

```
#which T Year and which city do highest Transactions?
```

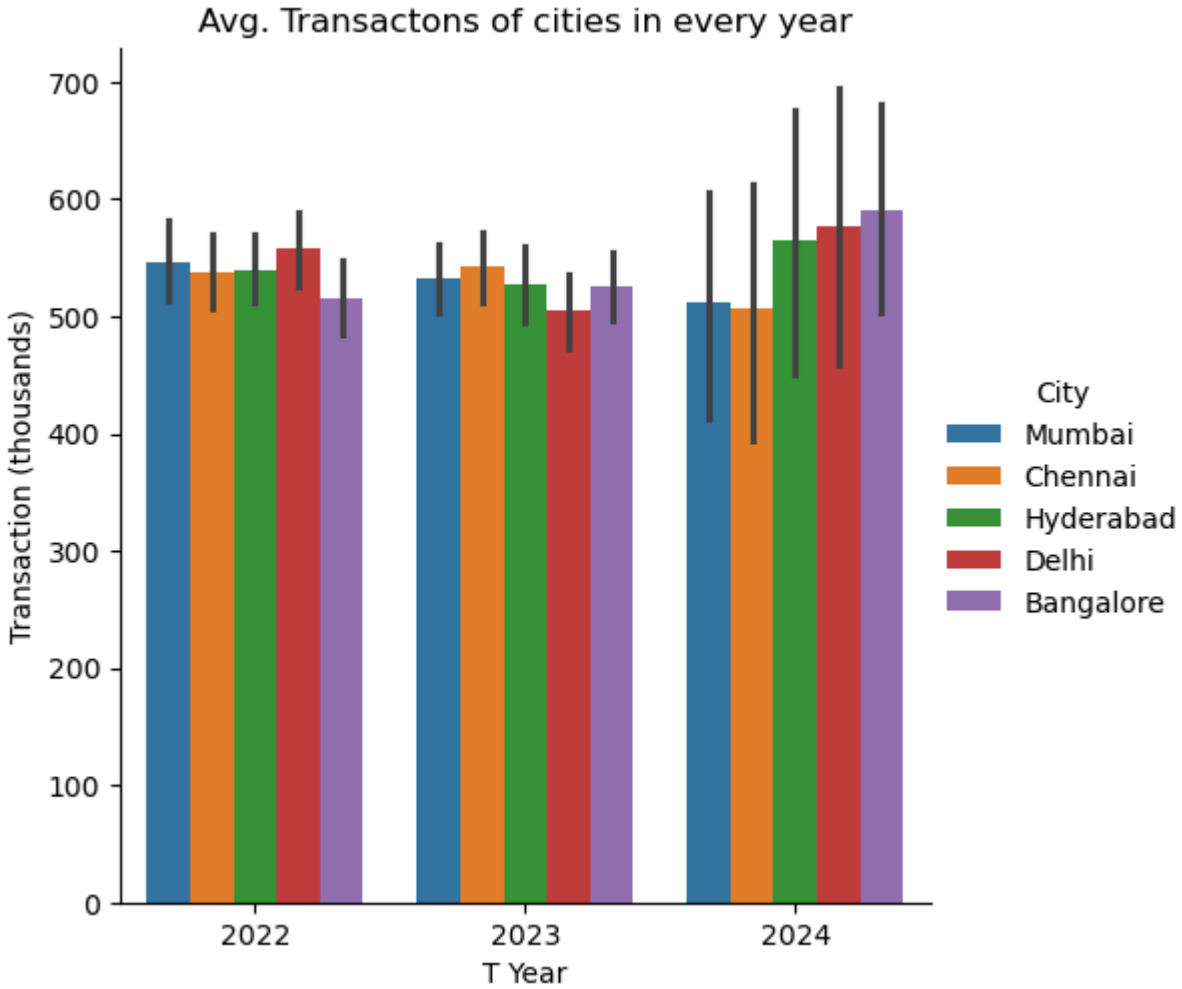
```
df.groupby(['T Year','City'])['Transaction (thousands)'].mean()
```

Out[109…

```
T Year  City
2022    Bangalore    515.775204
        Chennai      537.762501
        Delhi        558.549977
        Hyderabad    539.970760
        Mumbai       546.688030
2023    Bangalore    526.144529
        Chennai      542.556349
        Delhi        504.463746
        Hyderabad    527.955705
        Mumbai       533.108286
2024    Bangalore    591.177591
        Chennai      506.908527
        Delhi        577.482047
        Hyderabad    565.184530
        Mumbai       511.813531
Name: Transaction (thousands), dtype: float64
```

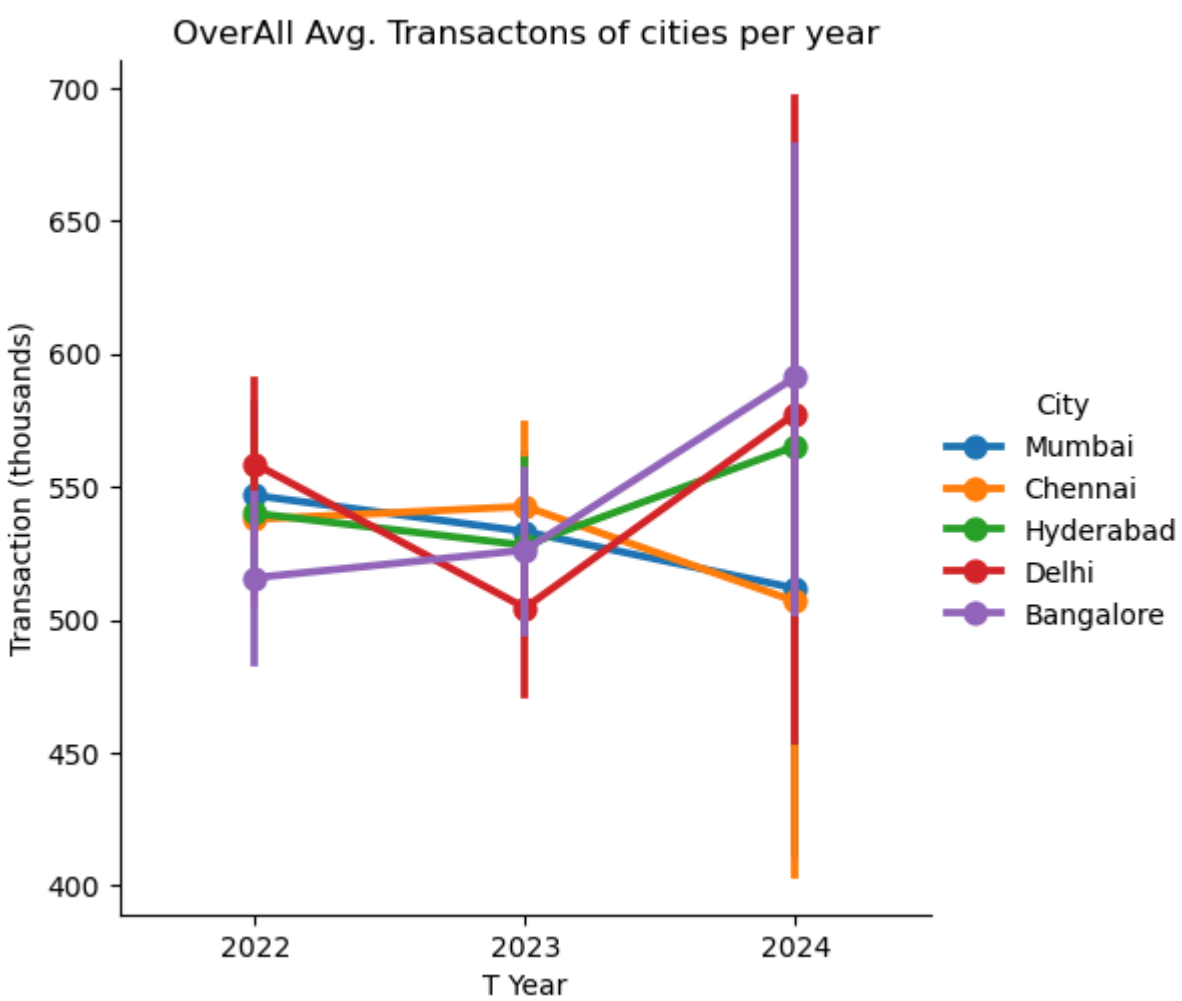
In [110…

```
sns.catplot(y="Transaction (thousands)",x="T Year",hue="City",data=df,kind="bar")
plt.title("Avg. Transactons of cities in every year")
plt.show()
```



In [111…

```
# relationship among all over cities transactions are increasing or decreasing?

sns.catplot(y="Transaction (thousands)",x="T Year",hue="City",data=df,kind="point")
plt.title(" OverAll Avg. Transactons of cities per year ")
plt.show()
```



Observations

- Bangalore kept increasing each year from 2022 to 2024.
- Delhi dropped in 2023 but rose again in 2024.
- Chennai stayed almost the same or slightly decreased in 2024.
- Hyderabad went up or down at different times — check the exact values.
- Mumbai stayed steady or changed just a little.

In [ ]:

In [112…

```
#Whether the  avg. transactions are increasing or decreasing every years for each bank?
```

```
df.groupby(['T Year','Sent From Bank'])['Transaction (thousands)'].mean()
```

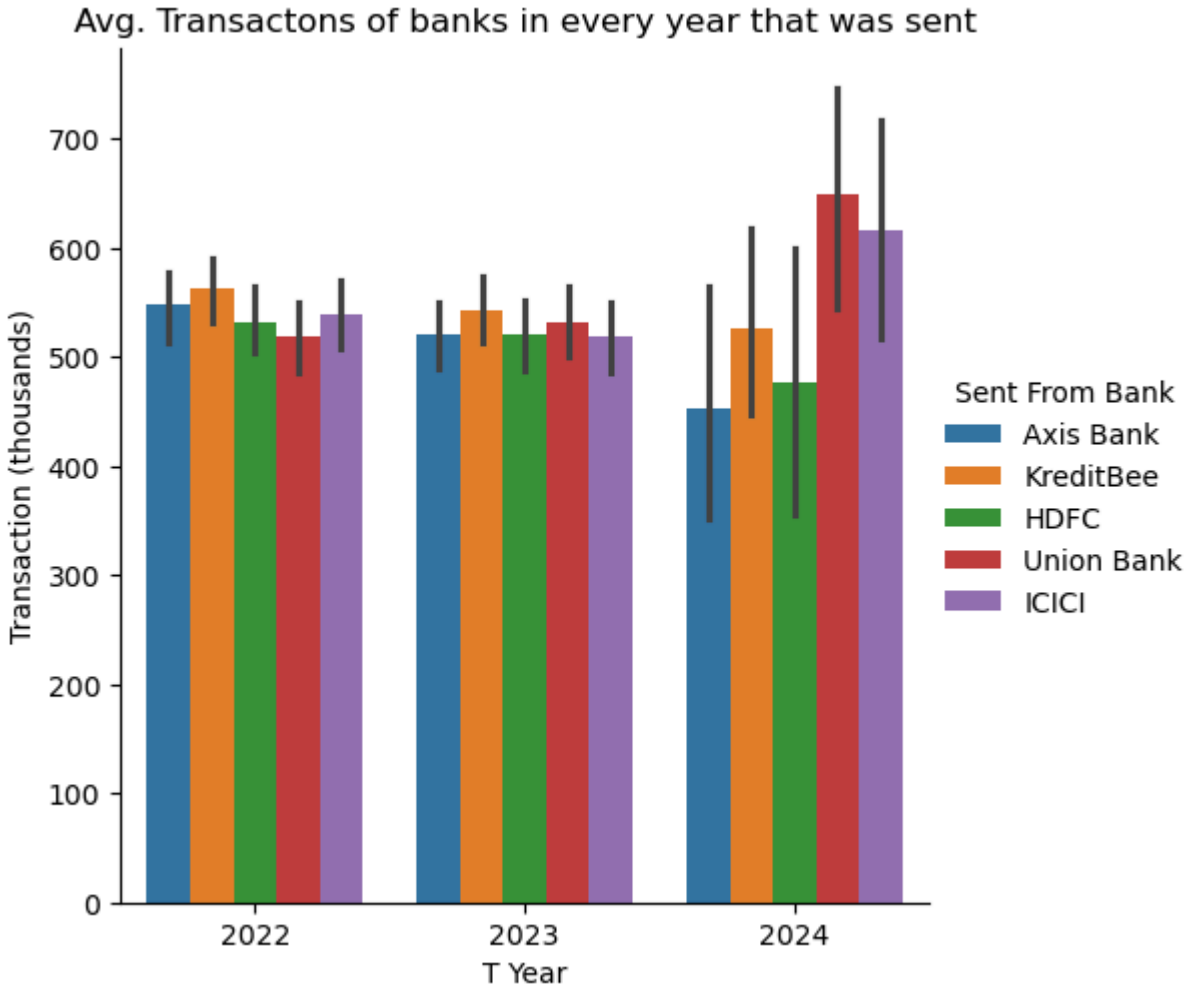
Out[112…

```
T Year  Sent From Bank
2022    Axis Bank         547.999420
        HDFC              531.723434
        ICICI             538.429231
        KreditBee         562.091426
        Union Bank        518.829073
2023    Axis Bank         519.847155
        HDFC              520.426107
        ICICI             519.153568
        KreditBee         542.669232
        Union Bank        531.744975
2024    Axis Bank         452.157526
        HDFC              477.037536
        ICICI             616.428171
        KreditBee         526.553156
        Union Bank        649.569902
Name: Transaction (thousands), dtype: float64
```

In [ ]:

In [113…

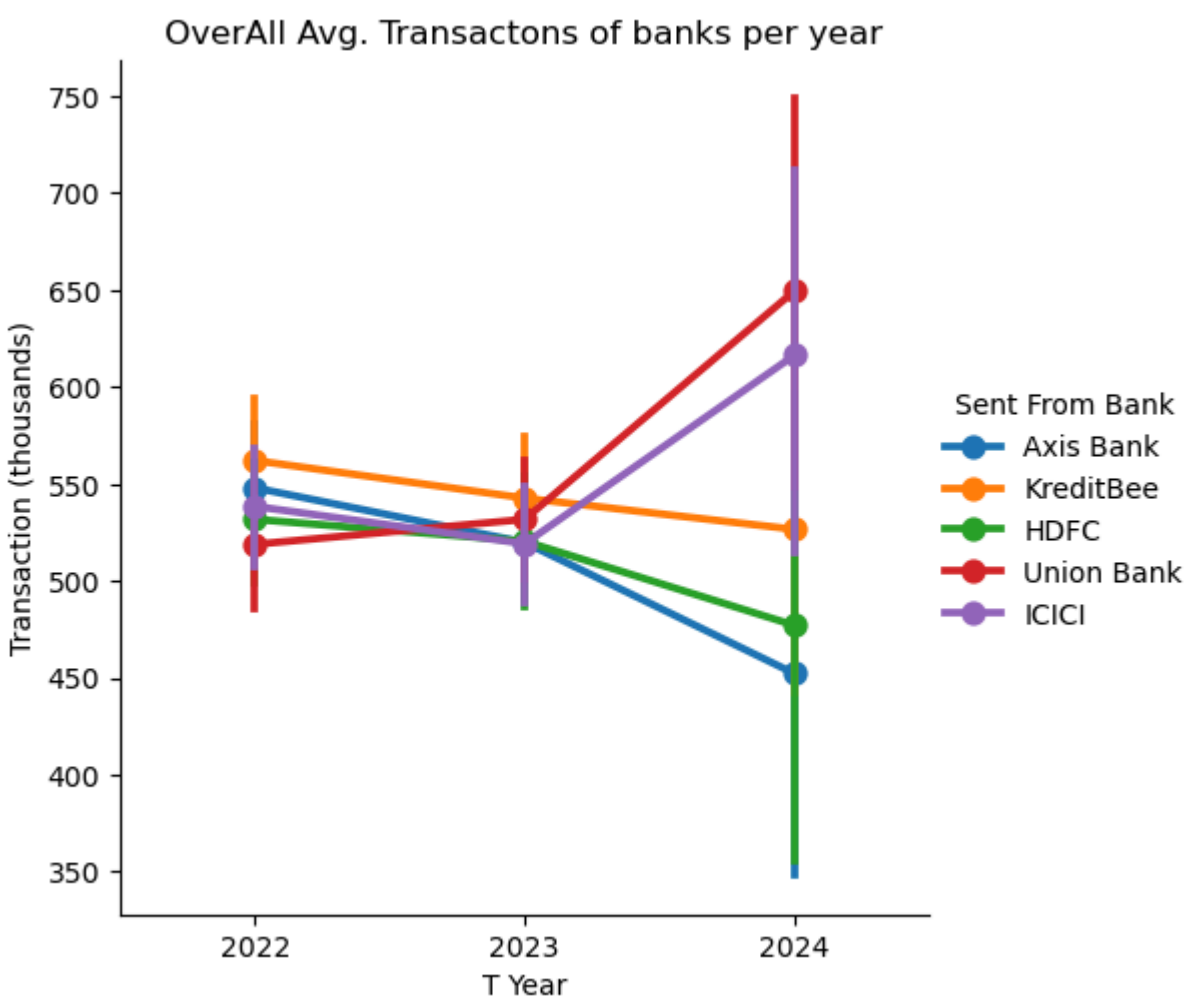
```
sns.catplot(y="Transaction (thousands)",x="T Year",hue="Sent From Bank",data=df,kind="bar")
plt.title("Avg. Transactons of banks in every year that was sent")
plt.show()
```



In [ ]:

In [114…

```
# relationship among all over year transactions are increasing or decreasing that was sent?

sns.catplot(y="Transaction (thousands)",x="T Year",hue="Sent From Bank",data=df,kind="point")
plt.title(" OverAll Avg. Transactons of banks per year ")
plt.show()
```



Union Bank and ICICI are growing in popularity for UPI transactions.

HDFC and Axis Bank are declining in transaction usage.

KreditBee maintains user base.

In [ ]:

In [115…

```
#Whether the transactions are increasing or decreasing over years that was received?

df.groupby(['T Year','Bank Received'])['Transaction (thousands)'].mean()
```

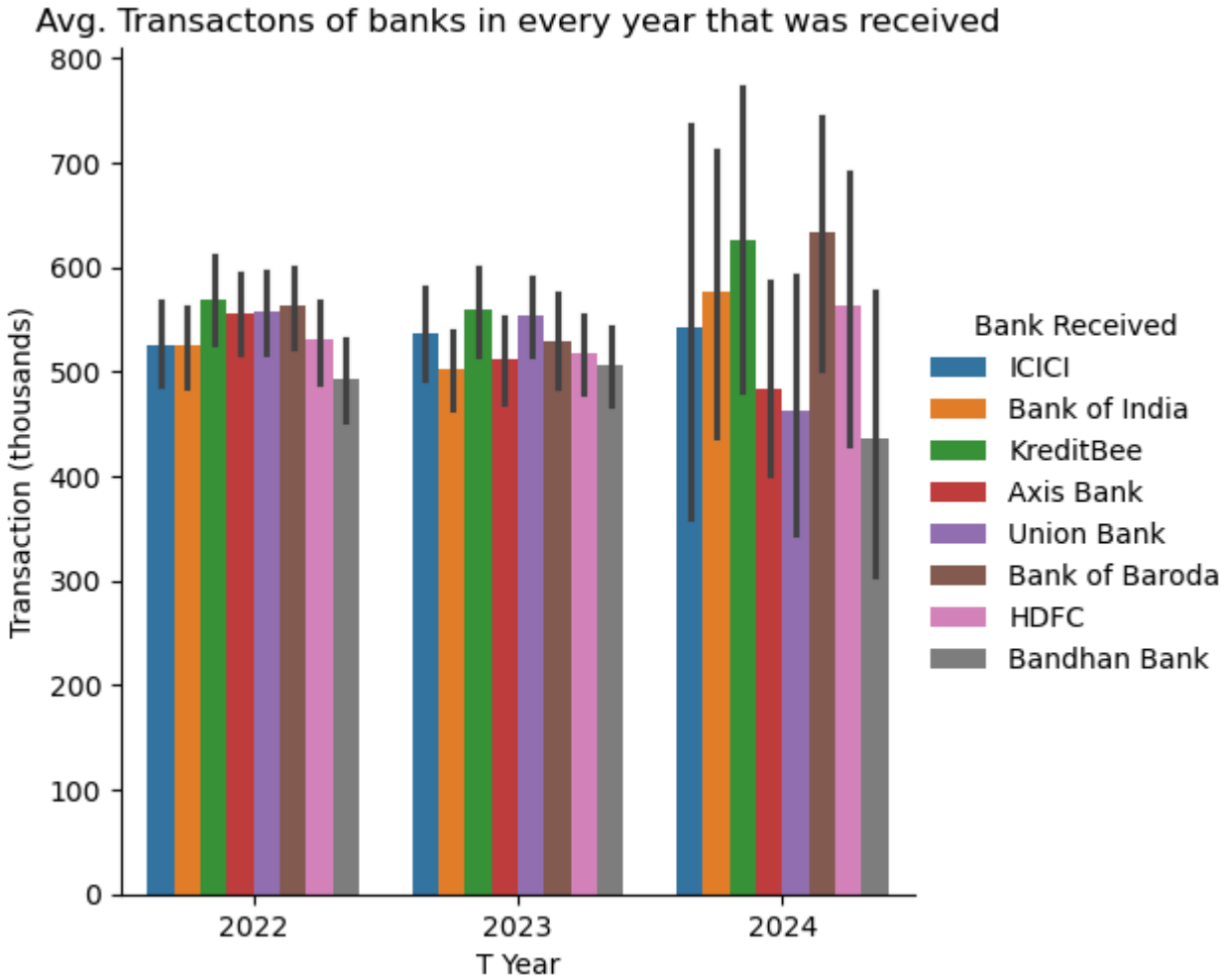
Out[115…

```
T Year  Bank Received
2022    Axis Bank          555.700000
        Bandhan Bank       492.425715
        Bank of Baroda     564.198796
        Bank of India      525.326085
        HDFC               530.615576
        ICICI              526.255016
        KreditBee          569.827503
        Union Bank         558.177940
2023    Axis Bank          512.887715
        Bandhan Bank       507.236290
        Bank of Baroda     528.780776
        Bank of India      501.804510
        HDFC               518.269667
        ICICI              537.623736
        KreditBee          558.912508
        Union Bank         553.735450
2024    Axis Bank          484.330756
        Bandhan Bank       435.867997
        Bank of Baroda     633.973004
        Bank of India      576.874716
        HDFC               563.315540
        ICICI              542.864813
        KreditBee          626.281157
        Union Bank         462.915492
Name: Transaction (thousands), dtype: float64
```

In [ ]:

In [116…

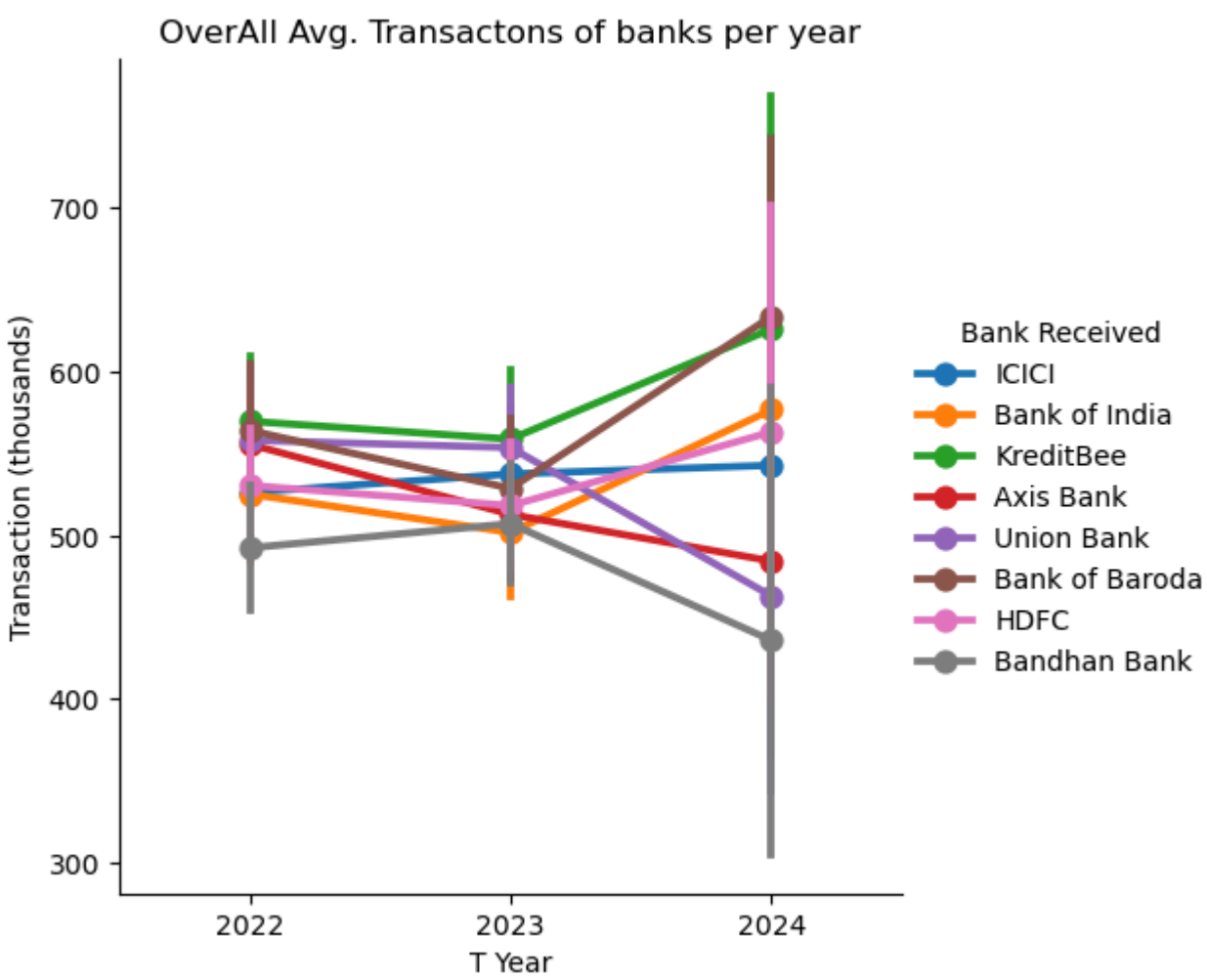
```
sns.catplot(y="Transaction (thousands)",x="T Year",hue="Bank Received",data=df,kind="bar")
plt.title("Avg. Transactons of banks in every year that was received")
plt.show()
```



In [ ]:

In [117…

```
# relationship among all over banks transactions are increasing or decreasing that eas receivedL?

sns.catplot(y="Transaction (thousands)",x="T Year",hue="Bank Received",data=df,kind="point")
plt.title(" OverAll Avg. Transactons of banks per year ")
plt.show()
```



Observations:-

- KreditBee bank has most preferred bank for receiving payments
- ICICI and Bank of Baroda also showed steady growth over the years
- Bandhan Bank witnessed a decline in transactions
- Bank of India showed a fluctuating trend in UPI transaction
- HDFC, Union Bank, and Axis Bank maintained relatively stable performance.

In [ ]:

Conclusions:-

- UPI transactions across all major cities were high in 2022 and 2023.
- A sharp decline in 2024 is visible in all cities, likely due to incomplete data or external factors
- Bangalore and Mumbai showed consistent growth initially before the 2024 drop.
- Hyderabad is the only city that shows a slight recovery in 2024 after a dip.
- Overall, while UPI usage was strong in earlier years, 2024 shows a concerning drop that needs further investigation.

In [ ]: