Data Visualization: Project

TRADE & BUSINESS

NEELMANI: 224161019

In [1]:
```
import pandas as pd
from plotnine import *
from pandas.api.types import CategoricalDtype
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import calendar
import numpy as np
```

In [2]:
```
# LOAD THE DATA
startup_df = pd.read_csv('startup_funding.csv')
startup_df.head(5)
```

Out[2]:

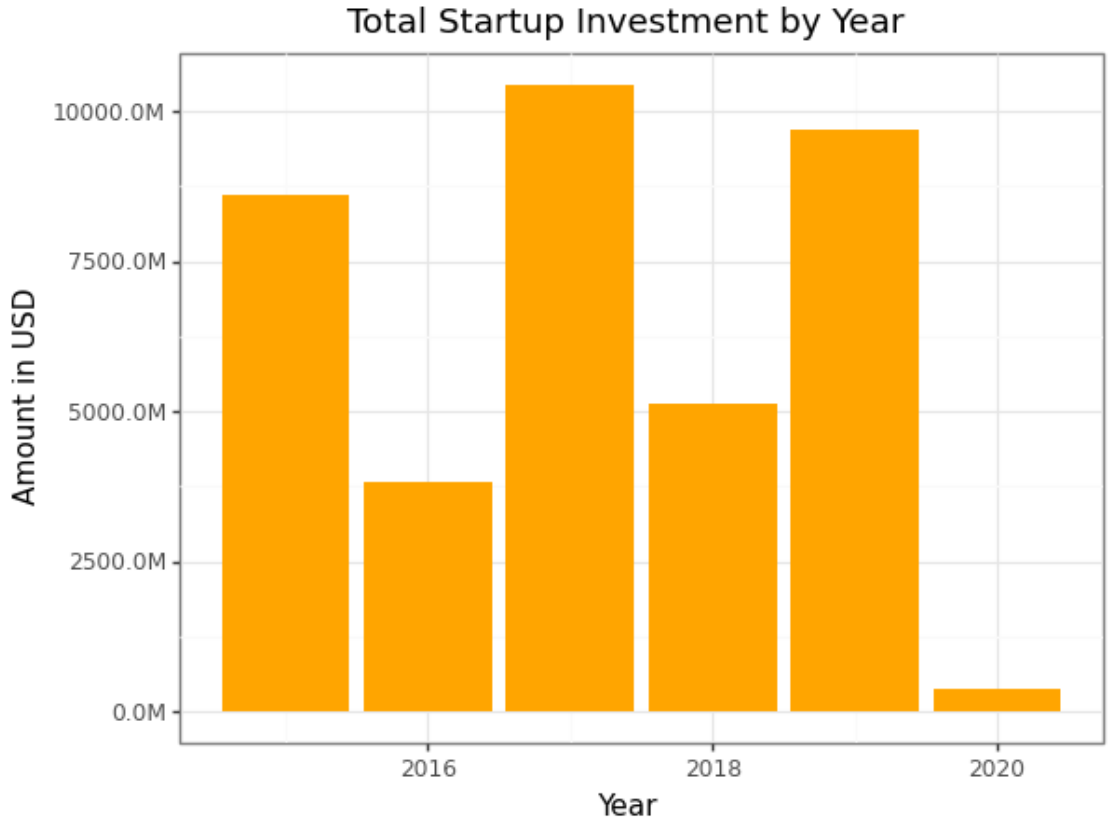
	Sr No	Date dd/mm/yyyy	Startup Name	Industry Vertical	SubVertical	City Location	Investors Name	InvestmentnType	Amount in USD	Remarks
0	1	09/01/2020	BYJU'S	E-Tech	E-learning	Bengaluru	Tiger Global Management	Private Equity Round	20,00,00,000	NaN
1	2	13/01/2020	Shuttl	Transportation	App based shuttle service	Gurgaon	Susquehanna Growth Equity	Series C	80,48,394	NaN
2	3	09/01/2020	Mamaearth	E-commerce	Retailer of baby and toddler products	Bengaluru	Sequoia Capital India	Series B	1,83,58,860	NaN
3	4	02/01/2020	https://www.wealthbucket.in/	FinTech	Online Investment	New Delhi	Vinod Khatumal	Pre-series A	30,00,000	NaN
4	5	02/01/2020	Fashor	Fashion and Apparel	Embroiled Clothes For Women	Mumbai	Sprout Venture Partners	Seed Round	18,00,000	NaN

In [3]:
```
# DATA CLEANING
startup_df['Date dd/mm/yyyy'] = pd.to_datetime(startup_df['Date dd/mm/yyyy'], format='%d/%m/%Y', errors='coerce')
startup_df['Date dd/mm/yyyy'] = startup_df['Date dd/mm/yyyy'].fillna(pd.to_datetime(startup_df['Date dd/mm/yyyy'], format='%d-%b
startup_df.dropna(subset=['Date dd/mm/yyyy'], inplace=True)

startup_df['Amount in USD'] = startup_df['Amount in USD'].str.replace(',', '')
startup_df['Amount in USD'] = pd.to_numeric(startup_df['Amount in USD'], errors='coerce')

startup_df.head(5)
```

Out[3]:

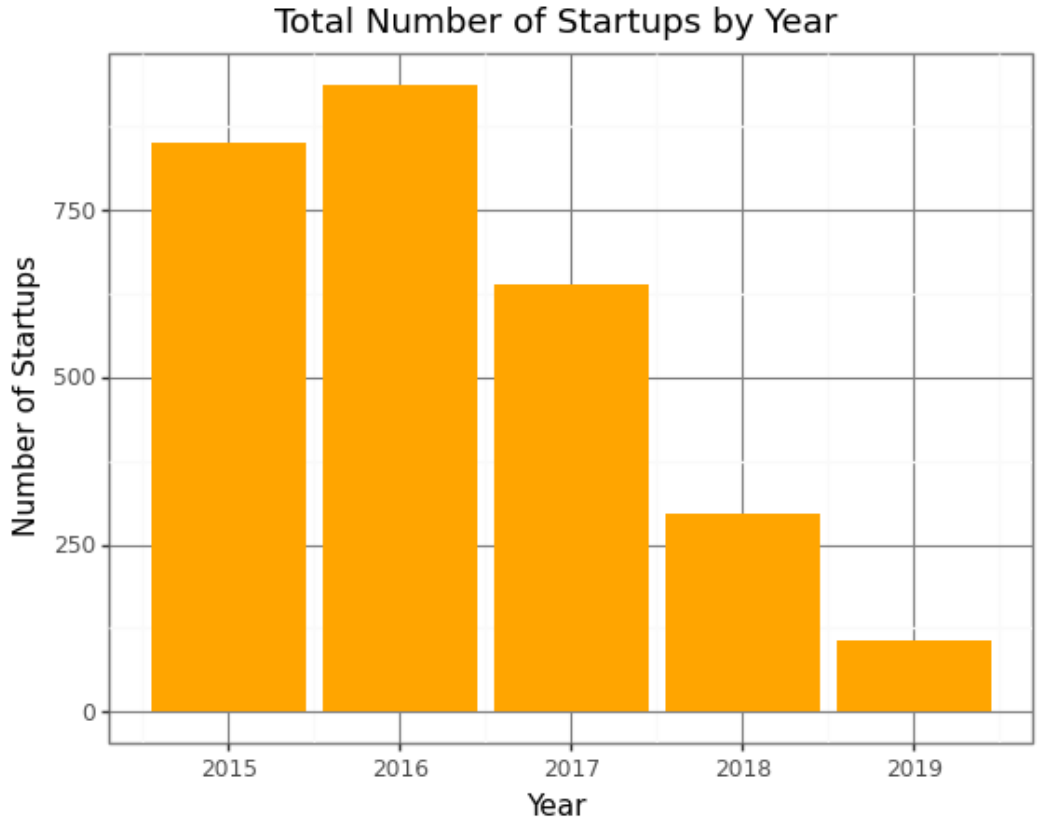
	Sr No	Date dd/mm/yyyy	Startup Name	Industry Vertical	SubVertical	City Location	Investors Name	InvestmentnType	Amount in USD	Remarks
0	1	2020-01-09	BYJU'S	E-Tech	E-learning	Bengaluru	Tiger Global Management	Private Equity Round	200000000.0	NaN
1	2	2020-01-13	Shuttl	Transportation	App based shuttle service	Gurgaon	Susquehanna Growth Equity	Series C	8048394.0	NaN
2	3	2020-01-09	Mamaearth	E-commerce	Retailer of baby and toddler products	Bengaluru	Sequoia Capital India	Series B	18358860.0	NaN
3	4	2020-01-02	https://www.wealthbucket.in/	FinTech	Online Investment	New Delhi	Vinod Khatumal	Pre-series A	3000000.0	NaN
4	5	2020-01-02	Fashor	Fashion and Apparel	Embroiled Clothes For Women	Mumbai	Sprout Venture Partners	Seed Round	1800000.0	NaN

In [4]:
```
# Extract the year from the date and create a new column
startup_df['Year'] = pd.DatetimeIndex(startup_df['Date dd/mm/yyyy']).year
startup_by_year = startup_df.query('Year != 2020').groupby('Year')['Startup Name'].nunique().reset_index()
investment_by_year = startup_df.groupby('Year')['Amount in USD'].sum().reset_index()

# Create a bar plot of total investment by year
(ggplot(investment_by_year, aes(x='Year', y='Amount in USD'))
 + geom_bar(stat='identity', fill='orange')
 + labs(title='Total Startup Investment by Year', x='Year', y='Amount in USD')
 + theme_bw()
 + scale_y_continuous(labels=lambda l: ['%.1fM' % (v / 1000000) for v in l])
)
```



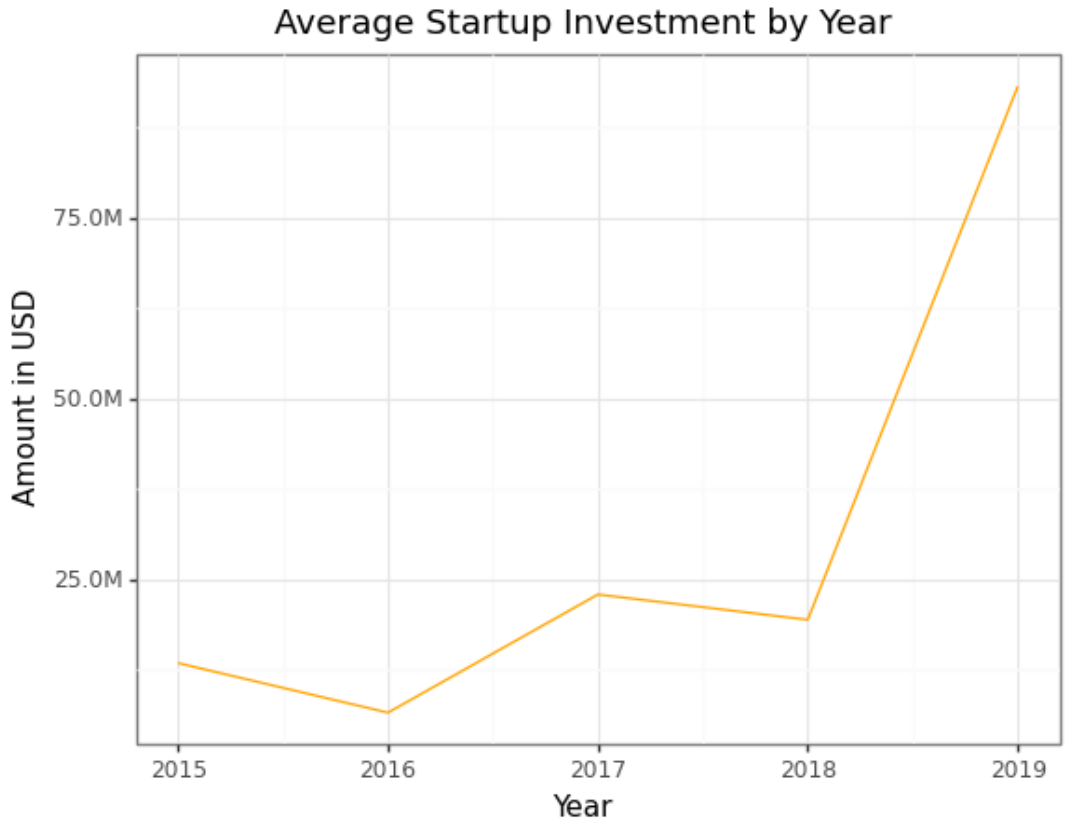
Out[4]: <ggplot: (159525118678)>

In [5]:
```
#Total Number of Startups by Year
(ggplot(startup_by_year, aes(x='Year', y='Startup Name')) +
geom_col(fill='orange') +
labs(title='Total Number of Startups by Year', x='Year', y='Number of Startups') +
theme_bw() +
theme(panel_grid_major=element_line(color='grey'))
)
```



Out[5]: <ggplot: (159525667063)>

In [6]:
```
# Group the data by year and calculate the average investment
investment_by_year = startup_df.query('Year != 2020').groupby('Year')['Amount in USD'].mean().reset_index()

(ggplot(investment_by_year, aes(x='Year', y='Amount in USD'))
 + geom_line(color='orange')
 + labs(title='Average Startup Investment by Year', x='Year', y='Amount in USD')
 + theme_bw()
 + scale_y_continuous(labels=lambda l: ['%.1fM' % (v / 1000000) for v in l])
)
```

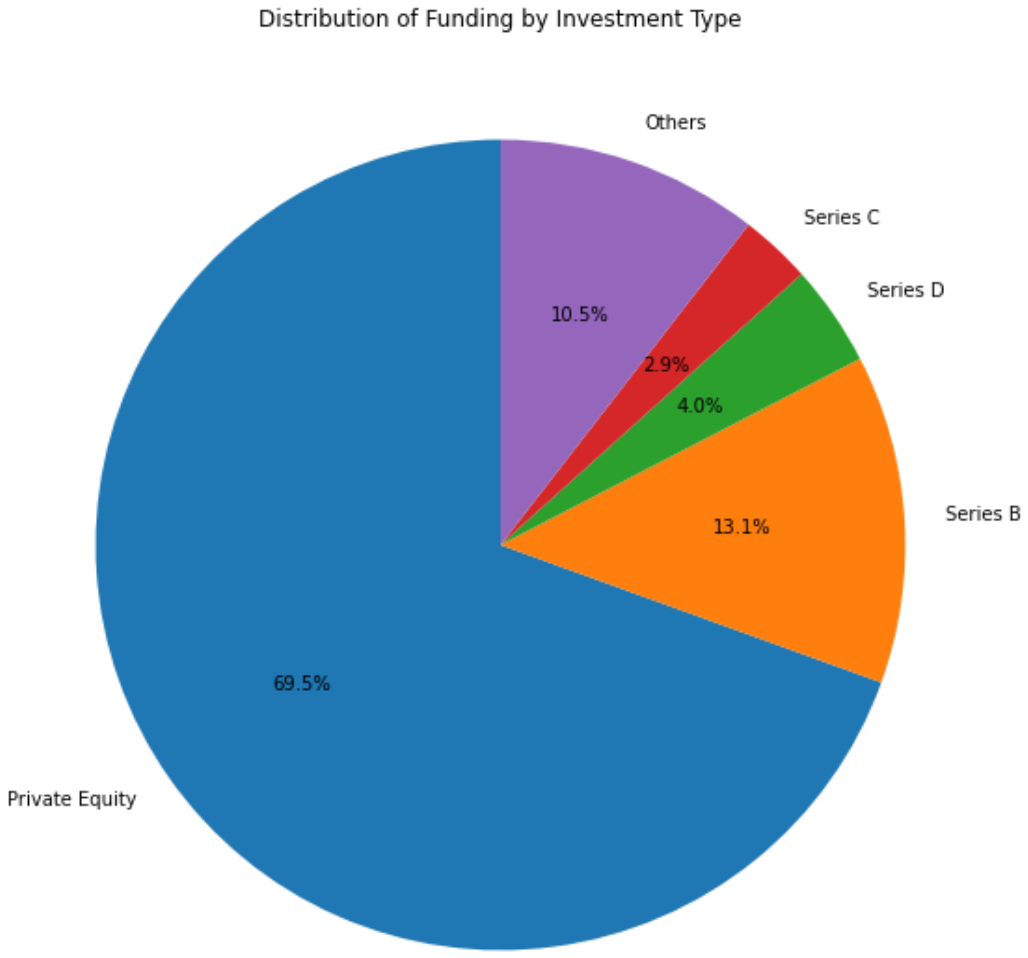


Out[6]: <ggplot: (159525711568)>

Obserbation: Based on the plot diagram, it appears that the average investment in startups has been steadily increasing over the years from 2015 to 2019. There is a slight dip in 2016(due to recession) and 2018, but overall the trend is upwards. It's also worth noting that the amount of

investment seems to be increasing at a faster rate in later years, with a sharper incline towards the end of the time period in 2018 and 2019. The

In [7]:
```
# DATA CLEANING
startup_df = startup_df.dropna(subset=['Industry Vertical'])
startup_df['Industry Vertical'] = startup_df['Industry Vertical'].replace('E-Commerce', 'eCommerce')
startup_df['Industry Vertical'] = startup_df['Industry Vertical'].replace('ECommerce', 'eCommerce')
startup_df['Industry Vertical'] = startup_df['Industry Vertical'].replace('Online Marketplace', 'eCommerce')
startup_df['Industry Vertical'] = startup_df['Industry Vertical'].replace('Ecommerce Marketplace', 'eCommerce')

# Calculate total funding for each industry vertical
funding_by_industry = startup_df.groupby('InvestmentnType')['Amount in USD'].sum().reset_index()

# Sort industries
funding_by_industry = funding_by_industry.sort_values('Amount in USD', ascending=False)

# Select the top 5 industries by funding and group the rest as "others"
top_industries = funding_by_industry.head(4)
other_industry = pd.DataFrame({
    'InvestmentnType': ['Others'],
    'Amount in USD': [funding_by_industry['Amount in USD'][4:].sum()]
})

funding_by_industry = pd.concat([top_industries, other_industry])

plt.figure(figsize=(10, 10))
plt.pie(funding_by_industry['Amount in USD'], labels=funding_by_industry['InvestmentnType'], autopct='%1.1f%%', startangle=90)
plt.title('Distribution of Funding by Investment Type')
plt.show()
```



Obserbation:

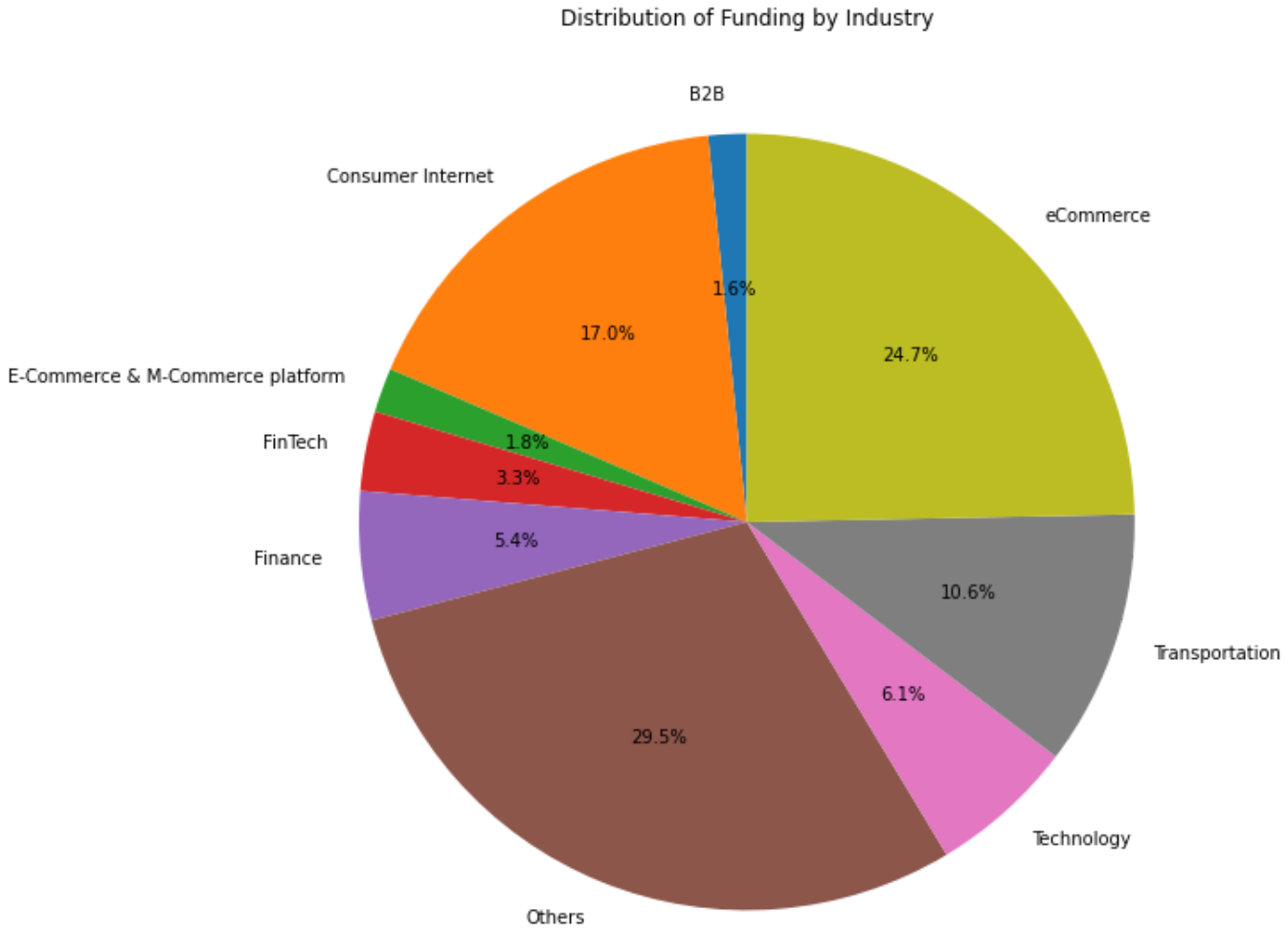
Based on the pie chart, it appears that the majority of funding for startups comes from two types of investments: Private Equity and Series B. These two types of funding combined account for over 80% of the total funding. The next two types of funding, Series c and Series D, each account for around 3-4% of the total funding.

The "Others" category, which includes all other types of investments not among the top 4, accounts for less than 12% of the total funding. This suggests that a small number of investment types dominate the funding landscape for startups, with most of the money coming from Private

Equity

In [8]:

```
funding_by_industry = startup_df.groupby('Industry Vertical')['Amount in USD'].sum().reset_index()
funding_by_industry = funding_by_industry.sort_values('Amount in USD', ascending=False)

# Create a new column to group industries as "Others"
funding_by_industry['Industry Group'] = funding_by_industry['Industry Vertical']
top_industries = funding_by_industry['Industry Vertical'][:8]
funding_by_industry.loc[~funding_by_industry['Industry Vertical'].isin(top_industries), 'Industry Group'] = 'Others'
funding_by_group = funding_by_industry.groupby('Industry Group')['Amount in USD'].sum().reset_index()

#Distribution of Funding by Industry
plt.figure(figsize=(10, 10))
plt.pie(funding_by_group['Amount in USD'], labels=funding_by_group['Industry Group'], autopct='%1.1f%%', startangle=90)
plt.title('Distribution of Funding by Industry')
plt.show()
```



Obserbation:

Based on the pie chart, it appears that eCommerce is the dominant industry for startup funding, accounting for almost 25% of the total funding. Consumer Internet and Transportation follow closely behind, each accounting for around 17% and 10.6% of the total funding respectively.

The other top industries by funding, in descending order, are Technology, Finance and FinTech. These industries each account for less than 10% of the total funding.

The "Others" category, which includes all other industries not among the top 8, accounts for just over 10% of the total funding. This suggests that a small number of industries dominate the funding landscape for startups, with most of the money going to eCommerce, Consumer Internet, Transport and Technology.

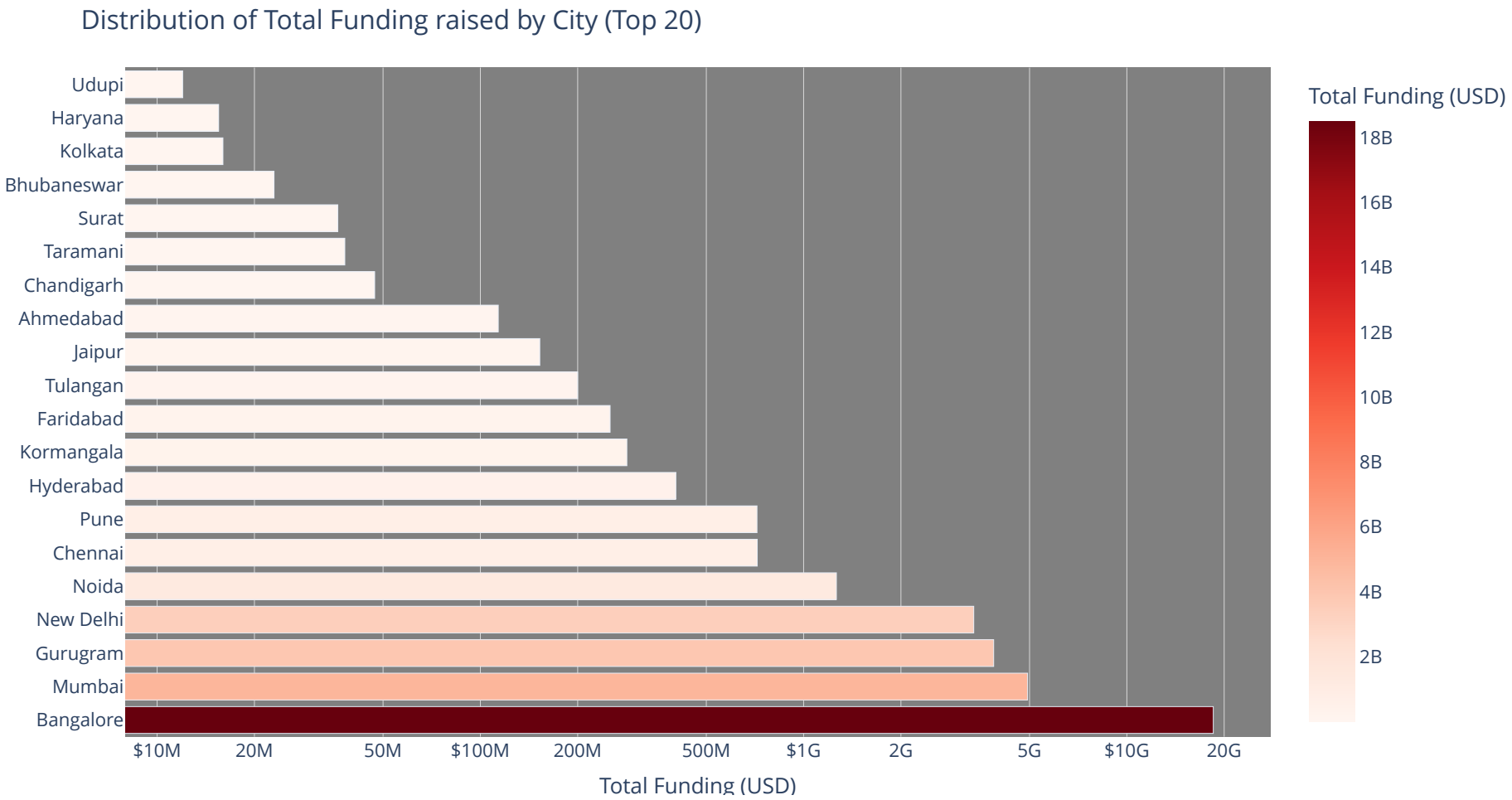
In [9]:

```
# DATA CLEANING of City Location
startup_df = startup_df.dropna(subset=['City  Location'])
startup_df['City  Location'] = startup_df['City  Location'].replace('Ahemadabad', 'Ahemdabad')
startup_df['City  Location'] = startup_df['City  Location'].replace('Ahemadabad', 'Ahemdabad')
startup_df['City  Location'] = startup_df['City  Location'].replace('Bangalore / Palo Alto', 'Bangalore')
startup_df['City  Location'] = startup_df['City  Location'].replace('New York, Bengaluru', 'Bangalore')
startup_df['City  Location'] = startup_df['City  Location'].replace('Bangalore / SFO', 'Bangalore')
startup_df['City  Location'] = startup_df['City  Location'].replace('SFO / Bangalore', 'Bangalore')
startup_df['City  Location'] = startup_df['City  Location'].replace('Bangalore / San Mateo', 'Bangalore')
startup_df['City  Location'] = startup_df['City  Location'].replace('Bangalore/ Bangkok', 'Bangalore')
startup_df['City  Location'] = startup_df['City  Location'].replace('Bangalore / USA', 'Bangalore')
startup_df['City  Location'] = startup_df['City  Location'].replace('Bhubneswar', 'Bhubaneswar')
startup_df['City  Location'] = startup_df['City  Location'].replace('Bengaluru', 'Bangalore')
startup_df['City  Location'] = startup_df['City  Location'].replace('Bengaluru and Gurugram', 'Bangalore')
startup_df['City  Location'] = startup_df['City  Location'].replace('Pune / US', 'Pune')
startup_df['City  Location'] = startup_df['City  Location'].replace('Nw Delhi', 'New Delhi')
startup_df['City  Location'] = startup_df['City  Location'].replace('Delhi', 'New Delhi')
startup_df['City  Location'] = startup_df['City  Location'].replace('New Delhi/ Houston', 'New Delhi')
startup_df['City  Location'] = startup_df['City  Location'].replace('New Delhi / California', 'New Delhi')
startup_df['City  Location'] = startup_df['City  Location'].replace('Gurgaon', 'Gurugram')
startup_df['City  Location'] = startup_df['City  Location'].replace('New Delhi / US', 'New Delhi')
startup_df['City  Location'] = startup_df['City  Location'].replace('Gurgaon / SFO', 'Gurugram')
startup_df['City  Location'] = startup_df['City  Location'].replace('Mumbai / Global', 'Mumbai')
startup_df['City  Location'] = startup_df['City  Location'].replace('Mumbai/Bengaluru', 'Mumbai')
startup_df['City  Location'] = startup_df['City  Location'].replace('Mumbai / UK', 'Mumbai')
startup_df['City  Location'] = startup_df['City  Location'].replace('Mumbai / NY', 'Mumbai')
startup_df['City  Location'] = startup_df['City  Location'].replace('Pune/Seattle', 'Pune')
startup_df['City  Location'] = startup_df['City  Location'].replace('Pune / Dubai', 'Pune')
startup_df['City  Location'] = startup_df['City  Location'].replace('Hyderabad/USA', 'Hyderabad')
startup_df['City  Location'] = startup_df['City  Location'].replace('Goa/Hyderabad', 'Hyderabad')
startup_df['City  Location'] = startup_df['City  Location'].replace('Dallas / Hyderabad', 'Hyderabad')
startup_df['City  Location'] = startup_df['City  Location'].replace('Chennai/ Singapore', 'Chennai')
startup_df['City  Location'] = startup_df['City  Location'].replace('Noida / Singapore', 'Noida')
startup_df['City  Location'] = startup_df['City  Location'].replace('Kerala', 'Trivandrum')
startup_df['City  Location'] = startup_df['City  Location'].replace('Delhi & Cambridge', 'New Delhi')
startup_df['City  Location'] = startup_df['City  Location'].replace('\\xc2\\xa0Bangalore', 'Bangalore')

startup_df = startup_df[~startup_df['City  Location'].isin(['Boston','London','Singapore','Burnsville','San Jose,', 'Menlo Park'
startup_df.head(5)
```

Out[9]:

	Sr No	Date dd/mm/yyyy	Startup Name	Industry Vertical	SubVertical	City Location	Investors Name	InvestmentnType	Amount in USD	Remarks	Year
0	1	2020-01-09	BYJU'S	E-Tech	E-learning	Bangalore	Tiger Global Management	Private Equity Round	200000000.0	NaN	2020
1	2	2020-01-13	Shuttl	Transportation	App based shuttle service	Gurugram	Susquehanna Growth Equity	Series C	8048394.0	NaN	2020
2	3	2020-01-09	Mamaearth	E-commerce	Retailer of baby and toddler products	Bangalore	Sequoia Capital India	Series B	18358860.0	NaN	2020
3	4	2020-01-02	https://www.wealthbucket.in/	FinTech	Online Investment	New Delhi	Vinod Khatumal	Pre-series A	3000000.0	NaN	2020
4	5	2020-01-02	Fashor	Fashion and Apparel	Embroiled Clothes For Women	Mumbai	Sprout Venture Partners	Seed Round	1800000.0	NaN	2020

In [10]:
```
# Calculate total funding for each city
funding_by_city = startup_df.groupby('City  Location')['Amount in USD'].sum().reset_index()
total_funding = funding_by_city['Amount in USD'].sum()
funding_by_city['Funding %'] = funding_by_city['Amount in USD'] / total_funding * 100

# Sort cities by total funding in descending order and select the top 20
funding_by_city = funding_by_city.sort_values('Amount in USD', ascending=False).head(20)

# BAR PLOT
fig = px.bar(funding_by_city, x='Amount in USD', y='City  Location', orientation='h',
             color='Amount in USD', color_continuous_scale='Reds',
             labels={'Amount in USD': 'Total Funding (USD)', 'City  Location': 'City'})

fig.update_layout(title='Distribution of Total Funding raised by City (Top 20)',
                  xaxis_type='log', xaxis_tickprefix='$', xaxis_tickformat='s',
                  yaxis_tickfont=dict(size=12), yaxis_title=None,
                  plot_bgcolor='grey', margin=dict(l=0, r=0, t=50, b=0))

fig.show()
```



Observation:

The chart also shows the distribution of total funding for the top 20 cities in India. The x-axis is in a logarithmic scale to accommodate the wide range of funding amounts. The color of each bar represents the total funding amount for each city, with darker shades indicating higher funding amounts.

Based on the horizontal bar chart, we can observe that Bangalore is the city with the highest total funding raised by startups, with over 18 billion USD in total funding. Mumbai follows behind with just under 5 billion USD in total funding.

Overall, the chart suggests that there are significant regional disparities in the funding received by startups in India, with a few cities dominating

Plots using data: Credit_Card_uses.csv

In [11]:
```
# Load the CSV file Credit_Card_uses.csv
credit_df = pd.read_csv('Credit_Card_uses.csv')
credit_df.head(5)
```

Out[11]:

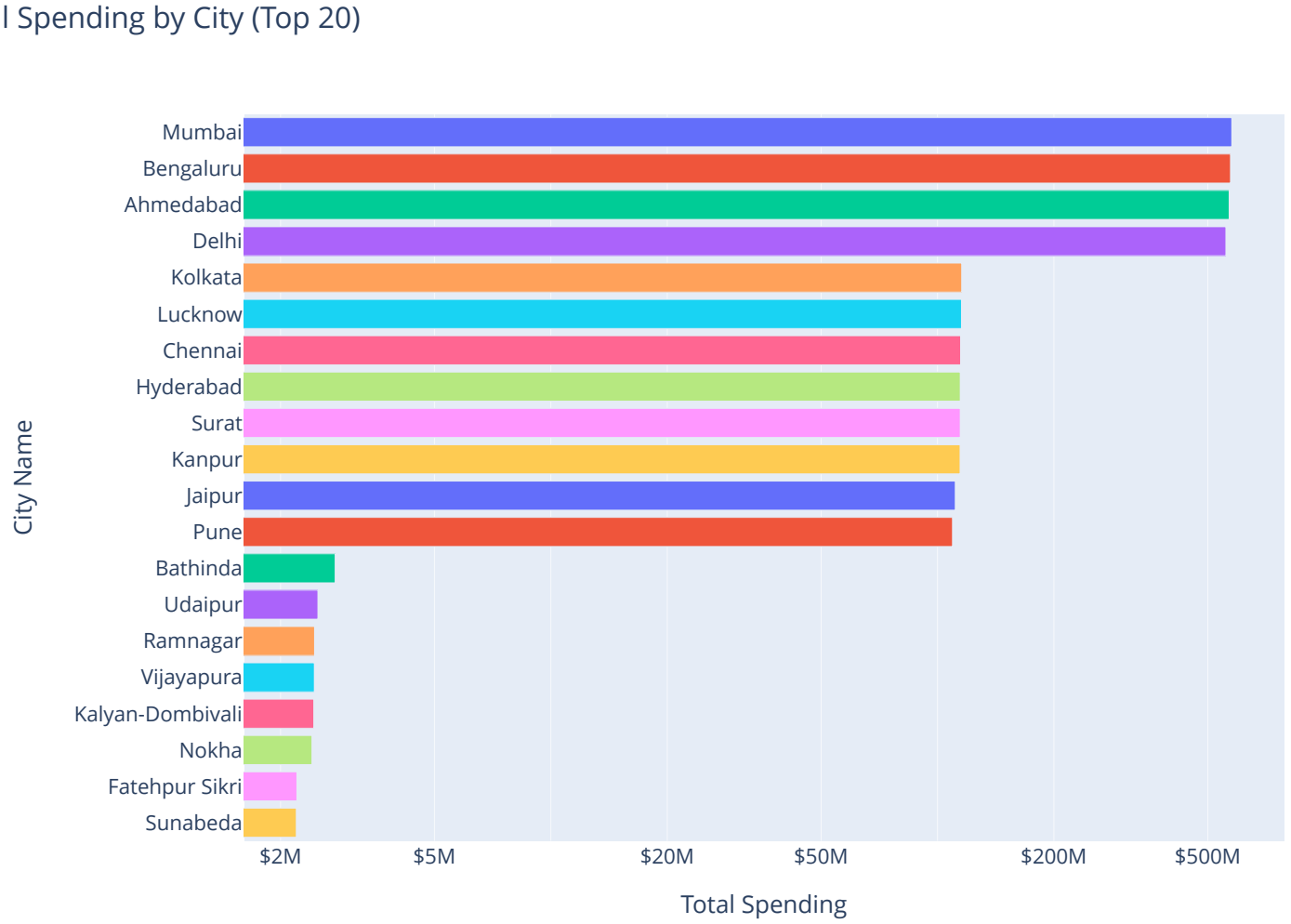
	index	City	Date	Card Type	Exp Type	Gender	Amount
0	0	Delhi, India	29-Oct-14	Gold	Bills	F	82475
1	1	Greater Mumbai, India	22-Aug-14	Platinum	Bills	F	32555
2	2	Bengaluru, India	27-Aug-14	Silver	Bills	F	101738
3	3	Greater Mumbai, India	12-Apr-14	Signature	Bills	F	123424
4	4	Bengaluru, India	05-May-15	Gold	Bills	F	171574

In [12]:
```
# DATA CLEANING
credit_df['City'] = credit_df['City'].str.split(',').str[0]
credit_df['City'] = credit_df['City'].replace('Greater Mumbai', 'Mumbai')
credit_df.head(5)
```

Out[12]:

	index	City	Date	Card Type	Exp Type	Gender	Amount
0	0	Delhi	29-Oct-14	Gold	Bills	F	82475
1	1	Mumbai	22-Aug-14	Platinum	Bills	F	32555
2	2	Bengaluru	27-Aug-14	Silver	Bills	F	101738
3	3	Mumbai	12-Apr-14	Signature	Bills	F	123424
4	4	Bengaluru	05-May-15	Gold	Bills	F	171574

In [13]:
```
total_spending = credit_df.groupby('City')['Amount'].sum().reset_index()
total_spending = total_spending.sort_values('Amount', ascending=False).head(20)

fig = px.bar(total_spending, x='Amount', y='City', orientation='h', color='City',
             title='Total Spending by City (Top 20)', labels={'Amount': 'Total Spending', 'City': 'City Name'})

fig.update_layout(xaxis_type='log', xaxis_tickprefix='$', xaxis_tickformat='$.1s',
                  yaxis_title='City Name', legend_title='City Name', showlegend=False, width=900, height=600)

fig.show()
```



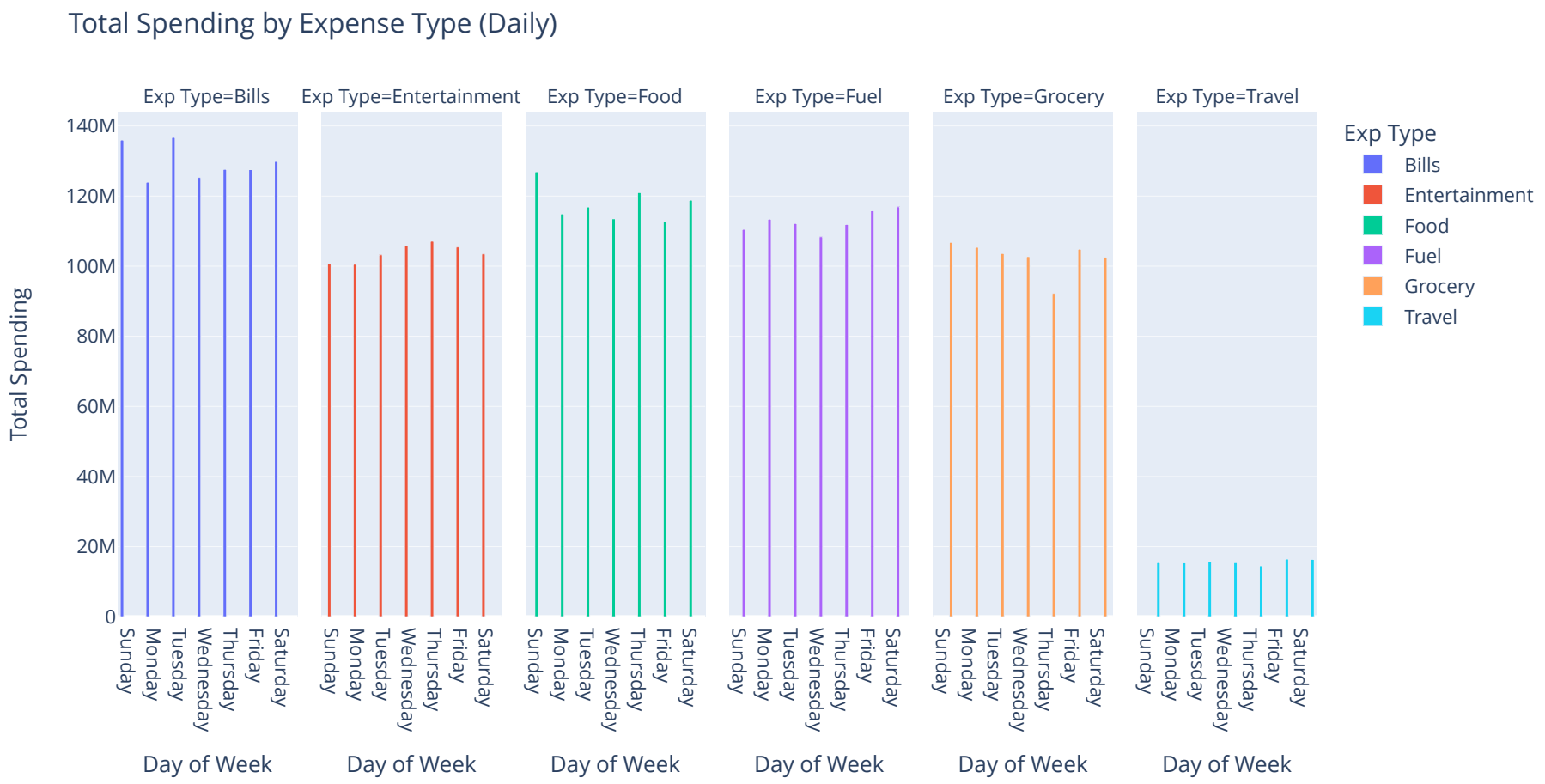
Observation:

The plot shows the top 20 cities with the highest total spending. The y-axis represents the city names while the x-axis represents the total amount spent in USD. Each bar is colored differently based on the city it represents.

From the plot, it can be observed that Mumbai has the highest total spending followed by Bengaluru, Ahmedabad and Delhi. It's also interesting to note that the spending drops off significantly after the top 4 cities.

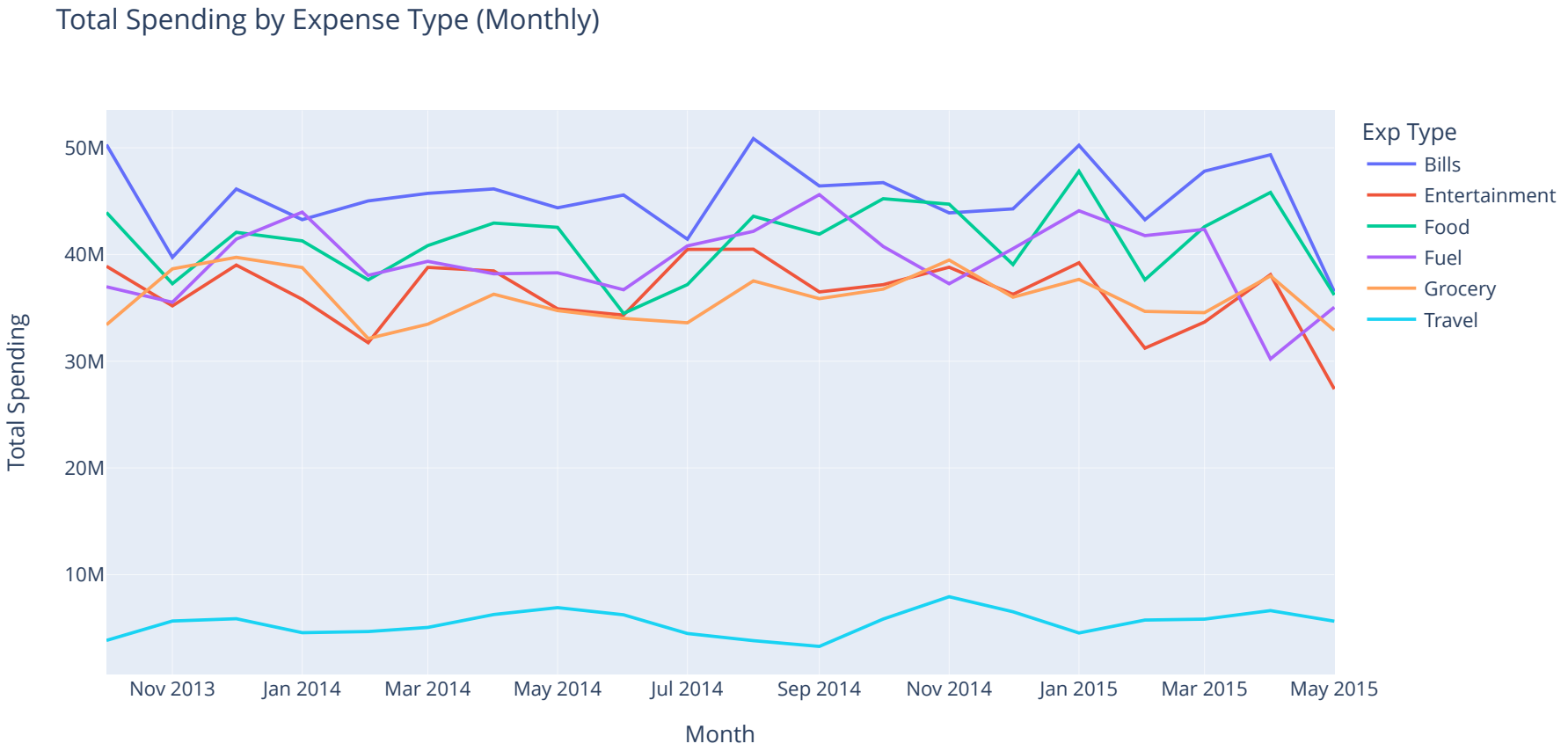
Overall, the plot provides a clear and positive corelation of the total spending and toatal investment in startups by city.

In [14]:
```
# Convert the "Date" column to a datetime object and extract the day of the week
credit_df['Day of Week'] = pd.to_datetime(credit_df['Date']).dt.day_name()

# Define the week days in the correct order
weekdays = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']

# Convert the day of the week column to a categorical variable with the week days in the correct order
credit_df['Day of Week'] = pd.Categorical(credit_df['Day of Week'], categories=weekdays, ordered=True)

# Calculate the total spending by expense type and day of the week
daily_spending = credit_df.groupby(['Day of Week', 'Exp Type'], as_index=False)['Amount'].sum()

# Create a grouped bar chart using Plotly
fig = px.bar(daily_spending, x='Day of Week', y='Amount', color='Exp Type', barmode='group',
             facet_col='Exp Type', title='Total Spending by Expense Type (Daily)',
             labels={'Amount': 'Total Spending', 'Day of Week': 'Day of Week'})

fig.show()
```

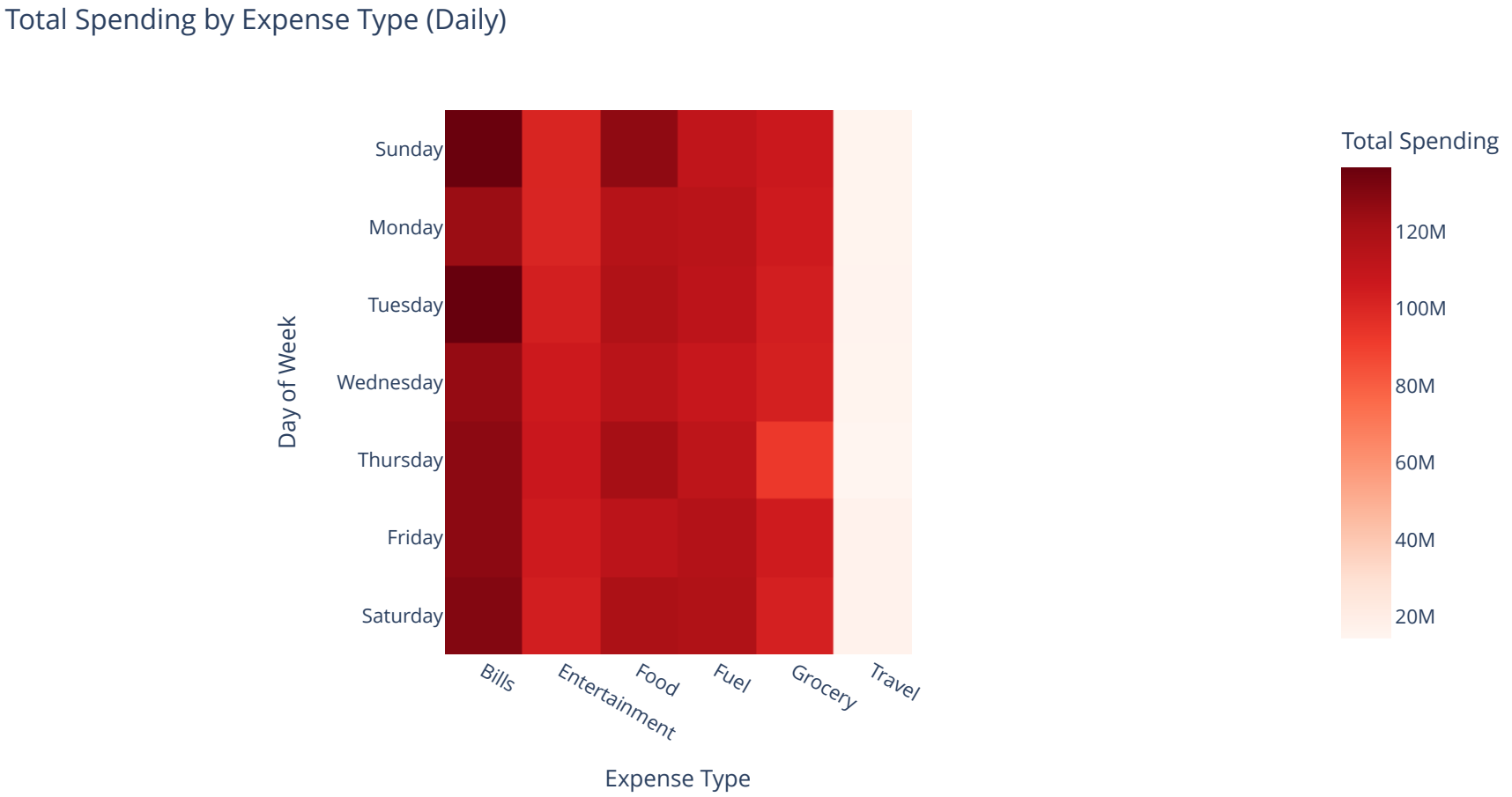

Total Spending by Expense Type (Daily)
Exp Type=Bills
Exp Type=Entertainment
Exp Type=Food
Exp Type=Fuel
Exp Type=Grocery
Exp Type=Travel
Total Spending
140M
120M
100M
80M
60M
40M
20M
0
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of Week
Exp Type
Bills
Entertainment
Food
Fuel
Grocery
Travel

In [15]:

```
# Convert the "Date" column to a datetime object and extract the month
credit_df['Month'] = pd.to_datetime(credit_df['Date']).dt.strftime('%Y-%m')

# Calculate the total spending by expense type and month
monthly_spending = credit_df.groupby(['Month', 'Exp Type'], as_index=False)['Amount'].sum()

# Create a line chart using Plotly
fig = px.line(monthly_spending, x='Month', y='Amount', color='Exp Type',
              title='Total Spending by Expense Type (Monthly)', labels={'Amount': 'Total Spending', 'Month': 'Month'})

fig.show()
```


Total Spending by Expense Type (Monthly)
Total Spending
50M
40M
30M
20M
10M
Nov 2013
Jan 2014
Mar 2014
May 2014
Jul 2014
Sep 2014
Nov 2014
Jan 2015
Mar 2015
May 2015
Month
Exp Type
Bills
Entertainment
Food
Fuel
Grocery
Travel

In [16]:
```
# Define the week days in the correct order
weekdays = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']

# Convert the day of the week column to a categorical variable with the week days in the correct order
credit_df['Day of Week'] = pd.Categorical(credit_df['Day of Week'], categories=weekdays, ordered=True)

# Calculate the total spending by expense type and day of the week
daily_spending = credit_df.groupby(['Day of Week', 'Exp Type'], as_index=False)['Amount'].sum()

# Pivot the data to create a matrix of daily spending by expense type
daily_spending_matrix = daily_spending.pivot(index='Day of Week', columns='Exp Type', values='Amount')

# Create a heatmap using Plotly
fig = px.imshow(daily_spending_matrix, x=daily_spending_matrix.columns, y=daily_spending_matrix.index,
                color_continuous_scale='Reds', title='Total Spending by Expense Type (Daily)',
                labels={'x': 'Expense Type', 'y': 'Day of Week', 'color': 'Total Spending'})

fig.show()
```



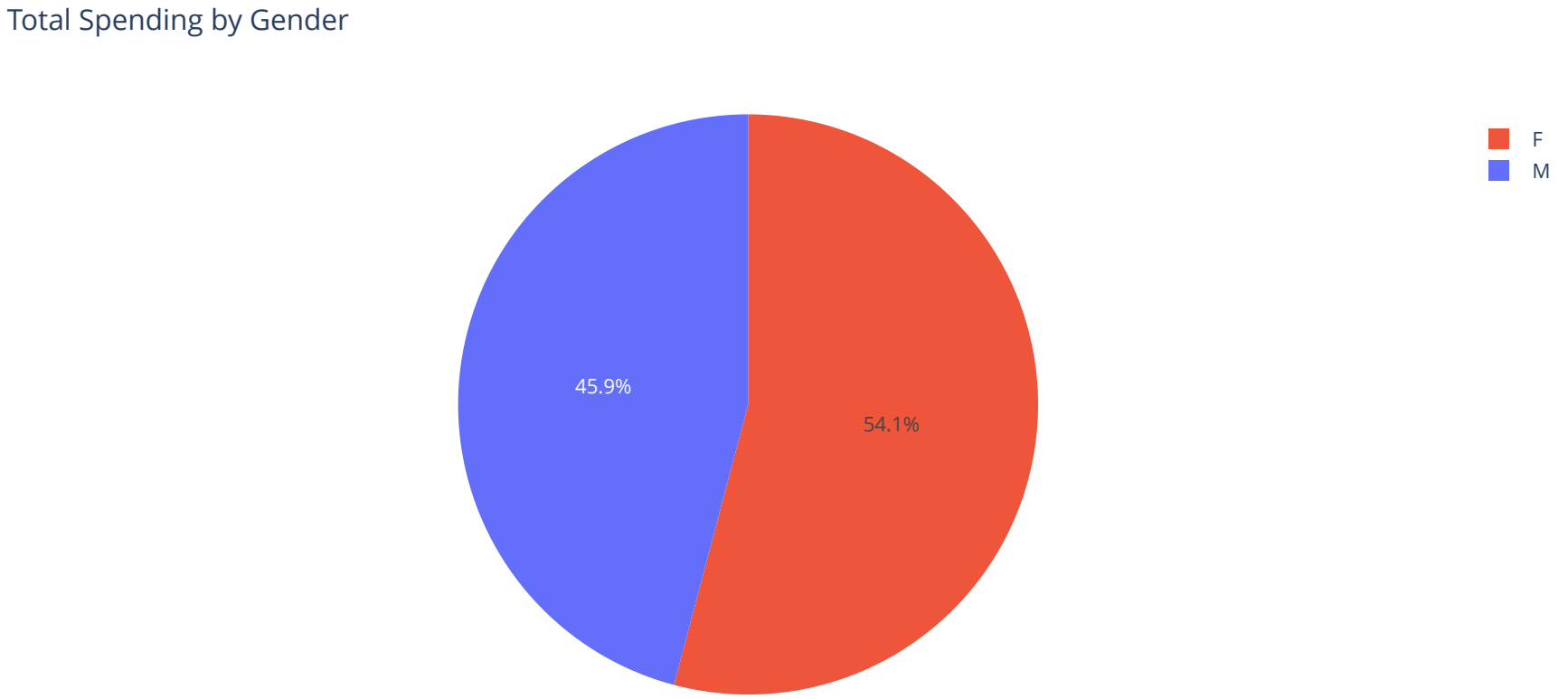
Observation: The heatmap shows the total spending by expense type for each day of the week. The expense types are listed on the x-axis and the days of the week are listed on the y-axis. The darker red colors indicate higher spending while the lighter colors indicate lower spending.

1.The highest spending is on Sunday and Tuesday, with the highest expenses being in Bills.

2.The lowest spending is in Travel and Entertainment categories.

3.Travel and Entertainment categories have relatively consistent spending across all days of the week.

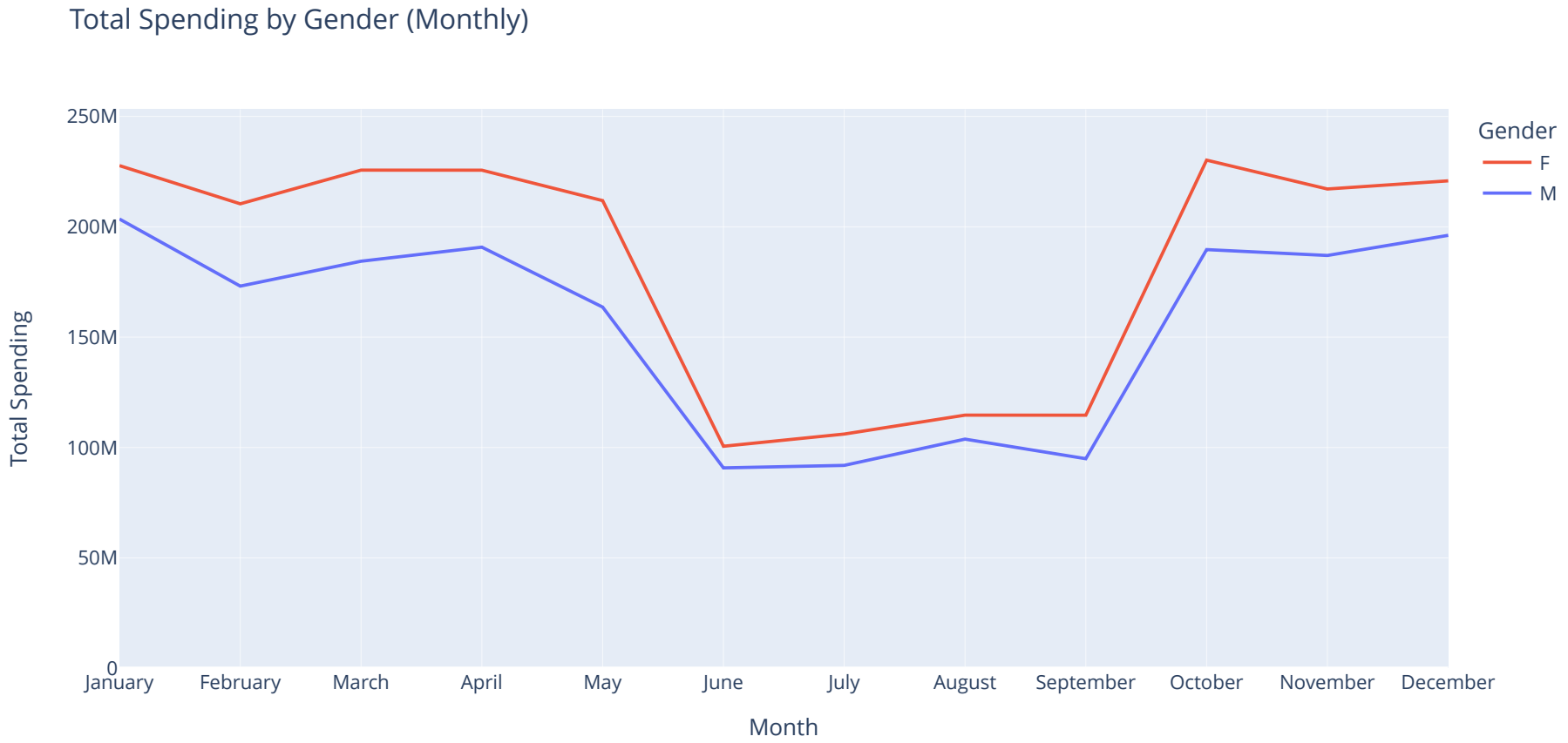
In [17]:
```
# Calculate the total spending by gender
gender_spending = credit_df.groupby('Gender', as_index=False)['Amount'].sum()

color_map = {'M': px.colors.qualitative.Plotly[0], 'F': px.colors.qualitative.Plotly[1]}

# Create a pie chart using Plotly
fig = px.pie(gender_spending, values='Amount', names='Gender',color='Gender',color_discrete_map=color_map,  title='Total Spendin

fig.show()
```



Observation:

The pie chart shows the distribution of total spending by gender in the dataset. The chart shows that males (M) have a lower total spending on credit card compared to females (F), with M accounting for approximately 45.9% of the total spending, while F accounting for approximately 54.1%. The color coding used in the chart helps to distinguish between the two genders easily.

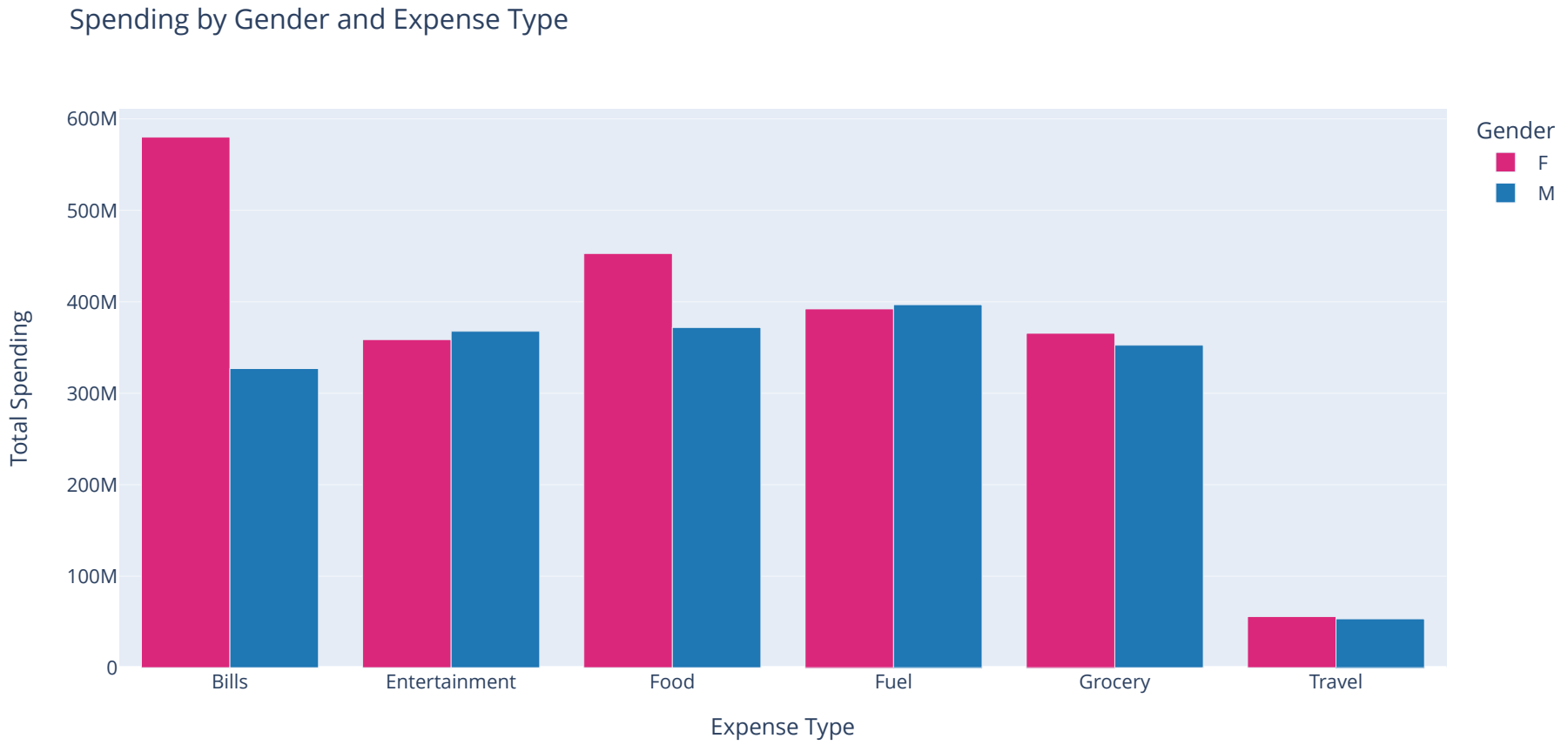
In [18]:
```
# Convert the "Date" column to a datetime object and extract the month
credit_df['Month'] = pd.to_datetime(credit_df['Date']).dt.month_name()

# Set the "Month" column to categorical data type with correct order of categories
month_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'De
credit_df['Month'] = pd.Categorical(credit_df['Month'], categories=month_order, ordered=True)

# Calculate the total spending by gender and month
monthly_spending = credit_df.groupby(['Month', 'Gender'], as_index=False)['Amount'].sum()

# Create a line chart using Plotly
color_map = {'M': px.colors.qualitative.Plotly[0], 'F': px.colors.qualitative.Plotly[1]}
fig = px.line(monthly_spending, x='Month', y='Amount', color='Gender',
              color_discrete_map=color_map, title='Total Spending by Gender (Monthly)',
              labels={'Amount': 'Total Spending'})

# Set the range of y-axis to start from zero
fig.update_layout(yaxis_range=[0, monthly_spending['Amount'].max() * 1.1])
fig.show()
```



Observation:

The line chart shows the total spending by gender on a monthly basis. The spending for both male and female is at its highest in festive seasons. The spending for males is slightly lower than females for all of the months.

In [19]:
```
# Calculate the total spending by gender and expense type
gender_spending = credit_df.groupby(['Gender', 'Exp Type'], as_index=False)['Amount'].sum()

female_spending = gender_spending[gender_spending['Gender'] == 'F']
male_spending = gender_spending[gender_spending['Gender'] == 'M']

colors = ['#db277b', '#1f77b4']
fig = px.bar(gender_spending, x='Exp Type', y='Amount', color='Gender', barmode='group',
             title='Spending by Gender and Expense Type', labels={'Exp Type': 'Expense Type', 'Amount': 'Total Spending'},
             color_discrete_sequence=colors)

fig.show()
```



Based on the given bar chart, it can be observed that females spent more than males in all expense types except for "Fuel" and "Entertainment". The highest difference in spending between males and females can be observed in the "Bills" and "Food" expense types. In terms of overall spending, the lowest spending for both males and females was in the "Travel".

Startups.csv

In [20]:
```
# Read the CSV file into a DataFrame
df = pd.read_csv('Startups.csv')
df.head(4)
```

Out[20]:

	Unnamed: 0	Company	City	Starting Year	Founders	Industries	Description	No. of Employees	Funding Amount in $	Funding Round	No. of Investors
0	0	Urban Company	Gurgaon	2014	Abhiraj Singh Bhal, Raghav Chandra, Varun Khaitan	Apps, Home Services, Marketplace, Service Indu...	Urban is a marketplace for independent contrac...	1001-5000	445920356	12	16
1	1	Classplus	Noida	2018	Bhaswat Agarwal, Bikash Dash, Mukul Rustagi, N...	B2B, E-Learning, EdTech, Education, Mobile App...	Classplus is a mobile-first SaaS platform that...	101-250	89506451	10	20
2	2	Paytm	Noida	2010	Akshay Khanna, Vijay Shekhar Sharma	E-Commerce, Finance, Financial Services, Inter...	Paytm is a payment gateway that allows users a...	501-1000	32448851	4	4
3	3	Apna	Mumbai	2019	Nirmit Parikh	Employment, Human Resources, Recruiting, Staff...	Apna is a professional networking and job-sear...	101-250	93450000	4	6

In [21]:
```
# DATA CLEANING
def clean_employees(x):
    if type(x) == str and '-' in x:
        # If the value is in the form "1001-5000"
        nums = x.split('-')
        mean = round((int(nums[0]) + int(nums[1])) / 2)
        return mean
    elif type(x) == str and x.endswith('+'):
        # If the value is in the form "10001+"
        num = int(x[:-1])
        return num
    elif type(x) == str and not x.isnumeric():
        # If the value is not numeric
        return np.nan
    else:
        # If the value is already numeric
        return x

df['No. of Employees'] = df['No. of Employees'].replace('Nov-50', 'Nan')
df['No. of Employees'] = df['No. of Employees'].replace('01-Oct', 'Nan')
df['No. of Employees'] = df['No. of Employees'].apply(clean_employees)

# Print the updated DataFrame
df.head(4)
```

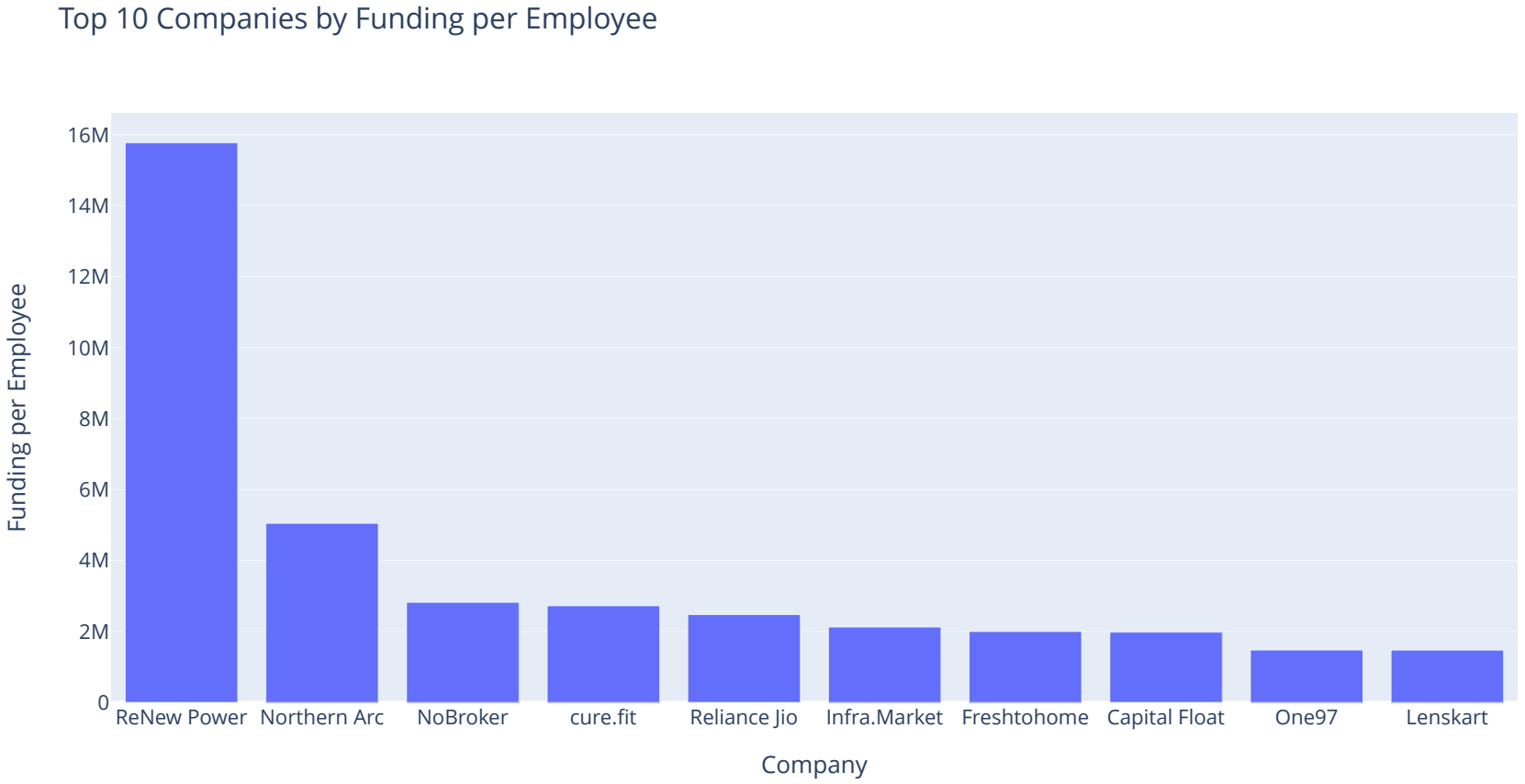
Out[21]:

	Unnamed: 0	Company	City	Starting Year	Founders	Industries	Description	No. of Employees	Funding Amount in $	Funding Round	No. of Investors
0	0	Urban Company	Gurgaon	2014	Abhiraj Singh Bhal, Raghav Chandra, Varun Khaitan	Apps, Home Services, Marketplace, Service Indu...	Urban is a marketplace for independent contrac...	3000.0	445920356	12	16
1	1	Classplus	Noida	2018	Bhaswat Agarwal, Bikash Dash, Mukul Rustagi, N...	B2B, E-Learning, EdTech, Education, Mobile App...	Classplus is a mobile-first SaaS platform that...	176.0	89506451	10	20
2	2	Paytm	Noida	2010	Akshay Khanna, Vijay Shekhar Sharma	E-Commerce, Finance, Financial Services, Inter...	Paytm is a payment gateway that allows users a...	750.0	32448851	4	4
3	3	Apna	Mumbai	2019	Nirmit Parikh	Employment, Human Resources, Recruiting, Staff...	Apna is a professional networking and job-sear...	176.0	93450000	4	6

The "clean_employees" function is defined to handle different formats of employee count values. If the value is in the form of "1001-5000", the function calculates the mean value. If the value is in the form of "10001+", the function returns the same number. If the value is not numeric or not in any of the given formats, the function returns NaN.

In [22]:

```
# Create a new column "Funding per Employee"
df['Funding per Employee'] = df['Funding Amount in $'] / df['No. of Employees']

# Sort the DataFrame by "Funding per Employee" in descending order
df_sorted = df.sort_values(by='Funding per Employee', ascending=False)

# Select the top 10 companies with highest "Funding per Employee"
df_top = df_sorted.head(10)

# Create a bar chart using Plotly
fig = px.bar(df_top, x='Company', y='Funding per Employee', title='Top 10 Companies by Funding per Employee')

# Show the chart
fig.show()
```



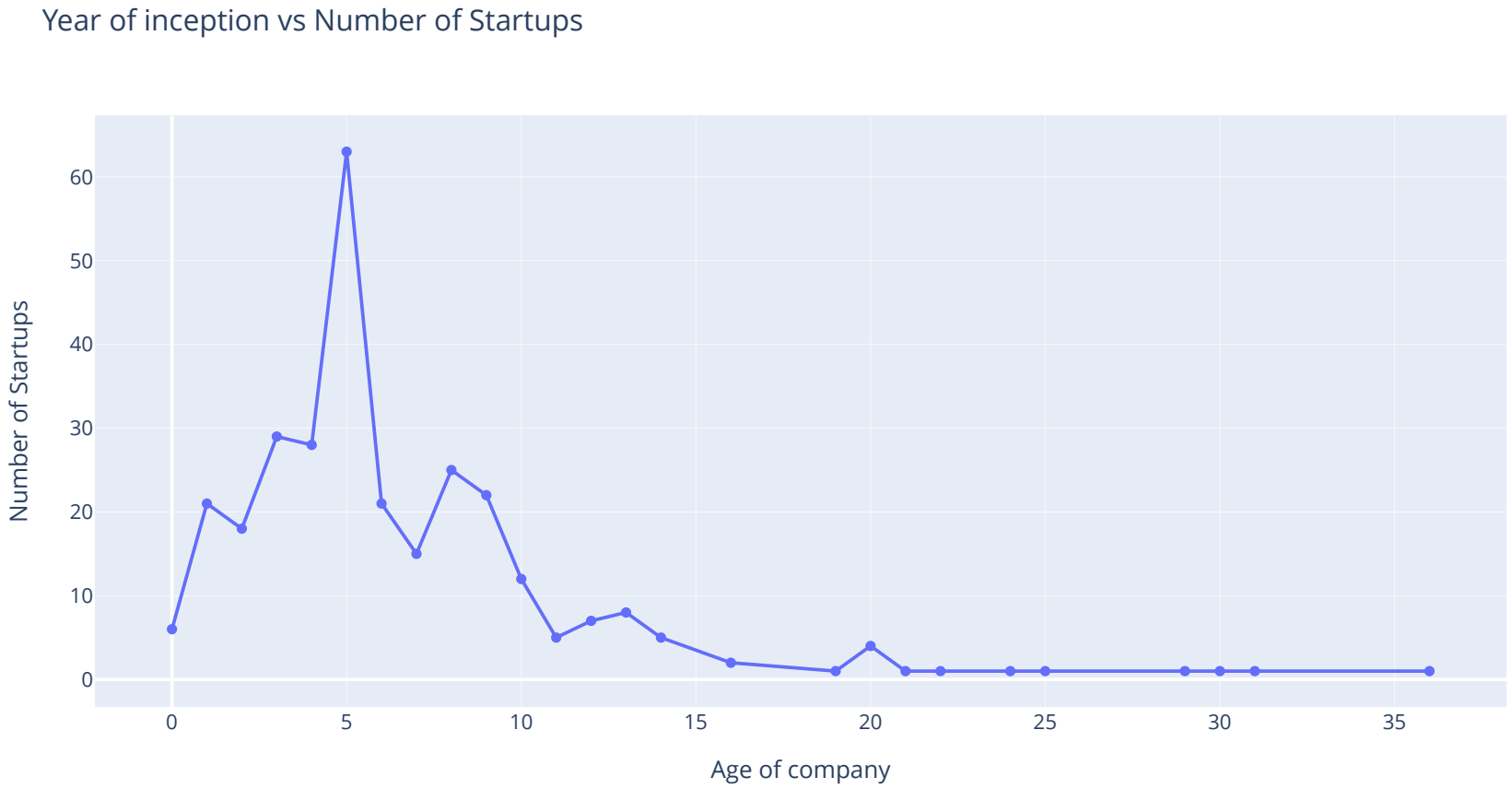
Funding per employee: This metric can be calculated by dividing the total funding raised by a company by the number of employees. This can help identify which companies are more efficient in utilizing their funding and have higher productivity levels.

The plot shows the top 10 companies with the highest funding per employee. It appears that the top company has a funding per employee of around $15.5 million, which is significantly higher than the other companies in the top 10.

Overall, the plot suggests that there are some companies that are able to secure a high amount of funding per employee, which could be an indicator of their ability to generate revenue or potential for growth.

In [23]:

```
import pandas as pd
import plotly.graph_objects as go

# Read the CSV file into a DataFrame
df = pd.read_csv('Startups.csv')
df['Age of company'] = 2020 - df['Starting Year']
df_grouped = df.groupby('Age of company').size().reset_index(name='Number of Startups')

# Create a line chart
fig = go.Figure()
fig.add_trace(go.Scatter(x=df_grouped['Age of company'], y=df_grouped['Number of Startups'], mode='lines+markers'))
fig.update_layout(title='Year of inception vs Number of Startups', xaxis_title='Age of company', yaxis_title='Number of Startups
fig.show()
```



Observation:

The plot shows a line chart of the number of startups getting investments based on their age of inception. It also indicate the preference choice of investers over a company age due ot various factors ie. risk in investment, stability and performance of the company. From the plot we can see that the investor rely more in the company of age between 2-10 years for investment.

Delhi.csv

In [24]:
```
# Load the CSV file
df_delhi = pd.read_csv("Delhi.csv")
df_delhi.head(5)
```

Out[24]:

	Area	BHK	Bathroom	Furnishing	Locality	Parking	Price	Status	Transaction	Type	Per_Sqft
0	800.0	3	2.0	Semi-Furnished	Rohini Sector 25	1.0	6500000	Ready_to_move	New_Property	Builder_Floor	NaN
1	750.0	2	2.0	Semi-Furnished	J R Designers Floors, Rohini Sector 24	1.0	5000000	Ready_to_move	New_Property	Apartment	6667.0
2	950.0	2	2.0	Furnished	Citizen Apartment, Rohini Sector 13	1.0	15500000	Ready_to_move	Resale	Apartment	6667.0
3	600.0	2	2.0	Semi-Furnished	Rohini Sector 24	1.0	4200000	Ready_to_move	Resale	Builder_Floor	6667.0
4	650.0	2	2.0	Semi-Furnished	Rohini Sector 24 carpet area 650 sqft status R...	1.0	6200000	Ready_to_move	New_Property	Builder_Floor	6667.0

In [25]:

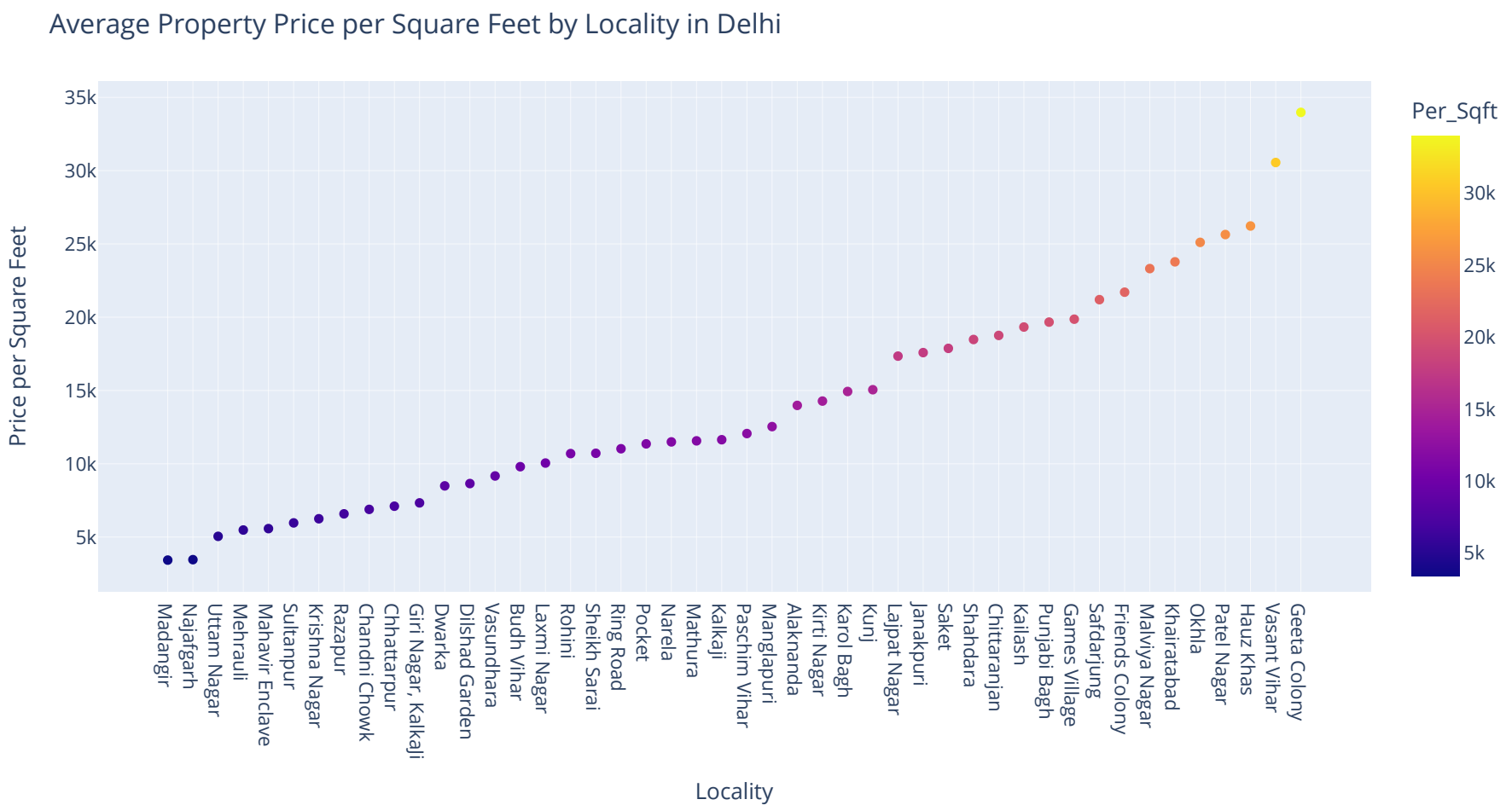
```
#DATA CLEANING
# Update the "Per_Sqft" column by dividing "Price" by "Area"
df_delhi["Per_Sqft"] = round(df_delhi["Price"] / df_delhi["Area"])

# Replacing string in "Locality"
df_delhi.loc[df_delhi['Locality'].str.contains('Lajpat Nagar'), 'Locality'] = 'Lajpat Nagar'
df_delhi.loc[df_delhi['Locality'].str.contains('Rohini Sector'), 'Locality'] = 'Rohini Sector'
df_delhi.loc[df_delhi['Locality'].str.contains('Dwarka'), 'Locality'] = 'Dwarka'
df_delhi.loc[df_delhi['Locality'].str.contains('Uttam Nagar'), 'Locality'] = 'Uttam Nagar'
df_delhi.loc[df_delhi['Locality'].str.contains('Karol Bagh'), 'Locality'] = 'Karol Bagh'
df_delhi.loc[df_delhi['Locality'].str.contains('Patel Nagar'), 'Locality'] = 'Patel Nagar'
df_delhi.loc[df_delhi['Locality'].str.contains('Laxmi Nagar'), 'Locality'] = 'Laxmi Nagar'
df_delhi.loc[df_delhi['Locality'].str.contains('Rohini'), 'Locality'] = 'Rohini'
df_delhi.loc[df_delhi['Locality'].str.contains('Kailash'), 'Locality'] = 'Kailash'
df_delhi.loc[df_delhi['Locality'].str.contains('Chhattarpur'), 'Locality'] = 'Chhattarpur'
df_delhi.loc[df_delhi['Locality'].str.contains('Narela'), 'Locality'] = 'Narela'
df_delhi.loc[df_delhi['Locality'].str.contains('Ring Road'), 'Locality'] = 'Ring Road'
df_delhi.loc[df_delhi['Locality'].str.contains('Hauz Khas'), 'Locality'] = 'Hauz Khas'
df_delhi.loc[df_delhi['Locality'].str.contains('Okhla'), 'Locality'] = 'Okhla'
df_delhi.loc[df_delhi['Locality'].str.contains('Punjabi Bagh'), 'Locality'] = 'Punjabi Bagh'
df_delhi.loc[df_delhi['Locality'].str.contains('Saket'), 'Locality'] = 'Saket'
df_delhi.loc[df_delhi['Locality'].str.contains('Kunj'), 'Locality'] = 'Kunj'
df_delhi.loc[df_delhi['Locality'].str.contains('Kirti Nagar'), 'Locality'] = 'Kirti Nagar'
df_delhi.loc[df_delhi['Locality'].str.contains('Paschim Vihar'), 'Locality'] = 'Paschim Vihar'
df_delhi.loc[df_delhi['Locality'].str.contains('Alaknanda'), 'Locality'] = 'Alaknanda'
df_delhi.loc[df_delhi['Locality'].str.contains('Shahdara'), 'Locality'] = 'Shahdara'
df_delhi.loc[df_delhi['Locality'].str.contains('Dilshad Garden'), 'Locality'] = 'Dilshad Garden'
df_delhi.loc[df_delhi['Locality'].str.contains('Pocket'), 'Locality'] = 'Pocket'
df_delhi.loc[df_delhi['Locality'].str.contains('Vasundhara'), 'Locality'] = 'Vasundhara'
df_delhi.loc[df_delhi['Locality'].str.contains('Friends Colony'), 'Locality'] = 'Friends Colony'
df_delhi.loc[df_delhi['Locality'].str.contains('Vasant Vihar'), 'Locality'] = 'Vasant Vihar'
df_delhi.loc[df_delhi['Locality'].str.contains('Kalkaji'), 'Locality'] = 'Kalkaji'
df_delhi.loc[df_delhi['Locality'].str.contains('Chandni Chowk'), 'Locality'] = 'Chandni Chowk'
df_delhi.loc[df_delhi['Locality'].str.contains('Sheikh Sarai'), 'Locality'] = 'Sheikh Sarai'
df_delhi.loc[df_delhi['Locality'].str.contains('Chittaranjan'), 'Locality'] = 'Chittaranjan'
df_delhi.loc[df_delhi['Locality'].str.contains('Safdarjung Enclave'), 'Locality'] = 'Safdarjung Enclave'
df_delhi.loc[df_delhi['Locality'].str.contains('Budh Vihar'), 'Locality'] = 'Budh Vihar'
df_delhi.loc[df_delhi['Locality'].str.contains('Mathura'), 'Locality'] = 'Mathura'
df_delhi.loc[df_delhi['Locality'].str.contains('Safdarjung Enclave'), 'Locality'] = 'Safdarjung'
df_delhi.loc[df_delhi['Locality'].str.contains('Mahavir Enclave'), 'Locality'] = 'Mahavir Enclave'
df_delhi.loc[df_delhi['Locality'].str.contains('Sultanpur'), 'Locality'] = 'Sultanpur'
df_delhi.loc[df_delhi['Locality'].str.contains('Mehrauli'), 'Locality'] = 'Mehrauli'
df_delhi.loc[df_delhi['Locality'].str.contains('Malviya Nagar'), 'Locality'] = 'Malviya Nagar'
df_delhi.loc[df_delhi['Locality'].str.contains('Games Village'), 'Locality'] = 'Games Village'
```

```
df_delhi.loc[df_delhi['Locality'].str.contains('Najafgarh'), 'Locality'] ='Najafgarh'
df_delhi.loc[df_delhi['Locality'].str.contains('Manglapuri'), 'Locality'] = 'Manglapuri'
df_delhi.loc[df_delhi['Locality'].str.contains('Madangir'), 'Locality'] = 'Madangir'
df_delhi.loc[df_delhi['Locality'].str.contains('Khairatabad'), 'Locality'] = 'Khairatabad'
df_delhi.loc[df_delhi['Locality'].str.contains('Krishna Nagar'), 'Locality'] = 'Krishna Nagar'
df_delhi.loc[df_delhi['Locality'].str.contains('Razapur'), 'Locality'] = 'Razapur'
df_delhi.loc[df_delhi['Locality'].str.contains('Janakpuri'), 'Locality'] = 'Janakpuri'

df_delhi = df_delhi[~df_delhi['Locality'].str.contains('mind.')]

df_delhi.head(5)
```

Out[25]:

	Area	BHK	Bathroom	Furnishing	Locality	Parking	Price	Status	Transaction	Type	Per_Sqft
0	800.0	3	2.0	Semi-Furnished	Rohini	1.0	6500000	Ready_to_move	New_Property	Builder_Floor	8125.0
1	750.0	2	2.0	Semi-Furnished	Rohini	1.0	5000000	Ready_to_move	New_Property	Apartment	6667.0
2	950.0	2	2.0	Furnished	Rohini	1.0	15500000	Ready_to_move	Resale	Apartment	16316.0
3	600.0	2	2.0	Semi-Furnished	Rohini	1.0	4200000	Ready_to_move	Resale	Builder_Floor	7000.0
4	650.0	2	2.0	Semi-Furnished	Rohini	1.0	6200000	Ready_to_move	New_Property	Builder_Floor	9538.0

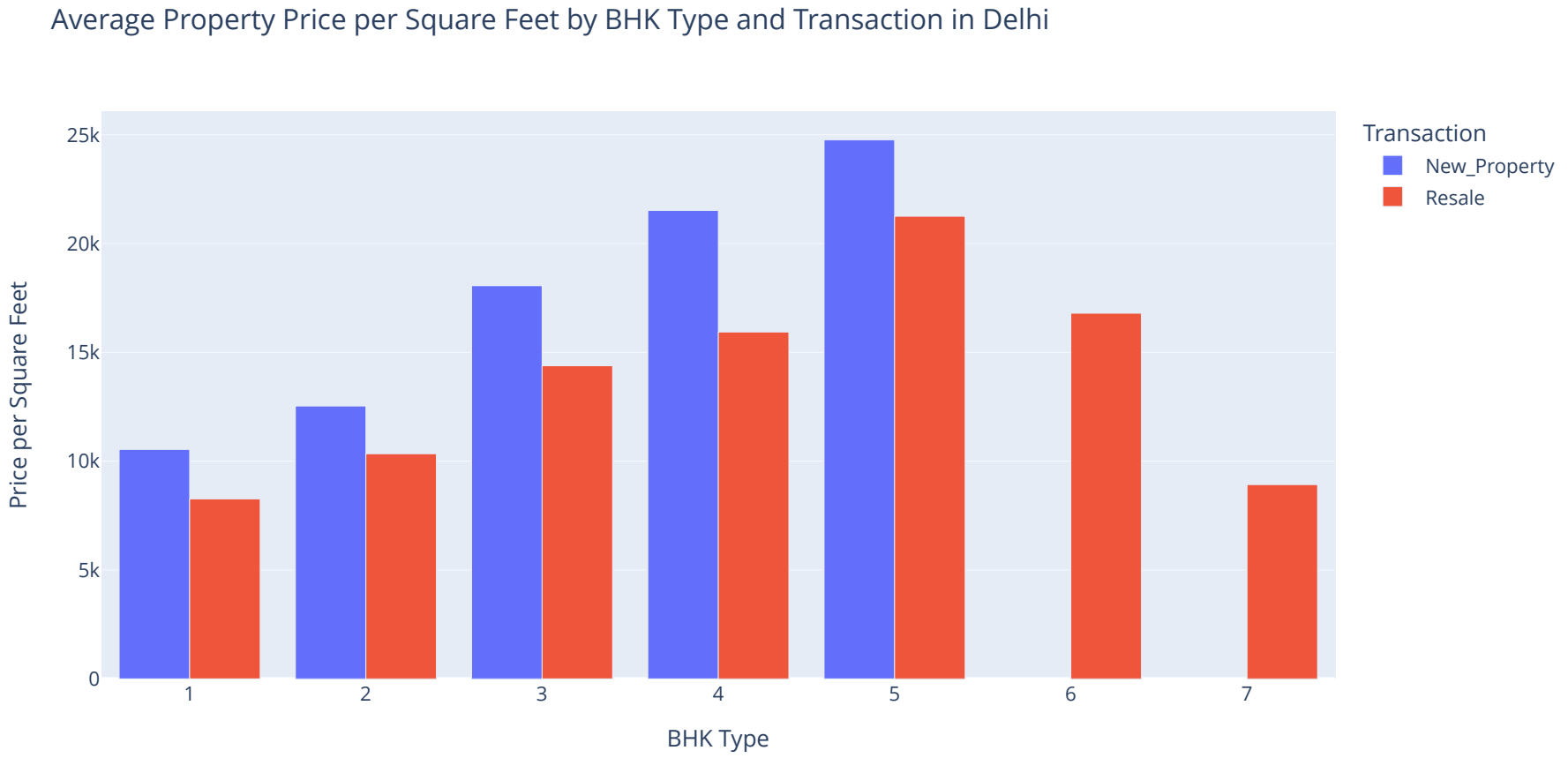
In [26]:
```
# Calculate the average "per sqft price"
avg_price = df_delhi.groupby("Locality")["Per_Sqft"].mean().reset_index().sort_values(by="Per_Sqft")

# Create a scatter plot using plotly
fig = px.scatter(avg_price, x="Locality", y="Per_Sqft", color="Per_Sqft")
fig.update_layout(title="Average Property Price per Square Feet by Locality in Delhi",
                  xaxis_title="Locality", yaxis_title="Price per Square Feet")
fig.show()
```



Observation: From the plot, we can observe that the average property price per square feet varies widely across different localities in Delhi. Some localities have an average price per square feet that is much higher than others. We can also see that there are a few localities with a very high

average price per square feet that stand out from the rest. Overall, the plot provides a good overview of the distribution of property prices across different localities in Delhi.

In [27]:
```
avg_price = df_delhi.groupby(["BHK", "Transaction"])["Per_Sqft"].mean().reset_index()

fig = px.bar(avg_price, x="BHK", y="Per_Sqft", color="Transaction", barmode="group",
             title="Average Property Price per Square Feet by BHK Type and Transaction in Delhi",
             labels={"BHK": "BHK Type", "Per_Sqft": "Price per Square Feet"})
fig.show()
```

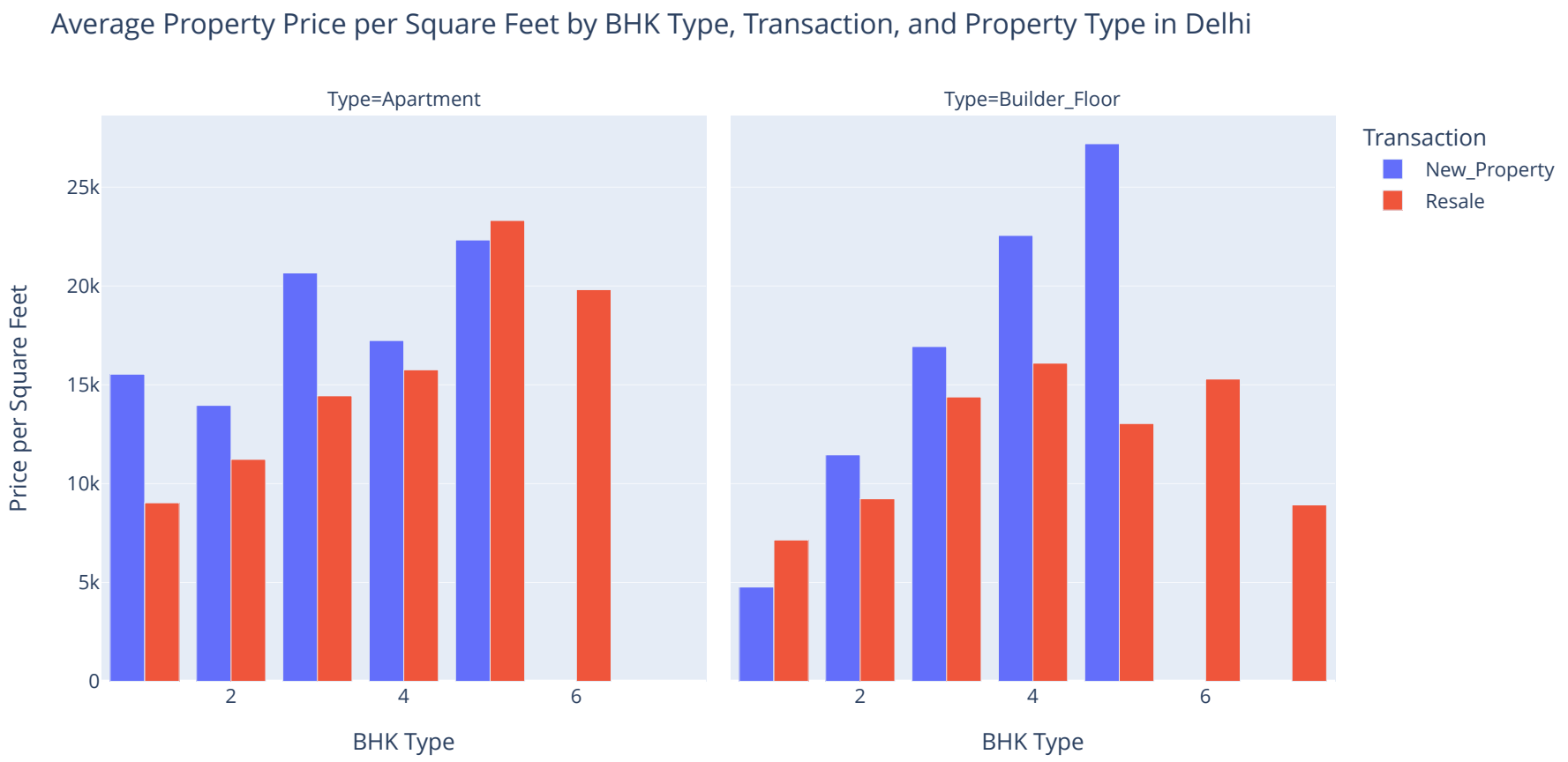


Observation: This plot shows the average property price per square foot by BHK type and transaction type in Delhi. The plot uses a grouped bar chart to display the data, with the x-axis showing the BHK type, the y-axis showing the average price per square foot, and the color of the bars indicating the transaction type (resale or new construction).

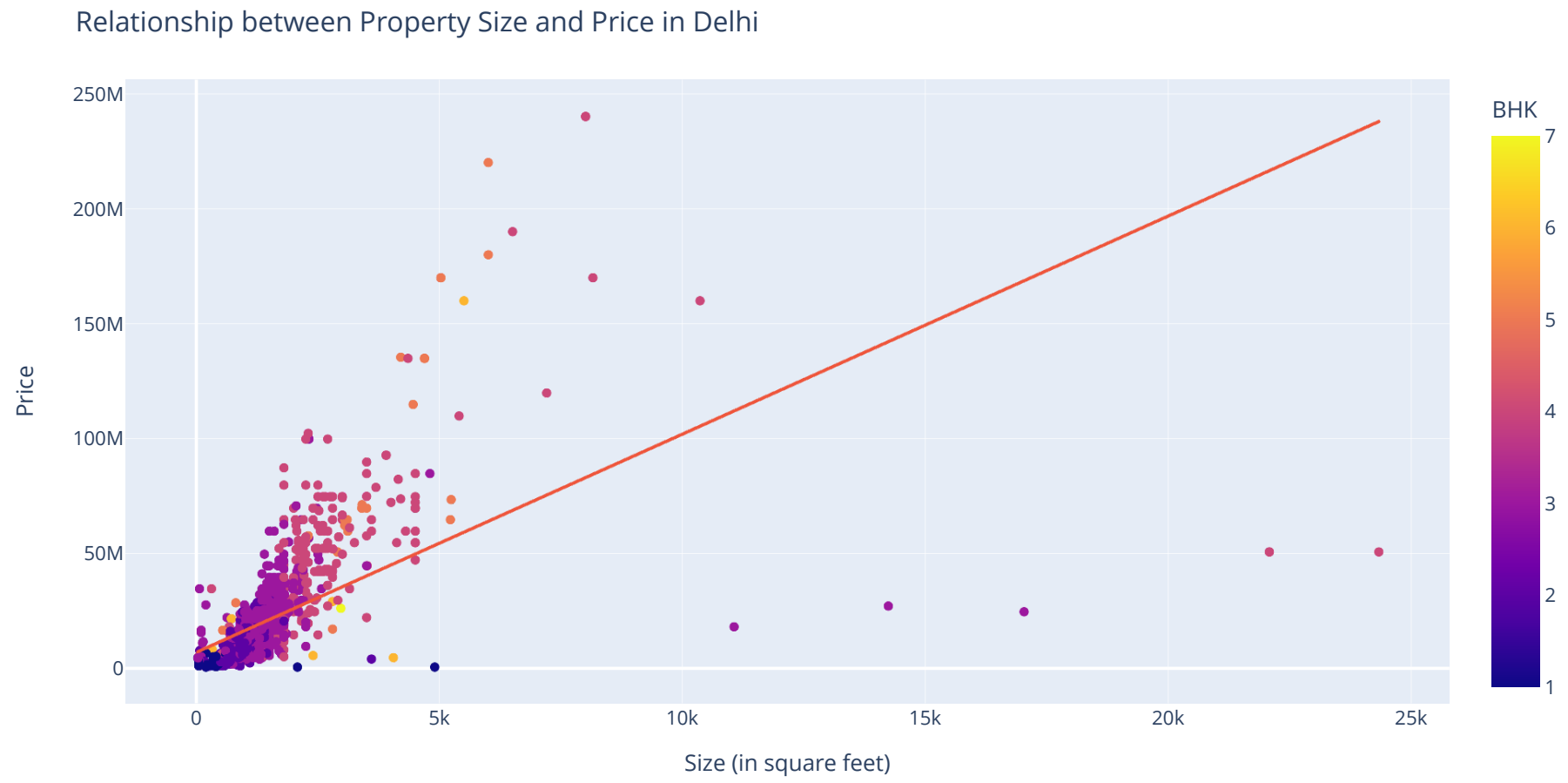
From the plot, we can observe that in general, as the number of BHKs increases, the average price per square foot also increases. Additionally, we can see that the average price per square foot for new construction properties is generally higher than for resale properties, across all BHK types.

We can observe from the plot that after 5BHK New Property is not availabe in delhi indicating the preference choice of society as they tend towards more nuclear families.

In [28]:
```
# Calculate the average "per sqft price" for each BHK type, Transaction type, and Property type
avg_price = df_delhi.groupby(["BHK", "Transaction", "Type"])["Per_Sqft"].mean().reset_index()

# Create a grouped bar chart
fig = px.bar(avg_price, x="BHK", y="Per_Sqft", color="Transaction", barmode="group",
             facet_col="Type",
             title="Average Property Price per Square Feet by BHK Type, Transaction, and Property Type in Delhi",
             labels={"BHK": "BHK Type", "Per_Sqft": "Price per Square Feet"})
fig.show()
```



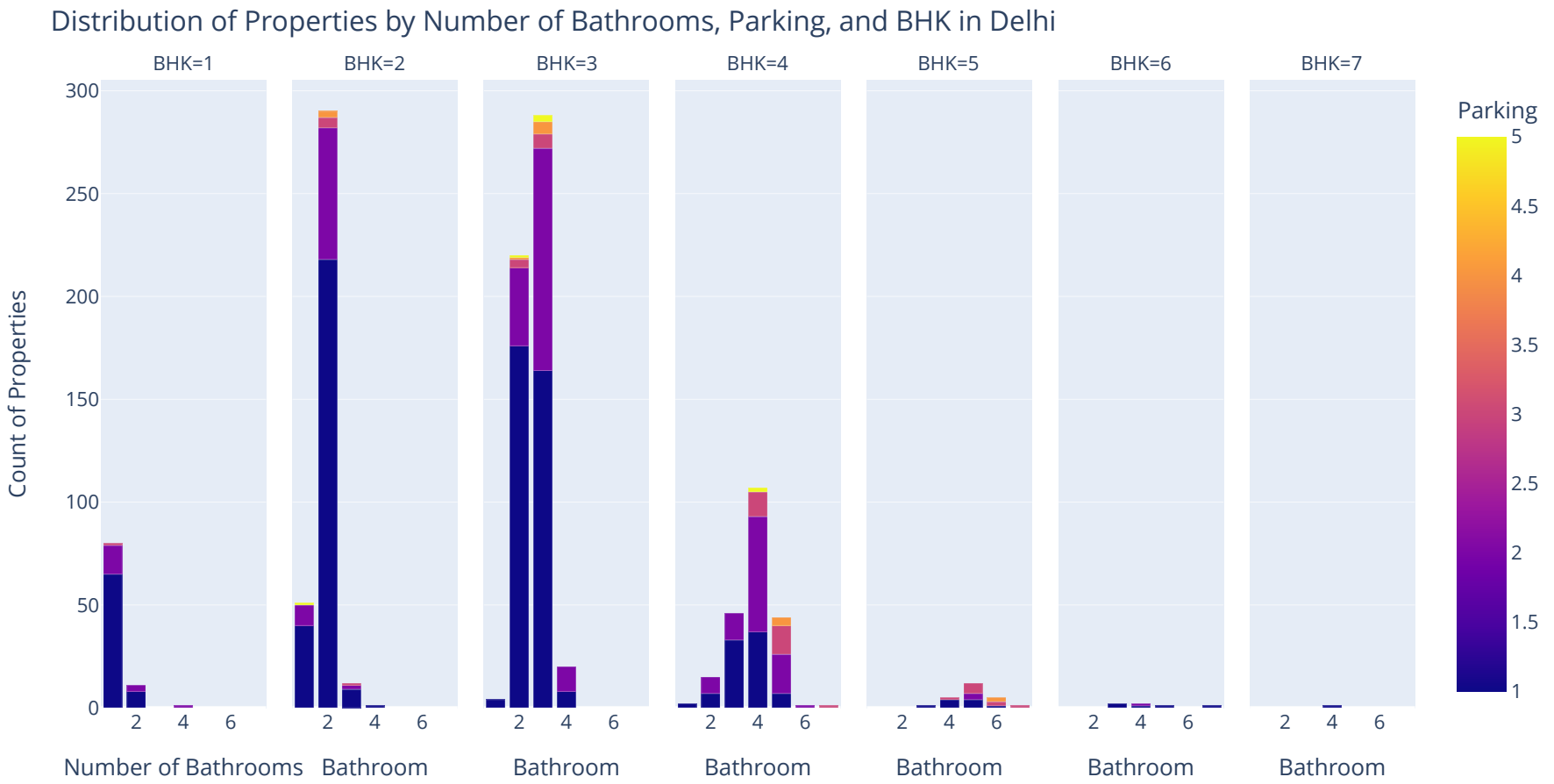
In [29]:
```
# Create a scatter plot using plotly
fig = px.scatter(df_delhi, x="Area", y="Price", trendline="ols", color="BHK")

# Set the title and axis labels
fig.update_layout(title="Relationship between Property Size and Price in Delhi",
                  xaxis_title="Size (in square feet)", yaxis_title="Price")

# Show the plot
fig.show()
```



The above plot using Plotly shows the relationship between property size (in square feet) and price in Delhi. A linear regression line is added to the plot, and the color="BHK" argument colors the data points by the number of bedrooms (BHK) in the property.

The plot can help to visualize the overall trend between size and price, as well as any variation in this trend based on the number of bedrooms in the property.

In [30]:
```
# Data cleaning in Parking
parking_mean = round(df_delhi["Parking"].mean())
df_delhi["Parking"] = np.where(df_delhi["Parking"] > 8, parking_mean, df_delhi["Parking"])

# count of properties
count_props = df_delhi.groupby(["BHK", "Bathroom", "Parking"])["Area"].count().reset_index()

fig = px.bar(count_props, x="Bathroom", y="Area", color="Parking",
             barmode="stack", facet_col="BHK")
fig.update_layout(title="Distribution of Properties by Number of Bathrooms, Parking, and BHK in Delhi",
                  xaxis_title="Number of Bathrooms", yaxis_title="Count of Properties")
fig.show()
```



Observation:

The plot shows the distribution of properties in Delhi based on the number of bathrooms, parking spaces, and BHK. The plot is divided into multiple facets based on the BHK types. Each facet has stacked bars, where the height of each bar indicates the count of properties with a specific number of bathrooms and parking spaces.

From the plot, we can observe that most of the properties have 2-3 bathrooms, and the number of parking spaces varies from 1 to 3. For 1 BHK properties, most of them have one parking space. For 2 BHK properties, the majority have either one or two parking spaces, while for 4 BHK properties, the majority have two or three parking spaces. There are also a few properties with four or more parking spaces, but they are relatively rare.

Overall, the plot provides an informative view of the distribution of properties in Delhi based on their size, amenities, and parking spaces.

CONCLUSION:

The Datasets along with the plots provide valuable insights into different industries and markets. By combining the information from all three files, one can gain a comprehensive understanding of the startup ecosystem, consumer spending behavior, and real estate market in a city, which can help make informed business decisions.