Startup Analysis in 2017

In [1]:
```
import pandas as pd
df_2017 = pd.read_excel('C:/Users/Snehal/Downloads/2017_data.xlsx')
df_2017.head()
```

Out[1]:

	Sr. No.	Date	Startup Name	Industry/Vertical	Sub-Vertical	City	Investor Name	Investment Type	Amount(in USD)	InvestmentType
0	1.0	2017-09-01	Aahaa Stores	eCommece	Online B2B store for office supplies	Chennai	YourNest Angel Fund	Private Equity	1,000,000	NaN
1	2.0	2017-09-01	MFine	Consumer Internet	Online Doctor Discovery platform	Bangalore	Stellaris Venture Partners, Mayur Abhaya, Rohi...	Private Equity	1,500,000	NaN
2	3.0	2017-09-01	Canvera	Consumer Internet	Online Photography platform	Mumbai	InfoEdge	Private Equity	1,300,000	NaN
3	4.0	2017-09-04	PrimaryIO	Technology	Application Performance Acceleration	Pune	Accel Partners, Exfinity Ventures, Partech Ven...	Private Equity	5,600,000	NaN
4	5.0	2017-09-05	Shubh Loans	Consumer Internet	online lending platform	Bangalore	SRI Capital, BeeNext, Pravega Ventures	Private Equity	1,500,000	NaN

In [2]:
```
# Renaming columns for our convinience

def renaming_columns(df_2017):
    df_2017.rename(columns={
    'Startup Name': 'Startup_Name',
    'City': 'Location',
    'Investor Name': 'Investors',
    'InvestmentType': 'Investment_Type',
    'Amount(in USD)': 'Amount($)',
    'Sub-Vertical': 'Sub_Industry',
    'Industry/Vertical':'Industry'
     }, inplace=True)
renaming_columns(df_2017)
```

In [3]:
```
# Extracting required columns
df_2017 = df_2017[['Date','Startup_Name','Industry','Sub_Industry','Location','Investors','Investment_Type','Am
```

In [4]:
```
# Dealing with date column to extract Year & Month
def date_opertion(df_2017):
    df_2017['Date'] = pd.to_datetime(df_2017['Date'], format="%d %B %Y")
    df_2017['Month'] = df_2017['Date'].dt.strftime('%B')
    df_2017['Year'] = df_2017['Date'].dt.year
date_opertion(df_2017)
```

In [5]:
```
# Dealing with duplicate rows
def duplicate_rows(data):
    duplicate_rows = data[data.duplicated()]
    if len(duplicate_rows) > 0:
        data = data.drop_duplicates()
        print('Droped',len(duplicate_rows),'Duplicate Rows.')
    else:
        print('No Duplicate Rows.')
duplicate_rows(df_2017)
```

```
Droped 9 Duplicate Rows.
```

In [6]:
```
# Dealing with Amount column data type
def amount_column(data):
    data['Amount($)'] = data['Amount($)'].fillna(0)
    data['Amount($)'] = data['Amount($)'].astype(str)
    data['Amount($)'] = data['Amount($)'].str.replace(',', '')
    data['Amount($)'] = data['Amount($)'].astype(float)
amount_column(df_2017)
```

In [7]:
```
# Editing Industry column
values_to_replace = {'eCommece' : 'E-Commerce',
                     'eCommerce' : 'E-Commerce',
                     'ECommerce' : 'E-Commerce',
                     'Ecommerce' : 'E-Commerce',
                     'Health Care' : 'Healthcare',}

def replace_values(df):
    df['Industry'] = df['Industry'].replace(values_to_replace)
replace_values(df_2017)
```

In [8]:
```
# Editing Location column

def replace_values(df):
    value_to_replace = {'Bengaluru': 'Bangalore', 'Nw Delhi' : 'New Delhi', 'Delhi':'New Delhi'}
    df['Location'] = df['Location'].replace(value_to_replace)
```

```
replace_values(df_2017)
```

In [9]:
```
# Editing Investor column
values_to_replace = {'Undisclosed investor' : 'Undisclosed Investors',
                     'Undisclosed investors' : 'Undisclosed Investors',
                      'Undisclosed Investor' : 'Undisclosed Investors',
                      'undisclosed investors' : 'Undisclosed Investors'
                     }
def replace_values(df):
    df['Investors'] = df['Investors'].replace(values_to_replace)
replace_values(df_2017)
```

In [10]:
```
df_2017.head()
```

Out[10]:

	Date	Startup_Name	Industry	Sub_Industry	Location	Investors	Investment_Type	Amount($)	Month	Year
0	2017-09-01	Aahaa Stores	E-Commerce	Online B2B store for office supplies	Chennai	YourNest Angel Fund	NaN	1000000.0	September	2017.0
1	2017-09-01	MFine	Consumer Internet	Online Doctor Discovery platform	Bangalore	Stellaris Venture Partners, Mayur Abhaya, Rohi...	NaN	1500000.0	September	2017.0
2	2017-09-01	Canvera	Consumer Internet	Online Photography platform	Mumbai	InfoEdge	NaN	1300000.0	September	2017.0
3	2017-09-04	PrimaryIO	Technology	Application Performance Acceleration	Pune	Accel Partners, Exfinity Ventures, Partech Ven...	NaN	5600000.0	September	2017.0
4	2017-09-05	Shubh Loans	Consumer Internet	online lending platform	Bangalore	SRI Capital, BeeNext, Pravega Ventures	NaN	1500000.0	September	2017.0

Summary of the year 2017

- Shape = (700, 10)
- Unique Industry = 12
- Unique Sub_Industry = 656
- Unique Location = 17
- Unique Investment_Type = 3

In [ ]:
```

```

In [11]:
```
# Total Startup count in 2017
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import pandas as pd

def startup_count(data):
    startup_count = data['Startup_Name'].nunique()
    return go.Indicator(
        mode="number",
        value=startup_count,
        title="Startup Count")

fig = make_subplots(rows=1, cols=1)

fig.add_trace(startup_count(df_2017))

fig.update_layout(title_text="Startup Count in 2017")
fig.show()
```

Startup Count in 2017

Startup Count

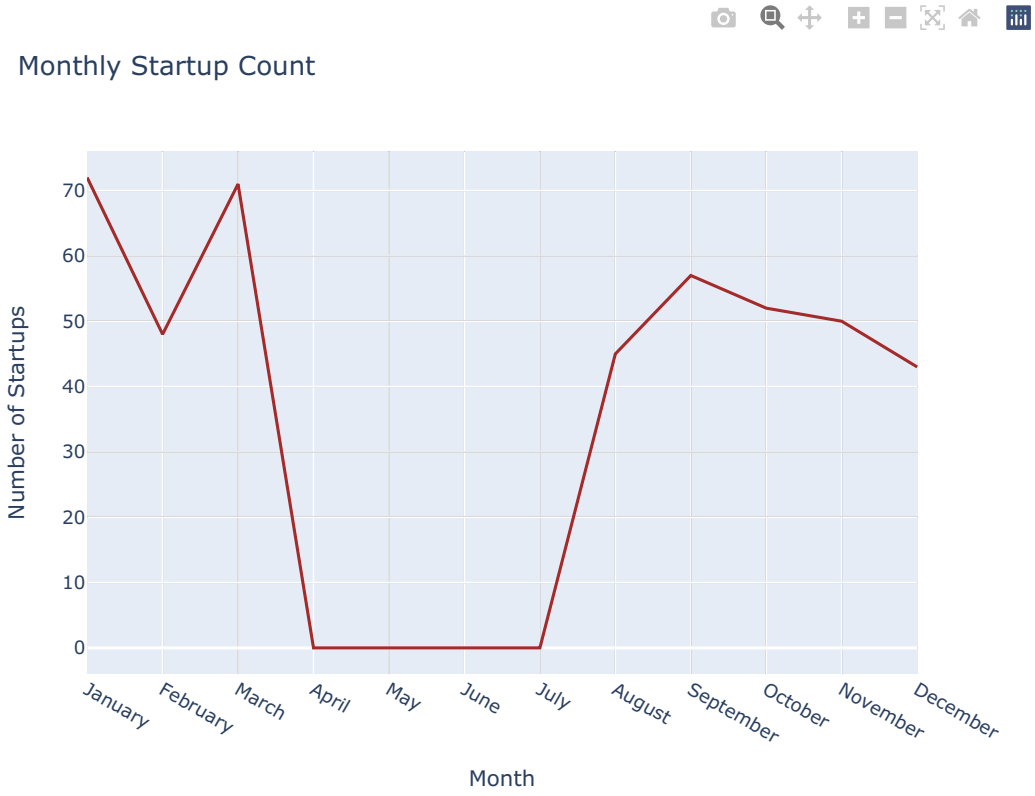
640

In [12]:
```
import pandas as pd
import ipywidgets as widgets
from IPython.display import display, HTML
import plotly.graph_objects as go

def update_subindustry_options(change):
    pass

def create_table(selected_industry):
    if selected_industry == 'All':
        display(HTML("<p>Select an industry to view the table.</p>"))
    else:
        filtered_df = df_2017[df_2017['Industry'] == selected_industry]

        if filtered_df.empty:
            display(HTML("<p>No data available for the selected industry.</p>"))
        else:
            trace = go.Table(
                header=dict(values=["Startup_Name", "Sub_Industry", "Investors", "Investment_Type", "Location",
                            fill=dict(color='#abb8e7'),
                            align=['left', 'center']),
                cells=dict(values=[filtered_df['Startup_Name'],
                                   filtered_df['Sub_Industry'],
                                   filtered_df['Investors'],
                                   filtered_df['Investment_Type'],
                                   filtered_df['Location'],
                                   filtered_df['Amount($)'],
                                   filtered_df['Month']],
                           fill=dict(color=['white', 'lightgray']),
                           align=['left', 'center'])
            )

            layout = dict(width=1000, height=800)
            fig = go.Figure(data=[trace], layout=layout)
            fig.update_layout(margin=dict(l=0, r=0, t=0, b=0))
            display(fig)

industry_dropdown = widgets.Dropdown(options=['All'] + sorted(df_2017['Industry'].dropna().unique()), value='Al
industry_dropdown.observe(update_subindustry_options, names='value')
widgets.interactive(create_table, selected_industry=industry_dropdown)
```

Out[12]:
```
interactive(children=(Dropdown(description='Industry:', options=('All', 'Consumer Internet', 'Consumer Portal'…
```

In [13]:
```
import plotly.express as px

# Month wise startups
def monthly_startup_count(data):
    month_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'Octo
    data['Month'] = pd.Categorical(data['Month'], categories=month_order, ordered=True)
    monthly_count = data.groupby('Month')['Startup_Name'].nunique().reset_index()
    fig = px.line(monthly_count, x='Month', y='Startup_Name', title='Monthly Startup Count', labels={'Startup_N
    fig.update_layout(xaxis=dict(title='Month'))
    fig.update_traces(line=dict(color='brown'))
    fig.show()
```

```
monthly_startup_count(df_2017)
```



In [14]:
```
import plotly.express as px

def top_10_location(data):
    location_counts = data.groupby('Location').size().reset_index(name='Count')
    top_10_locations = location_counts.sort_values(by='Count', ascending=False).head(10)
    fig = px.bar(top_10_locations, x='Location', y='Count', color='Count',
                 title='Top 10 Locations by Startups Count',
                 color_continuous_scale='sunset')
    fig.update_layout(xaxis_title='Location', yaxis_title='Count')
    fig.show()

top_10_location(df_2017)
```

In [15]:
```
import plotly.express as px

def Industry_wise_startup(df):
    industry_count = df['Industry'].value_counts().reset_index()
    industry_count.columns = ['Industry', 'Startup Count']
```

```
    fig = px.bar(industry_count, x='Industry', y='Startup Count',
             title='Industry-wise Count of Startups',
             labels={'Industry': 'Industry', 'Startup Count': 'Number of Startups'},color = 'Startup Count', te
             color_continuous_scale='sunset')
    fig.update_traces(textposition='outside')
    fig.show()

Industry_wise_startup(df_2017)
```

In [16]:

```
# Top 10 Sub-Industries by Startups Count

def sub_industry_top_10(data):
    top_sub_industries = data['Sub_Industry'].value_counts().nlargest(10)
    top_sub_df = top_sub_industries.reset_index()
    top_sub_df.columns = ['Sub_Industry', 'Count']
    fig = px.bar(top_sub_df, y='Sub_Industry', x='Count', title='Top 10 Sub-Industries by Startups Count', orie
                color_continuous_scale='sunset')
    fig.update_yaxes(categoryorder='total ascending')
    fig.show()

sub_industry_top_10(df_2017)
```

In [17]:
```
# Top 10 Invetors
def top_investors(data):
    top_investors = data['Investors'].value_counts().nlargest(10)
    top_investors_df = top_investors.reset_index()
    top_investors_df.columns = ['Investor', 'Investment Count']
    fig = px.bar(top_investors_df, x='Investor', y='Investment Count',
                 title='Top 10 Investors by Investment Count',
                 labels={'Investor': 'Investor Name', 'Investment Count': 'Number of Investments'},
                 color= 'Investment Count',
                        color_continuous_scale='sunset')

    fig.show()

top_investors(df_2017)
```

In [18]:
```
import pandas as pd
import ipywidgets as widgets
from IPython.display import display, HTML

# Location wise Startups
def create_table(selected_location):
    if selected_location == 'All':
        display(HTML("<p>Select an option from the dropdown to view the table.</p>"))
    else:
        filtered_df = df_2017[df_2017['Location'] == selected_location]

        if filtered_df.empty:
            display(HTML("<p>No data available for the selected location.</p>"))
        else:
            trace = go.Table(
                header=dict(values=["Startup Name", "Sub Industry", "Investor", "Investment Type", "Amount($)",
                            fill=dict(color='lightblue'),
                            align=['left', 'center']),
                cells=dict(values=[filtered_df['Startup_Name'],
                                   filtered_df['Sub_Industry'],
                                   filtered_df['Investors'],
                                   filtered_df['Investment_Type'],
                                   filtered_df['Amount($)'],
                                   filtered_df['Month']],
                           fill=dict(color=['white', 'lightgray']),
                           align=['left', 'center'])
            )

            layout = dict(width=1000, height=800)
            fig = go.Figure(data=[trace], layout=layout)
            display(fig)

location_dropdown = widgets.Dropdown(options=['All'] + sorted(df_2017['Location'].dropna().unique()), value='Al
widgets.interactive(create_table, selected_location=location_dropdown)
```

Out[18]:
```
interactive(children=(Dropdown(description='Location:', options=('All', 'Ahmedabad', 'Bangalore', 'Chennai', '…
```

In [ ]: