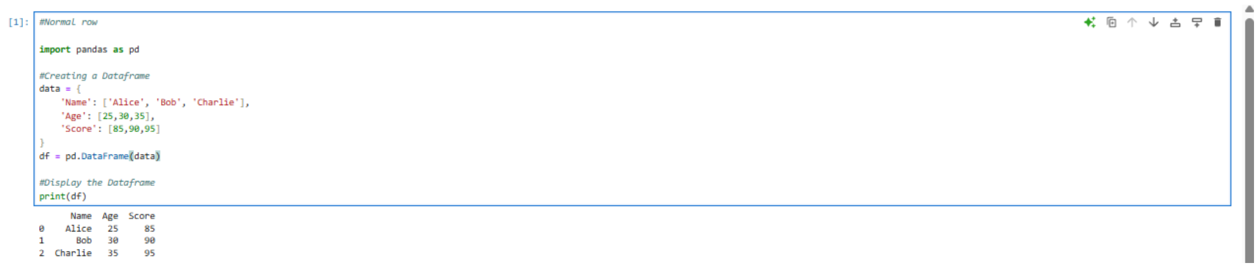
# Activity 2

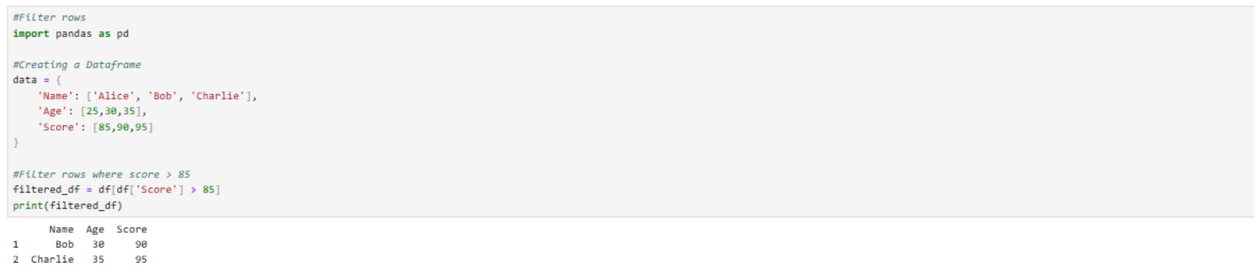
## Normal Row



```
[1]: #Normal row

     import pandas as pd

     #Creating a Dataframe
     data = {
         'Name': ['Alice', 'Bob', 'Charlie'],
         'Age': [25,30,35],
         'Score': [85,90,95]
     }
     df = pd.DataFrame(data)

     #Display the Dataframe
     print(df)
```

```
      Name  Age  Score
0    Alice   25     85
1      Bob   30     90
2  Charlie   35     95
```

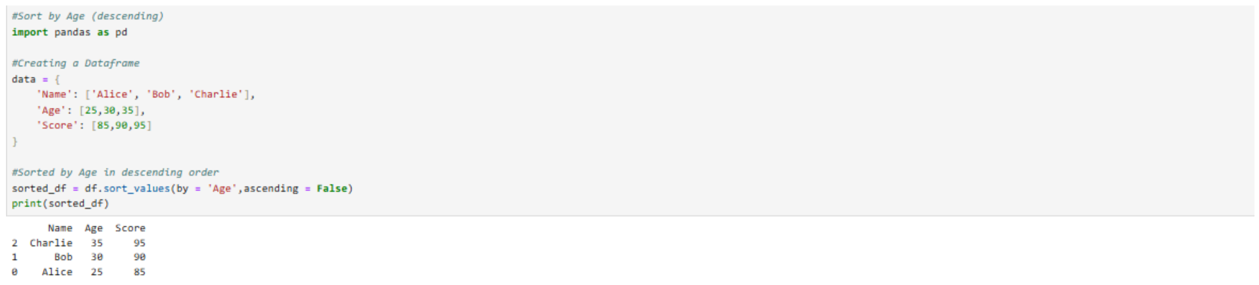
## Filter row:



```
#Filter rows
import pandas as pd

#Creating a Dataframe
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25,30,35],
    'Score': [85,90,95]
}

#Filter rows where score > 85
filtered_df = df[df['Score'] > 85]
print(filtered_df)
```

```
      Name  Age  Score
1      Bob   30     90
2  Charlie   35     95
```

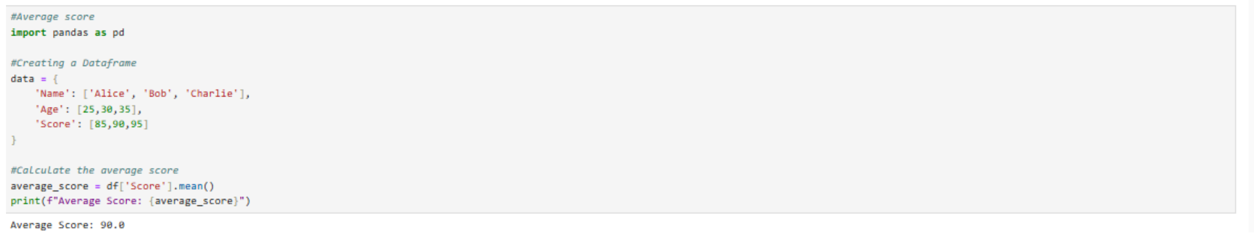
## Sort by Age:



```
#Sort by Age (descending)
import pandas as pd

#Creating a Dataframe
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25,30,35],
    'Score': [85,90,95]
}

#Sorted by Age in descending order
sorted_df = df.sort_values(by = 'Age',ascending = False)
print(sorted_df)
```

```
      Name  Age  Score
2  Charlie   35     95
1      Bob   30     90
0    Alice   25     85
```

## Average Score:



```
#Average score
import pandas as pd

#Creating a Dataframe
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25,30,35],
    'Score': [85,90,95]
}

#Calculate the average score
average_score = df['Score'].mean()
print(f"Average Score: {average_score}")
```

```
Average Score: 90.0
```

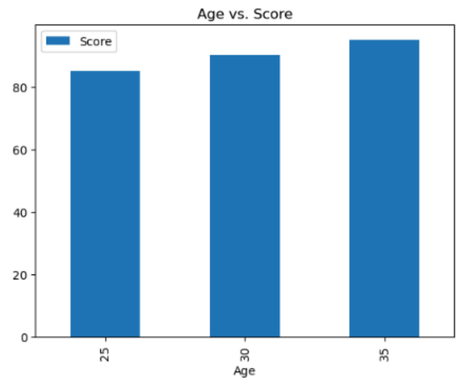
## Data Frame:

```
#Another Data Frame
import pandas as pd
new_data = {
    'Name': ['David', 'Eva'],
    'Age': [40,22],
    'Score':[88,92]
}
new_df = pd.DataFrame(new_data)

#Concatenate the two Dataframes
combined_df = pd.concat([df,new_df])
print(combined_df)
```

```
      Name  Age  Score
0    Alice   25     85
1      Bob   30     90
2  Charlie   35     95
0    David   40     88
1      Eva   22     92
```

## 1st matplotlib test:

```
import matplotlib.pyplot as plt

#Plot Age vs. Score
df.plot(x= 'Age', y= 'Score', kind='bar',title= "Age vs. Score")
plt.show()
```



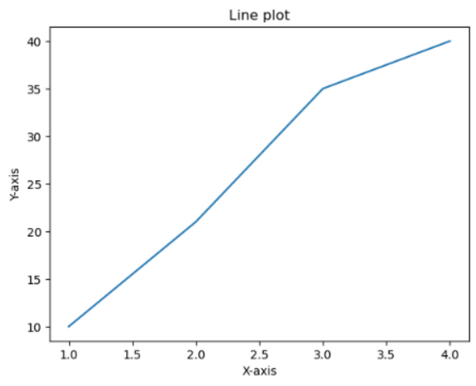
## 1st pandas test:

```
import pandas as pd

#Load as CSV file into a DataFrame
df = pd.read_csv('data.csv')

#Display
print(df.head())
```

```
   FirstName   LastName  Salary Abroad  Salary Locally Philippinre Office  \
0  Jess Dale  Dela Cruz          70000          150000            Ortigas   
1      Maria     Santos          65000          140000             Makati   
2       John        Doe          80000          160000             Taguig   
3       Anna     Garcia          75000          155000               Cebu   
4       Mark        Tan          72000          145000              Davao   

  Singapore Office  
0   Fortune Centre  
1    Raffles Place  
2      Shenton Way  
3    Tanjong Pagar  
4           Jurong  
```

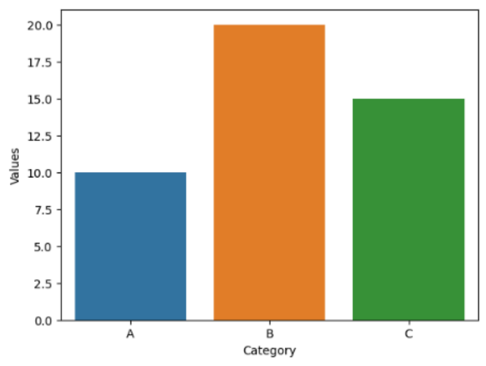
## Line plot:

```
import matplotlib.pyplot as plt

#simple line plot
X = [1,2,3,4]
Y = [10,21,35,40]
plt.plot (X,Y)
plt.title("Line plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```



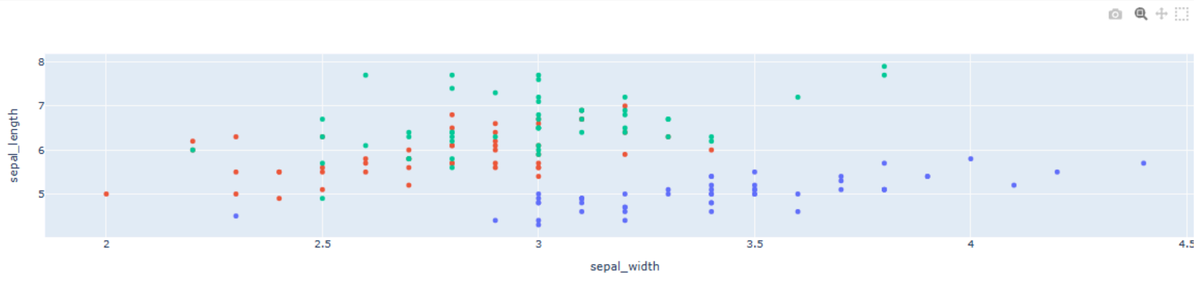
## Using seaborn,matplotlib,pandas:

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

#Example data
data = pd.DataFrame({'Category':['A','B','C'],'Values':[10,20,15]})

#Bar plot
sns.barplot(x='Category', y='Values', data=data)
plt.show()
```



## Using plotly:

```
[34]: import plotly.express as px

#Simple scatter plot
df = px.data.iris()
fig = px.scatter(df, x='sepal_width', y='sepal_length',color='species')
fig.show()
```



**Using  plotly, seaborn,matplotlib,pandas:**

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

#Load the Excel file
data = pd.read_csv('data.csv')

#Prepare data
data[ 'Employee'] = data[ 'FirstName'] + "" + data['LastName']

#Function to create a dashboard with slicers
def create_dashboard(first_name = None, last_name = None):
    filtered_data = data
    if first_name:
        filtered_data = filtered_data[filtered_data['FirstName'] == first_name]
    if last_name:
        filtered_data = filtered_data[filtered_data['LastName'] == last_name]

    if filtered_data.empty:
        print("No data available for the selected filters.")
        return

#Bar Chart: Comparison of Salaries Abroad vs Locally
plt.figure(figsize=(12,6))
sns.barplot(x='Employee', y = 'Salary Abroad', data=data, color = 'skyblue', label = 'Salary
Abroad')
sns.barplot(x='Employee', y = 'Salary Locally', data=data, color = 'lightgreen', label = 'Salary
Locally')
plt.xlabel('Employees')
plt.ylabel('Salary')
plt.title('Comparison of Salaries Abroad vs Locally')
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout
plt.show()

#Line Chart: Salaries Abroad and Locally by Employee
plt.figure(figsize=(12,6))
sns.lineplot(x='Employee', y ='Salary Abroad', data=data, marker = 'o', label = 'Salary Abroad')
sns.lineplot(x='Employee', y ='Salary Locally',data=data, marker = 's', label = 'Salary Locally',
color = 'orange')
plt.xlabel('Employees')
plt.ylabel('Salary')
plt.title('Trend of Salaries Abroad vs Locally')
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout
plt.show()

#Pie Chart: Proportion of Total Salaries Abroad vs Locally
total_salary_abroad = data['Salary Abroad'].sum()
total_salary_local = data['Salary Locally'].sum()
sizes = [total_salary_abroad, total_salary_local]
labels = ['Total Salary Abroad', 'Total Salary Locally']
fig = px.pie(
    names = labels,
    values = sizes,
    title='Proportion of Total Salaires Abroad vs. Locally',
    hole = 0.3
)

fig.show()

#Input slicers
first_name = input(f"Enter First Name (or press Enter to skip):
{list(data['FirstName'].unique())}\n")
last_name = input(f"Enter Last Name (or press Enter to skip):
{list(data['LastName'].unique())}\n")
create_dashboard(first_name if first_name else None, last_name if last_name else None)
```

## Creating dashboards with ipwidgets:

```
import pandas as pd
import plotly.express as px
import ipywidgets as widgets
from IPython.display import display

data = pd.read_csv('data.csv')

# Create full employee name column for display
data['Employee'] = data['FirstName'] + " " + data['LastName']
```

```
# Function to filter data based on dropdown selections
def filter_data(df, first_name=None, last_name=None):
    filtered_data = df
    if first_name:
        filtered_data = filtered_data[filtered_data['FirstName'] == first_name]
    if last_name:
        filtered_data = filtered_data[filtered_data['LastName'] == last_name]
    return filtered_data

# Function to create a dynamic dashboard
def create_dashboard(first_name=None, last_name=None):
    filtered_data = filter_data(data, first_name, last_name)

    # Bar Chart: Comparison of Salaries Abroad vs Locally
    fig_bar = px.bar(
        filtered_data,
        x='Employee',
        y=['Salary Abroad', 'Salary Locally'],
        barmode='group',
        title='Comparison of Salaries Abroad vs Locally',
        labels={'value': 'Salary', 'variable': 'Type'}
    )

    # Line Chart: Salaries Abroad and Locally
    fig_line = px.line(
        filtered_data,
        x='Employee',
        y=['Salary Abroad', 'Salary Locally'],
        title='Trend of Salaries Abroad vs Locally',
        labels={'value': 'Salary', 'variable': 'Type'}
    )

    # Pie Chart: Proportion of Total Salaries Abroad vs Locally
    total_salary_abroad = filtered_data['Salary Abroad'].sum()
    total_salary_local = filtered_data['Salary Locally'].sum()
    sizes = [total_salary_abroad, total_salary_local]
    labels = ['Total Salary Abroad', 'Total Salary Locally']
    fig_pie = px.pie(
        values=sizes,
        names=labels,
        title='Proportion of Total Salaries Abroad vs Locally'
    )
    # Display the figures
    fig_bar.show()
    fig_line.show()
    fig_pie.show()

# Unique dropdown options for First and Last names
first_names = [None] + list(data['FirstName'].unique())
last_names = [None] + list(data['LastName'].unique())

# Create dropdowns
first_name_dropdown = widgets.Dropdown(
    options=first_names,
    description='First Name:'
)

last_name_dropdown = widgets.Dropdown(
    options=last_names,
    description='Last Name:'
)

# Button to update the dashboard
update_button = widgets.Button(description="Update Dashboard")

# Function to handle button click
def on_button_click(b):
    create_dashboard(
        first_name=first_name_dropdown.value,
        last_name=last_name_dropdown.value
    )

update_button.on_click(on_button_click)

# Display dropdowns and button
display(first_name_dropdown, last_name_dropdown, update_button)

# Display initial dashboard with all data
create_dashboard()
```

## Map Dashboards:

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

```
from ipywidgets import widgets
from IPython.display import display

# Load data from Excel
data = pd.read_csv('data.csv')

# Prepare the data
data['Employee'] = data['FirstName'] + " " + data['LastName']

# Add some sample geolocation coordinates (replace these with actual coordinates if available)
data['Coordinates'] = data['Philippine Office'].map({
    'Makati': (14.5547, 121.0244),
    'Taguig': (14.5176, 121.0509),
    'Cebu': (10.3157, 123.8854),
    'Davao': (7.1907, 125.4553)
})

# Replace missing coordinates with default values
data['Coordinates'] = data['Coordinates'].apply(lambda x: x if pd.notnull(x) else (0, 0))

# Extract latitude and longitude from Coordinates
data[['Latitude', 'Longitude']] = pd.DataFrame(data['Coordinates'].tolist(), index=data.index)

# Function to filter data based on dropdown selections
def filter_data(df, first_name=None, last_name=None):
    filtered_data = df
    if first_name:
        filtered_data = filtered_data[filtered_data['FirstName'] == first_name]
    if last_name:
        filtered_data = filtered_data[filtered_data['LastName'] == last_name]
    return filtered_data

# Function to create a dashboard
def create_dashboard(first_name=None, last_name=None):
    filtered_data = filter_data(data, first_name, last_name)

    if filtered_data.empty:
        print("No data available for the selected filters.")
        return

    # Cards: Summary Statistics
    total_salary_abroad = filtered_data['Salary Abroad'].sum()
    total_salary_local = filtered_data['Salary Locally'].sum()
    num_employees = len(filtered_data)
```

```
fig_cards = go.Figure()
fig_cards.add_trace(go.Indicator(
    mode="number",
    value=num_employees,
    title="Number of Employees",
    domain={'x': [0, 0.33], 'y': [0, 1]}
))
fig_cards.add_trace(go.Indicator(
    mode="number",
    value=total_salary_abroad,
    title="Total Salary Abroad",
    domain={'x': [0.33, 0.66], 'y': [0, 1]}
))
fig_cards.add_trace(go.Indicator(
    mode="number",
    value=total_salary_local,
    title="Total Salary Locally",
    domain={'x': [0.66, 1], 'y': [0, 1]}
))
fig_cards.update_layout(title="Summary Cards", height=250)
fig_cards.show()

# World Map: Plot Locations
fig_map = px.scatter_geo(
    filtered_data,
    lat="Latitude",
    lon="Longitude",
    text="Employee",
    title="Employee Locations",
    projection="natural earth"
)
fig_map.update_traces(marker=dict(size=10, color="blue"))
fig_map.show()

# Bar Chart: Salaries Comparison
fig_bar = px.bar(
    filtered_data,
    x="Employee",
    y=["Salary Abroad", "Salary Locally"],
    barmode="group",
    title="Comparison of Salaries Abroad vs Locally",
    labels={"value": "Salary", "variable": "Type"}
)
```

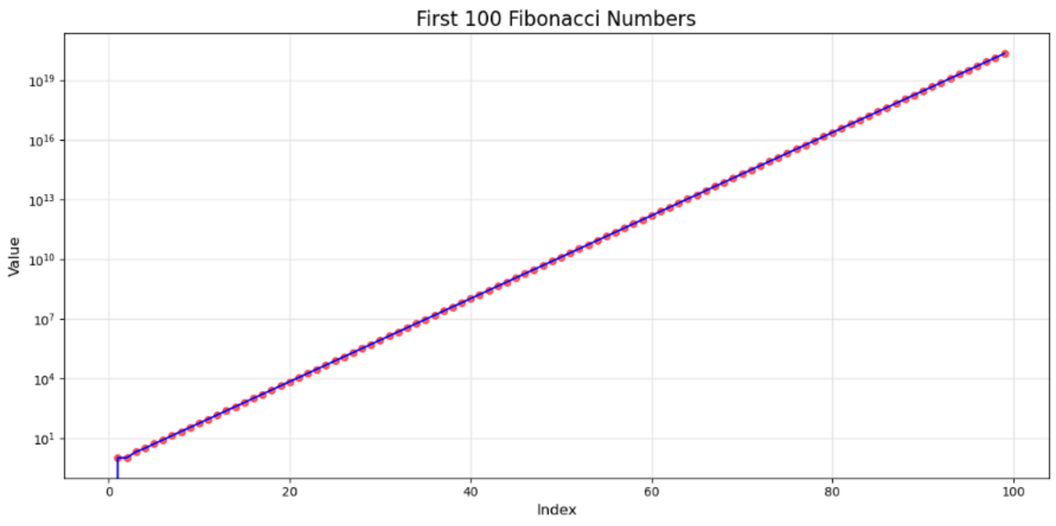
```
    fig_bar.show()

    # Line Chart: Salary Trends
    fig_line = px.line(
        filtered_data,
        x="Employee",
        y=["Salary Abroad", "Salary Locally"],
        title="Trend of Salaries Abroad vs Locally",
        labels={"value": "Salary", "variable": "Type"}
    )
    fig_line.show()

    # Pie Chart: Salary Proportions
    fig_pie = px.pie(
        names=["Total Salary Abroad", "Total Salary Locally"],
        values=[total_salary_abroad, total_salary_local],
        title="Proportion of Total Salaries Abroad vs Locally"
    )
    fig_pie.show()

# Unique dropdown options for first and last names
first_names = [None] + list(data['FirstName'].unique())
last_names = [None] + list(data['LastName'].unique())

# Dropdown-based filters
first_name_dropdown = widgets.Dropdown(
    options=first_names,
    description="First Name"
)

last_name_dropdown = widgets.Dropdown(
    options=last_names,
    description="Last Name"
)

# Button to update the dashboard
update_button = widgets.Button(description="Update Dashboard")

# Function to handle button click
def on_button_click(b):
    create_dashboard(
        first_name=first_name_dropdown.value,
        last_name=last_name_dropdown.value
    )
```

update_button.on_click(on_button_click)

# Display dropdowns and button
display(first_name_dropdown, last_name_dropdown, update_button)

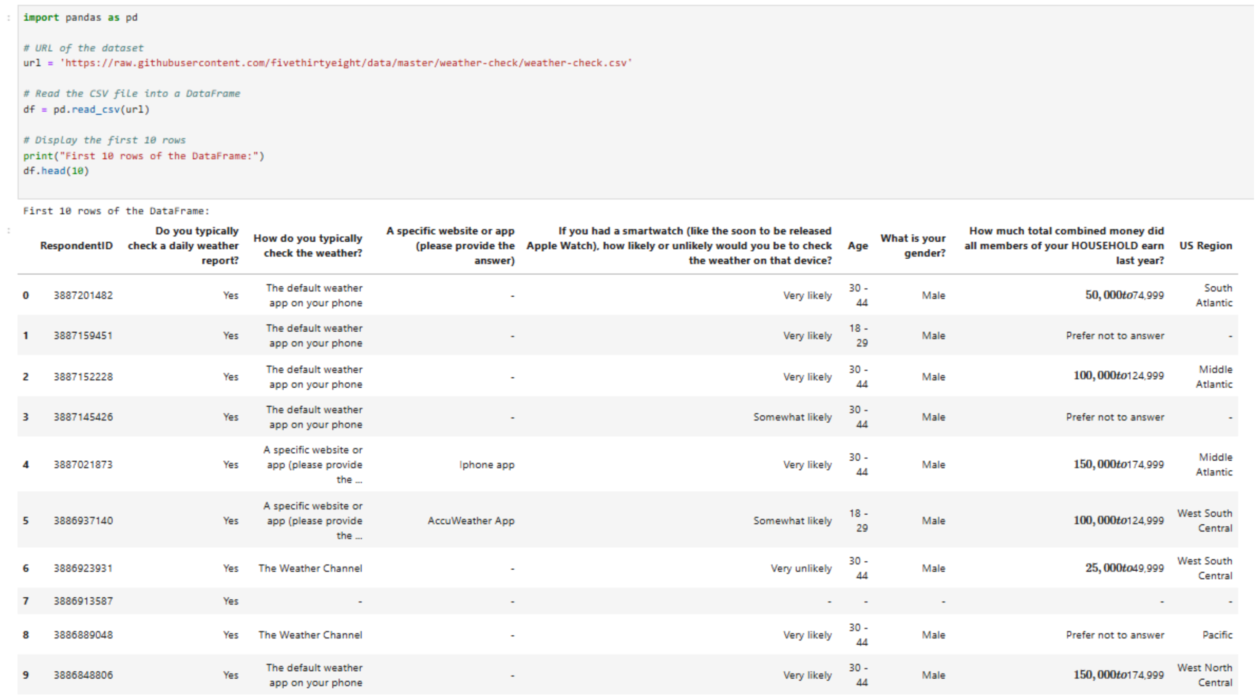
## Fibonacci Numbers

```
import numpy as np
import matplotlib.pyplot as plt

# Function to generate Fibonacci sequence
def fibonacci(n):
    fib_sequence = [0, 1]
    for i in range(2, n):
        fib_sequence.append(fib_sequence[i-1] + fib_sequence[i-2])
    return fib_sequence

# Generate first 100 Fibonacci numbers
n = 100
fib_numbers = fibonacci(n)

# Create the plot
plt.figure(figsize=(12, 6))
plt.plot(range(n), fib_numbers, 'b-', linewidth=1.5)
plt.scatter(range(n), fib_numbers, c='red', s=30, alpha=0.6)
plt.title('First 100 Fibonacci Numbers', fontsize=16)
plt.xlabel('Index', fontsize=12)
plt.ylabel('Value', fontsize=12)
plt.grid(True, alpha=0.3)

# Add a log scale y-axis to better visualize the exponential growth
plt.yscale('log')
plt.tight_layout()

# Show the plot
plt.show()

# Print the first few and last few numbers for reference
print(f"First 10 Fibonacci numbers: {fib_numbers[:10]}")
print(f"Last 5 Fibonacci numbers: {fib_numbers[-5:]}")
```



```
First 10 Fibonacci numbers: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
Last 5 Fibonacci numbers: [31940434634990099905, 51680708854858323072, 83621143489848422977, 135301852344706746049, 218922995834555169026]
```

## URL dataset:



```python
import pandas as pd

# URL of the dataset
url = 'https://raw.githubusercontent.com/fivethirtyeight/data/master/weather-check/weather-check.csv'

# Read the CSV file into a DataFrame
df = pd.read_csv(url)

# Display the first 10 rows
print("First 10 rows of the DataFrame:")
df.head(10)
```

```
First 10 rows of the DataFrame:
```

| | RespondentID | Do you typically check a daily weather report? | How do you typically check the weather? | A specific website or app (please provide the answer) | If you had a smartwatch (like the soon to be released Apple Watch), how likely or unlikely would you be to check the weather on that device? | Age | What is your gender? | How much total combined money did all members of your HOUSEHOLD earn last year? | US Region |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 3887201482 | Yes | The default weather app on your phone | - | Very likely | 30 - 44 | Male | $50,000to$74,999 | South Atlantic |
| **1** | 3887159451 | Yes | The default weather app on your phone | - | Very likely | 18 - 29 | Male | Prefer not to answer | - |
| **2** | 3887152228 | Yes | The default weather app on your phone | - | Very likely | 30 - 44 | Male | $100,000to$124,999 | Middle Atlantic |
| **3** | 3887145426 | Yes | The default weather app on your phone | - | Somewhat likely | 30 - 44 | Male | Prefer not to answer | - |
| **4** | 3887021873 | Yes | A specific website or app (please provide the ... | Iphone app | Very likely | 30 - 44 | Male | $150,000to$174,999 | Middle Atlantic |
| **5** | 3886937140 | Yes | A specific website or app (please provide the ... | AccuWeather App | Somewhat likely | 18 - 29 | Male | $100,000to$124,999 | West South Central |
| **6** | 3886923931 | Yes | The Weather Channel | - | Very unlikely | 30 - 44 | Male | $25,000to$49,999 | West South Central |
| **7** | 3886913587 | Yes | - | - | - | - | - | - | - |
| **8** | 3886889048 | Yes | The Weather Channel | - | Very likely | 30 - 44 | Male | Prefer not to answer | Pacific |
| **9** | 3886848806 | Yes | The default weather app on your phone | - | Very likely | 30 - 44 | Male | $150,000to$174,999 | West North Central |