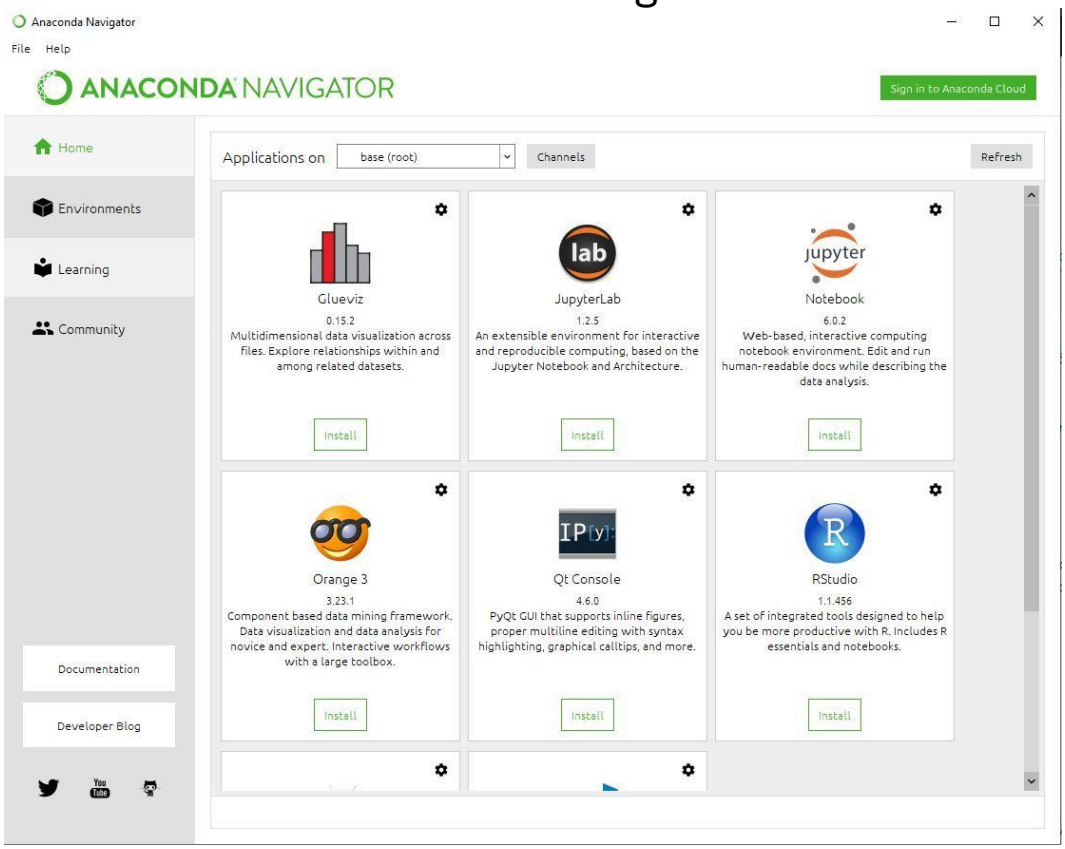
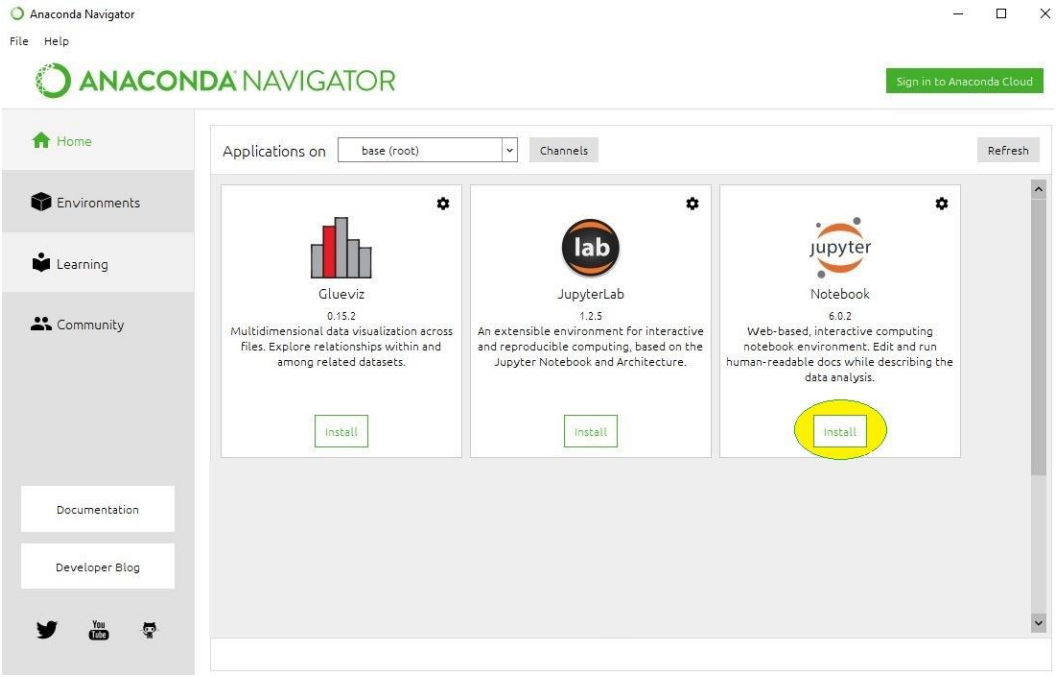
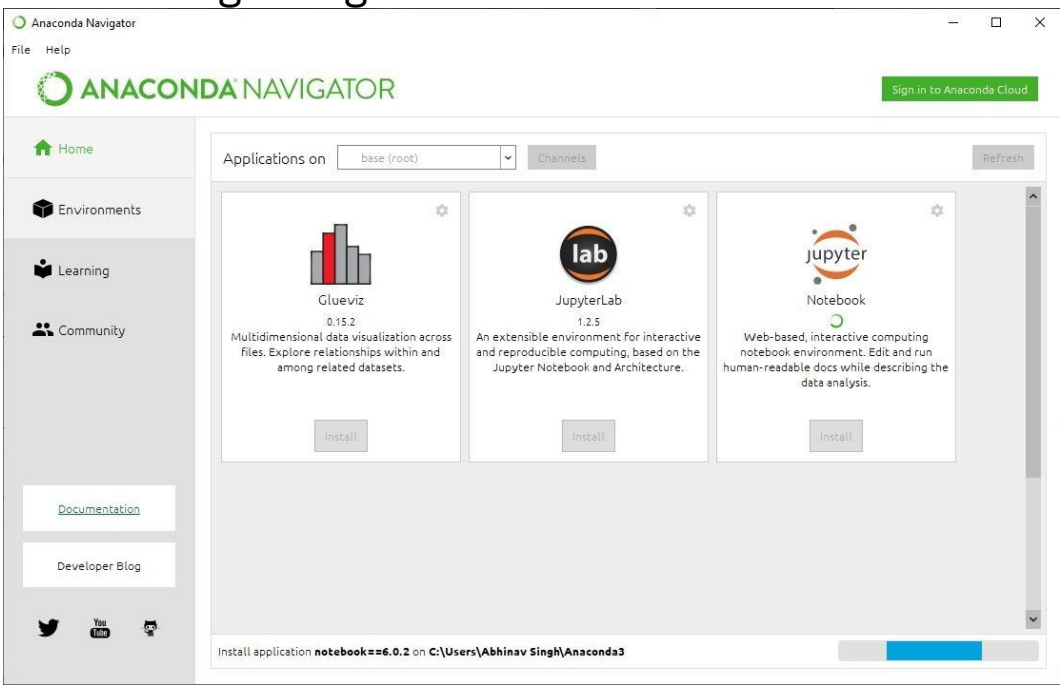
STEP 1: Launch Anaconda Navigator



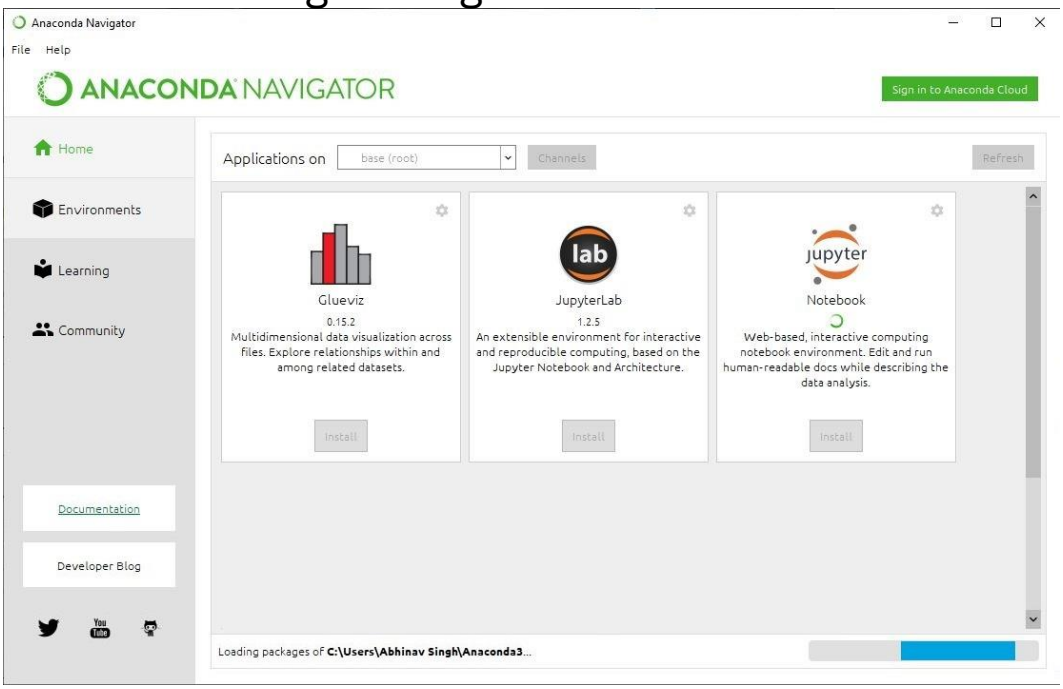
STEP 2: Click on the Install Jupyter Notebook Button:



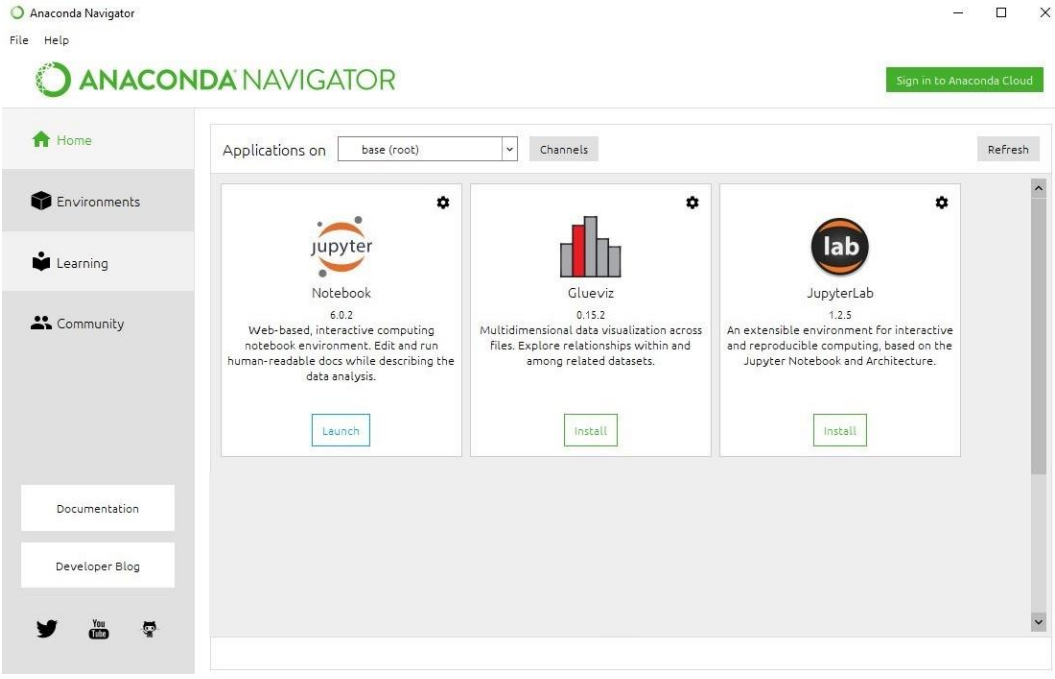
STEP 3: Beginning the Installation



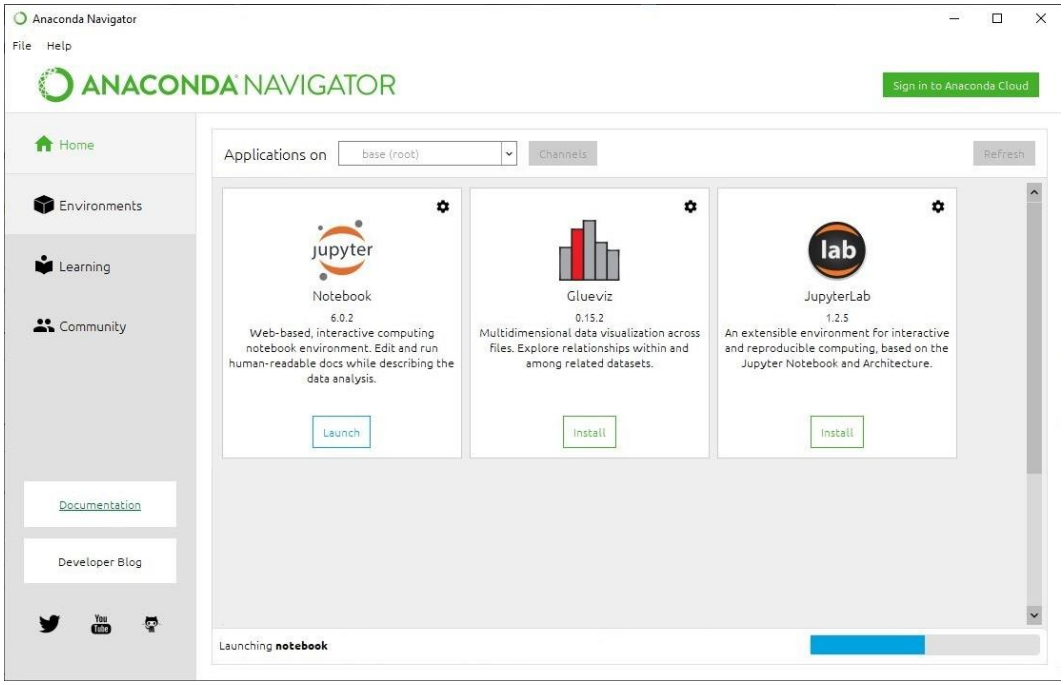
STEP 4: Loading Packages

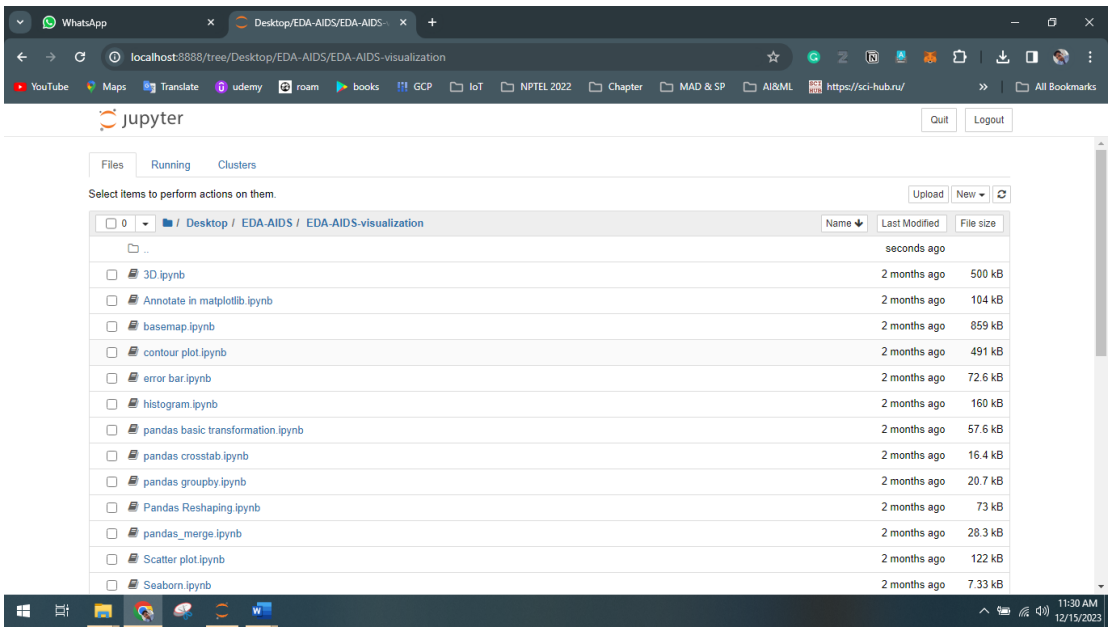


STEP 5: Finished



STEP 5: Launching Jupyter





In [1]:

```
# import numpy:

import numpy as np
```

In [2]:

```
# ndarray object:

arr = np.array([1,2,3,4,5])
print(arr)
print(type(arr))
```

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

In [3]:

```
# ndim and shape:

a = np.array([[[1,2],[3,4]],[[2,1],[4,5]]])
print(a)
print(a.ndim)
print(a.shape)
```

```
[[[1 2]
  [3 4]]

 [[2 1]
  [4 5]]]
3
(2, 2, 2)
```

In [4]:

```
# array indexing:

print(arr[0])
print(arr[0]+arr[2])
print(a[0,1,1])
print(a[-1][-1][-1])
```

```
1
4
4
5
```

In [5]:

```
# array slicing:

print(a[1:4])
print(a[0, 0:1])
```

```
[[[2 1]
  [4 5]]]
[[1 2]]
```

In [6]:

```
# data types:

b = np.array([1,2,3,4],dtype='i4')
print(b)
print(b.dtype)
```

```
[1 2 3 4]
int32
```

In [7]:

```
# type conversion:

arr = b.astype(float)
print(arr)
print(arr.dtype)
```

```
[1. 2. 3. 4.]
float64
```

In [8]:

```
# copy and view:

x = arr.copy()
y = arr.view()
print(x.base)
print(y.base)
```

```
None
[1. 2. 3. 4.]
```

In [9]:

```
# reshaping ndarray:

print(a)
arr = a.reshape(4,2)
print(arr)
print(arr.shape)
```

```
[[[1 2]
  [3 4]]

 [[2 1]
  [4 5]]]
[[1 2]
 [3 4]
 [2 1]
 [4 5]]
(4, 2)
```

In [10]:

```
# itterate ndarray:

for x in np.nditer(a):
    print(x,end=' ')
```

```
1 2 3 4 2 1 4 5
```

In [11]:

```
# enumerate ndarray

for idx, x in np.ndenumerate(a):
    print(idx,x)
```

```
(0, 0, 0) 1
(0, 0, 1) 2
(0, 1, 0) 3
(0, 1, 1) 4
(1, 0, 0) 2
(1, 0, 1) 1
(1, 1, 0) 4
(1, 1, 1) 5
```

In [12]:

```
# import pandas:

import pandas as pd
```

In [13]:

```
# pandas series:

a = pd.Series([1,7,2])
print(a)
```

```
0    1
1    7
2    2
dtype: int64
```

In [14]:

```
# set custom index:

a.index = ['x', 'y','z']
print(a.iloc[0])
print(a['x'])
```

```
1
1
```

In [15]:

```
# pandas dataframe:

data1 = {
    "Name": ["Alex","Allson"],
    "Age": [18,19],
}
df1 = pd.DataFrame(data1)
print(df1)
```

```
     Name  Age
0    Alex   18
1  Allson   19
```

In [23]:

```
# locate:

print(df1.iloc[0])
print(df1.iloc[1,0])
```

```
Name    Alex
Age       18
Name: s1, dtype: object
Allson
```

In [17]:

```
# set custom index:

df1.index = ['s1','s2']
print(df1)
```

```
      Name  Age
s1    Alex   18
s2  Allson   19
```

In [18]:

```
# read from csv or json file:

df = pd.read_csv('./dataset/temp.csv')
print(df)
df = pd.read_json('./dataset/temp.json')
print(df)
```

```
     Name  Age
0    Alex   18
1  Allson   19
     Name  Age
0    Alex   18
1  Allson   19
```

In [19]:

```
# describe data:

print(df1.describe())
```

```
             Age
count   2.000000
mean   18.500000
std     0.707107
min    18.000000
25%    18.250000
50%    18.500000
75%    18.750000
max    19.000000
```

In [20]:

```
# adding new data:

df1.loc[len(df1)] = ["Guru", 18]
print(df1)
```

```
      Name  Age
s1    Alex   18
s2  Allson   19
2     Guru   18
```

In [21]:

```
# insert column:

df1.insert(2,"Number",a.tolist())
print(df1)
```

```
      Name  Age  Number
s1    Alex   18       1
s2  Allson   19       7
2     Guru   18       2
```

In [22]:

```
# drop column:

df1 = df1.drop("Number",axis=1)
print(df1)
```

```
      Name  Age
s1    Alex   18
s2  Allson   19
2     Guru   18
```

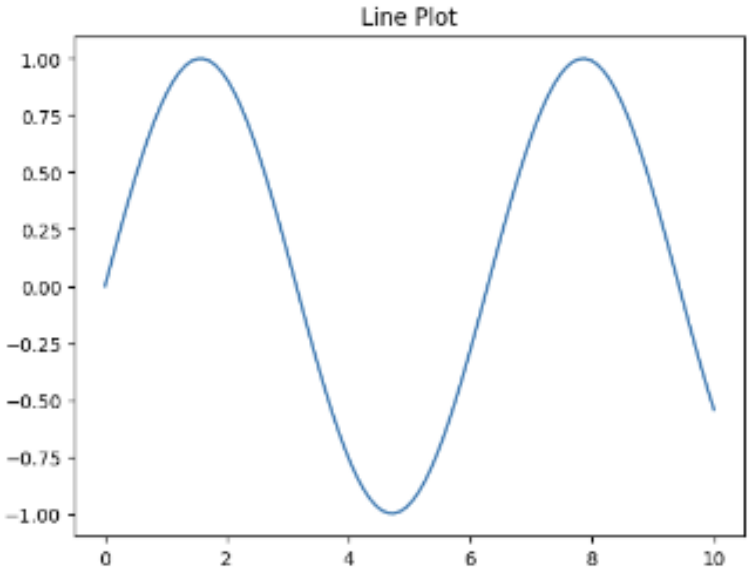
In [1]:

```
# import matplotlib:

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

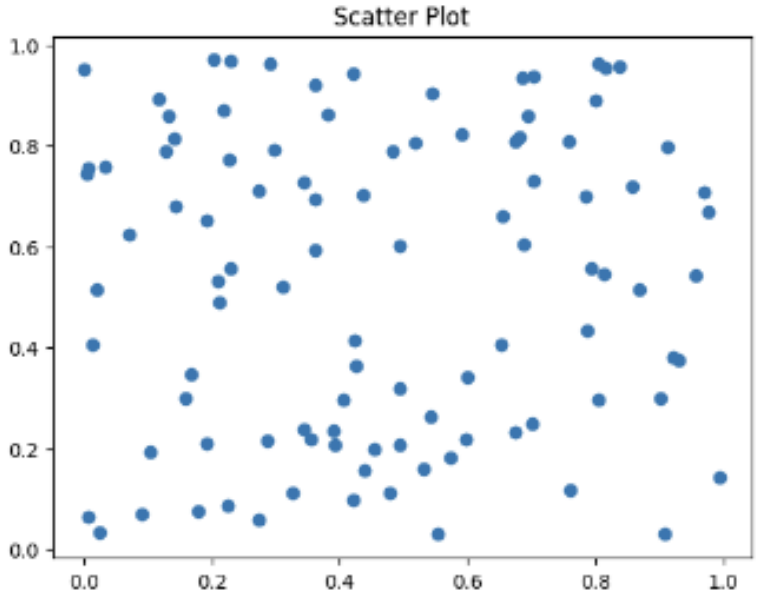
In [2]:

```
# Line Plot
x = np.linspace(0, 10, 100)
y = np.sin(x)
plt.plot(x, y)
plt.title("Line Plot")
plt.show()
```



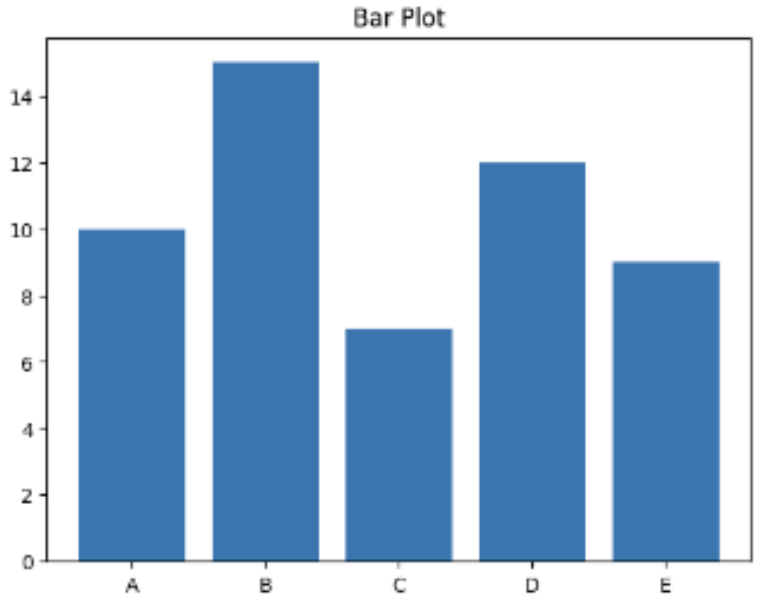
In [3]:

```
# Scatter Plot
x = np.random.rand(100)
y = np.random.rand(100)
plt.scatter(x, y)
plt.title("Scatter Plot")
plt.show()
```



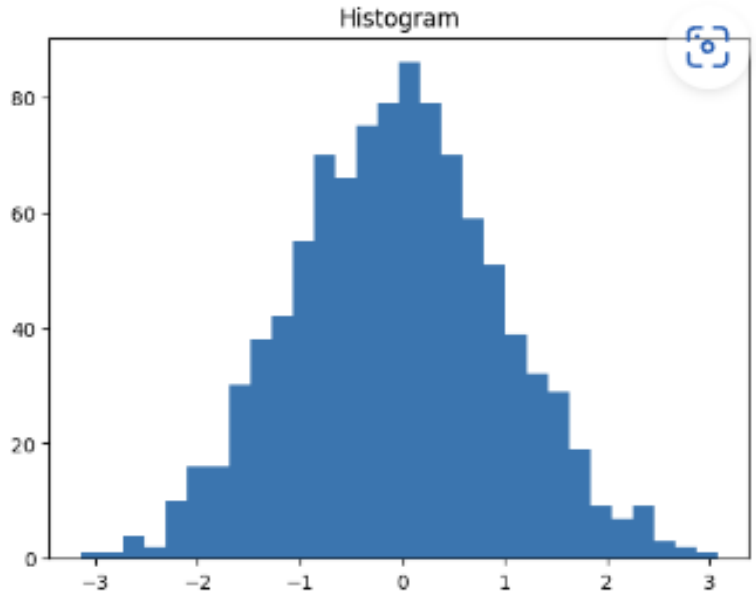
In [4]:

```
# Bar Plot
x = ["A", "B", "C", "D", "E"]
y = [10, 15, 7, 12, 9]
plt.bar(x, y)
plt.title("Bar Plot")
plt.show()
```



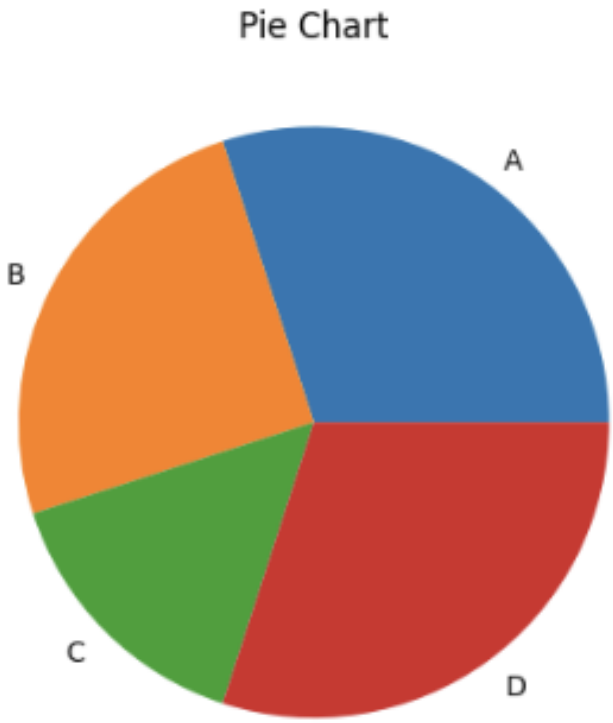
In [5]:

```
# Histogram
data = np.random.randn(1000)
plt.hist(data, bins=30)
plt.title("Histogram")
plt.show()
```



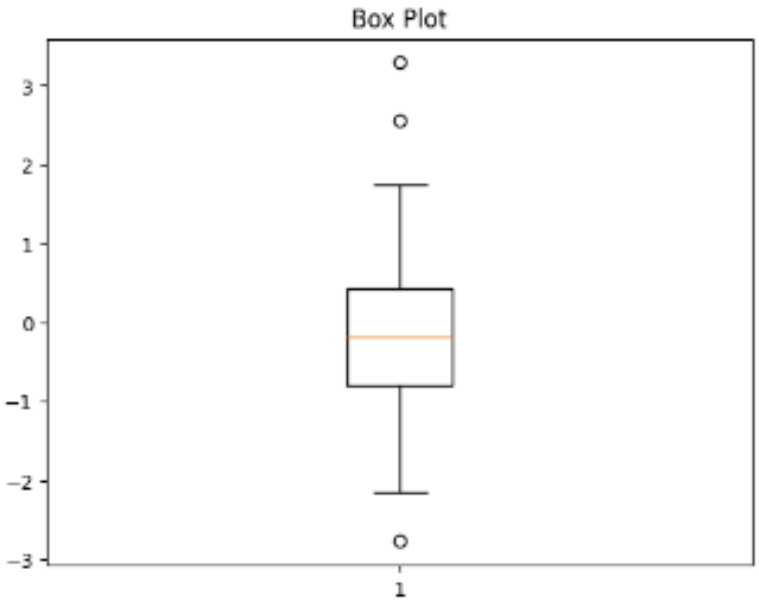
In [6]:

```
# Pie Chart
labels = ["A", "B", "C", "D"]
sizes = [30, 25, 15, 30]
plt.pie(sizes, labels=labels)
plt.title("Pie Chart")
plt.show()
```



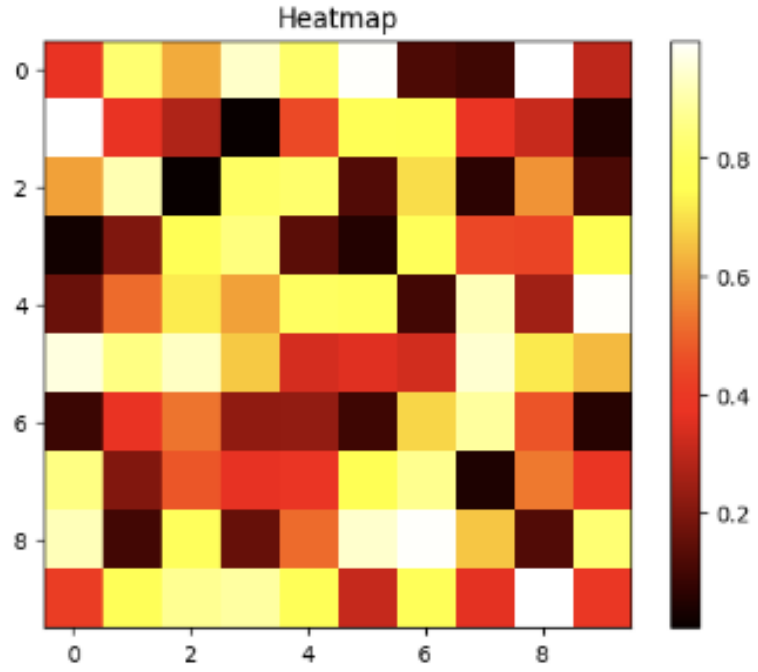
In [7]:

```
# Box Plot
data = np.random.randn(100)
plt.boxplot(data)
plt.title("Box Plot")
plt.show()
```



In [8]:

```
# Heatmap
data = np.random.rand(10, 10)
plt.imshow(data, cmap="hot")
plt.colorbar()
plt.title("Heatmap")
plt.show()
```



In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
from statsmodels.tsa.seasonal import seasonal_deco
mpose
from statsmodels.graphics.tsaplots import plot_acf
```

In [3]:

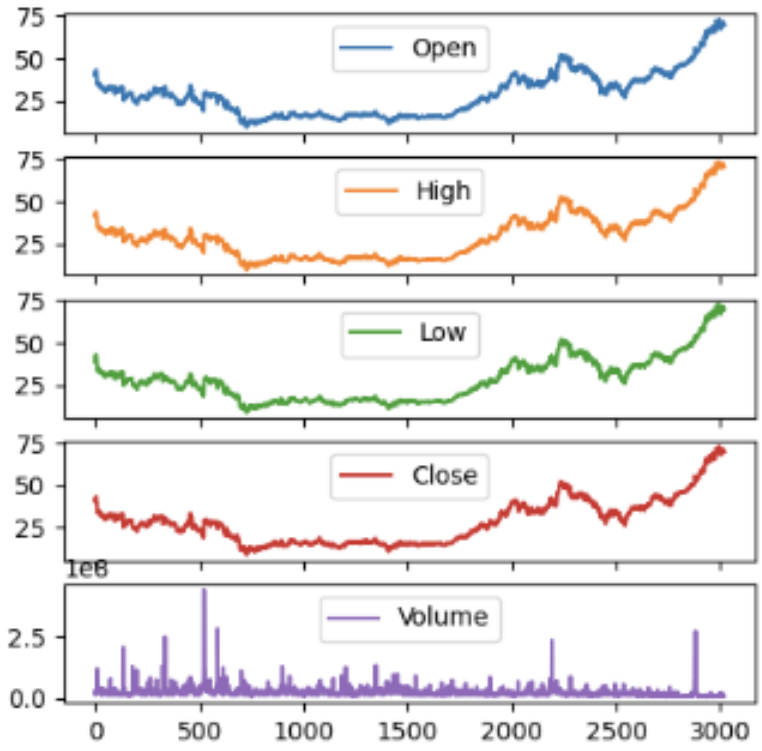
```
url = './dataset/stock_data.csv'
df = pd.read_csv(url, parse_dates=True)
df.drop(columns=['Unnamed: 0','Name'], inplace=Tru
e)
df.head()
```

Out[3]:

	Date	Open	High	Low	Close	Volun
0	1/3/2006	39.69	41.22	38.79	40.91	242327
1	1/4/2006	41.22	41.90	40.77	40.97	205534
2	1/5/2006	40.93	41.73	40.85	41.53	128296
3	1/6/2006	42.88	43.57	42.80	43.21	294228
4	1/9/2006	43.10	43.66	42.82	43.42	162683

In [4]:

```
df.plot(subplots=True, figsize=(5,5))
plt.show()
```

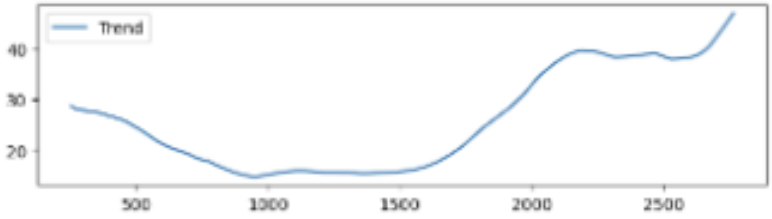


In [5]:

```
# decomposition
close = seasonal_decompose(df['Close'], model='mul
tiplicative', period = 500)
trend = close.trend
seasonal = close.seasonal
residual = close.resid
```

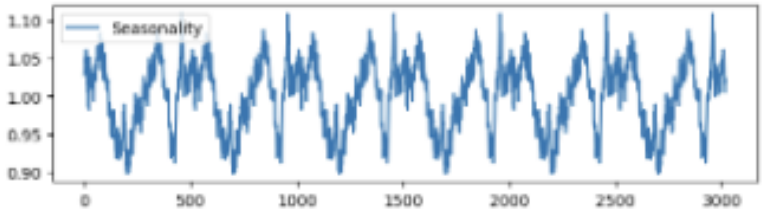
In [6]:

```
# trend analysis
plt.figure(figsize=(8,2))
plt.plot(trend, label='Trend')
plt.legend(loc='best')
plt.show()
```



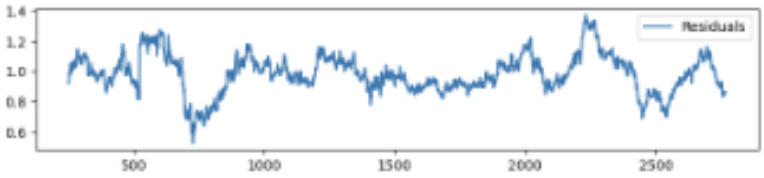
In [7]:

```
# seasonality analysis
plt.figure(figsize=(8,2))
plt.plot(seasonal,label='Seasonality')
plt.legend(loc='best')
plt.show()
```



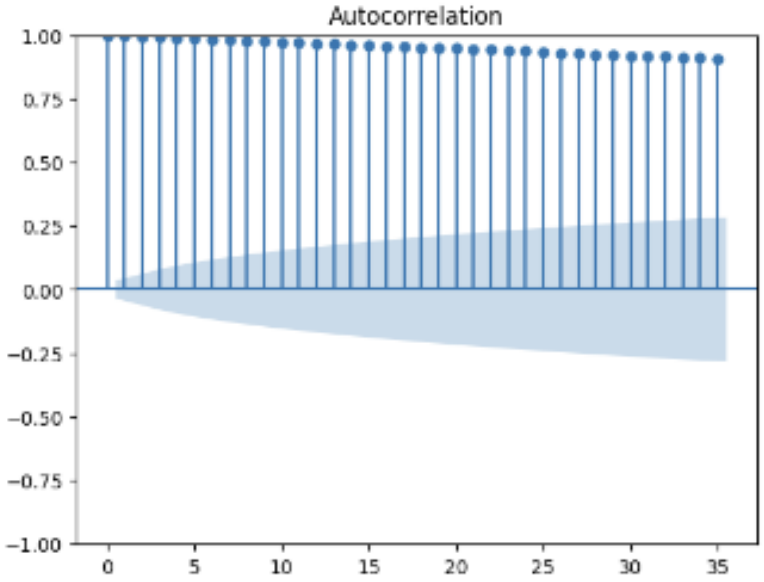
In [8]:

```
# residuals
plt.figure(figsize=(8,2))
plt.plot(residual, label='Residuals')
plt.legend(loc='best')
plt.tight_layout()
plt.show()
```



In [9]:

```
# autocorrelation
plot_acf(df['Close'])
plt.show()
```



In [1]:

```
import geopandas as gpd
import matplotlib.pyplot as plt
from shapely.geometry import Point
```

In [2]:

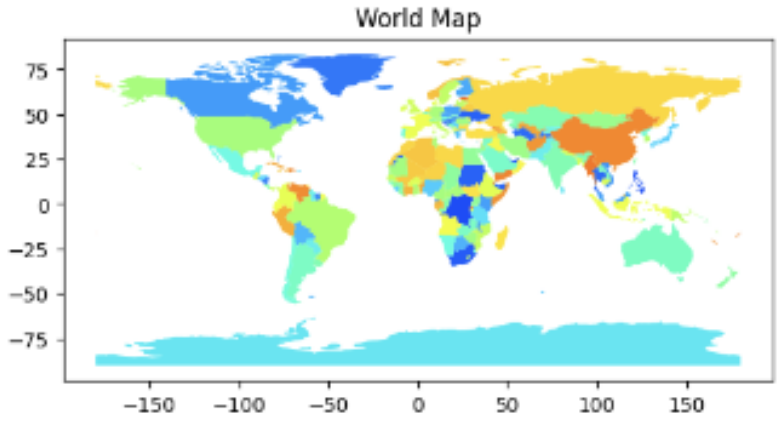
```
world = gpd.read_file("./ShapeFiles/ne_110m_admin_
0_countries.shp")
india = world[world['NAME'] == 'India']
```

In [3]:

```
# Political map:
world.boundary.plot(linewidth=0.4, color="black")
plt.title("World Political Map")
plt.axis('off')
plt.show()
```

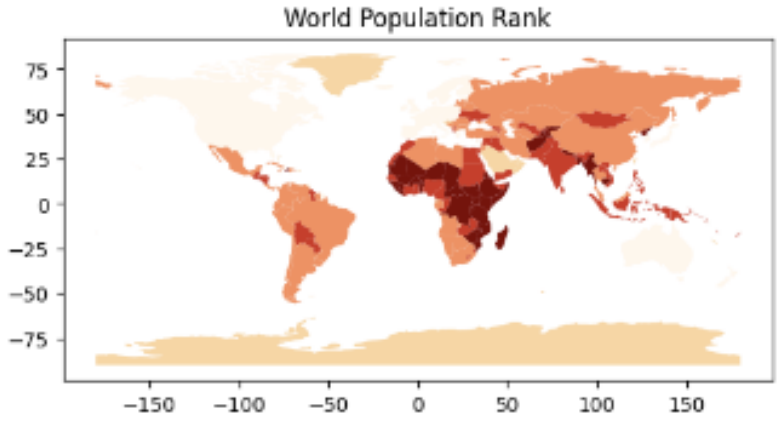
In [4]:

```
import numpy as np
vals = np.linspace(0.2,0.8,256)
np.random.shuffle(vals)
custom_cmap = plt.cm.colors.ListedColormap(plt.cm.
jet(vals))
world.plot(column='NAME', cmap=custom_cmap, legend
=False)
plt.title("World Map")
plt.show()
```



In [5]:

```
world.plot(column='INCOME_GRP', cmap='OrRd', legen
d=0)
plt.title("World Population Rank")
plt.show()
```

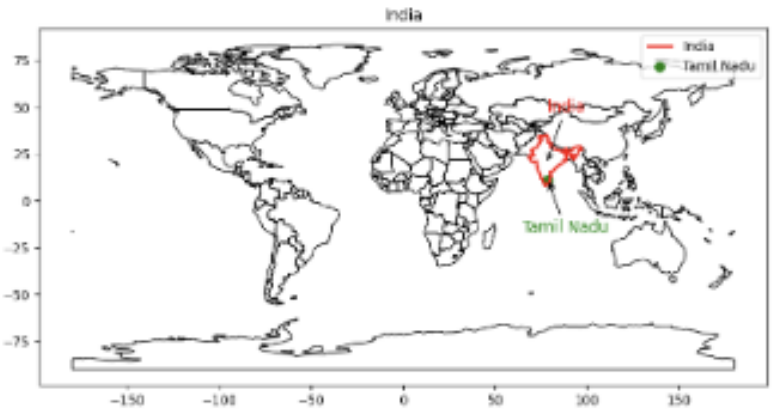


In [6]:

```
fig, ax = plt.subplots(figsize=(10, 10))
world.boundary.plot(ax=ax, linewidth=1, color='k')
india.boundary.plot(ax=ax, linewidth=2, color='r',
label='India')

tn_point = Point(78.6569, 11.1271)
tn = gpd.GeoDataFrame({'geometry': [tn_point]}, cr
s="EPSG:4326")
tn.plot(ax=ax, color='g', marker='o', markersize=5
0, label='Tamil Nadu')

ax.annotate('India', xy=(78, 20), xytext=(0, 40),
textcoords='offset points', arrowprops=dict(arrows
tyle="->", color='k', linewidth=1), color='r', fon
tsize=12)
ax.annotate('Tamil Nadu', xy=(78.6569, 11.1271), x
ytext=(-20, -40), textcoords='offset points', arro
wprops=dict(arrowstyle="->", color='k', linewidth=
1), color='g', fontsize=12)
plt.legend()
plt.title("India")
plt.show()
```



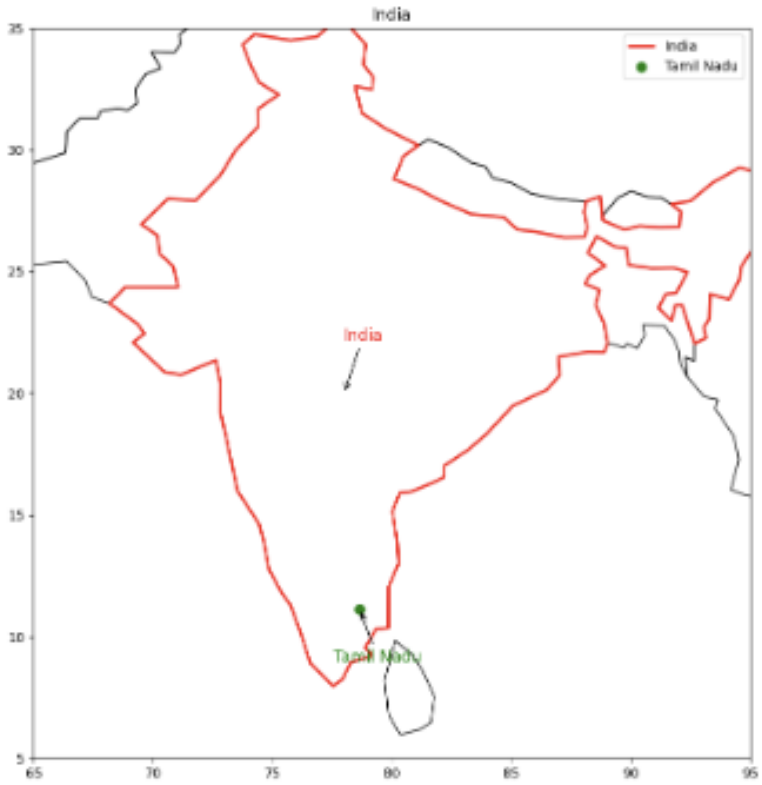
In [7]:

```
fig, ax = plt.subplots(figsize=(10, 10))
world.boundary.plot(ax=ax, linewidth=1, color='k')
india.boundary.plot(ax=ax, linewidth=2, color='r',
label='India')

tn_point = Point(78.6569, 11.1271)
tn = gpd.GeoDataFrame({'geometry': [tn_point]}, cr
s="EPSG:4326")
tn.plot(ax=ax, color='g', marker='o', markersize=5
0, label='Tamil Nadu')

ax.annotate('India', xy=(78, 20), xytext=(0, 40),
textcoords='offset points', arrowprops=dict(arrows
tyle="->", color='k', linewidth=1), color='r', fon
tsize=12)
ax.annotate('Tamil Nadu', xy=(78.6569, 11.1271), x
ytext=(-20, -40), textcoords='offset points', arro
wprops=dict(arrowstyle="->", color='k', linewidth=
1), color='g', fontsize=12)

ax.set_xlim(65, 95)
ax.set_ylim(5, 35)
plt.legend()
plt.title("India")
plt.show()
```



In [1]:

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

In [2]:

```
sns.set_style('darkgrid')
matplotlib.rcParams['font.size'] = 14
matplotlib.rcParams['figure.figsize'] = (10, 6)
matplotlib.rcParams['figure.facecolor'] = '#000000
00'
```

In [3]:

```
df = pd.read_csv("dataset/WineQT.csv")
df.head()
```

Out[3]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	c
0	7.4	0.70	0.00	1.9	0.076	
1	7.8	0.88	0.00	2.6	0.098	
2	7.8	0.76	0.04	2.3	0.092	
3	11.2	0.28	0.56	1.9	0.075	
4	7.4	0.70	0.00	1.9	0.076	

In [5]:

```
# show column names
df.columns
```

Out[5]:

```
Index(['fixed acidity', 'volatile aci
dity', 'citric acid', 'residual suga
r',
       'chlorides', 'free sulfur diox
ide', 'total sulfur dioxide', 'densit
y',
       'pH', 'sulphates', 'alcohol',
'quality', 'Id'],
      dtype='object')
```

In [6]:

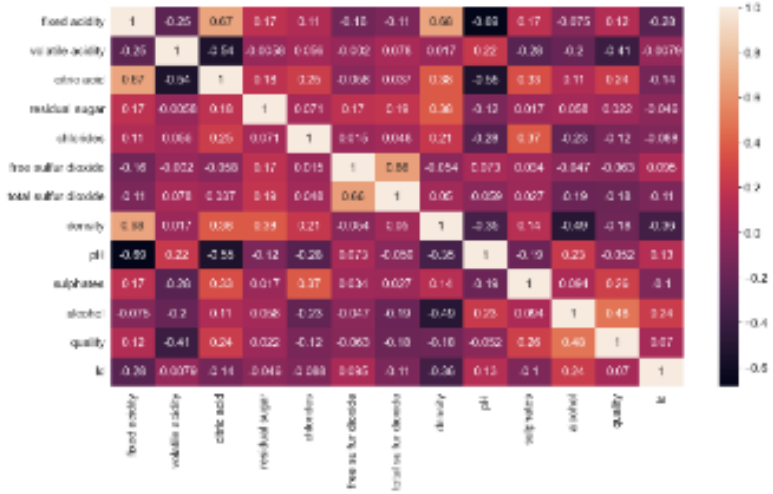
```
df.shape
```

Out[6]:

```
(1143, 13)
```

In [7]:

```
plt.figure(figsize=(15,8))
sns.heatmap(df.corr(),annot = True);
```



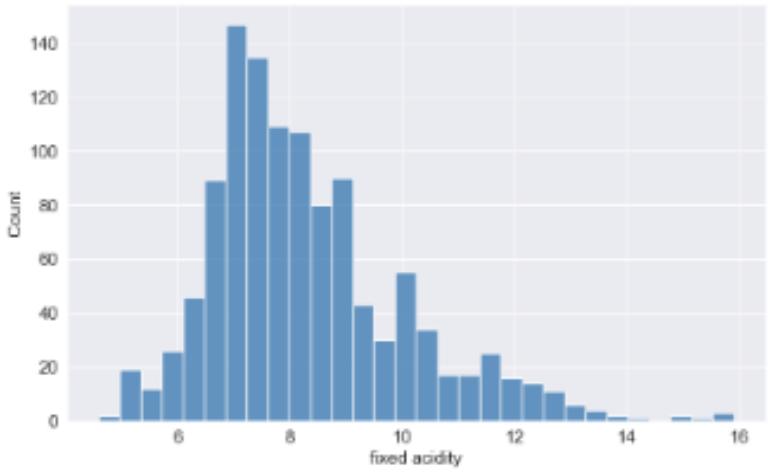
In [8]:

```
# Fixed acidity vs citric acid
sns.scatterplot(data=df, y='fixed acidity', x= 'ci
tric acid', hue= 'pH');
```

In [9]:

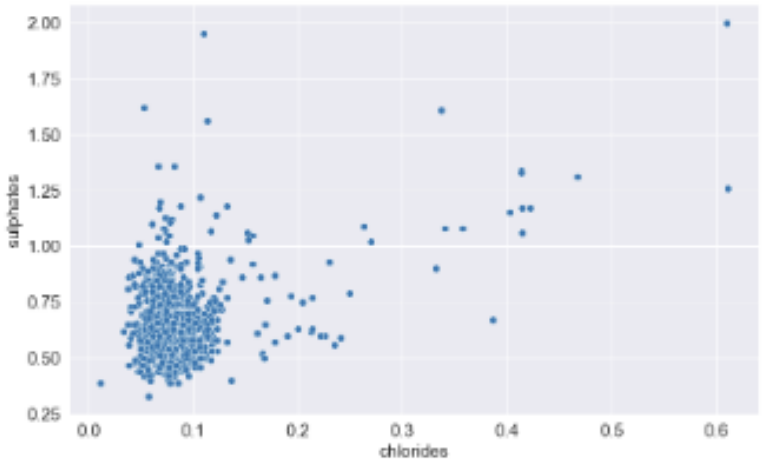
```
median_value = np.median(df['fixed acidity'])
print(f'{median_value} is median value.')
sns.histplot(data=df, x= 'fixed acidity');
```

```
7.9 is median value.
```



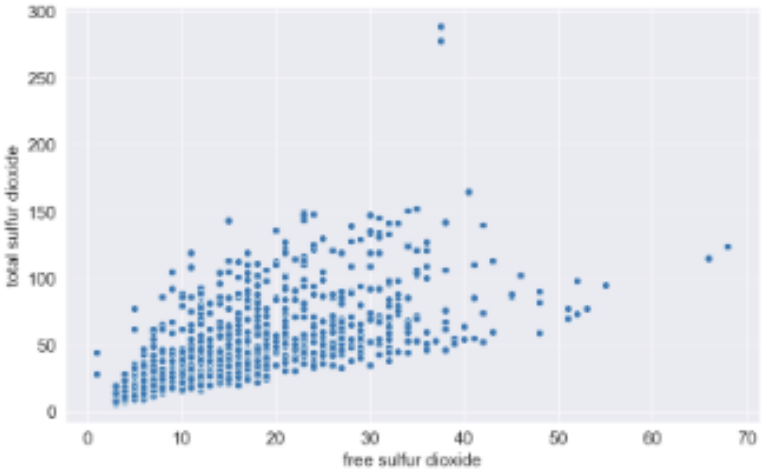
In [10]:

```
# chlorides vs sulphates
sns.scatterplot(data=df,x='chlorides',y='sulphate
s');
```



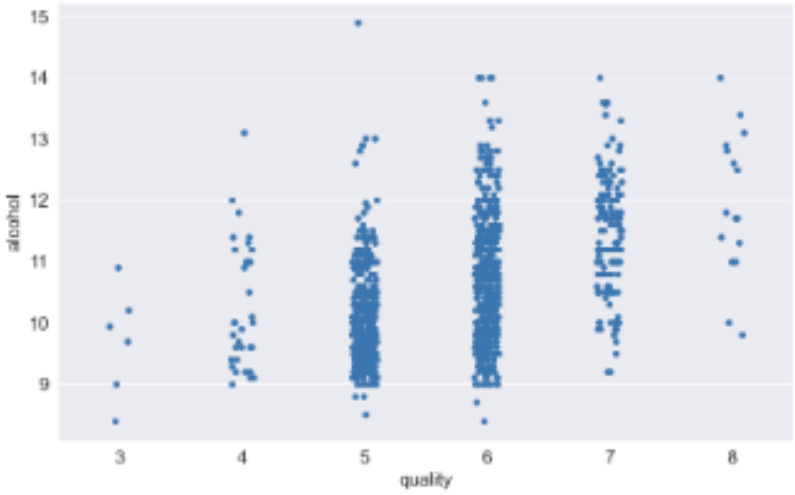
In [11]:

```
# free sulfur dioxide vs total sulfur dioxide
sns.scatterplot(data=df, x='free sulfur dioxide',
y='total sulfur dioxide');
```



In [12]:

```
# alcohol vs quality
sns.stripplot(df,y='alcohol',x='quality');
```



In [1]:

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

In [2]:

```
data = sns.load_dataset('iris')
data.head()
```

Out[2]:

	sepal_length	sepal_width	petal_length	peta
0	5.1	3.5	1.4	
1	4.9	3.0	1.4	
2	4.7	3.2	1.3	
3	4.6	3.1	1.5	
4	5.0	3.6	1.4	

In [3]:

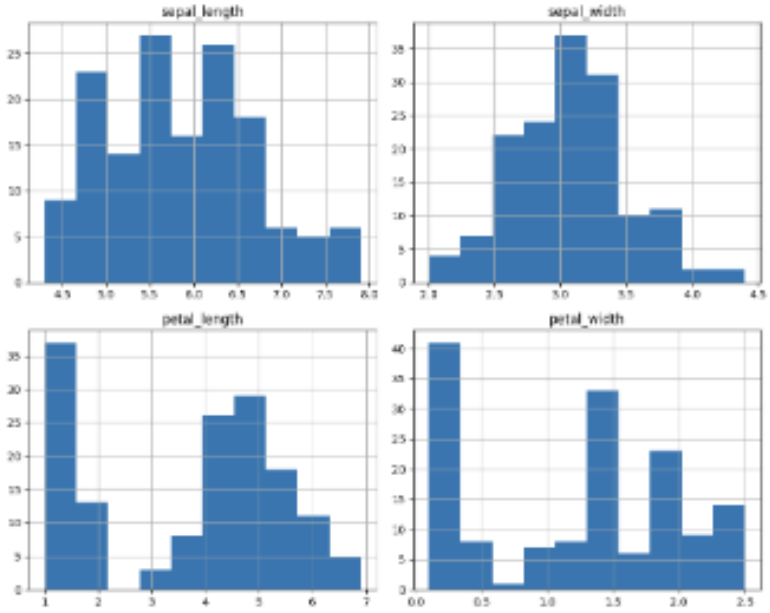
```
data.describe()
```

Out[3]:

	sepal_length	sepal_width	petal_length
count	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000
std	0.828066	0.435866	1.765298
min	4.300000	2.000000	1.000000
25%	5.100000	2.800000	1.600000
50%	5.800000	3.000000	4.350000
75%	6.400000	3.300000	5.100000
max	7.900000	4.400000	6.900000

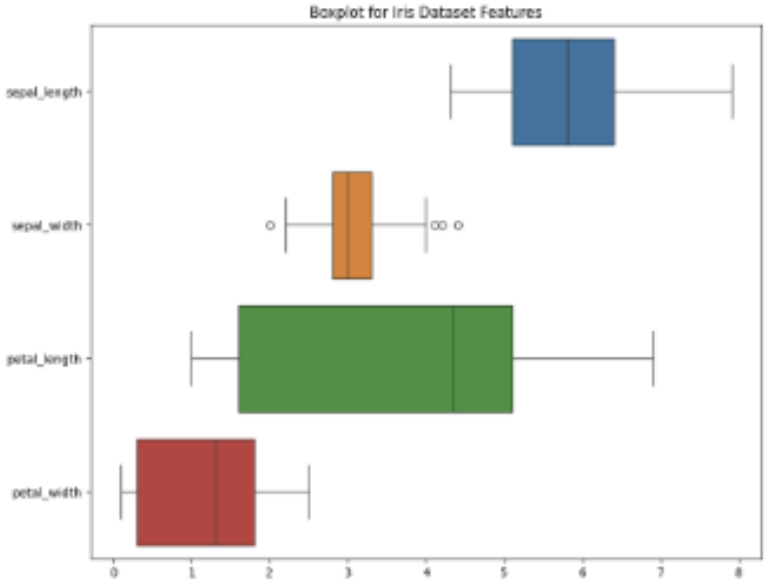
In [4]:

```
data.hist(figsize=(10, 8))
plt.tight_layout()
plt.show()
```



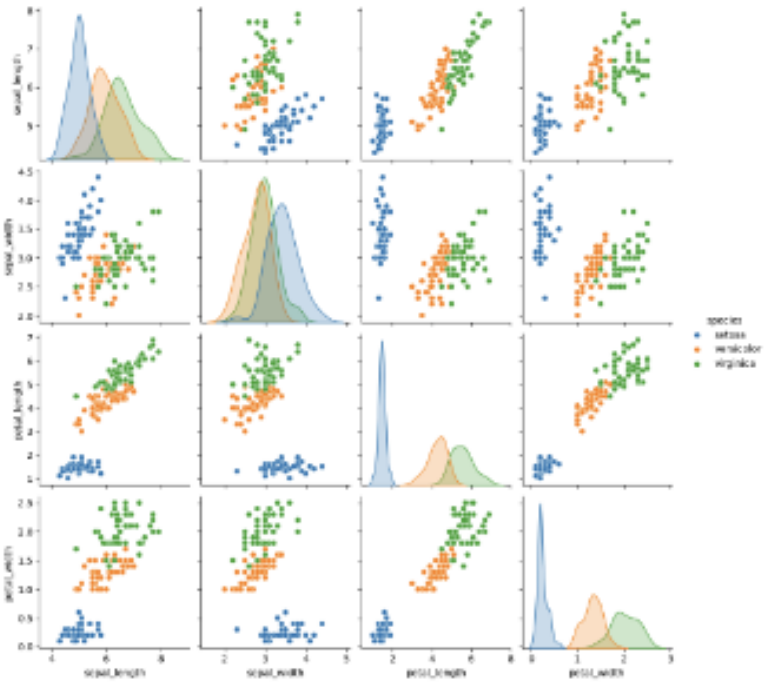
In [5]:

```
plt.figure(figsize=(10, 8))
sns.boxplot(data=data, orient="h")
plt.title("Boxplot for Iris Dataset Features")
plt.show()
```



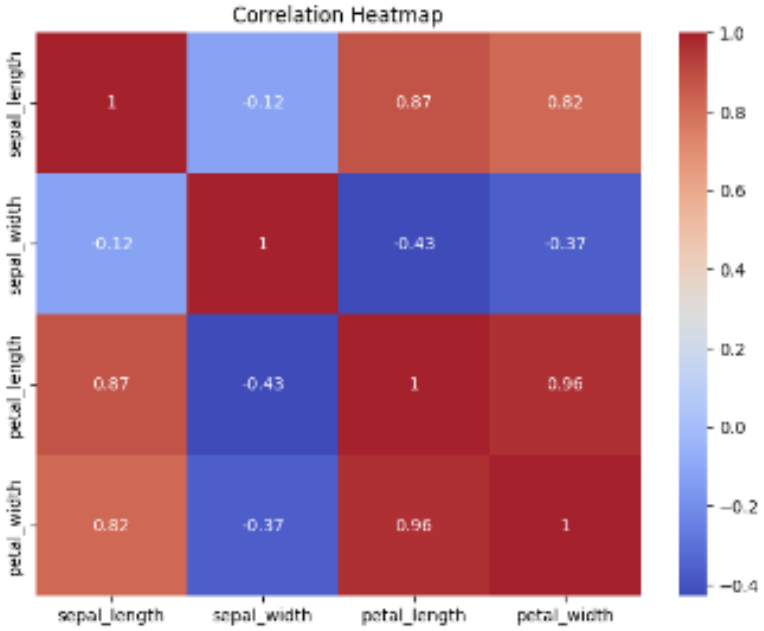
In [6]:

```
import warnings
warnings.filterwarnings('ignore')
sns.pairplot(data, hue="species", height=2.5)
plt.show()
```



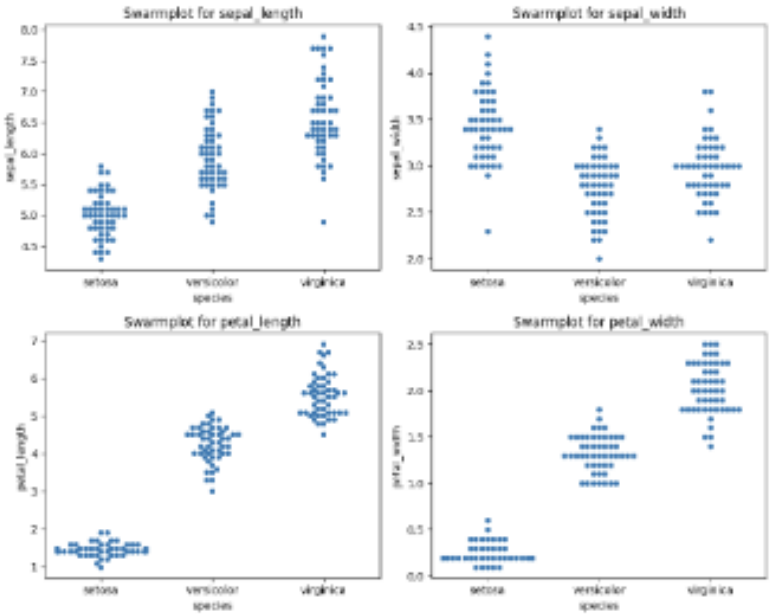
In [7]:

```
correlation = data.select_dtypes(include=[np.numbe
r]).corr()
plt.figure(figsize=(8, 6))
sns.heatmap(correlation, annot=True, cmap='coolwar
m')
plt.title("Correlation Heatmap")
plt.show()
```



In [8]:

```
plt.figure(figsize=(10, 8))
for i, feature in enumerate(data.columns[:-1]):
    plt.subplot(2, 2, i + 1)
    sns.swarmplot(x='species', y=feature, data=dat
a)
    plt.title(f'Swarmplot for {feature}')
plt.tight_layout()
plt.show()
```



In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:

```
df = pd.read_csv("./dataset/email.csv")
df.head()
```

Out[2]:

ead	Date	From	To
v4pl	2023-12-15	harry@google.com	tara@google.com
3wg	2023-12-16	tara@google.com	harry@google.com
v7le	2023-12-17	quinn@google.com	gina@google.com

In [3]:

```
df.describe(include='all')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 8 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   ID       50 non-null     object
 1   Thread   50 non-null     object
 2   Date     50 non-null     object
 3   From     50 non-null     object
 4   To       50 non-null     object
 5   Subject  50 non-null     object
 6   Snippet  50 non-null     object
 7   Labels   50 non-null     object
dtypes: object(8)
memory usage: 3.2+ KB
```

In [4]:

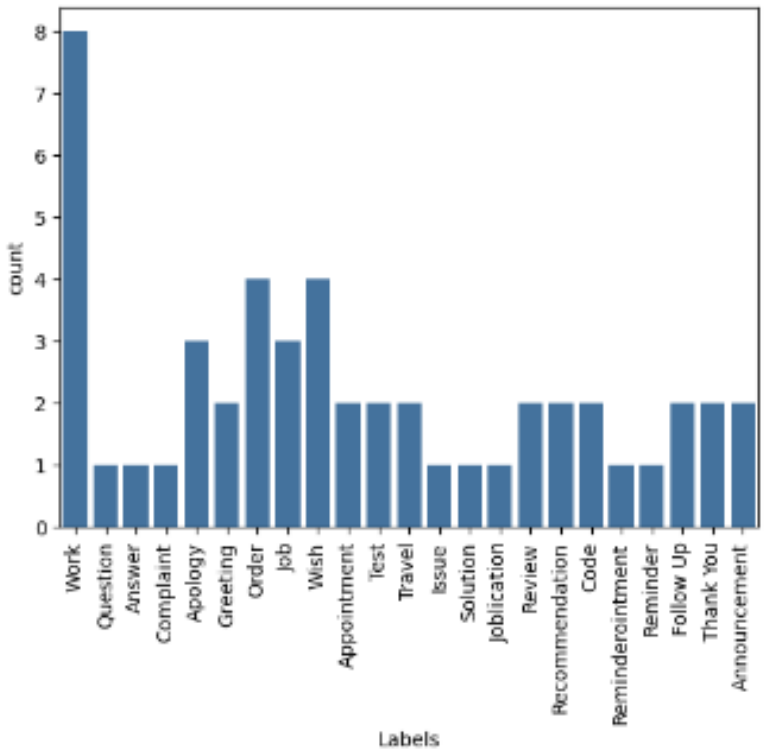
```
df.isnull().sum()
```

Out[4]:

```
ID         0
Thread     0
Date       0
From       0
To         0
Subject    0
Snippet    0
Labels     0
dtype: int64
```

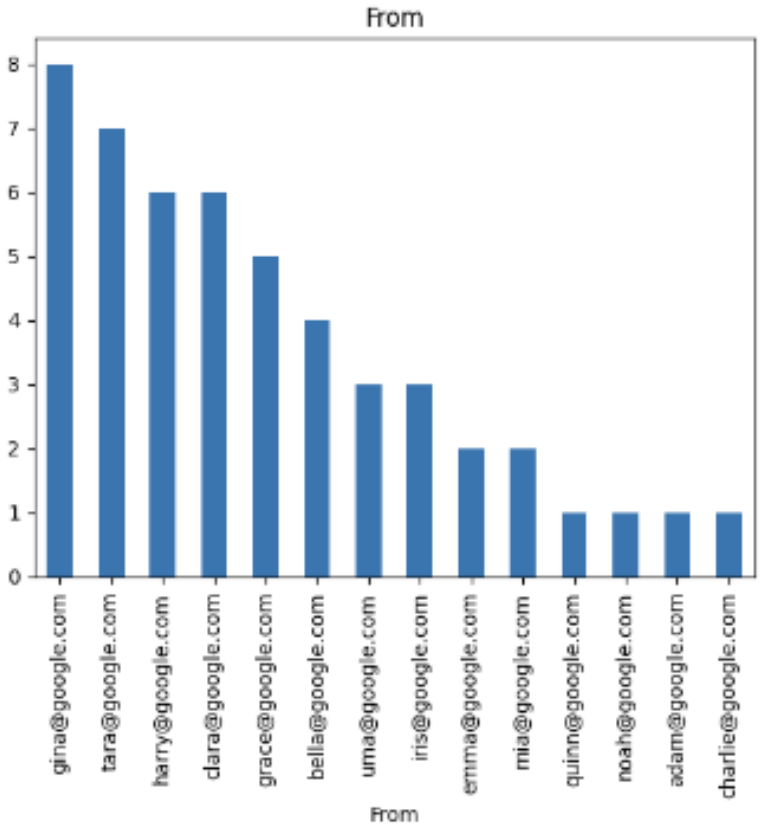
In [5]:

```
sns.countplot(x='Labels', data=df,)
plt.xticks(rotation=90)
plt.show()
```



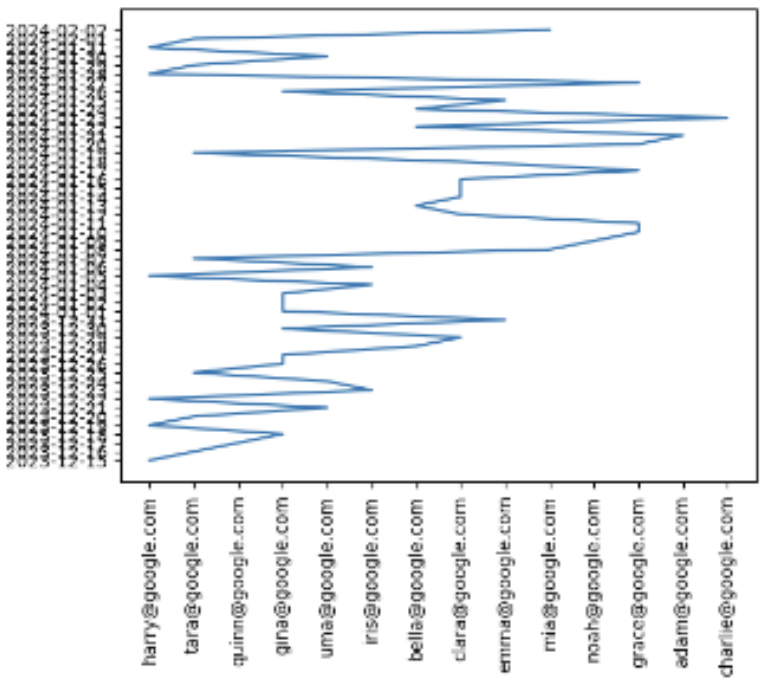
In [6]:

```
df['From'].value_counts().plot(kind='bar', title
='From')
plt.xticks(rotation=90)
plt.show()
```



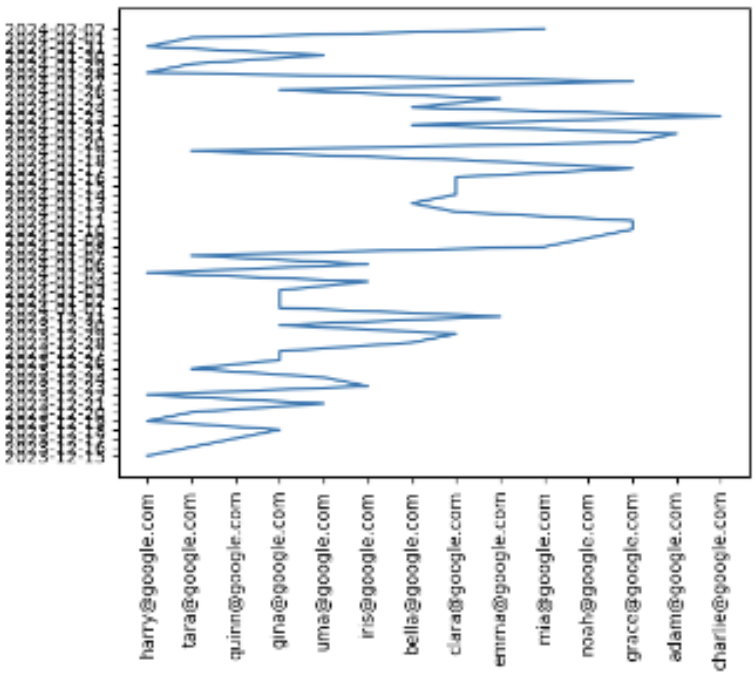
In [7]:

```
plt.plot(df['From'],df['Date'])
plt.xticks(rotation=90)
plt.show()
```



In [7]:

```
plt.plot(df['From'],df['Date'])
plt.xticks(rotation=90)
plt.show()
```



In [8]:

```
df['From'].value_counts().plot(kind='pie', autopct
='%1.1f%%')
plt.axis('equal')
plt.show()
```