Data Visualization With Matplotlib

Line Chart ¶

Data source: https://data.oecd.org/conversion/exchange-rates.htm (https://data.oecd.org/conversion/exchange-rates.htm)

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
plt.style.use('ggplot') # Other sytles to use; fivethirtyeight
```

In [3]:
```
ex_rate_df=pd.read_csv('Exchange_Rates.csv')
```

In [4]:
```
ex_rate_df.head()
```

Out[4]:

	LOCATION	INDICATOR	SUBJECT	MEASURE	FREQUENCY	TIME	Value	Flag Codes
0	AUS	EXCH	TOT	NATUSD	A	2000	1.724827	NaN
1	AUS	EXCH	TOT	NATUSD	A	2001	1.933443	NaN
2	AUS	EXCH	TOT	NATUSD	A	2002	1.840563	NaN
3	AUS	EXCH	TOT	NATUSD	A	2003	1.541914	NaN
4	AUS	EXCH	TOT	NATUSD	A	2004	1.359752	NaN

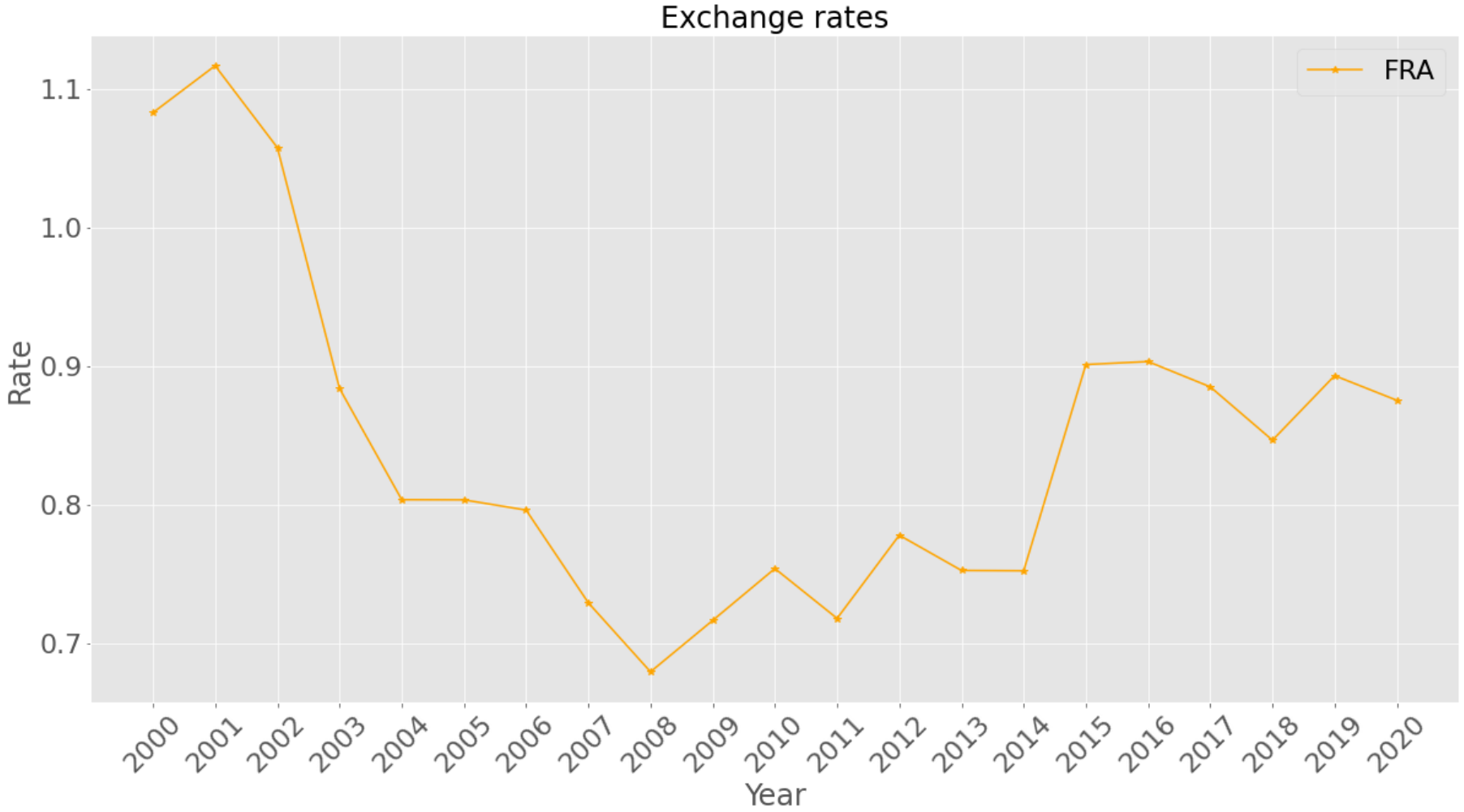
In [5]:
```
ex_rate_df['LOCATION'].unique()
```

Out[5]:
```
array(['AUS', 'AUT', 'BEL', 'CAN', 'CZE', 'DNK', 'FIN', 'FRA', 'DEU',
       'GRC', 'HUN', 'ISL', 'IRL', 'ITA', 'JPN', 'KOR', 'LUX', 'MEX',
       'NLD', 'NZL', 'NOR', 'POL', 'PRT', 'SVK', 'ESP', 'SWE', 'CHE',
       'TUR', 'GBR', 'USA', 'BRA', 'CHL', 'CHN', 'COL', 'EST', 'IND',
       'IDN', 'ISR', 'RUS', 'SVN', 'ZAF', 'LVA', 'LTU', 'SAU', 'EA19',
       'ARG', 'CRI', 'BGR', 'HRV', 'CYP', 'MLT', 'PER', 'ROU', 'MKD',
       'MDG', 'MAR', 'ZMB', 'SRB', 'HKG', 'EU27_2020'], dtype=object)
```

Let's Plot the Exchange Rate for FRA Currency

In [6]:
```
fra_x=ex_rate_df[ex_rate_df['LOCATION']=='FRA']['TIME'].astype(str)
fra_y=ex_rate_df[ex_rate_df['LOCATION']=='FRA']['Value']
```

In [7]:
```
plt.figure(figsize=(20,10)) # Set figure size
plt.rcParams.update({'font.size': 22}) # Set axes size
plt.plot(fra_x,fra_y,color='orange',marker='*') # Plot the chart
plt.title('Exchange rates',fontsize=24)
plt.xticks(rotation=45)
plt.xlabel('Year',fontsize=24)
plt.ylabel('Rate',fontsize=24)
plt.legend(['FRA'], loc='upper right')
plt.show()
```

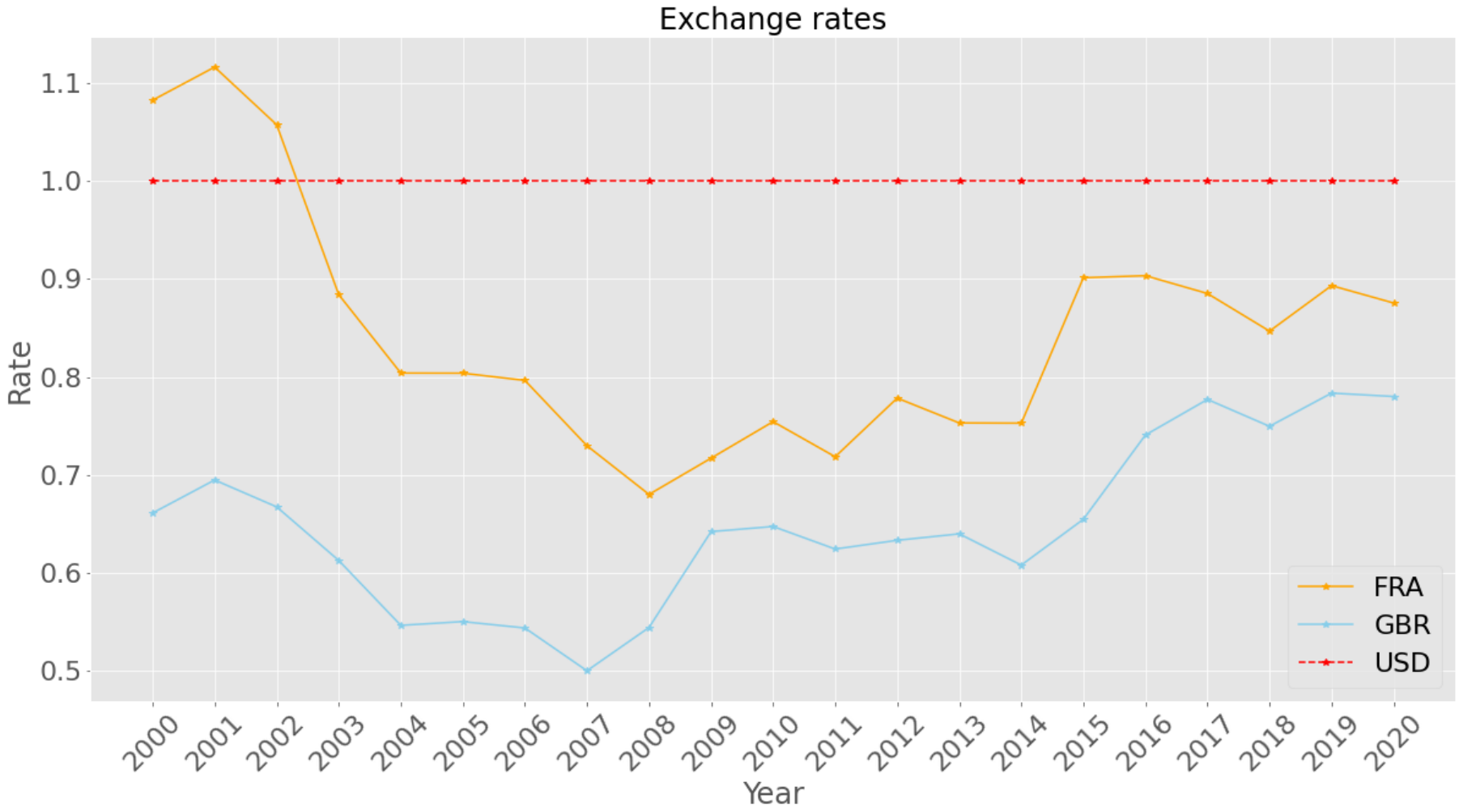


Multiple Lines in a Graph

Let's Add more Plots For the Exchange Rate for FRA, BGR and USD Baseline Currency

In [8]:
```
fra_x=ex_rate_df[ex_rate_df['LOCATION']=='FRA']['TIME'].astype(str)
fra_y=ex_rate_df[ex_rate_df['LOCATION']=='FRA']['Value']
gbr_x=ex_rate_df[ex_rate_df['LOCATION']=='GBR']['TIME'].astype(str)
gbr_y=ex_rate_df[ex_rate_df['LOCATION']=='GBR']['Value']
usd_x=ex_rate_df[ex_rate_df['LOCATION']=='USA']['TIME'].astype(str)
usd_y=ex_rate_df[ex_rate_df['LOCATION']=='USA']['Value']
```

In [9]:
```
plt.figure(figsize=(20,10)) # Set figure size
plt.rcParams.update({'font.size': 22}) # Set axes size
plt.plot(fra_x,fra_y,color='orange',marker='*',label='FRA') # Plot the chart for AUS
plt.plot(gbr_x,gbr_y,color='skyblue',marker='*',label='GBR') # Plot the chart for IDN
plt.plot(usd_x,usd_y,color='red',marker='*',label='USD',linestyle='--') # Plot the chart for HKG
plt.title('Exchange rates',fontsize=24)
plt.xticks(rotation=45)
plt.xlabel('Year',fontsize=24)
plt.ylabel('Rate',fontsize=24)
plt.legend(['FRA', 'GBR','USD','RUS'], loc='lower right')
plt.show()
```


Exchange rates
Rate
1.1
1.0
0.9
0.8
0.7
0.6
0.5
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
Year
FRA
GBR
USD

Bar Graph

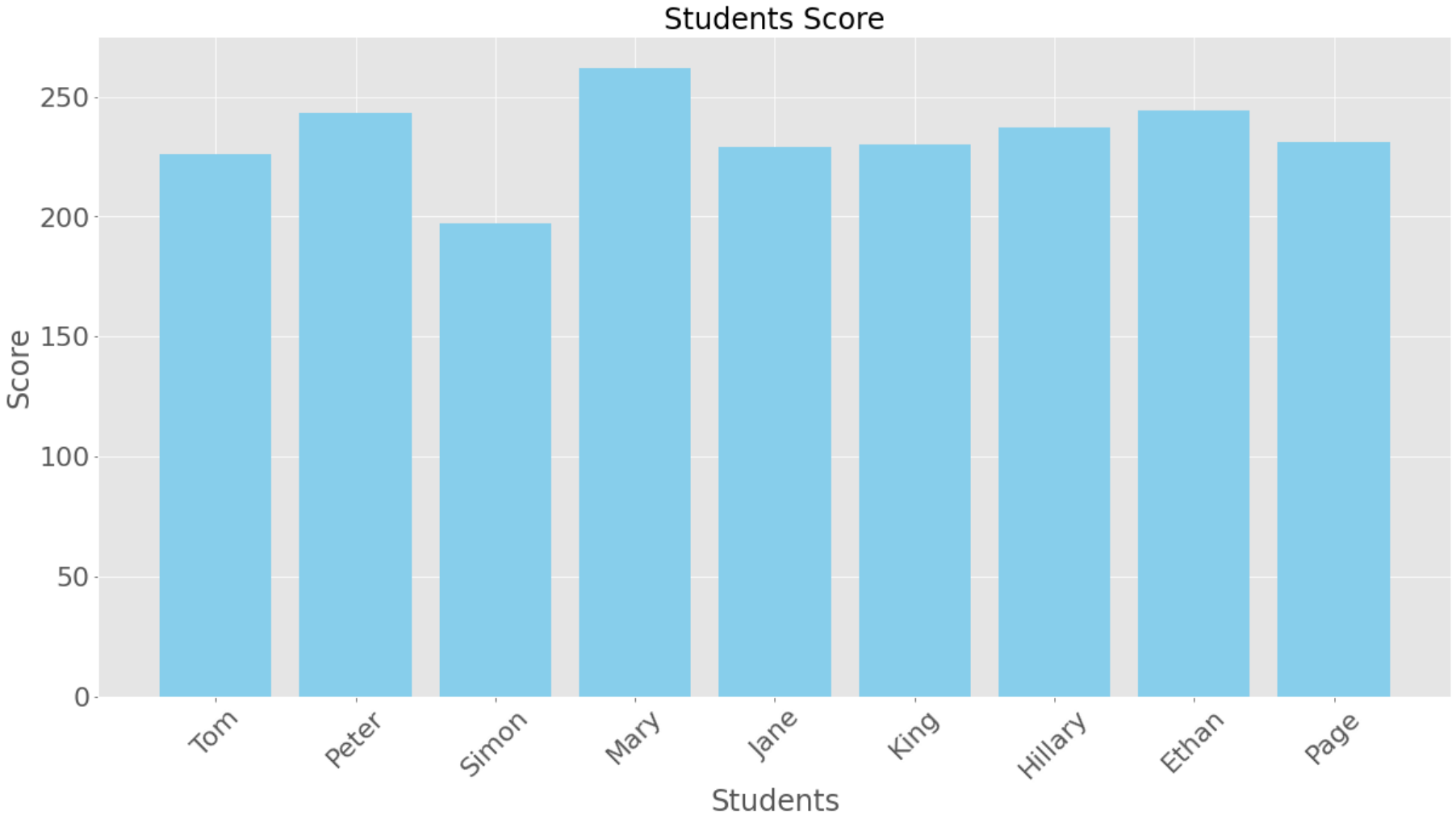
In [10]:
```
score_df = pd.DataFrame(
    {
        "Students": ["Tom", "Peter","Simon", "Mary", "Jane","King","Hillary","Ethan","Page"],
        "Math": [79.00, 67.00,80.00, 84.00, 70.00,60.00,90.00,76.00,75],
        "Physics":[63.00, 98, 60.00, 90,84.00, 77.00,55.00,70,66.00],
        "Computer":[84.00,78.00, 57.00, 88.00, 75.00,93.00,92.00,98.00,90.00],
    },
    index=["Tom", "Peter","Simon", "Mary", "Jane","King","Hillary","Ethan","Page"]
)

score_df['Total']=score_df[['Math','Physics','Computer']].apply(np.sum,axis=1)
score_df
```

Out[10]:

	Students	Math	Physics	Computer	Total
Tom	Tom	79.0	63.0	84.0	226.0
Peter	Peter	67.0	98.0	78.0	243.0
Simon	Simon	80.0	60.0	57.0	197.0
Mary	Mary	84.0	90.0	88.0	262.0
Jane	Jane	70.0	84.0	75.0	229.0
King	King	60.0	77.0	93.0	230.0
Hillary	Hillary	90.0	55.0	92.0	237.0
Ethan	Ethan	76.0	70.0	98.0	244.0
Page	Page	75.0	66.0	90.0	231.0

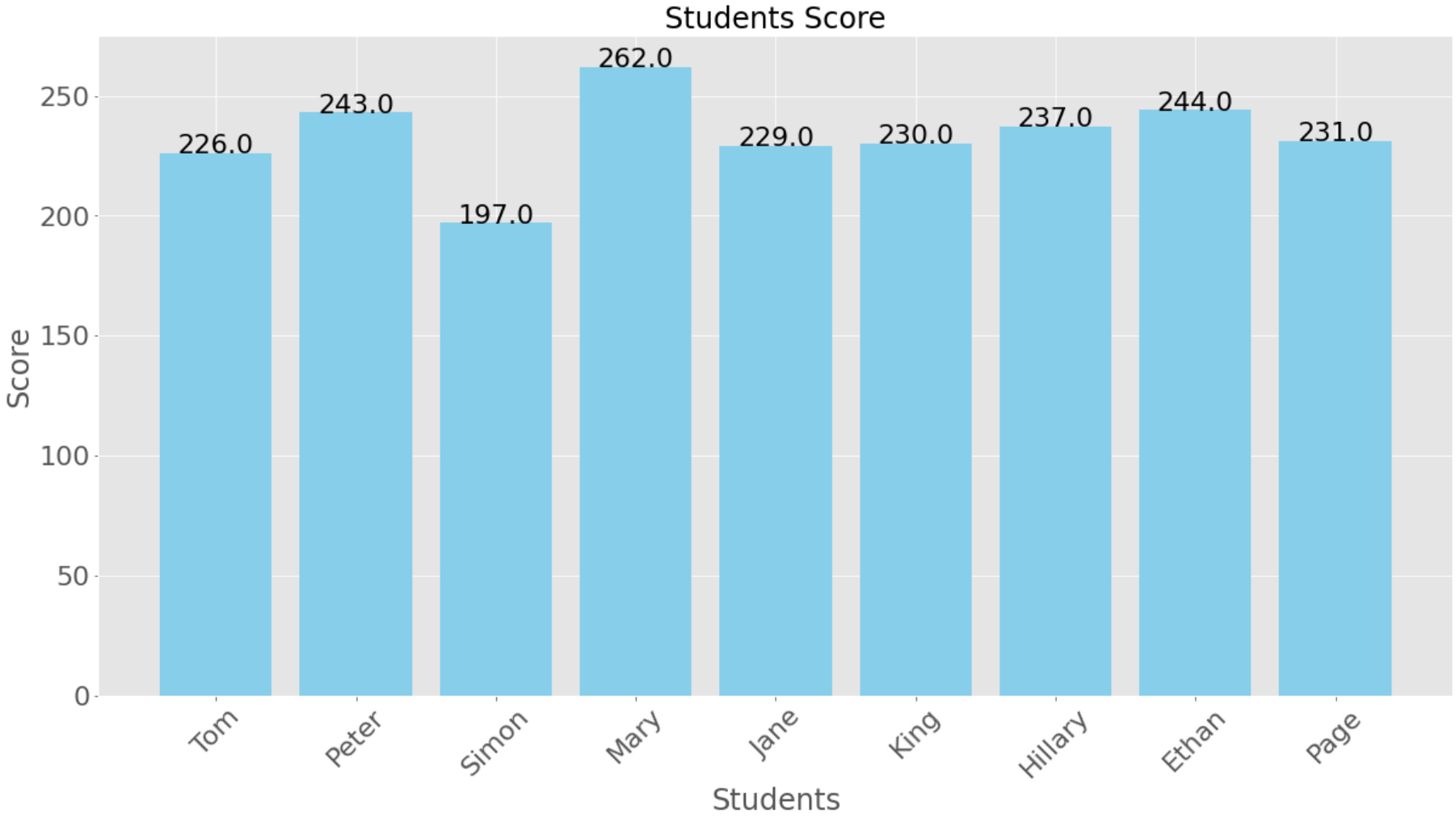
Simple Bar Graph

In [11]:
```
plt.figure(figsize=(20,10)) # Set figure size
plt.bar(score_df['Students'],score_df['Total'],color='skyblue')
plt.title('Students Score',fontsize=24)
plt.xticks(rotation=45)
plt.xlabel('Students',fontsize=24)
plt.ylabel('Score',fontsize=24)
plt.show()
```



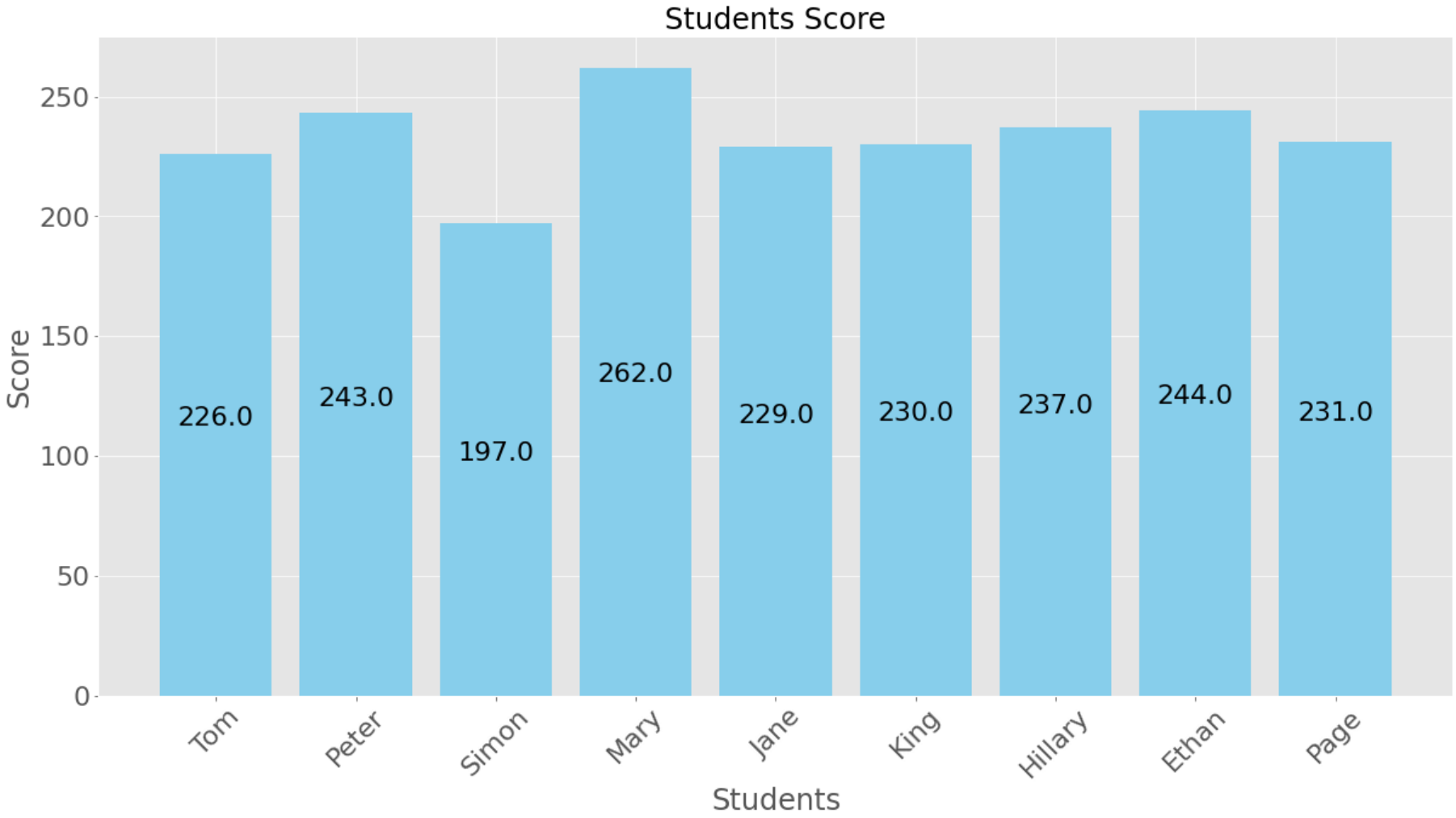
Add Labels

In [12]:
```
def addlabels(x,y):
    for i in range(len(x)):
        plt.text(i, y[i], y[i], ha = 'center')


plt.figure(figsize=(20,10)) # Set figure size
plt.bar(score_df['Students'],score_df['Total'],color='skyblue')
addlabels(score_df['Students'],score_df['Total'])
plt.title('Students Score',fontsize=24)
plt.xticks(rotation=45)
plt.xlabel('Students',fontsize=24)
plt.ylabel('Score',fontsize=24)
plt.show()
```



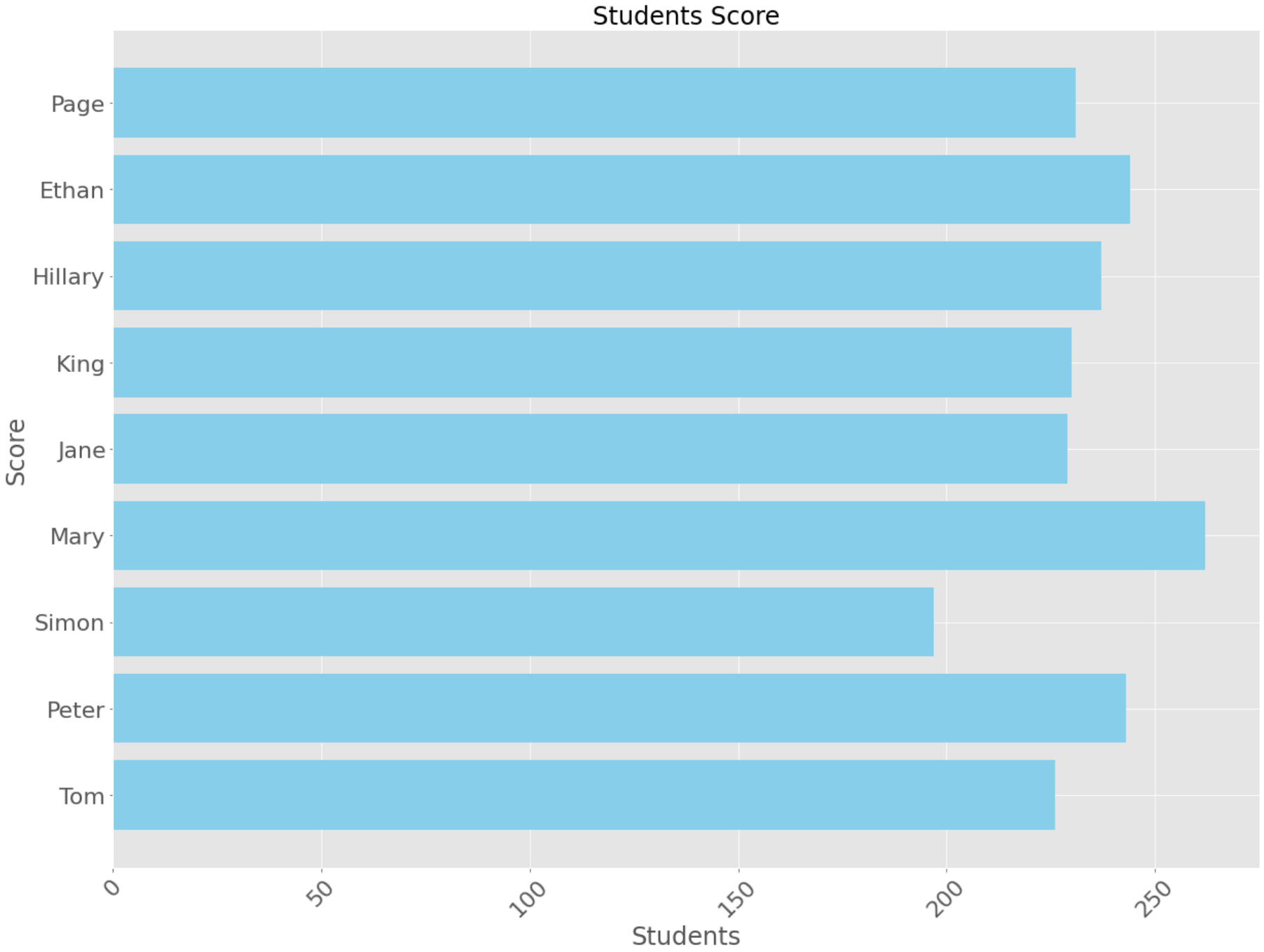
Add Labels at the middle

In [13]:
```
def addlabels(x,y):
    for i in range(len(x)):
        plt.text(i, y[i]//2, y[i], ha = 'center')


plt.figure(figsize=(20,10)) # Set figure size
plt.bar(score_df['Students'],score_df['Total'],color='skyblue')
addlabels(score_df['Students'],score_df['Total'])
plt.title('Students Score',fontsize=24)
plt.xticks(rotation=45)
plt.xlabel('Students',fontsize=24)
plt.ylabel('Score',fontsize=24)
plt.show()
```



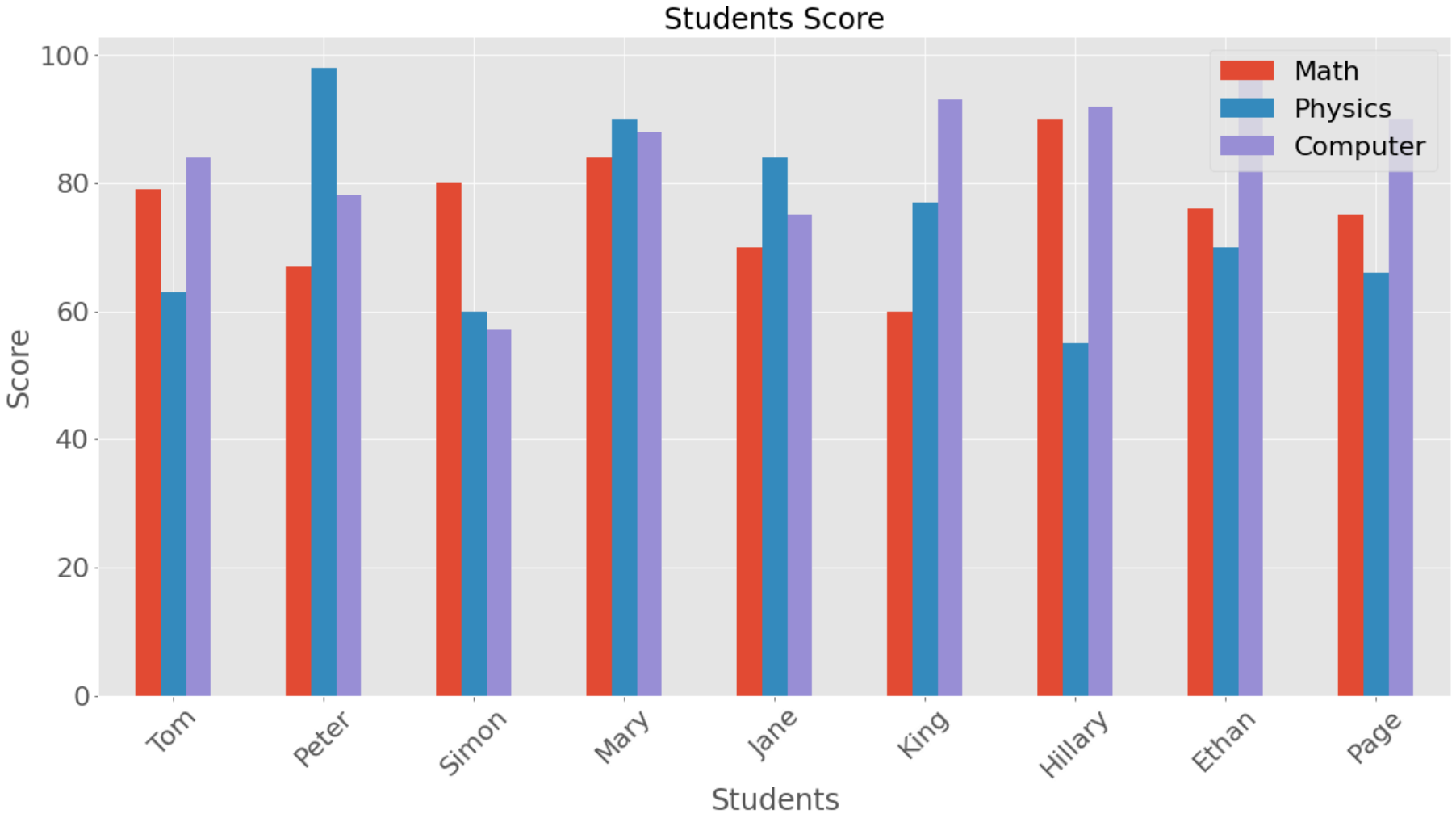
Horizontal Simple Bar Graph

In [14]:
```
plt.figure(figsize=(20,15)) # Set figure size
plt.barh(score_df['Students'],score_df['Total'],color='skyblue')
plt.title('Students Score',fontsize=24)
plt.xticks(rotation=45)
plt.xlabel('Students',fontsize=24)
plt.ylabel('Score',fontsize=24)
plt.show()
```


Students Score
Page
Ethan
Hillary
King
Jane
Mary
Simon
Peter
Tom
Score
0
50
100
150
200
250
Students

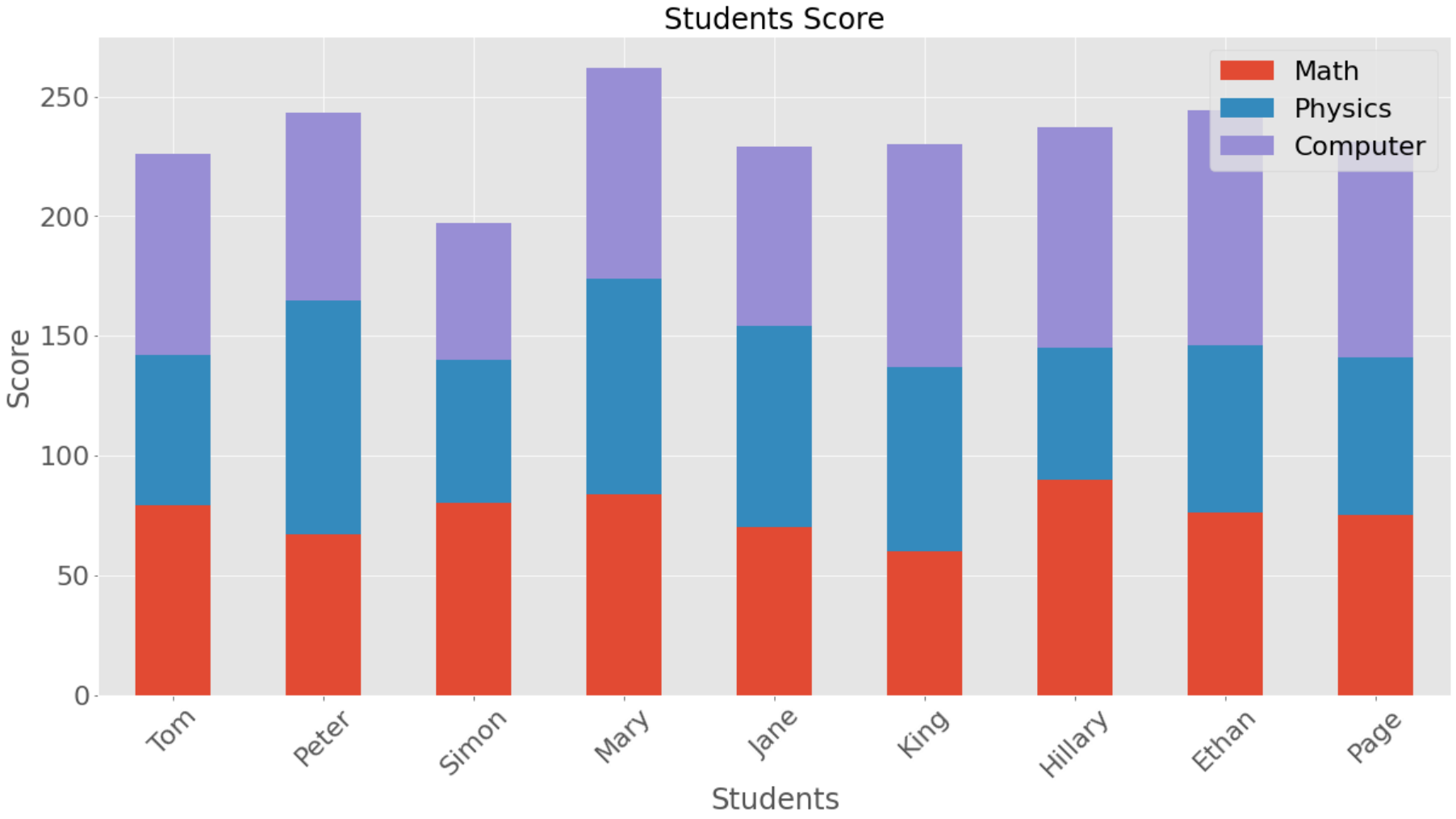
Grouped Bar Chart

In [15]:
```
score_df[['Math','Physics','Computer']].plot(kind="bar",figsize=(20, 10))

plt.title('Students Score',fontsize=24)
plt.xticks(rotation=45)
plt.xlabel('Students',fontsize=24)
plt.ylabel('Score',fontsize=24)
plt.show()
```



Stacked Bar Chart

In [16]:
```
score_df[['Math','Physics','Computer']].plot(kind="bar",stacked=True,figsize=(20, 10))
plt.title('Students Score',fontsize=24)
plt.xticks(rotation=45)
plt.xlabel('Students',fontsize=24)
plt.ylabel('Score',fontsize=24)
plt.show()
```



Scatter Plot

In [17]:
```
iris_df=pd.read_csv('iris.csv')
iris_df.head()
```

Out[17]:

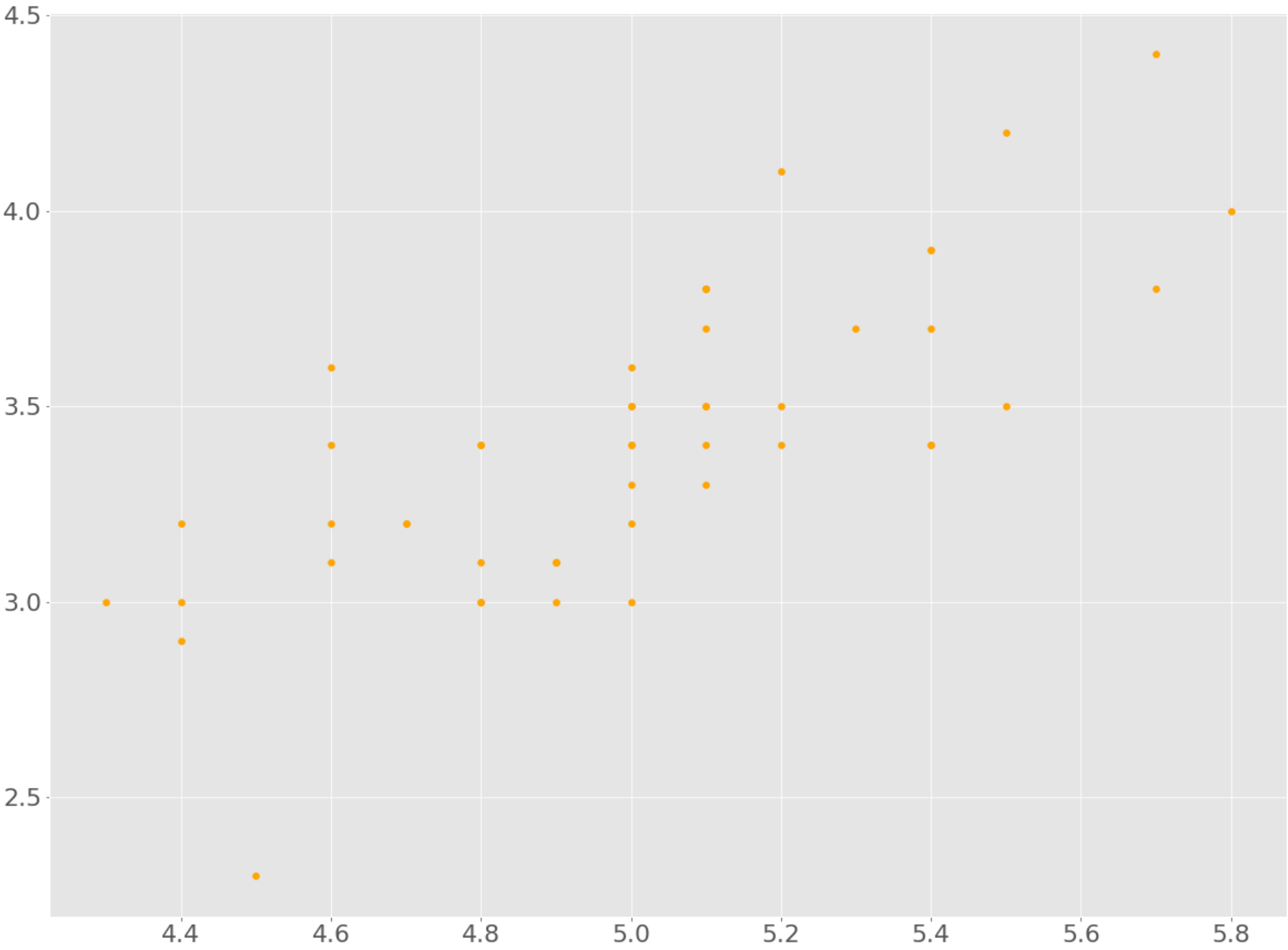
	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Simple Scatter Plot

Let's plot sepal length vs sepal width for Iris-setosa species

In [18]:
```
setosa_df=iris_df[iris_df['class']=='Iris-setosa']

plt.figure(figsize=(20,15)) # Set figure size
plt.scatter(setosa_df['sepal_length'],setosa_df['sepal_width'],c='orange')
```

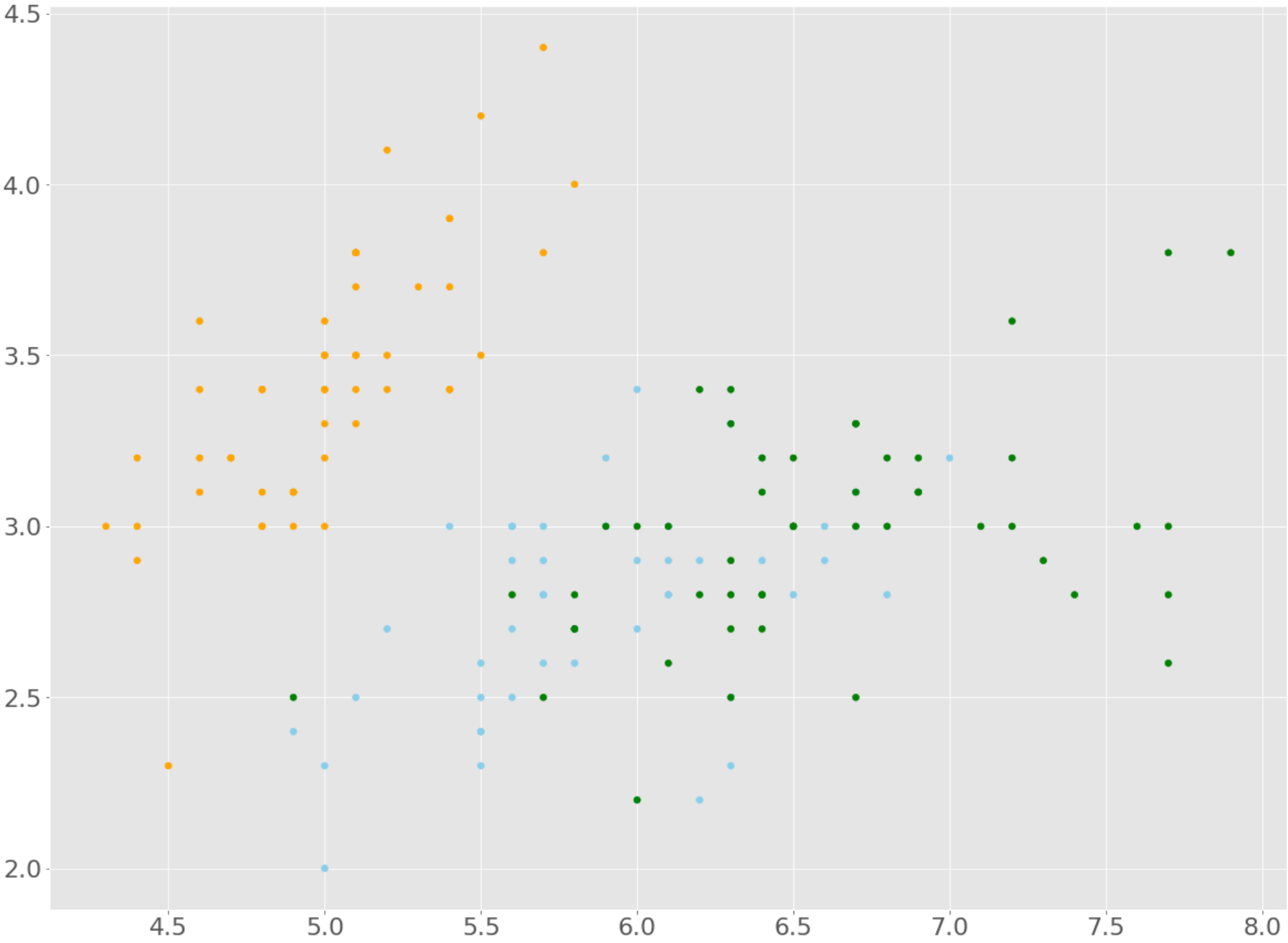
Out[18]: <matplotlib.collections.PathCollection at 0x236308917b8>



Add Classes

In [19]:
```
iris_df['class_color']=iris_df['class'].map({'Iris-setosa':'orange','Iris-versicolor':'skyblue','Iris-virginica':'gree
n'})
plt.figure(figsize=(20,15)) # Set figure size
plt.scatter(iris_df['sepal_length'],iris_df['sepal_width'],c=iris_df['class_color'])
```

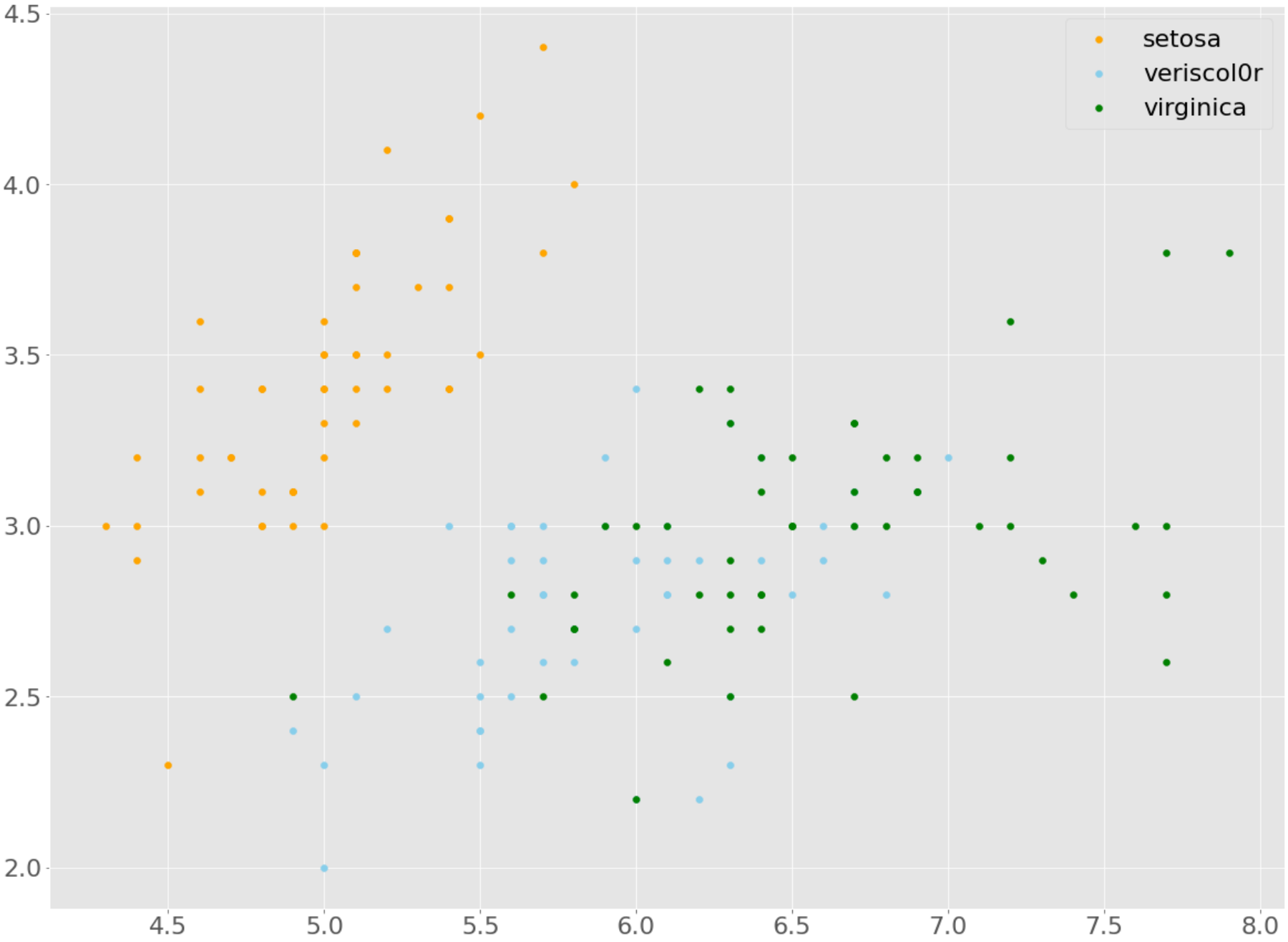
Out[19]: <matplotlib.collections.PathCollection at 0x236308d60b8>



Add Legend

In [20]:
```
setosa_df=iris_df[iris_df['class']=='Iris-setosa']
veriscolor_df=iris_df[iris_df['class']=='Iris-versicolor']
virginica_df=iris_df[iris_df['class']=='Iris-virginica']

plt.figure(figsize=(20,15)) # Set figure size
plt.scatter(setosa_df['sepal_length'],setosa_df['sepal_width'],c='orange')
plt.scatter(veriscolor_df['sepal_length'],veriscolor_df['sepal_width'],c='skyblue')
plt.scatter(virginica_df['sepal_length'],virginica_df['sepal_width'],c='green')
plt.legend(['setosa','veriscol0r','virginica'])
```

Out[20]: <matplotlib.legend.Legend at 0x23630d30160>



Pie Chart

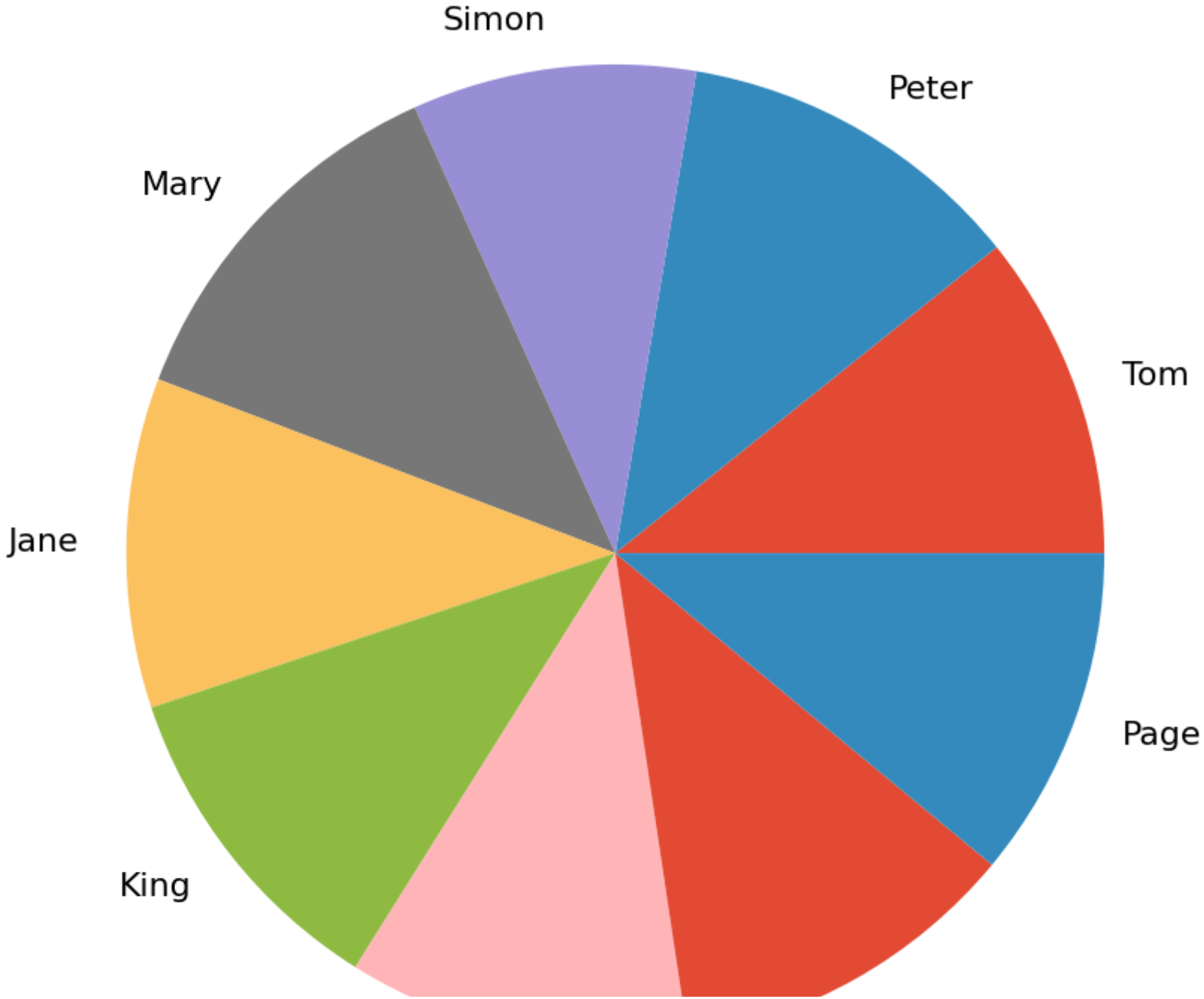
In [21]:
```
score_df = pd.DataFrame(
    {
        "Students": ["Tom", "Peter","Simon", "Mary", "Jane","King","Hillary","Ethan","Page"],
        "Math": [79.00, 67.00,80.00, 84.00, 70.00,60.00,90.00,76.00,75],
        "Physics":[63.00, 98, 60.00, 90,84.00, 77.00,55.00,70,66.00],
        "Computer":[84.00,78.00, 57.00, 88.00, 75.00,93.00,92.00,98.00,90.00],
    },
    index=["Tom", "Peter","Simon", "Mary", "Jane","King","Hillary","Ethan","Page"]
)

score_df['Total']=score_df[['Math','Physics','Computer']].apply(np.sum,axis=1)
score_df
```

Out[21]:

	Students	Math	Physics	Computer	Total
Tom	Tom	79.0	63.0	84.0	226.0
Peter	Peter	67.0	98.0	78.0	243.0
Simon	Simon	80.0	60.0	57.0	197.0
Mary	Mary	84.0	90.0	88.0	262.0
Jane	Jane	70.0	84.0	75.0	229.0
King	King	60.0	77.0	93.0	230.0
Hillary	Hillary	90.0	55.0	92.0	237.0
Ethan	Ethan	76.0	70.0	98.0	244.0
Page	Page	75.0	66.0	90.0	231.0

Simple Pie Chart

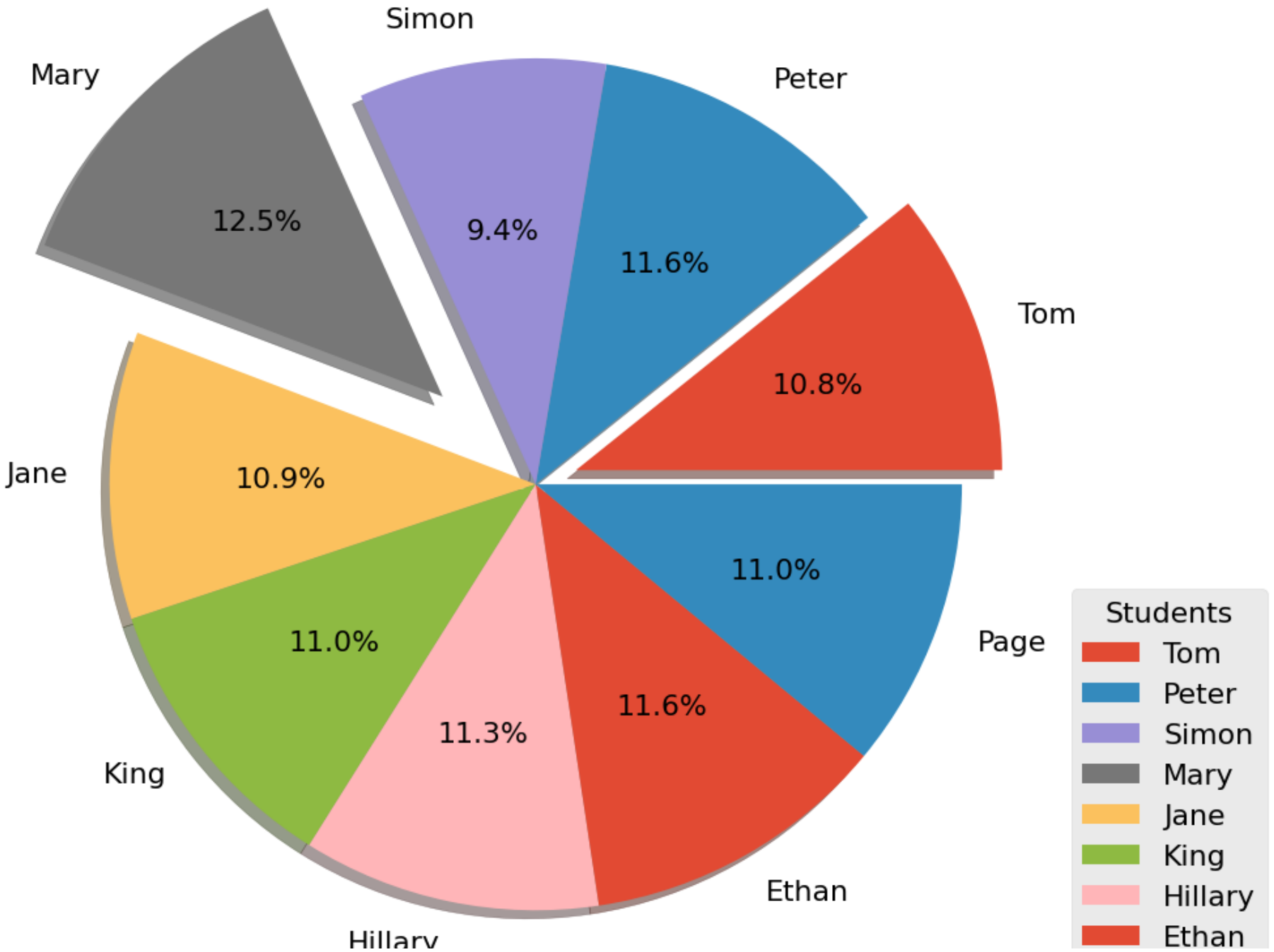
In [22]:
```
plt.figure(figsize=(20,15)) # Set figure size
plt.pie(score_df['Total'],labels=score_df['Students'])
plt.show()
```


Simon
Peter
Mary
Tom
Jane
Page
King

Format Pie Chart

First we add percentages with autopct to show the contribution of each students to the overal score. shadow=True adds shadow to the pie. Explode 1st and 4th pies.

In [23]:
```
plt.figure(figsize=(20,15)) # Set figure size
explode_student=[0.1,0,0,0.3,0,0,0,0,0]
plt.pie(score_df['Total'],labels=score_df['Students'],autopct='%1.1f%%',shadow=True,explode=explode_student)
plt.legend(loc='lower right',title='Students',bbox_to_anchor =(1, 0, 0.2, 1))
plt.show()
```

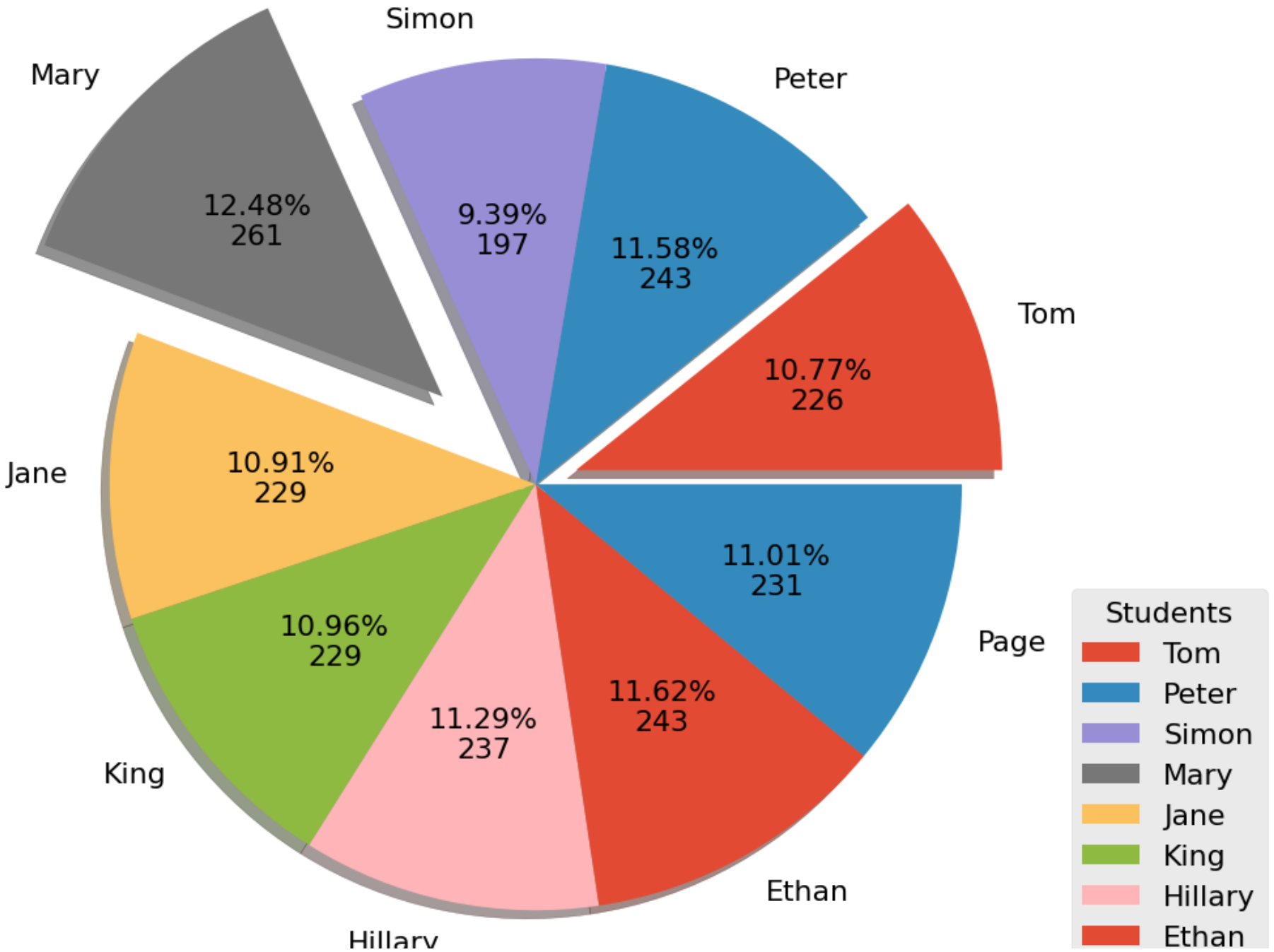

Simon
Mary
Peter
12.5%
9.4%
11.6%
Tom
10.8%
Jane
10.9%
11.0%
Page
11.0%
King
11.3%
11.6%
Ethan
Hillary
Students
Tom
Peter
Simon
Mary
Jane
King
Hillary
Ethan

Show percentage and Actual data

First we haev to create a custom function with percentanges and absolute values. Then apply the function to autopct with a lambda function

In [24]:
```
def percentage_and_actual_value(percentage, actual_value):
    actual = int(percentage / 100.*np.sum(actual_value))
    return "{:.2f}%\n{:d}".format(percentage, actual)
```

In [25]:
```
plt.figure(figsize=(20,15)) # Set figure size
explode_student=[0.1,0,0,0.3,0,0,0,0,0]
plt.pie(score_df['Total'],labels=score_df['Students'],autopct=lambda x: percentage_and_actual_value(x, score_df['Tota
l']),
        shadow=True,explode=explode_student)
plt.legend(loc='lower right',title='Students',bbox_to_anchor =(1, 0, 0.2, 1))
plt.show()
```

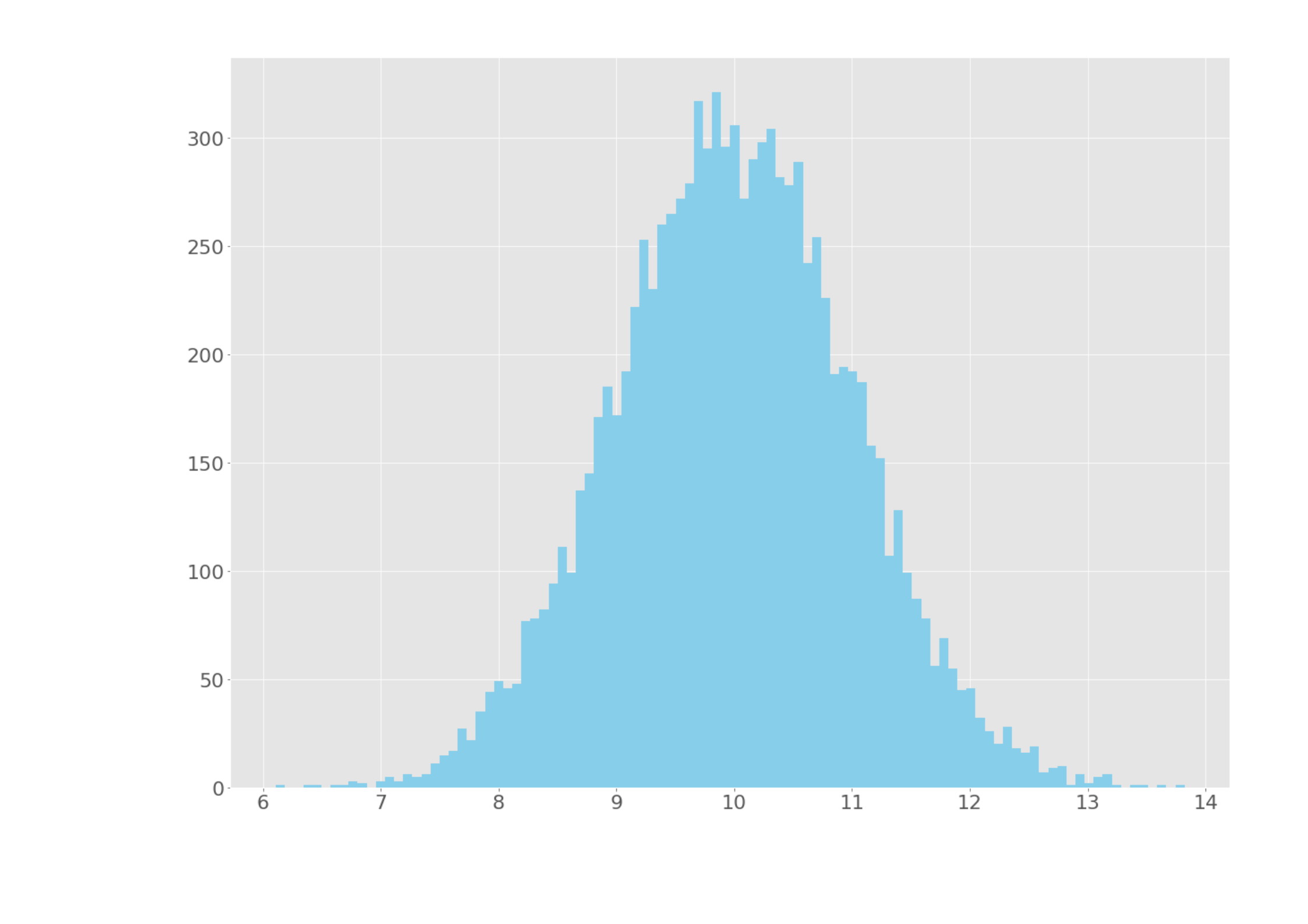

Simon
Mary
Peter
12.48%
261
9.39%
197
11.58%
243
Tom
10.77%
226
Jane
10.91%
229
11.01%
231
Page
10.96%
229
11.62%
243
11.29%
237
King
Ethan
Hillary
Students
Tom
Peter
Simon
Mary
Jane
King
Hillary
Ethan

Histogram

Histogram shows the distribution of a random variable

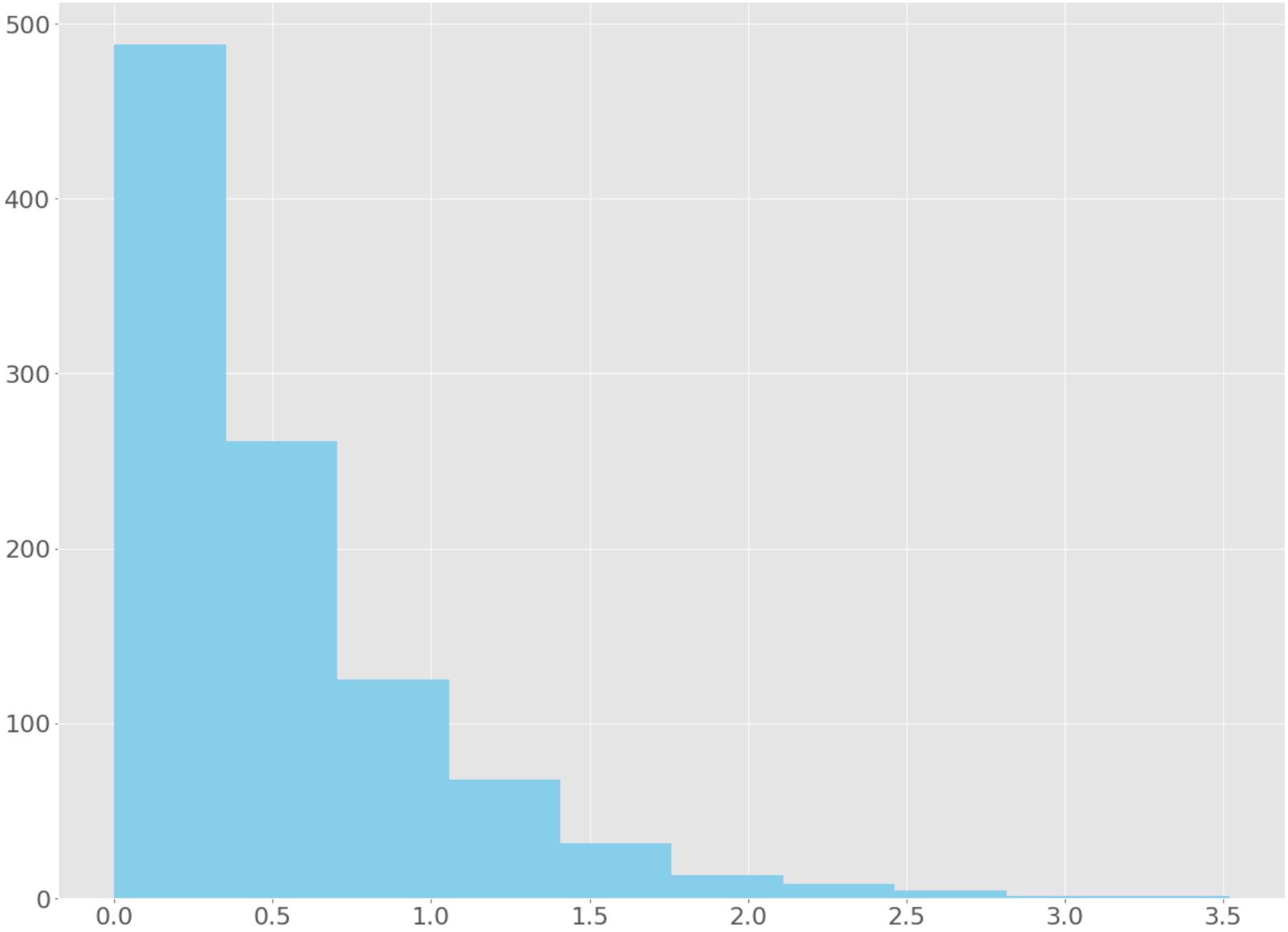
Normal/Gaussian data distribution

In [26]:
```
normal_distributed_variable=np.random.normal(10,1,10000)
plt.figure(figsize=(20,15)) # Set figure size
plt.hist(normal_distributed_variable,100,color = "skyblue")
plt.show()
```


300
250
200
150
100
50
0
6
7
8
9
10
11
12
13
14

Exponential Distribution

In [27]:
```
exponential_distributed_variable = np.random.exponential(scale=0.5, size=(1000, 1))
plt.figure(figsize=(20,15)) # Set figure size
plt.hist(exponential_distributed_variable,color = "skyblue")
plt.show()
```


500
400
300
200
100
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

Area Chart

In [28]:
```
ex_rate_df=pd.read_csv('Exchange_Rates.csv')
ex_rate_df.head()
```

Out[28]:

	LOCATION	INDICATOR	SUBJECT	MEASURE	FREQUENCY	TIME	Value	Flag Codes
0	AUS	EXCH	TOT	NATUSD	A	2000	1.724827	NaN
1	AUS	EXCH	TOT	NATUSD	A	2001	1.933443	NaN
2	AUS	EXCH	TOT	NATUSD	A	2002	1.840563	NaN
3	AUS	EXCH	TOT	NATUSD	A	2003	1.541914	NaN
4	AUS	EXCH	TOT	NATUSD	A	2004	1.359752	NaN

In [29]:
```
area_chart_ex_rate_df= pd.crosstab(ex_rate_df['TIME'],ex_rate_df['LOCATION'],values=ex_rate_df['Value'],aggfunc='mean'
)


area_chart_ex_rate_df.index=area_chart_ex_rate_df.index.astype(str)

area_chart_ex_rate_df.head()
```

Out[29]:

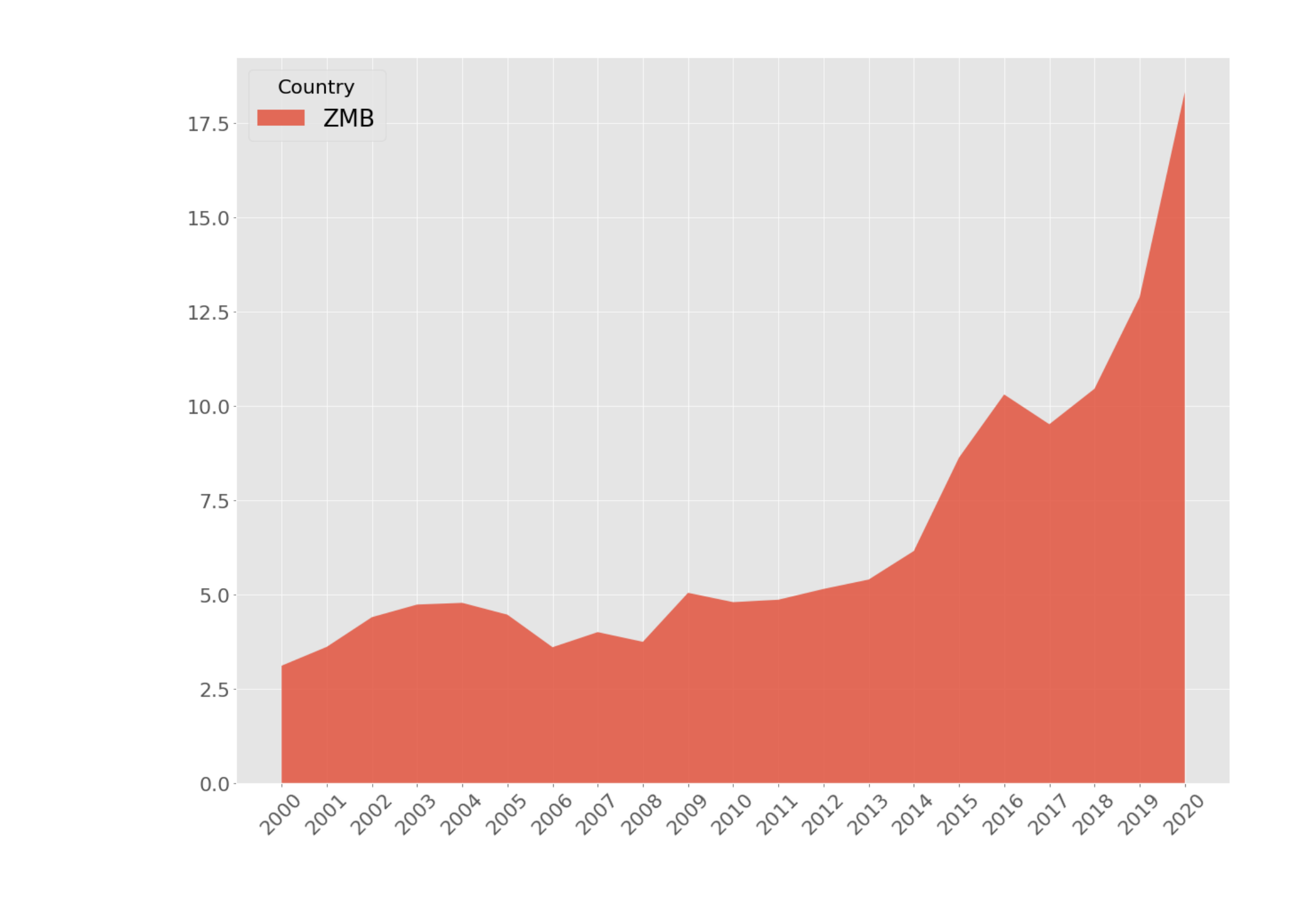
LOCATION	ARG	AUS	AUT	BEL	BGR	BRA	CAN	CHE	CHL	CHN	...	RUS	SAU
TIME													
2000	0.999500	1.724827	1.082705	1.082705	2.123275	1.829423	1.485394	1.688843	539.587500	8.278504	...	28.129167	3.75
2001	0.999500	1.933443	1.116533	1.116533	2.184708	2.349632	1.548840	1.687615	634.938333	8.277068	...	29.168525	3.75
2002	3.063257	1.840563	1.057559	1.057559	2.076975	2.920363	1.570343	1.558607	688.936667	8.276958	...	31.348483	3.75
2003	2.900629	1.541914	0.884048	0.884048	1.732702	3.077475	1.401015	1.346651	691.397500	8.277037	...	30.692025	3.75
2004	2.923301	1.359752	0.803922	0.803922	1.575109	2.925119	1.301282	1.243496	609.529167	8.276801	...	28.813742	3.75

5 rows × 60 columns

Basic Area Chart

Shows the change over time

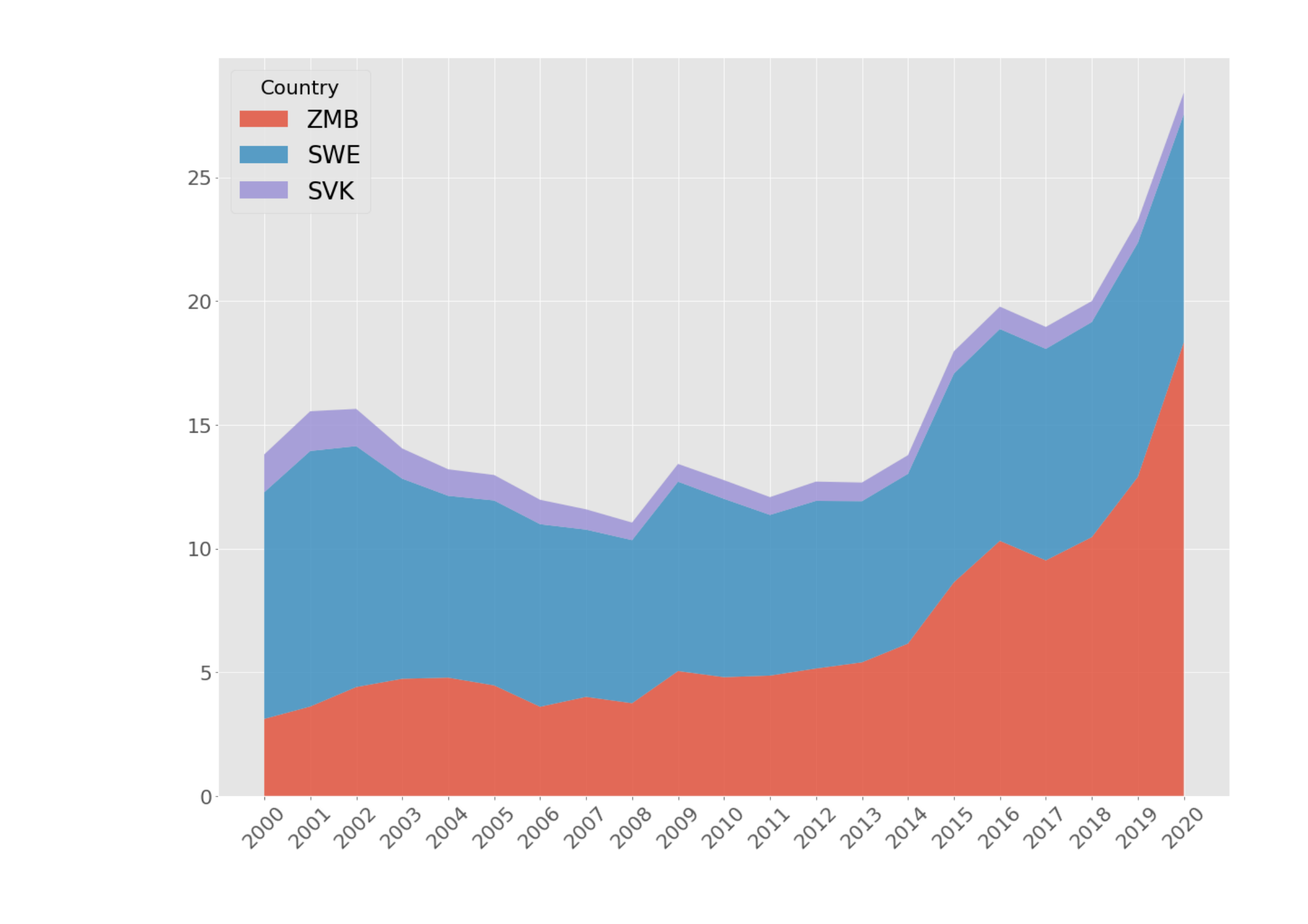
In [30]:
```
plt.figure(figsize=(20,15)) # Set figure size
plt.stackplot(area_chart_ex_rate_df.index, area_chart_ex_rate_df['ZMB'],
              labels=['ZMB'],
              alpha=0.8)

plt.legend(loc=2, fontsize='large',title='Country')
plt.xticks(rotation=45)
plt.show()
plt.show()
```


Country
ZMB
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020

Stacked Area Chart

Shows part-to-whole relationships, helping show how each category contributes to the cumulative total.

In [31]:
```
plt.figure(figsize=(20,15)) # Set figure size
plt.stackplot(area_chart_ex_rate_df.index, area_chart_ex_rate_df['ZMB'], area_chart_ex_rate_df['SWE'],
              area_chart_ex_rate_df['SVK'],
              labels=['ZMB', 'SWE', 'SVK'],
              alpha=0.8)

plt.legend(loc=2, fontsize='large',title='Country')
plt.xticks(rotation=45)
plt.show()
```


Country
ZMB
SWE
SVK
0
5
10
15
20
25
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020

Treemap in Matplotlib

To plot Treemap we need to install squarify. pip install squarify.

Use Treemap when; You have fewer categories You have Positive values You don't have enough space for multiple visualizations

In [32]:
```
# !pip install squarify # install squarify. Uncomment to install if using jupyter notebook
```

In [33]:
```
salary_df = pd.DataFrame(
    {
        "Staff": ["Tom", "Peter","Simon", "Mary", "Jane"],
        "Salary": [90000.00, 57000.00,40000.00, 34000.00, 12000.00]
    },
)

salary_df
```

Out[33]:

	Staff	Salary
0	Tom	90000.0
1	Peter	57000.0
2	Simon	40000.0
3	Mary	34000.0
4	Jane	12000.0

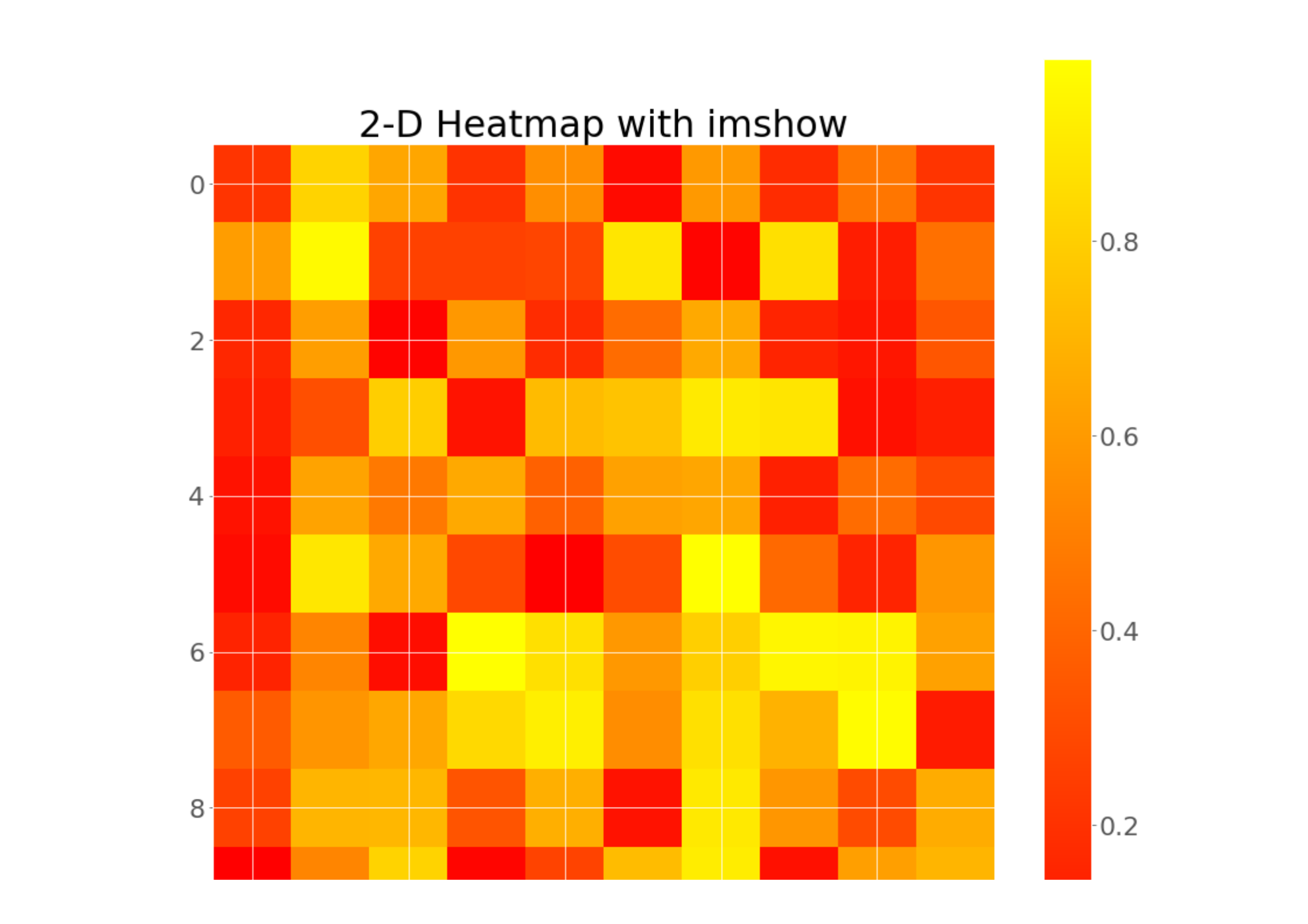
In [34]:
```
import squarify

plt.figure(figsize=(20,15)) # Set figure size
color = ['red', 'green', 'skyblue', 'orange','magenta']
squarify.plot(salary_df['Salary'], label = salary_df['Staff'], color=color, pad = True, alpha=.8)
plt.axis('off')
plt.show()
```


Peter
Jane
Mary
Tom
Simon

Heatmap in Matplotlib

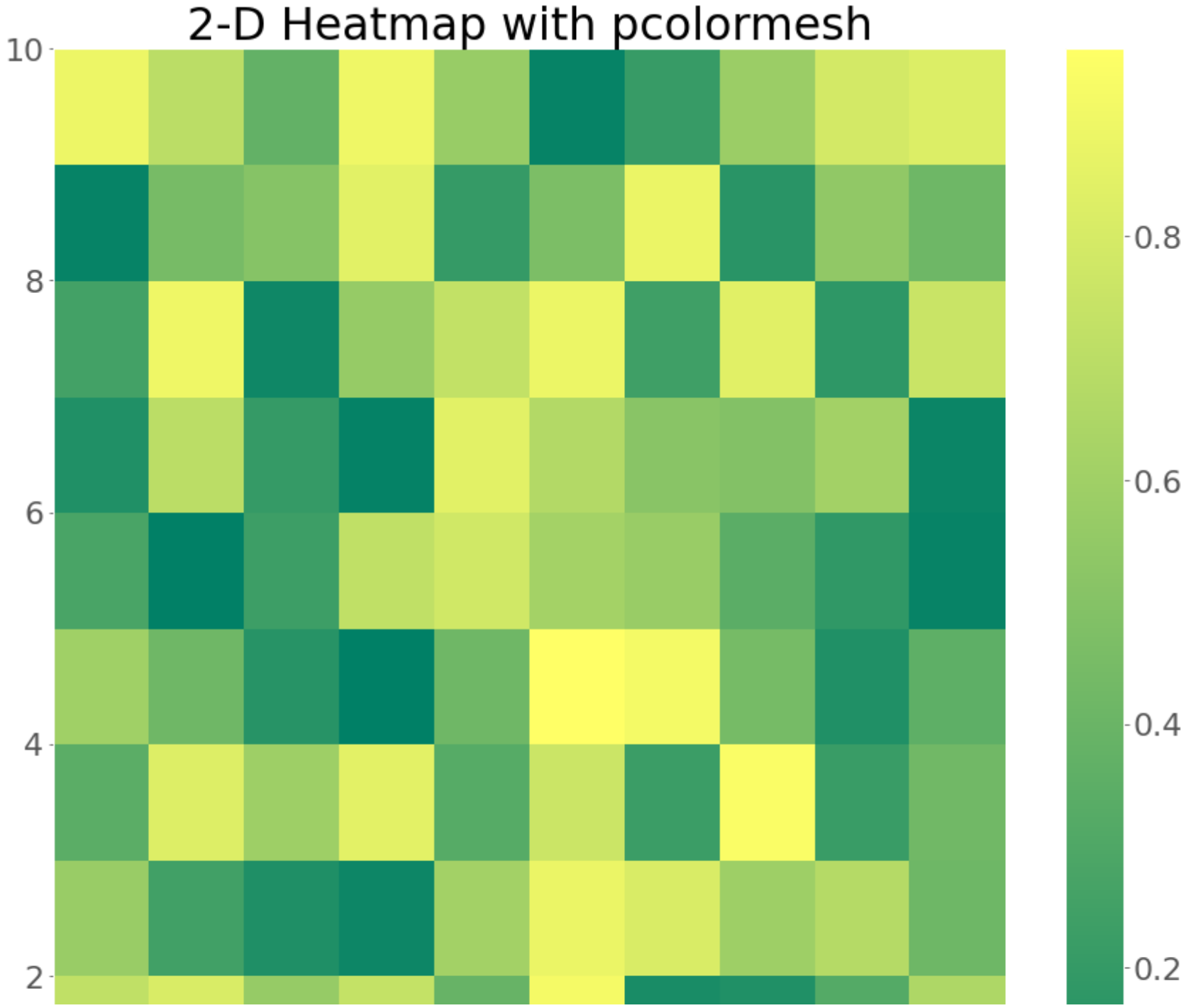
1. Creating Heatmap with imshow

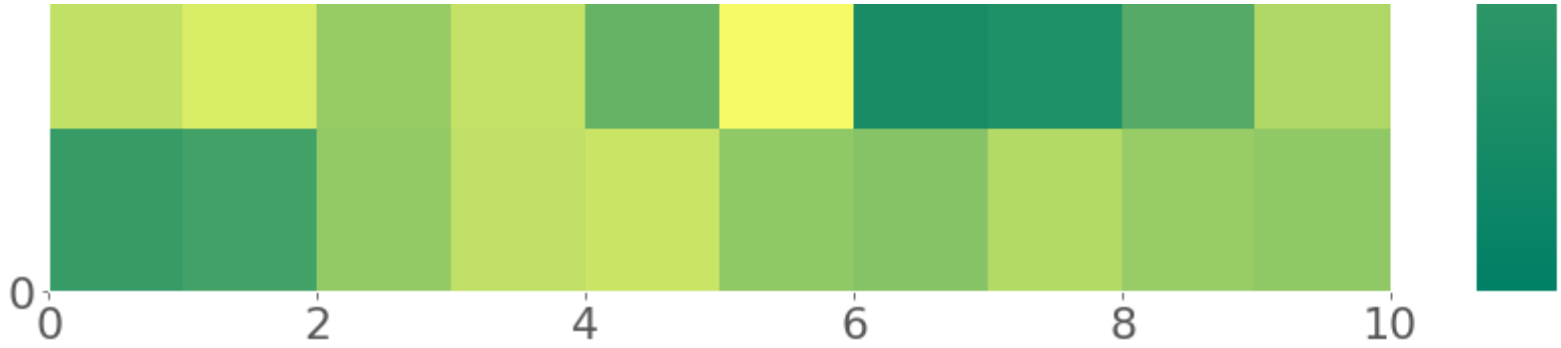
In [35]:
```
555555555555555555554

plt.figure(figsize=(15,15)) # Set figure size
plt.imshow(heatmap_data,cmap='autumn')
plt.colorbar()
plt.title('2-D Heatmap with imshow')
plt.show()
```


2-D Heatmap with imshow
0
2
4
6
8
0.8
0.6
0.4
0.2

Creating Heatmap with pcolormesh

In [36]:
```
heatmap_data = np.random.random(( 10 , 10 ))

plt.figure(figsize=(15,15)) # Set figure size
plt.pcolormesh(heatmap_data, cmap = 'summer')
plt.colorbar()
plt.title('2-D Heatmap with pcolormesh')
plt.show()
```


2-D Heatmap with pcolormesh
10
8
6
4
2
0.8
0.6
0.4
0.2



Creating Heatmap from DataFrame

In [37]:
```
ex_rate_df=pd.read_csv('Exchange_Rates.csv')
ex_rate_df.head()
```

Out[37]:

	LOCATION	INDICATOR	SUBJECT	MEASURE	FREQUENCY	TIME	Value	Flag Codes
0	AUS	EXCH	TOT	NATUSD	A	2000	1.724827	NaN
1	AUS	EXCH	TOT	NATUSD	A	2001	1.933443	NaN
2	AUS	EXCH	TOT	NATUSD	A	2002	1.840563	NaN
3	AUS	EXCH	TOT	NATUSD	A	2003	1.541914	NaN
4	AUS	EXCH	TOT	NATUSD	A	2004	1.359752	NaN

In [38]:
```
heatmap_ex_rate_df= pd.crosstab(ex_rate_df['TIME'],ex_rate_df['LOCATION'],values=ex_rate_df['Value'],aggfunc='mean')

heatmap_ex_rate_df.iloc[0:,0:13]
```

Out[38]:

LOCATION	ARG	AUS	AUT	BEL	BGR	BRA	CAN	CHE	CHL	CHN	COL	
TIME												
2000	0.999500	1.724827	1.082705	1.082705	2.123275	1.829423	1.485394	1.688843	539.587500	8.278504	2087.903842	308.
2001	0.999500	1.933443	1.116533	1.116533	2.184708	2.349632	1.548840	1.687615	634.938333	8.277068	2299.633156	328.
2002	3.063257	1.840563	1.057559	1.057559	2.076975	2.920363	1.570343	1.558607	688.936667	8.276958	2504.241331	359.
2003	2.900629	1.541914	0.884048	0.884048	1.732702	3.077475	1.401015	1.346651	691.397500	8.277037	2877.652458	398.
2004	2.923301	1.359752	0.803922	0.803922	1.575109	2.925119	1.301282	1.243496	609.529167	8.276801	2628.612903	437.
2005	2.903658	1.309473	0.803800	0.803800	1.574133	2.434390	1.211405	1.245177	559.767500	8.194317	2320.834177	477.
2006	3.054313	1.327973	0.796433	0.796433	1.559267	2.175327	1.134345	1.253843	530.275000	7.973438	2361.139407	511.
2007	3.095649	1.195073	0.729672	0.729672	1.429050	1.947058	1.074046	1.200366	522.464167	7.607532	2078.291837	516.
2008	3.144165	1.192178	0.679923	0.679923	1.337117	1.833767	1.067087	1.083090	522.461036	6.948655	1967.711309	526.
2009	3.710107	1.282189	0.716958	0.716958	1.406692	1.999428	1.141535	1.088142	560.859895	6.831416	2158.255903	573.
2010	3.896295	1.090159	0.754309	0.754309	1.477392	1.759227	1.030113	1.042906	510.249167	6.770269	1898.569636	525.
2011	4.110140	0.969463	0.718414	0.718414	1.406458	1.672829	0.989258	0.888042	483.667500	6.461461	1848.139470	505.
2012	4.536934	0.965801	0.778338	0.778338	1.522050	1.953069	0.999365	0.937684	486.471303	6.312333	1796.895912	502.
2013	5.459353	1.035843	0.752945	0.752945	1.473567	2.156089	1.030137	0.926904	495.272878	6.195758	1868.785327	499.
2014	8.075276	1.109363	0.752728	0.752728	1.474183	2.352952	1.104747	0.916151	570.348216	6.143434	2001.781048	538.
2015	9.233186	1.331090	0.901296	0.901296	1.764400	3.326904	1.278786	0.962381	654.124084	6.227489	2741.880855	534.
2016	14.758175	1.345214	0.903421	0.903421	1.768042	3.491313	1.325615	0.985394	676.957736	6.644478	3054.121673	544.
2017	16.562707	1.304758	0.885206	0.885206	1.735458	3.191389	1.297936	0.984692	648.833793	6.758755	2951.327402	567.
2018	28.094992	1.338412	0.846773	0.846773	1.657042	3.653825	1.295818	0.977883	641.276813	6.615957	2955.703970	576.
2019	48.147892	1.438507	0.893276	0.893276	1.747042	3.944471	1.326793	0.993775	702.897423	6.908385	3280.831631	587.
2020	70.539167	1.453085	0.875506	0.875506	1.716333	5.155179	1.341153	0.938895	792.727206	6.900767	3694.854072	584.

In [39]:
```
heatmap_ex_rate_df.iloc[0:,0:13].style.background_gradient(cmap='summer')
```

Out[39]:

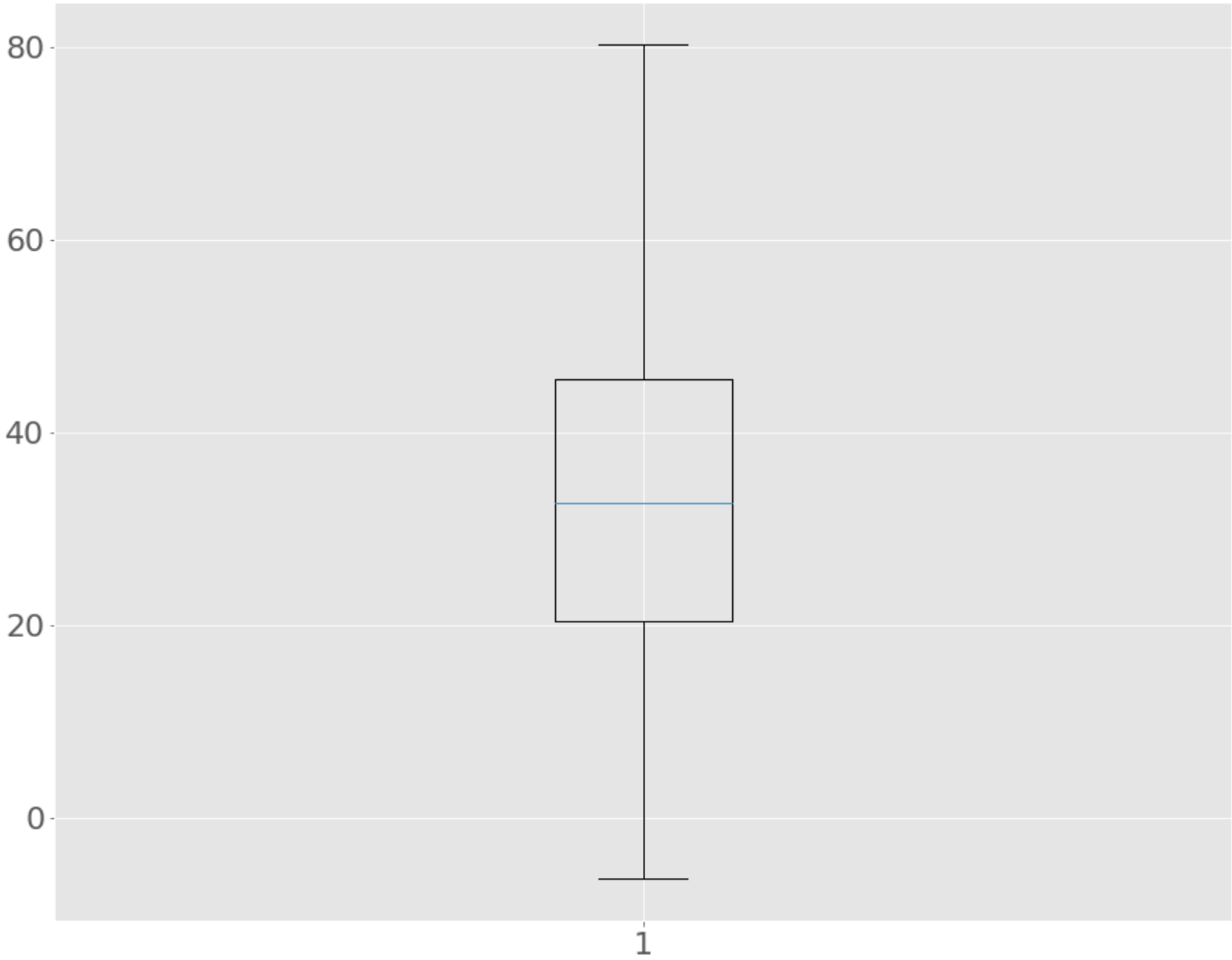
LOCATION	ARG	AUS	AUT	BEL	BGR	BRA	CAN	CHE	CHL	CHN	COL	CRI
TIME												
2000	0.999500	1.724827	1.082705	1.082705	2.123275	1.829423	1.485394	1.688843	539.587500	8.278504	2087.903842	308.
2001	0.999500	1.933443	1.116533	1.116533	2.184708	2.349632	1.548840	1.687615	634.938333	8.277068	2299.633156	328.
2002	3.063257	1.840563	1.057559	1.057559	2.076975	2.920363	1.570343	1.558607	688.936667	8.276958	2504.241331	359.
2003	2.900629	1.541914	0.884048	0.884048	1.732702	3.077475	1.401015	1.346651	691.397500	8.277037	2877.652458	398.
2004	2.923301	1.359752	0.803922	0.803922	1.575109	2.925119	1.301282	1.243496	609.529167	8.276801	2628.612903	437.
2005	2.903658	1.309473	0.803800	0.803800	1.574133	2.434390	1.211405	1.245177	559.767500	8.194317	2320.834177	477.
2006	3.054313	1.327973	0.796433	0.796433	1.559267	2.175327	1.134345	1.253843	530.275000	7.973438	2361.139407	511.
2007	3.095649	1.195073	0.729672	0.729672	1.429050	1.947058	1.074046	1.200366	522.464167	7.607532	2078.291837	516.
2008	3.144165	1.192178	0.679923	0.679923	1.337117	1.833767	1.067087	1.083090	522.461036	6.948655	1967.711309	526.
2009	3.710107	1.282189	0.716958	0.716958	1.406692	1.999428	1.141535	1.088142	560.859895	6.831416	2158.255903	573.
2010	3.896295	1.090159	0.754309	0.754309	1.477392	1.759227	1.030113	1.042906	510.249167	6.770269	1898.569636	525.
2011	4.110140	0.969463	0.718414	0.718414	1.406458	1.672829	0.989258	0.888042	483.667500	6.461461	1848.139470	505.
2012	4.536934	0.965801	0.778338	0.778338	1.522050	1.953069	0.999365	0.937684	486.471303	6.312333	1796.895912	502.
2013	5.459353	1.035843	0.752945	0.752945	1.473567	2.156089	1.030137	0.926904	495.272878	6.195758	1868.785327	499.
2014	8.075276	1.109363	0.752728	0.752728	1.474183	2.352952	1.104747	0.916151	570.348216	6.143434	2001.781048	538.
2015	9.233186	1.331090	0.901296	0.901296	1.764400	3.326904	1.278786	0.962381	654.124084	6.227489	2741.880855	534.
2016	14.758175	1.345214	0.903421	0.903421	1.768042	3.491313	1.325615	0.985394	676.957736	6.644478	3054.121673	544.
2017	16.562707	1.304758	0.885206	0.885206	1.735458	3.191389	1.297936	0.984692	648.833793	6.758755	2951.327402	567.
2018	28.094992	1.338412	0.846773	0.846773	1.657042	3.653825	1.295818	0.977883	641.276813	6.615957	2955.703970	576.
2019	48.147892	1.438507	0.893276	0.893276	1.747042	3.944471	1.326793	0.993775	702.897423	6.908385	3280.831631	587.
2020	70.539167	1.453085	0.875506	0.875506	1.716333	5.155179	1.341153	0.938895	792.727206	6.900767	3694.854072	584.

Boxplot in Matplotlib

Boxplot is atype of visualization used for summarizing the characteristics of groups of data. It shows important staistics such as Minimum values, Maximum, First Qquartile (25%), Median (Second Qquartile/50%) and Third Quartile (75%). It's also used for detecting outliers in data.

Simple Boxplot

In [40]:
```
boxplot_df1=np.random.normal(35, 20, 100)

plt.figure(figsize=(15,12)) # Set figure size
plt.boxplot(boxplot_df1)
plt.show()
```


80
60
40
20
0
1

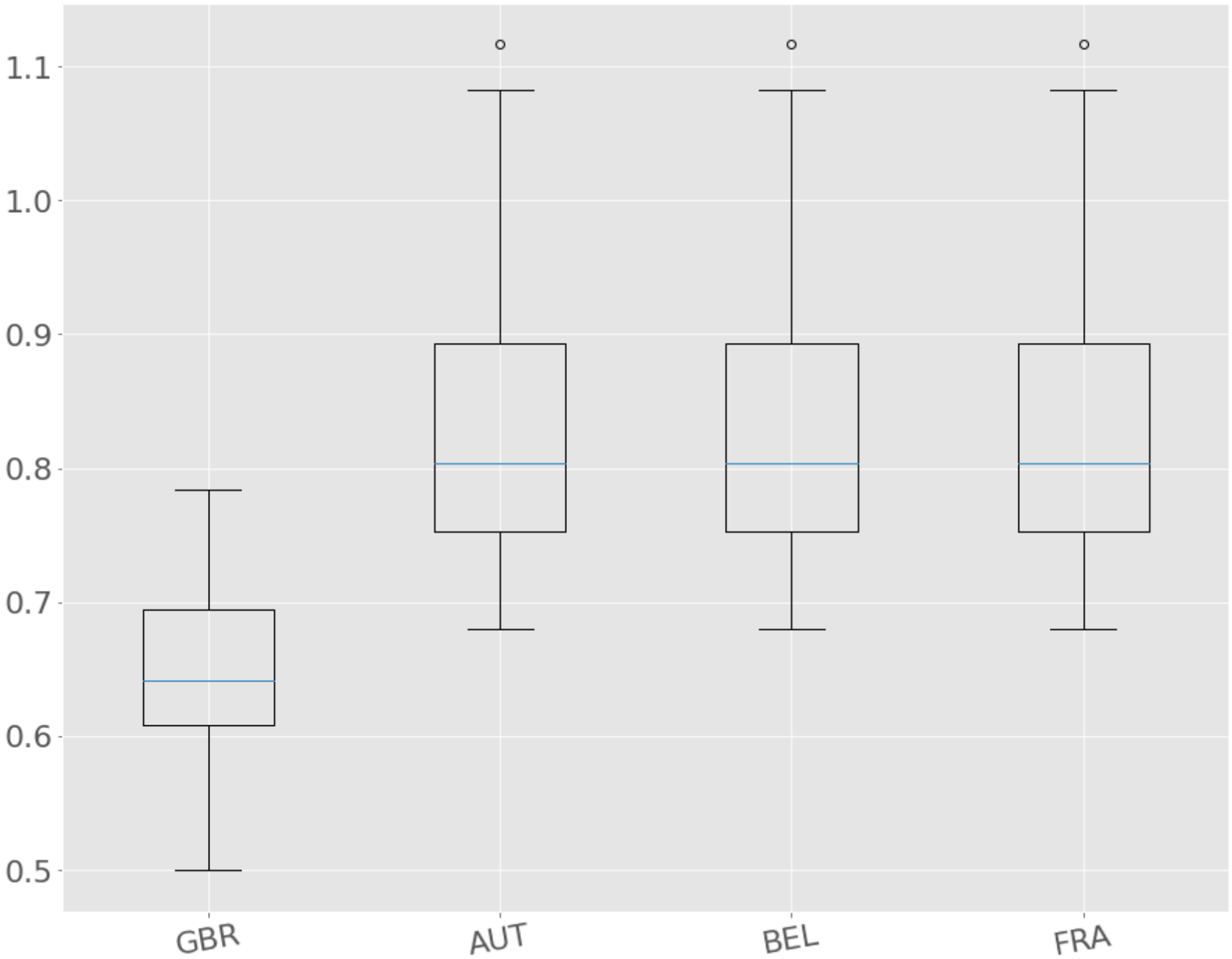
Boxplot with DataFrame data

In [41]:
```
ex_rate_df=pd.read_csv('Exchange_Rates.csv')
ex_rate_df=ex_rate_df[ex_rate_df['LOCATION'].isin(['AUT','BEL','FRA','GBR'])]
boxplot_ex_rate_df= pd.crosstab(ex_rate_df['TIME'],ex_rate_df['LOCATION'],values=ex_rate_df['Value'],aggfunc='mean')

boxplot_ex_rate_df.head()
```

Out[41]:

LOCATION	AUT	BEL	FRA	GBR
TIME				
2000	1.082705	1.082705	1.082705	0.660931
2001	1.116533	1.116533	1.116533	0.694655
2002	1.057559	1.057559	1.057559	0.667223
2003	0.884048	0.884048	0.884048	0.612472
2004	0.803922	0.803922	0.803922	0.546180

In [42]:
```
GBR=boxplot_ex_rate_df['GBR']
AUT=boxplot_ex_rate_df['AUT']
BEL=boxplot_ex_rate_df['BEL']
FRA=boxplot_ex_rate_df['FRA']

columns=[GBR,AUT,BEL,FRA]

plt.figure(figsize=(15,12)) # Set figure size
plt.boxplot(columns)
plt.xticks([1, 2, 3, 4], ["GBR", "AUT", "BEL", "FRA"], rotation=10)
plt.show()
```


1.1
1.0
0.9
0.8
0.7
0.6
0.5
GBR
AUT
BEL
FRA

Boxplot from Pandas DataFrame

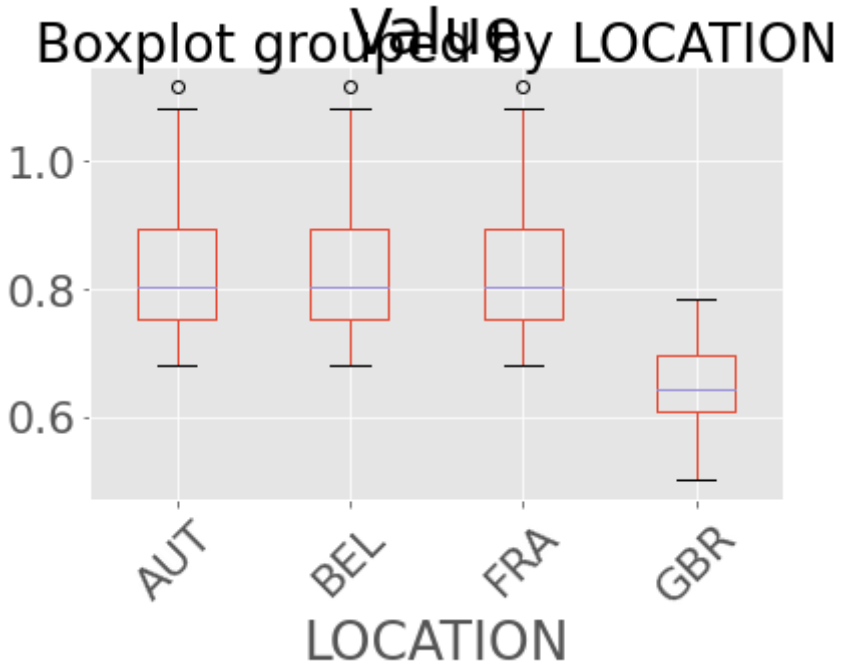
In [43]:
```
ex_rate_df.head()
```

Out[43]:

	LOCATION	INDICATOR	SUBJECT	MEASURE	FREQUENCY	TIME	Value	Flag Codes
21	AUT	EXCH	TOT	NATUSD	A	2000	1.082705	NaN
22	AUT	EXCH	TOT	NATUSD	A	2001	1.116533	NaN
23	AUT	EXCH	TOT	NATUSD	A	2002	1.057559	NaN
24	AUT	EXCH	TOT	NATUSD	A	2003	0.884048	NaN
25	AUT	EXCH	TOT	NATUSD	A	2004	0.803922	NaN

In [44]:
```
ex_rate_df.boxplot(column=['Value'],by="LOCATION",grid=True, rot=45)
```

Out[44]:
```
<AxesSubplot:title={'center':'Value'}, xlabel='LOCATION'>
```



Export Visualizations in Matplotlib

Create DataFrame

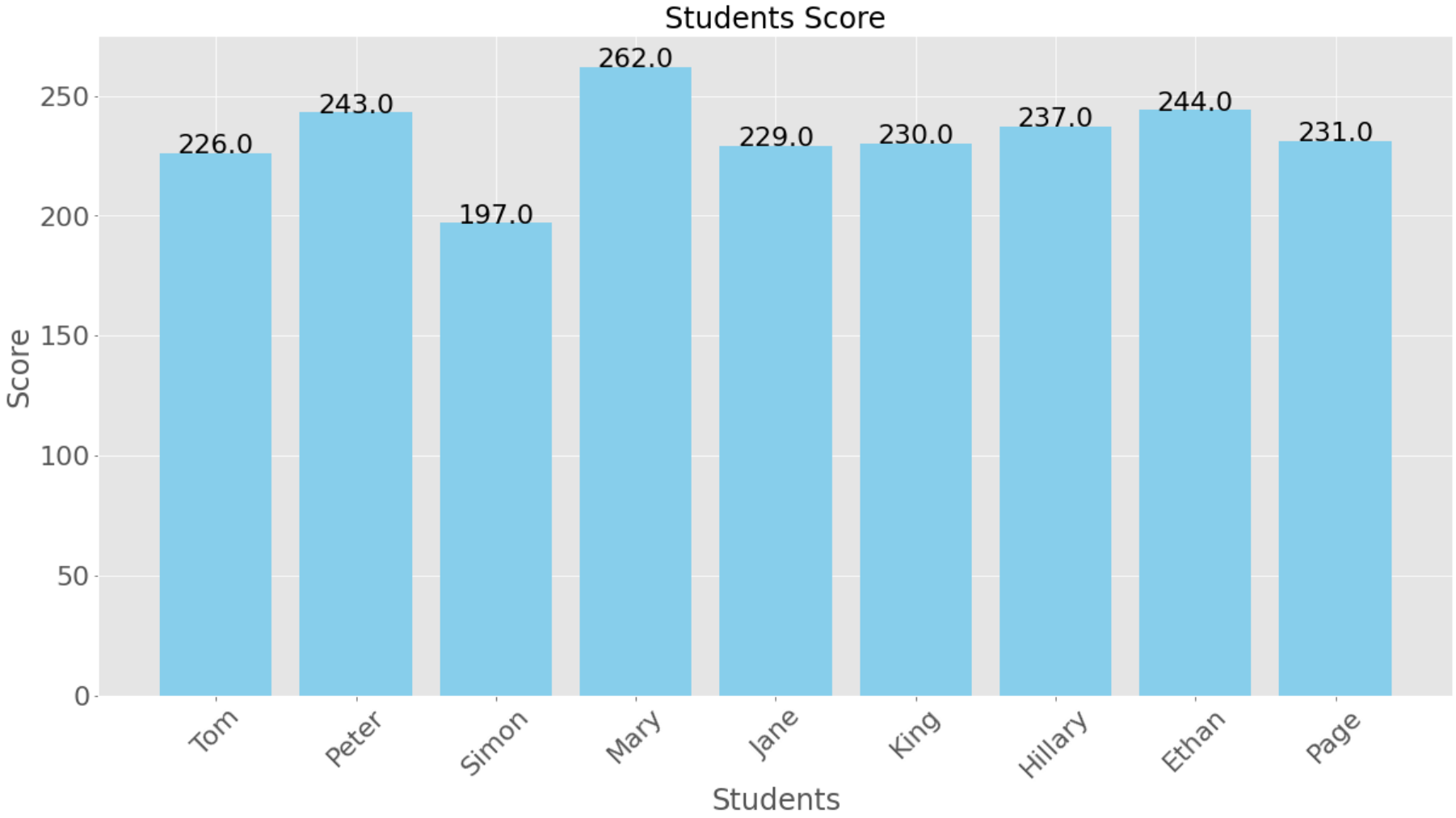
In [45]:
```
score_df = pd.DataFrame(
    {
        "Students": ["Tom", "Peter","Simon", "Mary", "Jane","King","Hillary","Ethan","Page"],
        "Math": [79.00, 67.00,80.00, 84.00, 70.00,60.00,90.00,76.00,75],
        "Physics":[63.00, 98, 60.00, 90,84.00, 77.00,55.00,70,66.00],
        "Computer":[84.00,78.00, 57.00, 88.00, 75.00,93.00,92.00,98.00,90.00],
    },
    index=["Tom", "Peter","Simon", "Mary", "Jane","King","Hillary","Ethan","Page"]
)

score_df['Total']=score_df[['Math','Physics','Computer']].apply(np.sum,axis=1)
score_df
```

Out[45]:

	Students	Math	Physics	Computer	Total
Tom	Tom	79.0	63.0	84.0	226.0
Peter	Peter	67.0	98.0	78.0	243.0
Simon	Simon	80.0	60.0	57.0	197.0
Mary	Mary	84.0	90.0	88.0	262.0
Jane	Jane	70.0	84.0	75.0	229.0
King	King	60.0	77.0	93.0	230.0
Hillary	Hillary	90.0	55.0	92.0	237.0
Ethan	Ethan	76.0	70.0	98.0	244.0
Page	Page	75.0	66.0	90.0	231.0

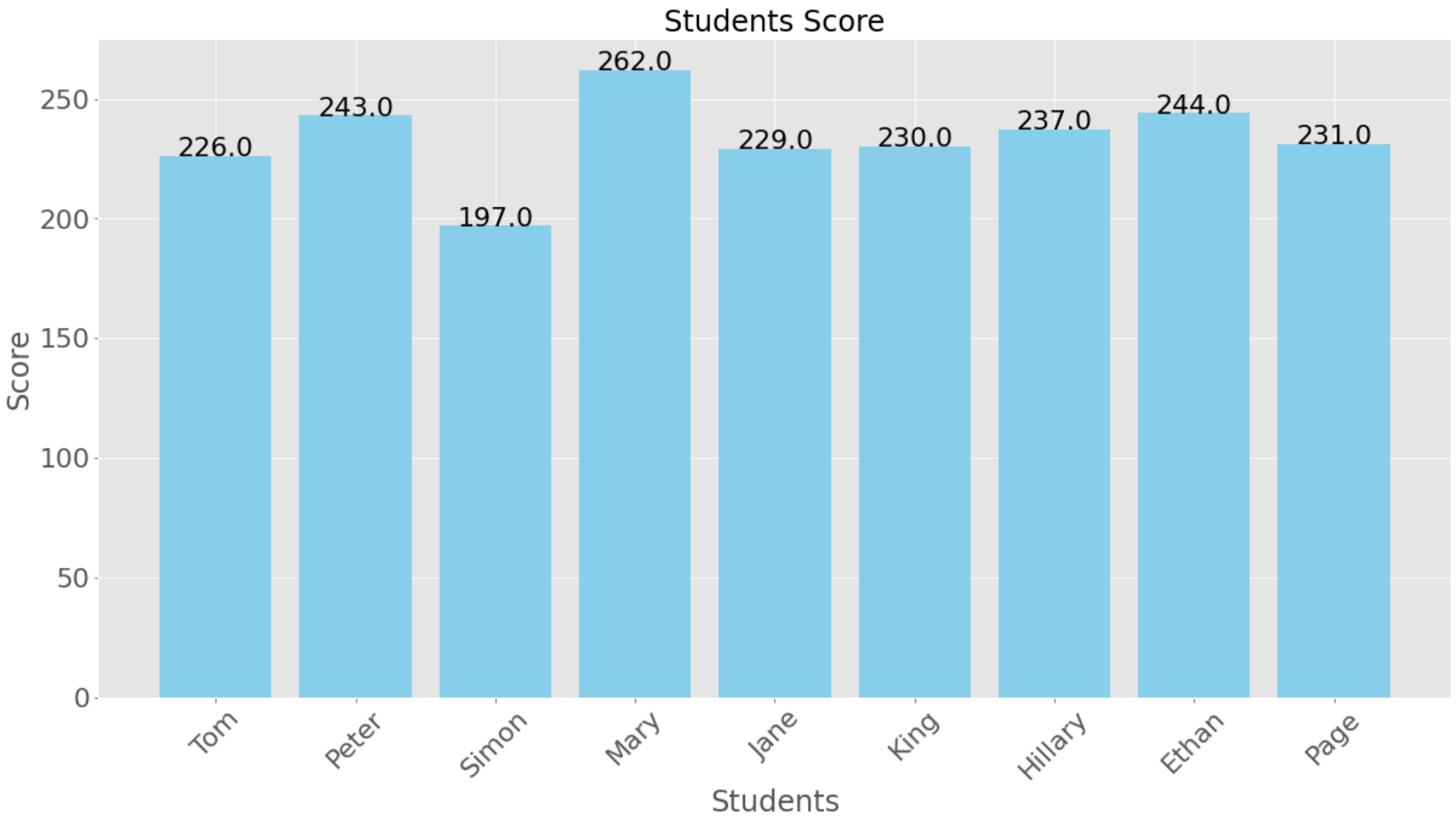
Save graphs as images

In [46]:
```
def addlabels(x,y):
    for i in range(len(x)):
        plt.text(i, y[i], y[i], ha = 'center')


plt.figure(figsize=(20,10)) # Set figure size
plt.bar(score_df['Students'],score_df['Total'],color='skyblue')
addlabels(score_df['Students'],score_df['Total'])
plt.title('Students Score',fontsize=24)
plt.xticks(rotation=45)
plt.xlabel('Students',fontsize=24)
plt.ylabel('Score',fontsize=24)
plt.savefig('Students_Score_Bar_Chart.png',bbox_inches='tight') # Save visualization as image
plt.show()
```


Students Score
Score
Students
0
50
100
150
200
250
226.0
243.0
197.0
262.0
229.0
230.0
237.0
244.0
231.0
Tom
Peter
Simon
Mary
Jane
King
Hillary
Ethan
Page

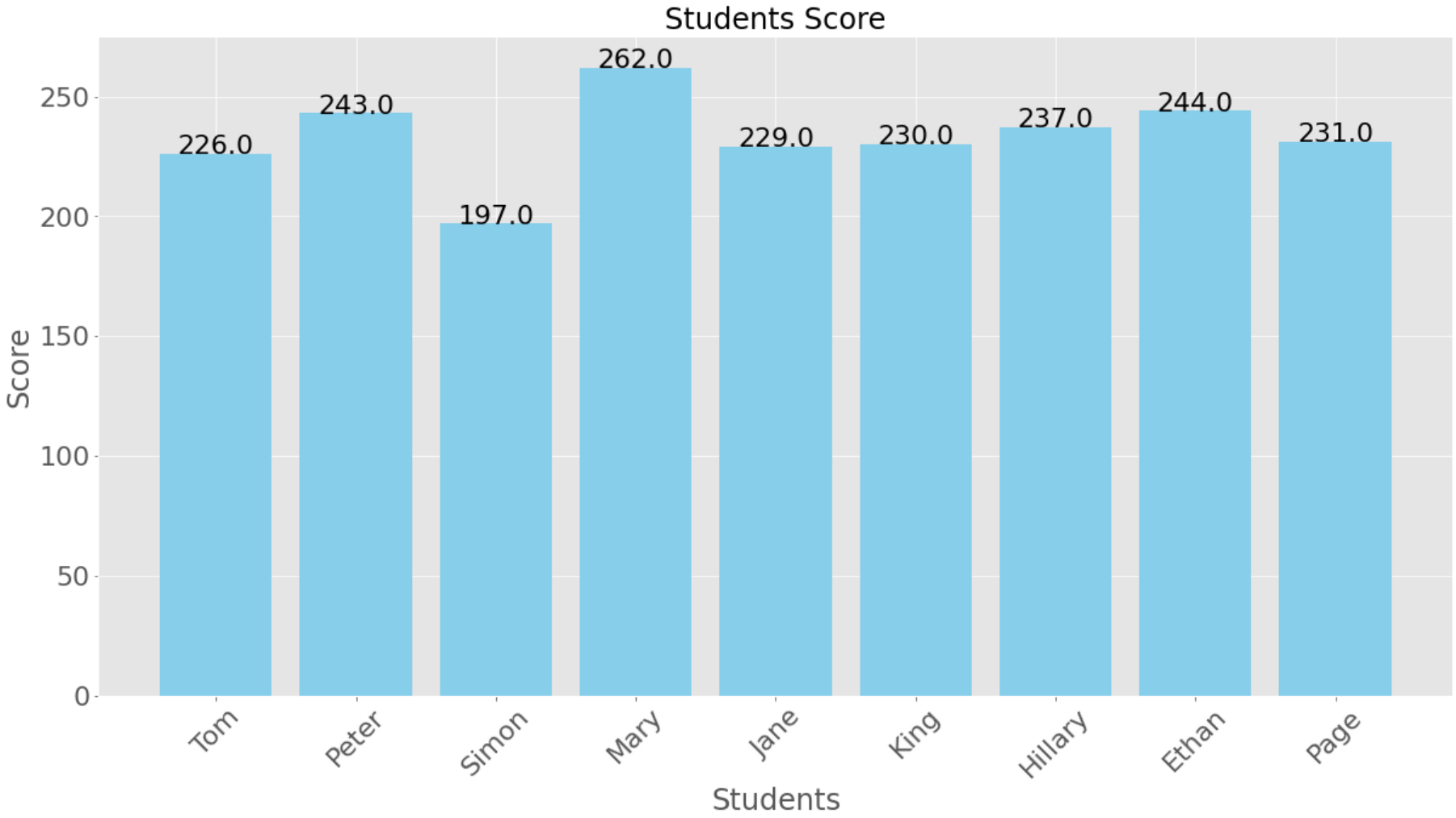
In [47]:
```
from IPython.display import Image
Image('Students_Score_Bar_Chart.png')
```

Out[47]:



Save graphs as PDF

In [48]:
```
def addlabels(x,y):
    for i in range(len(x)):
        plt.text(i, y[i], y[i], ha = 'center')


plt.figure(figsize=(20,10)) # Set figure size
plt.bar(score_df['Students'],score_df['Total'],color='skyblue')
addlabels(score_df['Students'],score_df['Total'])
plt.title('Students Score',fontsize=24)
plt.xticks(rotation=45)
plt.xlabel('Students',fontsize=24)
plt.ylabel('Score',fontsize=24)
plt.savefig('Students_Score_Bar_Chart.pdf',bbox_inches='tight') # Save visualization as image
plt.show()
```



Save graphs to Powerpoint

First we have to install python-pptx. If you're using anaconda open Anaconda Prompt terminal and run the command below: pip install python-pptx.

In [49]:
```
from pptx import Presentation
from pptx.util import Inches

ppt = Presentation() # ppt object
slide = ppt.slides.add_slide(ppt.slide_layouts[0]) # Add title slide
slide.shapes.title.text = "Students Score" # Add title
slide2 = ppt.slides.add_slide(ppt.slide_layouts[1]) # Add Bar chart slide
slide2.shapes.title.text = "Students Score Bar Graph" # Add title
slide2.shapes.add_picture('Students_Score_Bar_Chart.png', left= Inches(0.5), top = Inches(1.5),height = Inches(5))
ppt.save('Students_Score.pptx')
```