**Class 1**

**import** pandas **as** pd

who **=** pd**.**read\_csv("WHO.csv")

who **=** pd**.**read\_csv("WHO.csv",parse\_dates**=**["Date\_reported"])

|  |  |
| --- | --- |
| who | who**.**head() |
| type(who)  pandas.core.frame.DataFrame | type(who["Date\_reported"])  pandas.core.series.Series |
| who**.**ndim  2 | who["Date\_reported"]**.**ndim  1 |
| df**.**describe() |  |
| who**.**info() | who**.**Date\_reported **=** pd**.**to\_datetime(who["Date\_reported"])  0 2020-01-04  1 2020-01-05  2 2020-01-06  3 2020-01-07  4 2020-01-08  ...  31871 2020-07-31  31872 2020-07-31  31873 2020-07-31  31874 2020-07-31  31875 2020-07-31  Name: Date\_reported, Length: 31876, dtype: datetime64[ns] |
| who**.**Date\_reported**.**sort\_values() |
| who**.**shape  (31876, 6) | who**.**columns |
|  | poland **=** who[who**.**Country **==** "Poland"]  poland |

**Class 2**

|  |  |
| --- | --- |
| itay **=** who[who["Country"] **==** "Italy"]  **import** matplotlib.pyplot **as** plt  plt**.**plot(itay**.**Date\_reported,itay**.**Cumulative\_cases)  plt**.**show() | belaru **=** who[who**.**Country **==** "Belarus"]  **import** plotly.express **as** px  fig **=** px**.**line(belaru,"Date\_reported","New\_cases")  fig**.**update\_layout(hovermode**=**"x unified")  fig**.**show()  line\_chart("Belarus")  plt**.**plot(belaru**.**Date\_reported,belaru**.**New\_cases)  plt**.**show()    belaru**.**New\_cases**.**max()  1934  belaru**.**New\_cases**.**head() |
| **def** line\_chart(country):  c\_df **=** who[who["Country"]**==**country]  **if** country **==** "Italy":  plt**.**title(f"{country}:logistic convid commulaive cases")  **if** country **==** "India":  plt**.**title(f"{country}:Exponential convid commulaive cases")  **if** country **==** "Poland":  plt**.**title(f"{country}:Linear convid commulaive cases")  **if** country **==** "Italy":  plt**.**title(f"{country}:logrithmics convid commulaive cases")  plt**.**xlabel("Date")  plt**.**ylabel("Commulaive cases")  plt**.**ticklabel\_format(axis **=** "both")  plt**.**plot(c\_df**.**Date\_reported,c\_df**.**Cumulative\_cases)  plt**.**show() | plt**.**plot(uk**.**Date\_reported,uk**.**Cumulative\_cases,label **=** "The Uk")  plt**.**plot(France**.**Date\_reported,France**.**Cumulative\_cases,label **=** "The France")  plt**.**legend()  plt**.**show() |
|  | fig **=** px**.**line(France,"Date\_reported","New\_cases",title **=** "France:New Case", labels**=**{"Date\_reported":"Date","New\_cases":"New Cases"})  fig**.**update\_traces(mode**=**"markers+lines")  fig**.**update\_layout(hovermode**=**"x unified")  fig**.**show()  In [63]:  **import** plotly.graph\_objects **as** go  fig **=** go**.**Figure()    fig**.**add\_trace(go**.**Scatter(x **=** uk**.**Date\_reported,y **=** uk**.**Cumulative\_cases ,name **=** "UK"))  fig**.**add\_trace(go**.**Scatter(x **=** France**.**Date\_reported,y **=** France**.**Cumulative\_cases,name **=** "France"))  fig**.**show() |

**Class 3**

|  |
| --- |
| url **=** "https://archive.ics.uci.edu/static/public/275/bike+sharing+dataset.zip"  **import** requests  requests**=**requests**.**get(url) |
| **with** open("C:\\Users\\hp\\Desktop\\ned\\Data Visualization\\bike.zip","wb") **as** f:  f**.**write(requests**.**content) |
| **import** zipfile  **with** zipfile**.**ZipFile("C:\\Users\\hp\\Desktop\\ned\\Data Visualization\\bike.zip") **as** zipped:  zipped**.**extractall("C:\\Users\\hp\\Desktop\\ned\\Data Visualization\\bikeshared") |
| **import** json  **import** pandas **as** pd  In [14]:  path **=** "C:\\Users\\hp\\Desktop\\ned\\Data Visualization\\bikeshared\\day.csv"  bike **=** pd**.**read\_csv(path) |
| bike['dteday'] **=** pd**.**to\_datetime(bike**.**dteday) |
| **import** matplotlib.pyplot **as** plt  In [24]:  plt**.**plot(bike**.**dteday,bike**.**cnt)  plt**.**show() |
| plt**.**plot(bike**.**dteday,bike**.**casual,label **=** "casual")  plt**.**plot(bike**.**dteday,bike**.**registered,label **=** "registered")  plt**.**show() |
| plt**.**scatter(bike**.**temp,bike**.**cnt)  plt**.**show() |
| bike["cnt"]**.**corr(bike["temp"]) |
| correlaion **=** bike**.**corr()  correlaion["workingday"]  instant -0.004337  season 0.012485  yr -0.002013  mnth -0.005901  holiday -0.253023  weekday 0.035790  workingday 1.000000  weathersit 0.061200  temp 0.052660  atemp 0.052182  hum 0.024327  windspeed -0.018796  casual -0.518044  registered 0.303907  cnt 0.061156  Name: workingday, dtype: float64 |
| correlaion["workingday"][['casual','registered']]  casual -0.518044  registered 0.303907  Name: workingday, dtype: float64 |

**Class 4**

|  |  |
| --- | --- |
| plt**.**scatter(bike**.**workingday,bike**.**casual)  plt**.**show() | bike[bike["workingday"] **==** 0]  pd**.**pivot\_table(bike,index **=** "workingday" ,values **=** ["registered","casual"],aggfunc**=**"mean") |
| bike**.**index  RangeIndex(start=0, stop=731, step=1) | bike[bike['workingday'] **==** 0][["registered","casual"]]**.**mean() |
| w\_mean['casual']**.**plot**.**bar()  plt**.**xticks(ticks **=** [0,1],labels **=** ["non-working","working"],rotation**=**0 )  plt**.**show() | w\_mean**=**pd**.**pivot\_table(bike,index **=** "workingday" ,values **=** ["registered","casual"],aggfunc**=**"mean")  w\_mean |
| bike['dteday']**.**dt**.**day  bike[['dteday','weekday']] | bike['dteday']**.**dt**.**strftime("%A %B")    bike**.**pivot\_table(index **=**'workingday' ,values **=** ["registered","casual"],aggfunc **=** 'mean')    week\_group **=** bike**.**groupby("weekday")  pd**.**options**.**display**.**max\_rows **=** **None**  week\_group**.**groups  week\_group**.**get\_group(1) |
| bike**.**pivot\_table(index **=**'workingday' ,values **=** ["registered","casual"],aggfunc **=** 'mean') | week **=** week\_group[['registered','casual']]**.**mean()  week |
| week['registered']**.**plot**.**bar()  plt**.**show() | bike**.**weathersit**.**unique()  (bike**.**weathersit**.**value\_counts(normalize**=**'True')**\***100)**.**plot**.**bar()  plt**.**show() |
| (bike**.**weathersit**.**value\_counts(normalize**=**'True')**\***100)**.**plot**.**pie()  plt**.**show() | bike**.**yr**.**map({0:2011,1:2012})  bike**.**yr**.**replace({0:2011,1:2012},inplace **=** **True**) |
| Bifurcated\_Data**=** bike**.**groupby("yr")["weathersit"]**.**value\_counts()  Bifurcated\_Data    Bifurcated\_Data[2012]**.**plot**.**bar() | Bifurcated\_Data[2011]    Bifurcated\_Data[2011]**.**plot**.**bar() |
| **import** seaborn **as** sns  bike["weathersit\_cat"] **=**bike**.**weathersit**.**map({1:"Clear",2:"Cloudy",3:"Rain"})  sns**.**countplot(data**=**bike,x**=**"yr",hue**=** "weathersit\_cat")  plt**.**show()  bike**.**cnt**.**unique()**.**size | sns**.**countplot(data**=**bike,x**=**"yr",hue**=** "weathersit")  plt**.**show()    bike**.**cnt**.**unique()**.**size  696  bike**.**cnt**.**value\_counts(bins**=**10) |
| bike**.**cnt**.**value\_counts(bins**=**10)**.**sort\_index() | bike**.**cnt**.**value\_counts(bins**=**10)**.**sort\_index()**.**plot**.**hist() |
| cnt **=** bike**.**cnt**.**value\_counts(bins**=**10)**.**sort\_index()  plt**.**hist(bike['cnt'])  plt**.**show() |  |

**Class 5**

|  |  |
| --- | --- |
| df[df["Salary"]**>**111] | df[["Salary","Age"]] |
| laptop**.**isnull()**.**sum() | laptop["year"]**.**fillna(2023,inplace**=True**)  laptop["date"] **=** pd**.**to\_datetime(laptop["date"],"%Y-%M-%D")  print(laptop["date"]) |
| **laptop.iloc[(rows),(columns)]**  laptop**.**iloc[500:600,1:2]  **for** i **in** laptop**.**columns:  print(i, ":" , laptop[i]**.**unique()) | group **=** laptop**.**groupby("TypeName")["Company"]  In [52]:  pd**.**DataFrame(group,columns **=**["TypeName","Company"]) |
| laptop["Weight"] **=** laptop["Weight"]**.**str**.**replace("kg","")  laptop["Weight"] **=** laptop["Weight"]**.**astype(float)  laptop["Weight"] | pd**.**DataFrame(laptop**.**groupby("TypeName")["Weight"]**.**sum()**.**reset\_index()) |
| ids **=** {'Apple':1, 'HP':2, 'Acer':3, 'Asus':4, 'Dell':5, 'Lenovo':6, 'Chuwi':7, 'MSI':8,  'Microsof':9, 'Toshiba':10, 'Huawei':11, 'Xiaomi':12, 'Vero':13, 'Razer':14,  'Mediacom':15, 'Samsung':16, 'Google':17, 'Fujitsu':18, 'LG':19}  print(ids) | laptop**.**rename(columns **=** {"Company":"New\_Compnay"},inplace**=** **True**)  laptop**.**drop(columns **=** {"Unnamed: 0"},inplace**=** **True**) |
| df1 **=** laptop[["New\_Compnay","TypeName"]]  df2 **=** laptop[["Inches","ScreenResolution"]]  pd**.**concat([df1,df2] , axis **=** 1)  df2**.**rename(columns **=**{"Inches":"New\_Compnay","ScreenResolution":"TypeName"},inplace**=** **True**)  pd**.**concat([df1,df2] , axis **=** 0) | **import** datetime **as** dt  laptop["date"]**.**dt**.**day |
| url **=** "https://archive.ics.uci.edu/static/public/483/behavior+of+the+urban+traffic+of+the+city+of+sao+paulo+in+brazil.zip"  **import** requests  responce **=** requests**.**get(url)  responce | **with** open("city.zip" ,"wb") **as** f:  f**.**write(responce**.**content)  In [95]:  pwd |
| **import** zipfile  In [105]:  **with** zipfile**.**ZipFile("city.zip") **as** zipped:  zipped**.**extractall("brazil\_city")  In [106]:  **import** pandas **as** pd  In [108]:  taffic **=** pd**.**read\_csv(r"C:\Users\hp\Desktop\ned\Data Visualization\brazil\_city\Behavior of the urban traffic of the city of Sao Paulo in Brazil\Behavior of the urban traffic of the city of Sao Paulo in Brazil.csv")  In [110]:  taffic**.**head() | taffic **=** pd**.**read\_csv(r"C:\Users\hp\Desktop\ned\Data Visualization\brazil\_city\Behavior of the urban traffic of the city of Sao Paulo in Brazil\Behavior of the urban traffic of the city of Sao Paulo in Brazil.csv",sep**=**';')  In [112]:  taffic**.**head()  taffic**.**isnull()**.**sum() |
| taffic**.**index  RangeIndex(start=0, stop=135, step=1) | taffic**.**columns |
| taffic**.**isnull()**.**sum(axis **=** "index") | taffic**.**isnull()**.**sum(axis **=** "columns") |
| taffic**.**isnull()**.**sum(axis **=** 1) | taffic["Slowness in traffic (%)"]**.**str**.**replace(",",".",inplace **=True**)  Error  taffic["Slowness in traffic (%)"] **=** taffic["Slowness in  traffic(%)"]**.**str**.**replace(",",".")**.**astype(float)  taffic["Slowness in traffic (%)"] |
| taffic**.**dtypes | taffic["Slowness in traffic (%)"]**.**value\_counts()    taffic["Slowness in traffic (%)"]**.**value\_counts(bins**=**10) |
| taffic["Slowness in traffic (%)"]**.**value\_counts(bins**=**10)**.**sort\_index() | taffic["Slowness in traffic (%)"]**.**plot**.**hist() |
| **import** matplotlib.pyplot **as** plt  plt**.**hist(taffic["Slowness in traffic (%)"])  plt**.**show() | incidence **=** taffic**.**drop(['Hour (Coded)','Slowness in traffic (%)'] ,axis **=**1)**.**copy() *# axis column*  incidence |
| incidence**.**sum()    incidence**.**sum()**.**sum()   numeriac values -->scatter plotcategorical and numeric -->barplotcontinious values -->histogrmtime against value -->line chart | incidence**.**sum()**.**plot**.**barh()    taffic**.**corr()['Slowness in traffic (%)']**.**drop("Hour (Coded)")**.**abs()    taffic**.**corr()['Slowness in traffic (%)']**.**drop("Hour (Coded)")**.**sort\_values(ascending **=** **False**) |
| plt**.**scatter(taffic["Slowness in traffic (%)"],taffic["Lack of electricity"]) | plt**.**scatter(taffic["Slowness in traffic (%)"],taffic["Point of flooding"]) |

**Class 6**

|  |
| --- |
| df["Slowness in traffic (%)"] **=** df["Slowness in traffic(%)"]**.**str**.**replace(",",".")**.**astype(float) |
| df\_20 **=** df[df["Slowness in traffic (%)"] **>**20] |
| df\_20**.**drop(["Slowness in traffic (%)","Hour (Coded)"],axis **=** 1) |
| df\_20**.**drop(["Slowness in traffic (%)","Hour (Coded)"],axis **=** 1)**.**sum()**.**sort\_values(ascending **=** **False**) |
| iloc- Last loaction not includedloc - last location is included df.loc[:27]  df[["Hour (Coded)","Immobilized bus"]] |
| for i in range(0,len(df),27):  print(i)  for i in zip(["waqar","waqas","Amara"],[1,2,3]):  print(i)  ('waqar', 1)  ('waqas', 2)  ('Amara', 3)  for i,j in zip(["waqar","waqas","Amara"],[1,2,3]):  print(i)  waqar  waqas  Amara  days = ["Mon","Tue","Wed","Thu","Fri"]  traffic\_per\_day ={}  for day,j in zip (days,range(0,len(df),27)):  print(day,j)  Mon 0  Tue 27  Wed 54  Thu 81  Fri 108  days = ["Mon","Tue","Wed","Thu","Fri"]  traffic\_per\_day ={}  for day,i in zip(days,range(0,len(df),27)):  daily\_data = df.iloc[i:i+27]    traffic\_per\_day[day] = daily\_data |
| for df in traffic\_per\_day.values():  plt.plot(df["Hour (Coded)"],df["Slowness in traffic (%)"])  plt.show()    for day,df in traffic\_per\_day.items():  plt.plot(df["Hour (Coded)"],df["Slowness in traffic (%)"],label =day)  plt.legend()  plt.ylim(0,25)  plt.show() |
| plt.subplot(3,2,2)  plt.subplot(3,2,3)  plt.subplot(3,2,6)  plt.show()    plt.figure(figsize=(10,12))  plt.subplot(3,2,2)  plt.subplot(3,2,3)  plt.subplot(3,2,6)  plt.show()    plt.figure(figsize=(10,12))  for i,df in zip(range(0,5), traffic\_per\_day.values()):  plt.subplot(3,2,i+1)  plt.plot(df["Hour (Coded)"],df["Slowness in traffic (%)"])  plt.title(days[i])  plt.ylim(0,25)  plt.show()      plt.figure(figsize=(10,12))  for i,df in zip(range(0,5), traffic\_per\_day.values()):  plt.subplot(3,2,i+1)  plt.plot(df["Hour (Coded)"],df["Slowness in traffic (%)"])  plt.title(days[i])  plt.ylim(0,25)    plt.subplot(3,2,6)  for day,df in traffic\_per\_day.items():  plt.plot(df["Hour (Coded)"],df["Slowness in traffic (%)"],label =day)  plt.legend()  plt.ylim(0,25)  plt.title("comparision")  plt.savefig("traffic\_variation.png")  plt.show() |
| pro **=** pd**.**read\_csv("property.csv")  In [69]: pro**.**head()  **import** seaborn **as** sns  sns**.**relplot(data**=**pro,x**=**'Gr Liv Area',y**=**'SalePrice')  plt**.**show() |
| sns**.**set\_theme()  sns**.**relplot(data**=**pro,x**=**'Gr Liv Area',y**=**'SalePrice')  plt**.**show() |
| sns**.**relplot(data**=**pro,x**=**'Gr Liv Area',y**=**'SalePrice',  hue**=**'Overall Qual',palette**=**"RdYlGn")  plt**.**show() |
| sns**.**relplot(data**=**pro,x**=**'Gr Liv Area',y**=**'SalePrice',  hue**=**'Overall Qual',palette**=**"RdYlGn",  size**=** 'Garage Area')  plt**.**show() |
| sns**.**relplot(data**=**pro,x**=**'Gr Liv Area',y**=**'SalePrice',  hue**=**'Overall Qual',palette**=**"RdYlGn",  size**=** 'Garage Area',sizes**=**(1,300))  plt**.**show() |
| sns**.**relplot(data**=**pro,x**=**'Gr Liv Area',y**=**'SalePrice',  hue**=**'Overall Qual',palette**=**"RdYlGn",  size**=** 'Rooms')  plt**.**show() |
| sns**.**relplot(data**=**pro,x**=**'Gr Liv Area',y**=**'SalePrice',  hue**=**'Overall Qual',palette**=**"RdYlGn",  size**=** 'Rooms',sizes**=**[200,30])  plt**.**show() |
| sns**.**relplot(data**=**pro,x**=**'Gr Liv Area',y**=**'SalePrice',  hue**=**'Overall Qual',palette**=**"RdYlGn",  size**=** 'Garage Area',sizes**=**(1,300)  ,style**=**"Rooms")  plt**.**show() |
| sns**.**relplot(data**=**pro,x**=**'Gr Liv Area',y**=**'SalePrice',  hue**=**'Overall Qual',palette**=**"RdYlGn",  size**=** 'Garage Area',sizes**=**(1,300)  ,style**=**"Rooms",markers **=**["^","v"] )  plt**.**show() |
| sns**.**relplot(data**=**pro,x**=**'Gr Liv Area',y**=**'SalePrice',  hue**=**'Overall Qual',palette**=**"RdYlGn",  size**=** 'Garage Area',sizes**=**(1,300)  ,style**=**"Rooms",col **=** "Year" )  plt**.**show() |
| pro["Land Slope"]**.**unique()  sns**.**relplot(data**=**pro,x**=**'Gr Liv Area',y**=**'SalePrice',  hue**=**'Overall Qual',palette**=**"RdYlGn",  size**=** 'Garage Area',sizes**=**(1,300)  ,style**=**"Rooms",col **=** "Year", row **=** "Land Slope")  plt**.**show() |

Class 7

|  |
| --- |
| url **=** "https://archive.ics.uci.edu/static/public/492/metro+interstate+traffic+volume.zip"  **import** requests  responce **=** requests**.**get(url) |
| **with** open("Metro Interstate Traffic Volume.zip","wb") **as** f:  f**.**write(responce**.**content) |
| **import** zipfile  **with** zipfile**.**ZipFile("Metro Interstate Traffic Volume.zip") **as** zipped:  zipped**.**extractall("Metro Interstate Traffic Volume") |
| **import** pandas **as** pd  i\_94 **=** pd**.**read\_csv(r"Metro\_Interstate\_Traffic\_Volume.csv") |
| new\_df **=** i\_94[**~**i\_94["holiday"]**.**str**.**contains("None")]  new\_df |
| **import** matplotlib.pyplot **as** plt  i\_94**.**traffic\_volume**.**plot**.**hist()  plt**.**show() |
| i\_94**.**date\_time **=** pd**.**to\_datetime(i\_94**.**date\_time)  i\_94**.**date\_time**.**dtype |
| day\_bol **=** (i\_94**.**date\_time**.**dt**.**hour **>=** 7 ) **&** (i\_94**.**date\_time**.**dt**.**hour **<** 19 )  day **=** i\_94[day\_bol]  day    day**.**date\_time**.**max()  Timestamp('2018-09-30 18:00:00')  day**.**date\_time**.**dt**.**time    In [36]:  day**.**date\_time**.**dt**.**time**.**min()  Out[37]:  datetime.time(7, 0)  In [38]:  day**.**date\_time**.**dt**.**time**.**max()  Out[38]:  datetime.time(18, 0)  In [39]:  night\_bol **=** (i\_94**.**date\_time**.**dt**.**hour **>=** 19 ) **|** (i\_94**.**date\_time**.**dt**.**hour **<** 7 )  night**=**i\_94[night\_bol]  In [40]:  night**.**date\_time**.**dt**.**hour**.**unique()  Out[44]:  array([19, 20, 21, 22, 23, 0, 1, 2, 3, 4, 5, 6], dtype=int64)  In [45]:  day**.**date\_time**.**dt**.**hour**.**unique()  Out[45]:  array([ 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 8, 7], dtype=int64) |
| plt**.**figure(figsize**=**(11,4))  plt**.**subplot(1,2,1)  plt**.**hist(day["traffic\_volume"])  plt**.**title("traffic volumne :day")  plt**.**xlim(0,8000)  plt**.**ylim(0,8500)  plt**.**xlabel("traffic volume")  plt**.**ylabel("number of incident")  plt**.**subplot(1,2,2)  plt**.**hist(night["traffic\_volume"])  plt**.**title("traffic volumne :night")  plt**.**xlim(0,8000)  plt**.**ylim(0,8500)  plt**.**xlabel("traffic volume")  plt**.**show() |
| night**.**traffic\_volume**.**value\_counts(bins**=**10)**.**sort\_index() |
| day["month"] **=**day**.**date\_time**.**dt**.**strftime("%B")  day |
| group **=** day**.**groupby("month")  by\_month **=** group["traffic\_volume"]**.**mean()  by\_month |
| by\_month**.**plot**.**line()  plt**.**show() |
| day["year"] **=** day**.**date\_time**.**dt**.**year  only\_july **=** day[day["month"]**==**"July"]  yearly\_group **=** only\_july**.**groupby("year")["traffic\_volume"]**.**mean()  yearly\_group    only\_july**.**groupby("year")["traffic\_volume"]**.**mean()**.**plot**.**line()  plt**.**show() |

Class 8

|  |
| --- |
| **import** pandas **as** pd  i\_94 **=** pd**.**read\_csv("Metro\_Interstate\_Traffic\_Volume.csv")  i\_94**.**date\_time **=** pd**.**to\_datetime(i\_94**.**date\_time)  i\_94**.**date\_time**.**dt**.**strftime("%A") |
| avrageDayofWeek **=** i\_94**.**groupby("dayofweek")["traffic\_volume"]**.**mean()  avrageDayofWeek    avrageDayofWeek**.**plot**.**line()    avrageDayofWeek**.**plot(kind**=**"line") |
| i\_94["hours"] **=** i\_94**.**date\_time**.**dt**.**hour  day\_bol **=** (i\_94**.**date\_time**.**dt**.**hour **>=** 7) **&** (i\_94**.**date\_time**.**dt**.**hour **<**19)  day **=** i\_94[day\_bol]  day  day["dayinwords"] **=** day**.**date\_time**.**dt**.**strftime("%A") |
| avrageDayofWeek **=** day**.**groupby("dayofweek")["traffic\_volume"]**.**mean()  avrageDayofWeek  avrageDayofWeek**.**plot**.**line() |
| weekend **=** day[**~**(day["dayofweek"]**<=**4)]  weekend  weekdays**.**pivot\_table(index**=**"hours",values**=**"traffic\_volume",) |
| **import** matplotlib.pyplot **as** plt  plt**.**figure(figsize**=**(11,4))  plt**.**subplot(1,2,1)  Hourly\_weekday\_avg **=** weekdays**.**pivot\_table(index**=**"hours",values **=** "traffic\_volume")  plt**.**plot(Hourly\_weekday\_avg**.**index,Hourly\_weekday\_avg)  plt**.**ylim(1500,6500)  plt**.**subplot(1,2,2)  Hourly\_weekend\_avg **=** weekend**.**pivot\_table(index**=**"hours",values **=** "traffic\_volume")  plt**.**plot(Hourly\_weekend\_avg**.**index,Hourly\_weekend\_avg)  plt**.**ylim(1500,6500)  plt**.**show() |
| day**.**select\_dtypes(include**=**"number")    day**.**select\_dtypes(include**=**"number")**.**corr()["traffic\_volume"]    plt**.**scatter(day["temp"],day**.**traffic\_volume)    day**.**groupby("weather\_main")["traffic\_volume"]**.**mean()    day**.**groupby("weather\_main")["traffic\_volume"]**.**mean()**.**plot**.**barh()  plt**.**figure(figsize**=**(6,11))  day**.**groupby(["weather\_main","weather\_description"])["traffic\_volume"]**.**mean()**.**plot**.**barh()    plt**.**figure(figsize**=**(6,11))  day**.**groupby(["weather\_main","weather\_description"])["traffic\_volume"]**.**mean()**.**plot**.**barh()    plt**.**figure(figsize**=**(5,8))  day**.**groupby("weather\_description")["traffic\_volume"]**.**mean()**.**plot**.**barh()    day["weather\_description"]**.**value\_counts()**.**sort\_values(ascending**=** **False**)    day["weather\_description"] **=** day["weather\_description"]**.**str**.**replace("Sky is Clear","sky is clear")  day["weather\_description"]**.**value\_counts()**.**sort\_values(ascending**=** **False**) |

Class 9

|  |  |
| --- | --- |
| **import** pandas **as** pd  **import** matplotlib.pyplot **as** plt  In [3]:  death **=** pd**.**read\_csv("deaths.csv")  death | |
| **design princilple.. gendrate the disign option.. choose among those option**   * data element * structral elements * decoration   **-->familiarty principle**  **-->maximizig the data ink ratio =(data ink)/(data +sturutal + decotion)**  **sturutal + decotion== non dat ink**  **most customization offered by matplotlib**  **color of company logo make that color of graph** | |
| plt**.**barh(death**.**Country\_Other,death**.**Total\_Deaths)*#functinal interface*  plt**.**show() | *#oo inteface*  fig,ax **=** plt**.**subplots()*#oo interface*  ax**.**barh(death**.**Country\_Other,death**.**Total\_Deaths)*#functinal interface*  plt**.**show() |
| *#oo interface*  fig,ax **=** plt**.**subplots(figsize**=**(4.5,6)) *#oo iterface*  ax**.**barh(death**.**Country\_Other,death**.**Total\_Deaths)*#functinal interface*  plt**.**show() | *#oo inteface remove tick*  fig,ax **=** plt**.**subplots() *#oo iterface*  ax**.**barh(death**.**Country\_Other,death**.**Total\_Deaths)*#functinal interface*  ax**.**tick\_params(bottom**=False**,left **=** **False**)  plt**.**show() |
| **for** spines **in** ax**.**spines:  print(spines)  *#oo inteface remove spines*  fig,ax **=** plt**.**subplots() *#oo iterface*  ax**.**barh(death**.**Country\_Other,death**.**Total\_Deaths,height **=** 0.45)*#functinal interface*  ax**.**tick\_params(bottom**=False**,left **=** **False**)  **for** spines\_val **in** ax**.**spines**.**values():  spines\_val**.**set\_visible(**False**)  ax**.**set\_xticks([0,150000,300000])  ax**.**xaxis**.**tick\_top()  ax**.**tick\_params(top**=False**)  plt**.**show()   |  | | --- | |  | | *#oo inteface remove spines*  fig,ax **=** plt**.**subplots() *#oo iterface*  ax**.**barh(death**.**Country\_Other,death**.**Total\_Deaths,height **=** 0.45)*#functinal interface*  ax**.**tick\_params(bottom**=False**,left **=** **False**)  **for** spines\_val **in** ax**.**spines**.**values():  spines\_val**.**set\_visible(**False**)  ax**.**set\_xticks([0,150000,300000])  ax**.**xaxis**.**tick\_top()  ax**.**tick\_params(top**=False**)  plt**.**title("The death trol world wide is 1.5M+")  plt**.**show() |
| *#oo inteface remove spines*  fig,ax **=** plt**.**subplots() *#oo iterface*  ax**.**barh(death**.**Country\_Other,death**.**Total\_Deaths,height **=** 0.45)*#functinal interface*  ax**.**tick\_params(bottom**=False**,left **=** **False**)  **for** spines\_val **in** ax**.**spines**.**values():  spines\_val**.**set\_visible(**False**)  ax**.**set\_xticks([0,150000,300000])  ax**.**xaxis**.**tick\_top()  ax**.**tick\_params(top**=False**)  ax**.**text(x**=-**80000,y**=**23.5,s**=**"The death trol world wide is 1.5M+",size**=**17,weight**=** "bold")  *#plt.title("The death trol world wide is 1.5M+")*  plt**.**show() | *#oo inteface remove spines*  fig,ax **=** plt**.**subplots() *#oo iterface*  ax**.**barh(death**.**Country\_Other,death**.**Total\_Deaths,height **=** 0.45 )*#functinal interface*  ax**.**tick\_params(bottom**=False**,left **=** **False**)  **for** spines\_val **in** ax**.**spines**.**values():  spines\_val**.**set\_visible(**False**)  ax**.**set\_xticks([0,150000,300000])  ax**.**set\_xticklabels(["0","150,000","300,000"])  ax**.**xaxis**.**tick\_top()  ax**.**tick\_params(top**=False**)  ax**.**text(x**=-**80000,y**=**23.5,s**=**"The death trol world wide is 1.5M+",size**=**17,weight**=** "bold")  ax**.**text(x**=-**80000,y**=**22.5,s**=**"Top 20 Countires by death Tool(December 2020)",size**=**12)  *#plt.title("The death trol world wide is 1.5M+")*  plt**.**show() |
| *#oo inteface remove spines*  fig,ax **=** plt**.**subplots() *#oo iterface*  ax**.**barh(death**.**Country\_Other,death**.**Total\_Deaths,height **=** 0.45 )*#functinal interface*  ax**.**tick\_params(bottom**=False**,left **=** **False**)  **for** spines\_val **in** ax**.**spines**.**values():  spines\_val**.**set\_visible(**False**)  ax**.**set\_xticks([0,150000,300000])  ax**.**set\_xticklabels(["0","150,000","300,000"])  ax**.**set\_yticklabels([])  ax**.**xaxis**.**tick\_top()  ax**.**tick\_params(top**=False**)  ax**.**text(x**=-**80000,y**=**23.5,s**=**"The death trol world wide is 1.5M+",size**=**17,weight**=** "bold")  ax**.**text(x**=-**80000,y**=**22.5,s**=**"Top 20 Countires by death Tool(December 2020)",size**=**12)  **for** i,c **in** enumerate(death**.**Country\_Other):  ax**.**text(x**=-**80000,y**=**i**-**0.15,s**=**c)  *#plt.title("The death trol world wide is 1.5M+")*  plt**.**show() | |
| *#oo inteface remove spines*  fig,(ax1,ax2) **=** plt**.**subplots(nrows**=**1,ncols**=**2,figsize**=**(15 , 6)) *#oo iterface*  ax1**.**barh(death**.**Country\_Other,death**.**Total\_Deaths)*#functinal interface*  ax2**.**barh(death**.**Country\_Other,death**.**Total\_Deaths,height **=** 0.45 )*#functinal interface*  ax2**.**tick\_params(bottom**=False**,left **=** **False**)  **for** spines\_val **in** ax2**.**spines**.**values():  spines\_val**.**set\_visible(**False**)  ax2**.**set\_xticks([0,150000,300000])  ax2**.**set\_xticklabels(["0","150,000","300,000"])  ax2**.**set\_yticklabels([])  ax2**.**xaxis**.**tick\_top()  ax2**.**tick\_params(top**=False**)  ax2**.**text(x**=-**80000,y**=**23.5,s**=**"The death trol world wide is 1.5M+",size**=**17,weight**=** "bold")  ax2**.**text(x**=-**80000,y**=**22.5,s**=**"Top 20 Countires by death Tool(December 2020)",size**=**12)  **for** i,c **in** enumerate(death**.**Country\_Other):  ax2**.**text(x**=-**80000,y**=**i**-**0.15,s**=**c)  ax2**.**axvline(150000,ymin**=**0.045,ymax**=**1,alpha **=** 0.5,c**=**"grey")  plt**.**subplots\_adjust(wspace**=**0.5)  *#plt.title("The death trol world wide is 1.5M+")*  plt**.**show() | |

Class 10

|  |
| --- |
| **import** pandas **as** pd  **import** matplotlib.pyplot **as** plt  In [45]:  death **=** pd**.**read\_csv("avg\_deaths.csv")  death |
| **import** calendar  calendar**.**monthrange(2020,1)  Out[49]:  (2, 31)  In [48]:  \_,num\_day **=**calendar**.**monthrange(2020,1)  In [49]:  num\_day  Out[49]:  31 |
| death**.**columns **=** ['Year','Month','Num\_days' ,'New\_deaths', ]  death |
| **def** days(year,month):  \_,num\_days **=** calendar**.**monthrange(year,month)  **return** num\_days  days **=**death[["Year","Month"]]**.**apply(**lambda** row:days(row["Year"],row["Month"]) ,axis**=**1)  days |
| death**.**loc[11,"Num\_days"]**//=**2 |
| death["Monthly\_deaths"] **=** death**.**Num\_days**.**mul(death**.**New\_deaths) |
| death**.**rename({"New\_deaths":"Avg\_deaths"}, axis**=**1, inplace**=True**) |
| death["Cum\_deaths"] **=** death**.**Monthly\_deaths**.**cumsum() |
| fig,(ax1,ax2,ax3,ax4 ) **=** plt**.**subplots(nrows **=** 4,ncols **=** 1,figsize**=**(6,8))  axes **=** [ax1,ax2,ax3,ax4]  **for** ax **in** axes:  ax**.**plot(death["Month"],death["Avg\_deaths"],alpha**=**0.1)  ax**.**set\_xticklabels([])  ax**.**set\_yticklabels([])  ax**.**tick\_params(bottom **=False**,left**=False**)  **for** spines **in** ax**.**spines:  ax**.**spines[spines]**.**set\_visible(**False**)  ax1**.**plot(death["Month"][:3],death["Avg\_deaths"][:3])  ax1**.**text(x**=**0.5,y**=-**80,s**=**"0",alpha **=**0.5)  ax1**.**text(x**=**3.5,y**=** 2000,s**=**"1844",alpha **=**0.5)  ax1**.**text(x**=**11.5,y**=** 2400,s**=**"2247",alpha **=**0.5)  ax1**.**text(x**=**1.5,y**=** **-**300,s**=**"Jan-Mar",rotation**=**3)  ax1**.**text(0.5,3500,"The virius kill 900 people every day",size**=**14, weight **=** "bold")  ax1**.**text(0.5,3100,"The Avarge number of daily death" ,size**=**12)  ax2**.**plot(death["Month"][2:6],death["Avg\_deaths"][2:6])  ax2**.**text(x**=**3.7,y**=** 800,s**=**"Mar-Jun")  ax3**.**plot(death["Month"][5:10],death["Avg\_deaths"][5:10])  ax3**.**text(x**=**7.1,y**=** 500,s**=**"Jun-Oct")  ax4**.**plot(death["Month"][9:],death["Avg\_deaths"][9:])  ax4**.**text(x**=**10.5,y**=** 660,s**=**"Oct-Dec",rotation**=**45)  **for** ax **in** axes:  ax**.**axhline(y**=**2000,xmin**=**0.5,xmax**=**0.8,c**=**"blue",linewidth**=**6,alpha**=** 0.5)  plt**.**show() |
| cum\_cases **=** [death**.**loc[2,"Cum\_deaths"],death**.**loc[5,"Cum\_deaths"],death**.**loc[9,"Cum\_deaths"],death**.**loc[11,"Cum\_deaths"]  ]  In [97]:  cum\_cases  Out[97]:  [2387, 126145, 227055, 297510]  In [105]:  propotion **=** [round(i**/**cum\_cases[**-**1],2) **for** i **in** cum\_cases]  propotion  Out[105]:  [0.01, 0.42, 0.76, 1.0]  In [110]:  propotion **=** []  **for** i **in** cum\_cases:  v **=** i**/**297510  final **=** round(v,2)  propotion**.**append(final)  In [111]:  propotion  Out[111]:  [0.01, 0.42, 0.76, 1.0] |