Action	Code	Explantion

Mount Drive	#mounting drive
	from google.colab import drive
	drive.mount('/content/drive')	

Import Panda	import pandas as pd	

Upload CSV file	df1=pd.read_csv("/content/drive/MyDrive/wk10_pandas_learner_vs1902/Resources/Flight Delays/Flights_Delay_Cause_2019-2020.csv")
	df2=pd.read_csv("/content/drive/MyDrive/wk10_pandas_learner_vs1902/Resources/Flight Delays/Flights_Delay_Cause_2020-2021.csv")
	df3=pd.read_csv("/content/drive/MyDrive/wk10_pandas_learner_vs1902/Resources/Flight Delays/Flights_Delay_Cause_2021-2022.csv")	

To see 1st 5 & last 5 lines	df1.head()
	df1tail()	

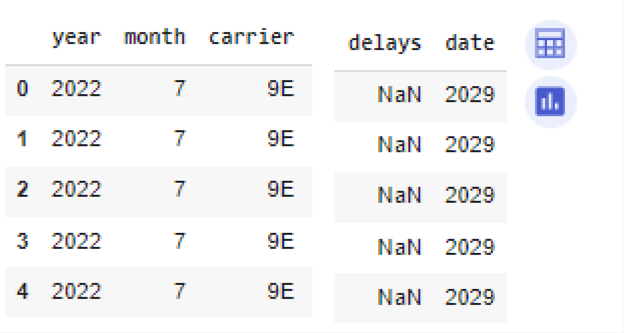
To combine multiple CSV file in 1 dataset	#use function .concat()
	df= pd.concat([df1, df2, df3], ignore_index=True)
	#setting the parameterignore_index=True will reasign new indexes to the whole dataset, the intial indexes of each document will be ignored	

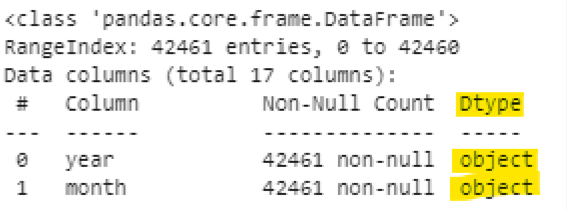
Explore wholedataset	df.head(10)	

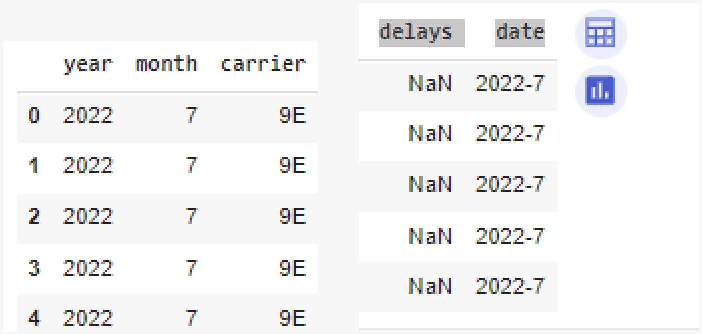
Explore total rows &Columns	df.info()
	<class 'pandas.core.frame.DataFrame'>
	RangeIndex: 42461 entries, 0 to 42460
	Data columns (total 16 columns):	

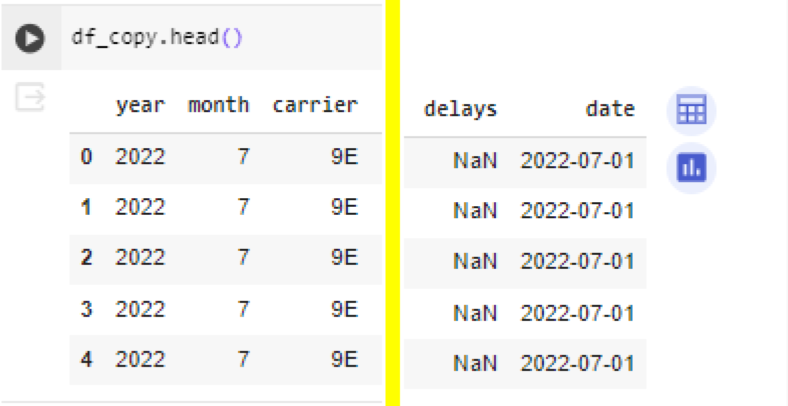
Attribution step	df.shape (42461, 16)	

Datatype	df.dtypes year int64 month int64 carrier object carrier_name object airport object airport_name object arr_flights float64 arr_del15 float64 arr_cancelled float64 arr_diverted float64 arr_delay float64 carrier_delay float64 weather_delay float64 nas_delay float64 security_delay float64 delays float64 dtype: object	
Describe	df.decribe()	
Convert data into a date datatype	.todatetime()	
Copy original dataset to avoid errors	df_copy= df.copy(deep=True)	This is our new copied database name from now on.
Add “Date” column	the "month" and "year" from our dataset. We can concatenate our 2 columns month and year using the "+" sign: df_copy['date']= df_copy['month']+df_copy['year'] df_copy.head()	Error spotted in date column

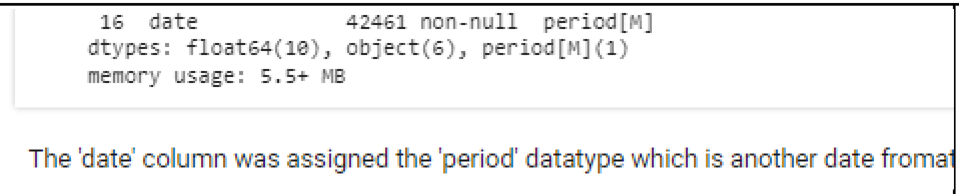
	 year month carrier delays date 0 2022 7 9E NaN 2029 1 2022 7 9E NaN 2029 2 2022 7 9E NaN 2029 3 2022 7 9E NaN 2029 4 2022 7 9E NaN 2029	

Change the month & year datatype to string	.astype()
	
	`#First, let's convert the month and year into a string datatype so we can concatenate them afterall:`
	`#We need to use the .astype() methodf to convert data in Pandas`
	`df_copy=df_copy.astype({"month": str, "year":str, "date":str})`
	`df_copy.info()`
	
	
	`<class 'pandas.core.frame.DataFrame'>`
	`RangeIndex: 42461 entries, 0 to 42460`
	`Data columns (total 17 columns):`
	` #   Column          Non-Null Count  Dtype `
	`---  ------          --------------  ----- `
	` 0   year            42461 non-null  object`
	` 1   month           42461 non-null  object`
	
	OR
	
	`#alternative method to convert datatypes in Pandas:`
	`df_copy['month'].astype(str)`
	`df_copy['year'].astype(str)`
	`df_copy['date'].astype(str)`
		Datatype changed to object in pandas

	`df_copy['date']= df_copy['year']+"-"+df_copy['month']`
	`df_copy.head()`
	
	`    year  month  carrier  delays    date`
	`0   2022      7       9E     NaN  2022-7`
	`1   2022      7       9E     NaN  2022-7`
	`2   2022      7       9E     NaN  2022-7`
	`3   2022      7       9E     NaN  2022-7`
	`4   2022      7       9E     NaN  2022-7`	

Transform the date into datetime datatype:	to_datetime()
	`df_copy['date']= pd.to_datetime(df_copy['date'])`
	`df_copy.info()`
	
	`df_copy.head()`
	`    year  month  carrier  delays        date`
	`0   2022      7       9E     NaN  2022-07-01`
	`1   2022      7       9E     NaN  2022-07-01`
	`2   2022      7       9E     NaN  2022-07-01`
	`3   2022      7       9E     NaN  2022-07-01`
	`4   2022      7       9E     NaN  2022-07-01`	

Correct date format	`df_copy['date']=`
	`pd.to_datetime(df_copy['date']).dt.to_period('M')`	

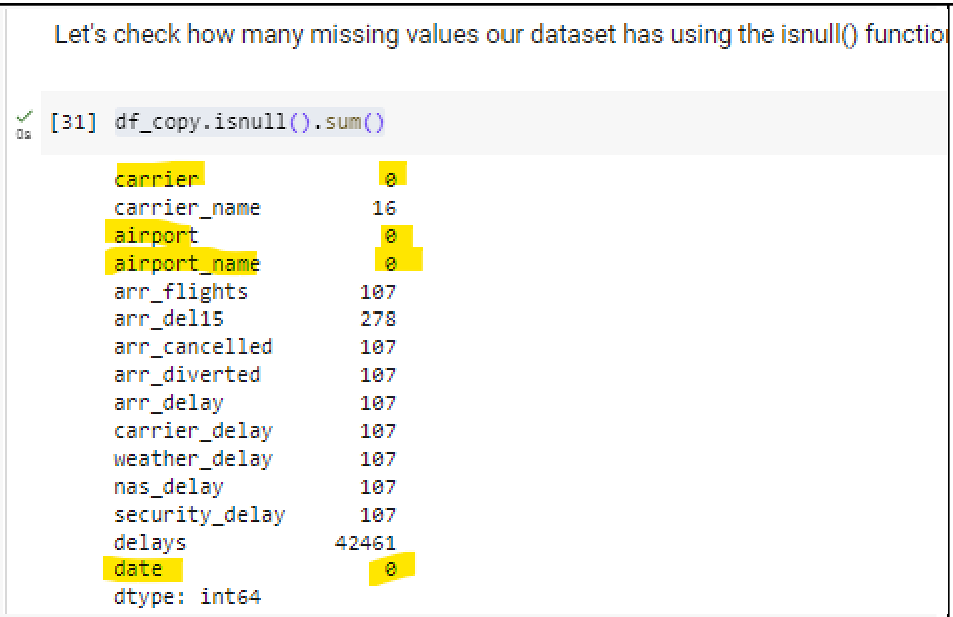
	 16 date 42461 non-null period[M] dtypes: float64(10), object(6), period[M](1) memory usage: 5.5+ MB The 'date' column was assigned the 'period' datatype which is another date fromat	

Save added “Date” column in our database	df_copy.to_csv('flights_dataset.csv')
	and mount the drive
	
	from google.colab import drive
	drive.mount('/content/drive')
		To save in different location
		from google.colab import drive; drive.mount('/content/drive').
		Then, use a path within your Drive, like /content/drive/MyDrive/data/flights_dataset.csv.

Clear unwanted columns	df_copy=df_copy.drop(['month', 'year'], axis=1)
	df_copy.head()	

3. DATA CLEANING		

Find missing values	isnull()
	df_copy.isnull().sum()	

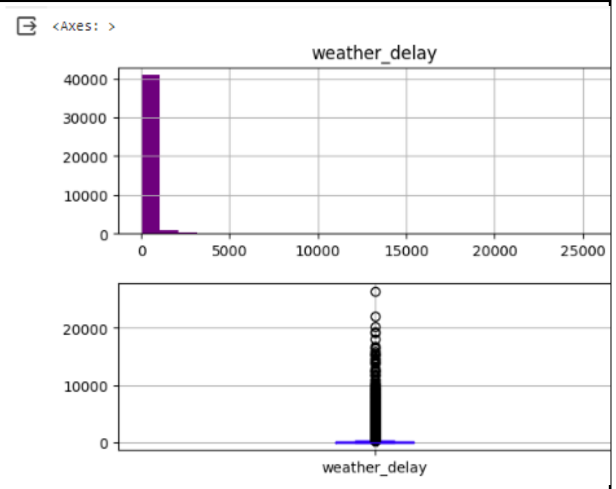
	 Let's check how many missing values our dataset has using the isnull() functio [31] df_copy.isnull().sum() carrier 0 carrier_name 16 airport 0 airport_name 0 arr_flights 107 arr_del15 278 arr_cancelled 107 arr_diverted 107 arr_delay 107 carrier_delay 107 weather_delay 107 nas_delay 107 security_delay 107 delays 42461 date 0 dtype: int64	

Drop column	df_copy= df_copy.dropna(subset=['carrier_name'])
	df_copy.isnull().sum()	

Get column names	Df_copy.columns Index(['carrier', 'carrier_name', 'airport', 'airport_name', 'arr_flights', 'arr_del15', 'arr_cancelled', 'arr_diverted', 'arr_delay', 'carrier_delay', 'weather_delay', 'nas_delay', 'security_delay', 'delays', 'date'], dtype='object')	
Built in box option	.box_plot df_copy.boxplot('weather_delay') #We create a function that will return a box plot:	# Identify and remove the row with the outlier

Name the chart X Axis	def box_plot(flights):
	    return df_copy.boxplot(flights)
	
	#We call our function passing a data variable as argument:
	box_plot('nas_delay')
		outlier_index= df_copy[df_copy['nas_delay']>175000].index
		df_cleaned= df_copy.drop(outlier_index)
		
		Int64Index([363], dtype='int64') #row number

	df_cleaned.boxplot('nas_delay')	

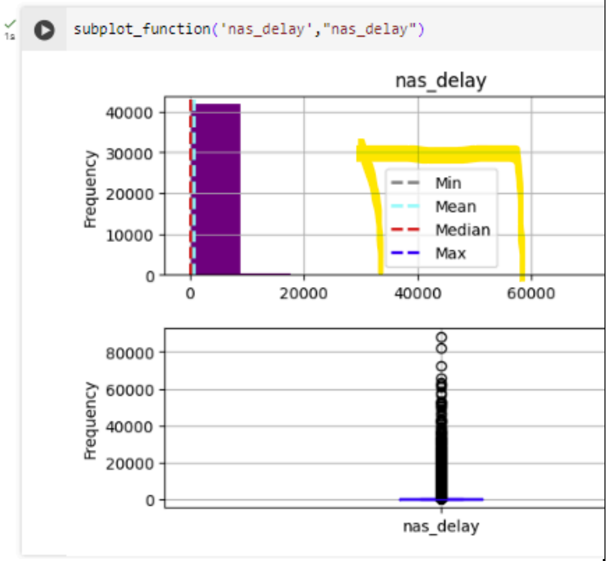
Python marplotlib visualisation	Subplots()
	
	import matplotlib.pyplot as plt
	figure, ax = plt.subplots(nrows=2, ncols=1) # subplot layer has 2 rows an 1 column which means 2 figures
	df_cleaned.hist('weather_delay',ax=ax[0], bins=25, color="purple")
	df_cleaned.boxplot("weather_delay",ax=ax[1], color="blue")
		
		<Axes: >

	def subplot_function(data1, data2): # I added one parameter per plot
	    figure, ax = plt.subplots(nrows=2, ncols=1)
	
	    # Get statistics
	    min_val= df_cleaned[data1].min()
	    max_val= df_cleaned[data1].max()
	    mean_val= df_cleaned[data1].mean()
	    med_val= df_cleaned[data1].median()
		def subplot_function(data1, data2): # Define a function to create a multi-plot figure
		    """
		    Creates a figure with two subplots: a histogram for data1 and a boxplot for data2.
		    Adds reference lines for min, mean

```
    # Add lines for the min, mean and median, and max
    ax[0].axvline(x=min_val, color = 'gray', linestyle='dashed', linewidth = 2)
    ax[0].axvline(x=mean_val, color = 'cyan', linestyle='dashed', linewidth = 2)
    ax[0].axvline(x=med_val, color = 'red', linestyle='dashed', linewidth = 2)
    ax[0].axvline(x=max_val, color = 'blue', linestyle='dashed', linewidth = 2)

    #Change labels- 1st plot
    ax[0].set_ylabel('Frequency')

    df_cleaned.hist(data1,ax=ax[0], bins=10, color="purple")


    #Change labels- 2nd plot
    ax[1].set_xlabel('')
    ax[1].set_ylabel('Frequency')


    df_cleaned.boxplot(data2,ax=ax[1], color="blue")
```

```
, median, and max values in the histogram.

    Args:
        data1 (str): Name of the column in df_cleaned to be used for the histogram.
        data2 (str): Name of the column in df_cleaned to be used for the boxplot.
    """

    figure, ax = plt.subplots(nrows=2, ncols=1)  # Create a figure with 2 rows and 1 column of subplots

    # Get statistics for data1 to add reference lines
    min_val = df_cleaned[data1].min()
    max_val = df_cleaned[data1].max()
    mean_val = df_cleaned[data1].mean()
    med_val = df_cleaned[data1].median()

    # Add vertical reference lines for min, mean, median, and max in the first subplot
    ax[0].axvline(x=min_val, color='gray', linestyle='dashed', linewidth=2, label='Min')
    ax[0].axvline(x=mean_val, color='cyan', linestyle='dashed', linewidth=2, label='Mean')
    ax[0].axvline(x=med_val, color='red', linestyle='dashed', linewidth=2, label='Median')
    ax[0].axvline(x=max_val, color='blue', linestyle='dashed', linewidth=2, label='Max')
    ax[0].legend()  # Add a legend to
```

		explain the reference lines
		
		# Create histogram for data1 in the first subplot
		ax[0].set_ylabel('Frequency') # Set y-axis label
		df_cleaned.hist(data1, ax=ax[0], bins=10, color="purple")
		
		# Create boxplot for data2 in the second subplot
		ax[1].set_xlabel('') # Remove x-axis label for cleaner presentation
		ax[1].set_ylabel('Frequency') # Set y-axis label
		df_cleaned.boxplot(data2, ax=ax[1], color="blue")

	subplot_function('nas_delay',"nas_delay")	
Add legend in chart	# Same plot with Labels def subplot_function(data1, data2): # Define a function to create a multi-plot figure """ Creates a figure with two subplots: a histogram for data1 and a boxplot for data2. Adds reference lines for min, mean, median, and max values in the histogram. Args: data1 (str): Name of the column in df_cleaned to be used for the histogram. data2 (str): Name of the column in df_cleaned to be used for the boxplot. """	 subplot_function('nas_delay',"nas_delay") nas_delay 40000 30000 20000 10000 0 Frequency 0 20000 40000 60000 Min Mean Median Max 80000 60000 40000 20000 0 Frequency nas_delay

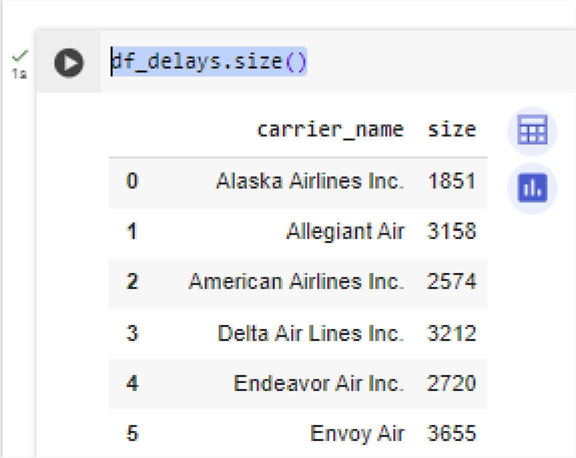
```
    figure, ax = plt.subplots(nrows=2, ncols=1)   # Create a figure with 2 rows and 1 column of subplots

    # Get statistics for data1 to add reference lines
    min_val = df_cleaned[data1].min()
    max_val = df_cleaned[data1].max()
    mean_val = df_cleaned[data1].mean()
    med_val = df_cleaned[data1].median()

    # Add vertical reference lines for min, mean, median, and max in the first subplot
    ax[0].axvline(x=min_val, color='gray', linestyle='dashed', linewidth=2, label='Min')
    ax[0].axvline(x=mean_val, color='cyan', linestyle='dashed', linewidth=2, label='Mean')
    ax[0].axvline(x=med_val, color='red', linestyle='dashed', linewidth=2, label='Median')
    ax[0].axvline(x=max_val, color='blue', linestyle='dashed', linewidth=2, label='Max')
    ax[0].legend()   # Add a legend to explain the reference lines

    # Create histogram for data1 in the first subplot
    ax[0].set_ylabel('Frequency')   # Set y-axis label
    df_cleaned.hist(data1, ax=ax[0], bins=10, color="purple")

    # Create boxplot for data2 in the second subplot
    ax[1].set_xlabel('')   # Remove x-axis label for cleaner presentation
    ax[1].set_ylabel('Frequency')   # Set y-axis label
    df_cleaned.boxplot(data2, ax=ax[1], color="blue")
```

4 Data Analysis

Q1-Which Airlines have the most delayed flights?

We can group the airline data and calculate the average of delayed flights per airline

```
df_delays=df_cleaned.groupby(by="carrier_name", as_index=False)
#using the argument as_index=False in the groupby function will allow us to return a pandas dataframe, otherwise a pandas series will be returned.
```

	`.size()`
	`df_delays.size()`
	
	`df_delays.size()`
	`  carrier_name  size`
	`0  Alaska Airlines Inc.  1851`
	`1  Allegiant Air  3158`
	`2  American Airlines Inc.  2574`
	`3  Delta Air Lines Inc.  3212`
	`4  Endeavor Air Inc.  2720`
	`5  Envoy Air  3655`
		we can see how many airlines has been grouped with the function .size():

Mean	`df_delays=df_delays["arr_del15"].mean()`
	`40.9383…`
		`df_delays # here we get the top 5 of airlines with the most delays`

	plot the data using the Matplotlib and Seaborn visualisation libraries
	`import matplotlib.pyplot as plt`
	`import seaborn as sns`	