PANDAS Zero2Hero

```
import pandas as pd
# LOAD THE CSV FILE INTO A DATAFRAME. A DATAFRAME IS THE SQL EQUIVALENT OF A TABLE
df=pd.read_csv("titanic.csv")
# THIS WILL PRINT THE FIRST TOP 10 ROWS OF THE DATAFRAME
df.head(10)
```

	Survived	Pclass	Name	Sex	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare
0	0	3	Mr. Owen Harris Braund	male	22.0	1	0	7.2500
1	1	1	Mrs. John Bradley (Florence Briggs Thayer) Cum...	female	38.0	1	0	71.2833
2	1	3	Miss. Laina Heikkinen	female	26.0	0	0	7.9250
3	1	1	Mrs. Jacques Heath (Lily May Peel) Futrelle	female	35.0	1	0	53.1000
4	0	3	Mr. William Henry Allen	male	35.0	0	0	8.0500
5	0	3	Mr. James Moran	male	27.0	0	0	8.4583
6	0	1	Mr. Timothy J McCarthy	male	54.0	0	0	51.8625
7	0	3	Master. Gosta Leonard Palsson	male	2.0	3	1	21.0750
8	1	3	Mrs. Oscar W (Elisabeth Vilhelmina Berg) Johnson	female	27.0	0	2	11.1333
9	1	2	Mrs. Nicholas (Adele Achem) Nasser	female	14.0	1	0	30.0708

NEW TASK→2

```
# A DATAFRAME CONSISTS OF SERIES WHICH ARE THE EQUIVALENT OF FIELDS IN SQL TABLE i.e. Name, Sex, Age.
# TO PRINT ONE SERIES
df["Name"]
```

```
0                              Mr. Owen Harris Braund
1      Mrs. John Bradley (Florence Briggs Thayer) Cum...
2                                Miss. Laina Heikkinen
3            Mrs. Jacques Heath (Lily May Peel) Futrelle
4                              Mr. William Henry Allen
                             ...
882                              Rev. Juozas Montvila
883                          Miss. Margaret Edith Graham
884                        Miss. Catherine Helen Johnston
885                                Mr. Karl Howell Behr
886                                  Mr. Patrick Dooley
Name: Name, Length: 887, dtype: object
```

NEW TASK→3

```
# TO PRINT MULTIPLE SERIES YOU PASS A LIST OF THE SERIES INTO THE DATAFRAME
df[['Name','Sex']]
```

	Name	Sex
0	Mr. Owen Harris Braund	male
1	Mrs. John Bradley (Florence Briggs Thayer) Cum...	female
2	Miss. Laina Heikkinen	female
3	Mrs. Jacques Heath (Lily May Peel) Futrelle	female
4	Mr. William Henry Allen	male
...	...	...
882	Rev. Juozas Montvila	male
883	Miss. Margaret Edith Graham	female
884	Miss. Catherine Helen Johnston	female
885	Mr. Karl Howell Behr	male
886	Mr. Patrick Dooley	male

887 rows × 2 columns

NEW TASK→4

```
# RETRIEVE A SINGLE ROW OF DATA
df.loc[4]
```

```
Survived                                0
Pclass                                  3
Name                     Mr. William Henry Allen
Sex                                     male
Age                                     35.0
Siblings/Spouses Aboard                 0
Parents/Children Aboard                 0
Fare                                    8.05
Name: 4, dtype: object
```

NEW TASK→5

```
# RETRIEVE ALL ROWS BASED ON A CLAUSE
df[(df["Age"]>= 10) & (df["Age"]<=30)]
```

	Survived	Pclass	Name	Sex	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare
0	0	3	Mr. Owen Harris Braund	male	22.0	1	0	7.2500
2	1	3	Miss. Laina Heikkinen	female	26.0	0	0	7.9250
5	0	3	Mr. James Moran	male	27.0	0	0	8.4583
8	1	3	Mrs. Oscar W (Elisabeth Vilhelmina Berg) Johnson	female	27.0	0	2	11.1333
9	1	2	Mrs. Nicholas (Adele Achem) Nasser	female	14.0	1	0	30.0708
...	...	...	...	...	...	...	...	...
879	0	2	Mr. Frederick James Banfield	male	28.0	0	0	10.5000
880	0	3	Mr. Henry Jr Sutehall	male	25.0	0	0	7.0500
882	0	2	Rev. Juozas Montvila	male	27.0	0	0	13.0000
883	1	1	Miss. Margaret Edith Graham	female	19.0	0	0	30.0000
885	1	1	Mr. Karl Howell Behr	male	26.0	0	0	30.0000

454 rows × 8 columns

NEW TASK→6

```
# RETRIEVE ALL PASSENGERS NOT 30 OR 35 (USING A TILDE ~)
# ALTERNATIVELY
# df[(df["Age"]!="30") | (df["Age"]!="35")]
df[~df["Age"].isin(["30","35"])]
```

	Survived	Pclass	Name	Sex	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare
0	0	3	Mr. Owen Harris Braund	male	22.0	1	0	7.2500
1	1	1	Mrs. John Bradley (Florence Briggs Thayer) Cum...	female	38.0	1	0	71.2833
2	1	3	Miss. Laina Heikkinen	female	26.0	0	0	7.9250
3	1	1	Mrs. Jacques Heath (Lily May Peel) Futrelle	female	35.0	1	0	53.1000
4	0	3	Mr. William Henry Allen	male	35.0	0	0	8.0500
...	...	...	...	...	...	...	...	...
882	0	2	Rev. Juozas Montvila	male	27.0	0	0	13.0000
883	1	1	Miss. Margaret Edith Graham	female	19.0	0	0	30.0000
884	0	3	Miss. Catherine Helen Johnston	female	7.0	1	2	23.4500
885	1	1	Mr. Karl Howell Behr	male	26.0	0	0	30.0000
886	0	3	Mr. Patrick Dooley	male	32.0	0	0	7.7500

887 rows × 8 columns

NEW TASK→7

```
# WHAT WOULD BE EACH PASSENGER'S AGE DEVIATION FROM THE AVERAGE AGE IN A NEW FIELD?
df["AGE DEVIATION"] = df["Age"] - df["Age"].mean()
df
```

	Survived	Pclass	Name	Sex	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare	AGE DEVIATION
0	0	3	Mr. Owen Harris Braund	male	22.0	1	0	7.2500	-7.471443
1	1	1	Mrs. John Bradley (Florence Briggs Thayer) Cum...	female	38.0	1	0	71.2833	8.528557
2	1	3	Miss. Laina Heikkinen	female	26.0	0	0	7.9250	-3.471443
3	1	1	Mrs. Jacques Heath (Lily May Peel) Futrelle	female	35.0	1	0	53.1000	5.528557
4	0	3	Mr. William Henry Allen	male	35.0	0	0	8.0500	5.528557
...	...	...	...	...	...	...	...	...	...
882	0	2	Rev. Juozas Montvila	male	27.0	0	0	13.0000	-2.471443
883	1	1	Miss. Margaret Edith Graham	female	19.0	0	0	30.0000	-10.471443
884	0	3	Miss. Catherine Helen Johnston	female	7.0	1	2	23.4500	-22.471443
885	1	1	Mr. Karl Howell Behr	male	26.0	0	0	30.0000	-3.471443
886	0	3	Mr. Patrick Dooley	male	32.0	0	0	7.7500	2.528557

887 rows × 9 columns

NEW TASK→8

```
# RESET THE INDEX OF A FILTERED SUBSET
# RUN ALL CODES TO COMPARE RESULTS
# newdf = df[df["Age"]==30]
# newdf = df[df["Age"]==30].reset_index()
newdf = df[df["Age"]==30].reset_index(drop=True)
newdf.head(10)
```

	Survived	Pclass	Name	Sex	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare	AGE DEVIATION
0	0	3	Mr. William John Rogers	male	30.0	0	0	8.050	0.528557
1	1	3	Miss. Elizabeth Dowdell	female	30.0	0	0	12.475	0.528557
2	0	3	Mr. Harry Corn	male	30.0	0	0	8.050	0.528557
3	0	2	Mr. Reginald Hale	male	30.0	0	0	13.000	0.528557
4	0	2	Mr. Hans Kristensen Givard	male	30.0	0	0	13.000	0.528557
5	0	2	Mr. Walter Harris	male	30.0	0	0	10.500	0.528557
6	0	3	Mr. Sleiman Attalah	male	30.0	0	0	7.225	0.528557
7	0	3	Mr. William Arthur Lobb	male	30.0	1	0	16.100	0.528557
8	1	1	Miss. Gladys Cherry	female	30.0	0	0	86.500	0.528557
9	1	3	Mr. Theodore de Mulder	male	30.0	0	0	9.500	0.528557

NEW TASK→9

```
# SORTING DATA AS IN SQL ORDER BY Sex AND THEN MY Age in ascending order
df.sort_values(by=["Sex","Age"], ascending=True)
```

	Survived	Pclass	Name	Sex	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare	AGE DEVIATION
466	1	3	Miss. Helene Barbara Baclini	female	0.75	2	1	19.2583	-28.721443
641	1	3	Miss. Eugenie Baclini	female	0.75	2	1	19.2583	-28.721443
171	1	3	Miss. Eleanor Ileen Johnson	female	1.00	1	1	11.1333	-28.471443
379	1	3	Miss. Maria Nakid	female	1.00	0	2	15.7417	-28.471443
118	0	3	Miss. Ellis Anna Maria Andersson	female	2.00	4	2	31.2750	-27.471443
...	...	...	...	...	...	...	...	...	...
115	0	3	Mr. Patrick Connors	male	70.50	0	0	7.7500	41.028557
95	0	1	Mr. George B Goldschmidt	male	71.00	0	0	34.6542	41.528557
490	0	1	Mr. Ramon Artagaveytia	male	71.00	0	0	49.5042	41.528557
847	0	3	Mr. Johan Svensson	male	74.00	0	0	7.7750	44.528557
627	1	1	Mr. Algernon Henry Wilson Barkworth	male	80.00	0	0	30.0000	50.528557

887 rows × 9 columns

NEW TASK→10

```
# RETRIEVE TOP 10 NAMES WITH JOHN. MIND THE SPACE AFTER JOHN TO EXCLUDE JOHNSON
df[df["Name"].str.contains("John ")].head(10)
```

	Survived	Pclass	Name	Sex	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare	AGE DEVIATION
1	1	1	Mrs. John Bradley (Florence Briggs Thayer) Cum...	female	38.0	1	0	71.2833	8.528557
41	0	2	Mrs. William John Robert (Dorothy Ann Wonnacot...	female	27.0	1	0	21.0000	-2.471443
44	0	3	Mr. William John Rogers	male	30.0	0	0	8.0500	0.528557
97	1	2	Mrs. John T (Ada Julia Bone) Doling	female	34.0	0	1	23.0000	4.528557
111	0	3	Mr. David John Barton	male	22.0	0	0	8.0500	-7.471443
116	0	2	Mr. William John Robert Turpin	male	29.0	1	0	21.0000	-0.471443
159	0	3	Mr. John Hatfield Cribb	male	44.0	0	1	16.1000	14.528557
161	0	3	Mr. John Viktor Bengtsson	male	26.0	0	0	7.7750	-3.471443
164	1	3	Master. Frank John William Goldsmith	male	9.0	0	2	20.5250	-20.471443
167	0	1	Mr. John D Baumann	male	60.0	0	0	25.9250	30.528557

NEW TASK→11

```
# EXPORT TO A NEW .CSV FILE
df.to_csv("titanic_new_info.csv")
```

```
# MATRIX RELOAD
df=pd.read_csv("titanic.csv")
df.dtypes
```

```
Survived                   int64
Pclass                     int64
Name                      object
Sex                       object
Age                      float64
Siblings/Spouses Aboard    int64
Parents/Children Aboard    int64
Fare                     float64
dtype: object
```

NEW TASK→12

```
# CREATE A DATAFRAME BASED ON THE TITANIC FRAME
person_survived = {"Survived":[1],
                   "Pclass":[1],
                   "Name":["Mr. Leo Miaris"],
                   "Sex":["male"],
                   "Age":[78.0],
                   "Siblings/Spouses Aboard":[0],
                   "Parents/Children Aboard":[0],
                   "Fare":[100.0]}
person_to_add = pd.DataFrame(columns=df.columns, data=person_survived)
person_to_add
```

	Survived	Pclass	Name	Sex	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare
0	1	1	Mr. Leo Miaris	male	78.0	0	0	100.0

NEW TASK→13

```
# CREATE A DATAFRAME FROM SCRATCH
lookup_data = {"Pclass":[1,2,3],"Total cabin amount":[371,105,84]}
additional_info = pd.DataFrame(columns = ['Pclass','Total cabin amount'], data=lookup_data)
additional_info
```

	Pclass	Total cabin amount
0	1	371
1	2	105
2	3	84

NEW TASK→14

```
# GET ALL THE DATAFRAME STATISTICS FOR DATA WRANGLING ETC
df.describe()
```

	Survived	Pclass	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare
count	887.000000	887.000000	887.000000	887.000000	887.000000	887.00000
mean	0.385569	2.305524	29.471443	0.525366	0.383315	32.30542
std	0.487004	0.836662	14.121908	1.104669	0.807466	49.78204
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.00000
25%	0.000000	2.000000	20.250000	0.000000	0.000000	7.92500
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.45420
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.13750
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.32920

NEW TASK→16

```
# AGGREGATE BASED ON SERIES
df["Age"].value_counts()
```

```
22.00    39
28.00    37
18.00    36
21.00    34
24.00    34
         ..
0.92      1
23.50     1
36.50     1
55.50     1
74.00     1
Name: Age, Length: 89, dtype: int64
```

NEW TASK→17

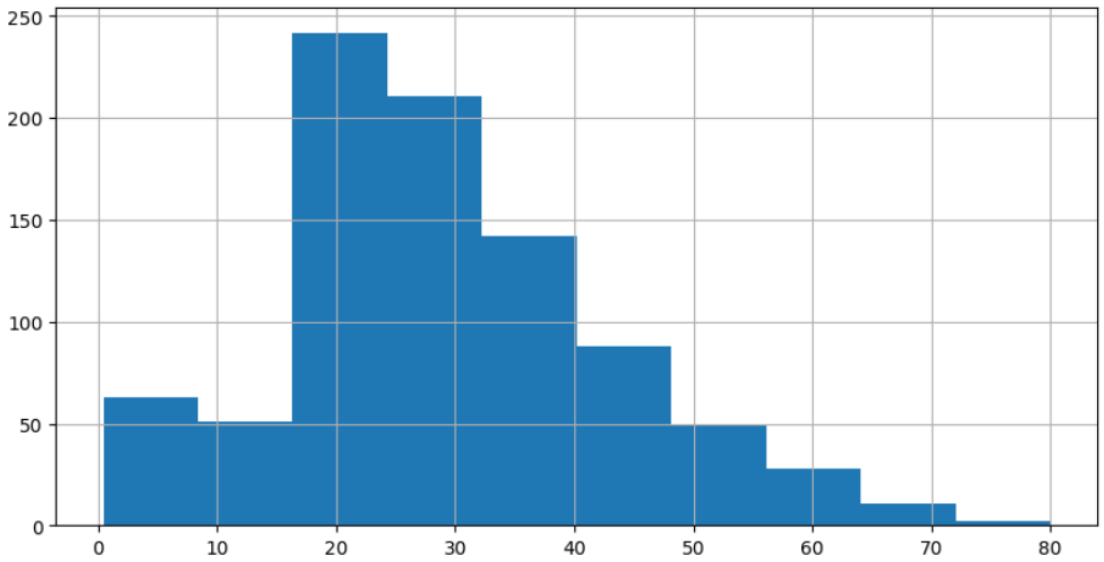
```
# GROUP BY AGGREGATIION
df.groupby("Sex")["Age"].mean()
```

```
Sex
female    27.719745
male      30.431361
Name: Age, dtype: float64
```

NEW TASK→18

```
# PLOT DATA
import matplotlib
df["Age"].hist(figsize=(10,5))
```

```
<AxesSubplot:>
```



NEW TASK→19

```
import pandas as pd
# THIS DATAFRAME HAS AGE BUT NOT EXPERTID AND 4 RECORDS
df1 = pd.read_csv("Students.csv")
df1
```

	StudentID	Name	Age
0	ST01	John	20
1	ST02	Alex	30
2	ST03	Misra	22
3	ST04	Jenny	28

NEW TASK→20

```
# THIS DATAFRAME HAS EXPERTIN BUT NOT AGE AND 5 RECORDS
df2 = pd.read_csv("Students2.csv")
df2
```

	StudentID	Name	ExpertIn
0	ST01	John	Excel
1	ST02	Alex	SQL
2	ST05	Misra	Pandas
3	ST06	Jenny	NumPy
4	ST07	Glenn	Power BI

NEW TASK→21

```
# AN INNER MERGE WILL MATCH THE COMMON STUDENTID WILL ALSO RETURN THE RIGHT TABLE NAME
df1.merge(df2, how="inner", on=["StudentID"])
```

	StudentID	Name_x	Age	Name_y	ExpertIn
0	ST01	John	20	John	Excel
1	ST02	Alex	30	Alex	SQL

```
# AN INNER MERGE WITH STUDENTID AND NAME WILL RETURN ONLY THE AGE AND EXPERTIN
df1.merge(df2, how="inner", on=["StudentID","Name"])
```

	StudentID	Name	Age	ExpertIn
0	ST01	John	20	Excel
1	ST02	Alex	30	SQL

NEW TASK→22

```
# AND OUTER JOIN WILL RETURN ALL WITH NaN FOR FIELDS THAT DO NOT MATCH
df1.merge(df2, how="outer")
```

	StudentID	Name	Age	ExpertIn
0	ST01	John	20.0	Excel
1	ST02	Alex	30.0	SQL
2	ST03	Misra	22.0	NaN
3	ST04	Jenny	28.0	NaN
4	ST05	Misra	NaN	Pandas
5	ST06	Jenny	NaN	NumPy
6	ST07	Glenn	NaN	Power BI

NEW TASK→23

```
# A LEFT WILL RETURN ALL DF1 AND ANY MATCHING DATA FROM DF2
df1.merge(df2, how="left")
```

	StudentID	Name	Age	ExpertIn
0	ST01	John	20	Excel
1	ST02	Alex	30	SQL
2	ST03	Misra	22	NaN
3	ST04	Jenny	28	NaN

NEW TASK→24

```
# A JOIN WORKS WELL WHEN THE COLUMN NAMES ARE INDEXED
df1.set_index("StudentID").join(df2.set_index("StudentID"), how="outer", lsuffix="_df1", rsuffix="_df2")
```

	Name_df1	Age	Name_df2	ExpertIn
StudentID				
ST01	John	20.0	John	Excel
ST02	Alex	30.0	Alex	SQL
ST03	Misra	22.0	NaN	NaN
ST04	Jenny	28.0	NaN	NaN
ST05	NaN	NaN	Misra	Pandas
ST06	NaN	NaN	Jenny	NumPy
ST07	NaN	NaN	Glenn	Power BI

NEW TASK→25

```
# STACKS THE DATAFRAMES ON TOP OF EACH OTHER
# BY DEFAULT THE JOIN TYPE IS OUTER
pd.concat([df1,df2])
```

	StudentID	Name	Age	ExpertIn
0	ST01	John	20.0	NaN
1	ST02	Alex	30.0	NaN
2	ST03	Misra	22.0	NaN
3	ST04	Jenny	28.0	NaN
0	ST01	John	NaN	Excel
1	ST02	Alex	NaN	SQL
2	ST05	Misra	NaN	Pandas
3	ST06	Jenny	NaN	NumPy
4	ST07	Glenn	NaN	Power BI

NEW TASK→26

```
# AXIS 1 WILL PUT THE DATAFRAMES SIDE BY SIDE RATHER THAN ON TOP OF EACH OTHER
# THERE ARE 4 RECORDS 0->3 FOR DF1 AND 5 RECORDS FOR DF2 0->4
# THE DF1 WITH NO 5TH RECORD WILL BE NaN
pd.concat([df1,df2], join="outer", axis=1)
```

	StudentID	Name	Age	StudentID	Name	ExpertIn
0	ST01	John	20.0	ST01	John	Excel
1	ST02	Alex	30.0	ST02	Alex	SQL
2	ST03	Misra	22.0	ST05	Misra	Pandas
3	ST04	Jenny	28.0	ST06	Jenny	NumPy
4	NaN	NaN	NaN	ST07	Glenn	Power BI

NEW TASK→27

```
df3 = df1.merge(df2, how="outer")
df3
```

	StudentID	Name	Age	ExpertIn
0	ST01	John	20.0	Excel
1	ST02	Alex	30.0	SQL
2	ST03	Misra	22.0	NaN
3	ST04	Jenny	28.0	NaN
4	ST05	Misra	NaN	Pandas
5	ST06	Jenny	NaN	NumPy
6	ST07	Glenn	NaN	Power BI

NEW TASK→28

```
df4 = pd.read_csv("StudentsDetails.csv")
df4
```

	StudentID	GCSE_Grade	Alevels_Grade	Salary	InEmployment_Yrs
0	ST01	7	8	30000	2
1	ST02	4	6	25000	5
2	ST03	8	8	42000	3
3	ST04	5	5	23000	7
4	ST05	9	9	40000	4
5	ST06	7	7	50000	2
6	ST07	6	7	28000	1

NEW TASK→29

```
# MDF: MASTER DATAFRAME
mdf = df3.merge(df4, how = "outer")
mdf
```

	StudentID	Name	Age	ExpertIn	GCSE_Grade	Alevels_Grade	Salary	InEmployment_Yrs
0	ST01	John	20.0	Excel	7	8	30000	2
1	ST02	Alex	30.0	SQL	4	6	25000	5
2	ST03	Misra	22.0	NaN	8	8	42000	3
3	ST04	Jenny	28.0	NaN	5	5	23000	7
4	ST05	Misra	NaN	Pandas	9	9	40000	4
5	ST06	Jenny	NaN	NumPy	7	7	50000	2
6	ST07	Glenn	NaN	Power BI	6	7	28000	1

NEW TASK→30

```
# IF NO SKILL THEN EXPERT IN ADMIN
mdf["ExpertIn"].fillna("Admin",inplace=True)
mdf
```

	StudentID	Name	Age	ExpertIn	GCSE_Grade	Alevels_Grade	Salary	InEmployment_Yrs
0	ST01	John	20.0	Excel	7	8	30000	2
1	ST02	Alex	30.0	SQL	4	6	25000	5
2	ST03	Misra	22.0	Admin	8	8	42000	3
3	ST04	Jenny	28.0	Admin	5	5	23000	7
4	ST05	Misra	NaN	Pandas	9	9	40000	4
5	ST06	Jenny	NaN	NumPy	7	7	50000	2
6	ST07	Glenn	NaN	Power BI	6	7	28000	1

NEW TASK→31

```
# IF NO AGE THEN AGE IS THE MIDDLE AGE VALUE
mdf["Age"].fillna(mdf["Age"].median(),inplace=True)
mdf
```

	StudentID	Name	Age	ExpertIn	GCSE_Grade	Alevels_Grade	Salary	InEmployment_Yrs
0	ST01	John	20.0	Excel	7	8	30000	2
1	ST02	Alex	30.0	SQL	4	6	25000	5
2	ST03	Misra	22.0	Admin	8	8	42000	3
3	ST04	Jenny	28.0	Admin	5	5	23000	7
4	ST05	Misra	25.0	Pandas	9	9	40000	4
5	ST06	Jenny	25.0	NumPy	7	7	50000	2
6	ST07	Glenn	25.0	Power BI	6	7	28000	1

NEW TASK→32

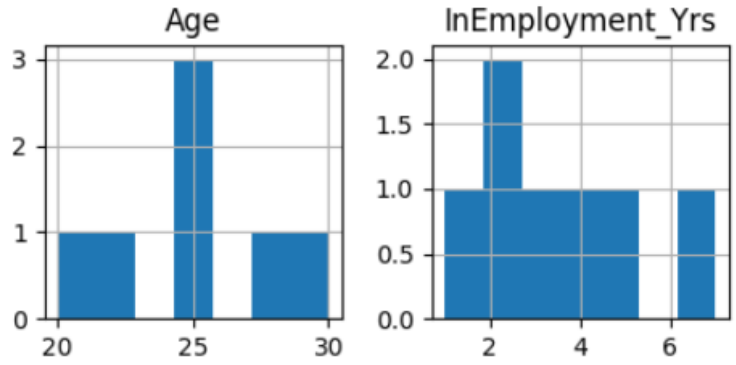
```
stats = mdf.describe()
stats
```

	Age	GCSE_Grade	Alevels_Grade	Salary	InEmployment_Yrs
count	7.000000	7.000000	7.000000	7.000000	7.000000
mean	25.000000	6.571429	7.142857	34000.000000	3.428571
std	3.366502	1.718249	1.345185	10082.988975	2.070197
min	20.000000	4.000000	5.000000	23000.000000	1.000000
25%	23.500000	5.500000	6.500000	26500.000000	2.000000
50%	25.000000	7.000000	7.000000	30000.000000	3.000000
75%	26.500000	7.500000	8.000000	41000.000000	4.500000
max	30.000000	9.000000	9.000000	50000.000000	7.000000

NEW TASK→33

```
# PANDAS NEEDS MATPLOTLIB LIBRARY TO RUN
# BY ADJUSTING BINS WE SHAPE THE HISTOGRAM TO SHOW THAT
# HAS NORMAL DISTRIBUTION CURVE WHILE EMPLOYMENT DOES NOT
import matplotlib as plt
mdf[["Age","InEmployment_Yrs"]].hist(figsize=(5,2), bins=7)
```

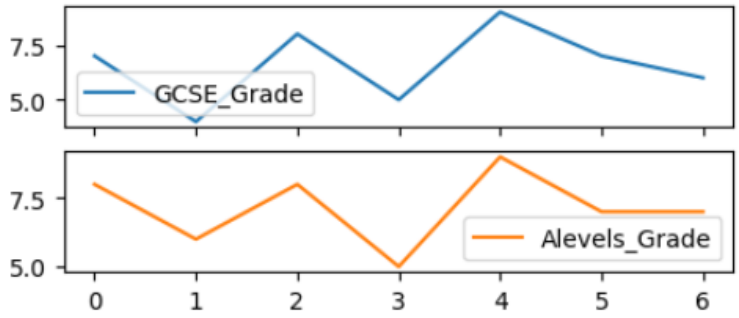
```
array([[<AxesSubplot:title={'center':'Age'}>,
        <AxesSubplot:title={'center':'InEmployment_Yrs'}>]], dtype=object)
```



NEW TASK→34

```
# HIGH GCSE SCORE YIELDS HIGH ALEVEL SCORE
mdf[["GCSE_Grade","Alevels_Grade"]].plot(kind="line", subplots=True, figsize=(5,2))
```

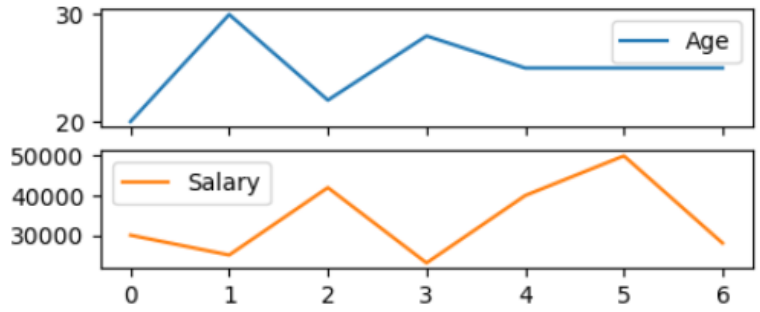
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



NEW TASK→35

```
# SALARIES ARE DISPROPORTIONAL TO AGE
# IN THIS MODEL YOUNGER AGES YIELD HIGHER SALARIES
# KNOWLEDGE OF NEWER TECHNOLOGIES PAY MORE THAT ACCUMULATED INDUSTRY EXPERIENCE
mdf[["Age","Salary"]].plot(kind="line", subplots=True, figsize=(5,2))
```

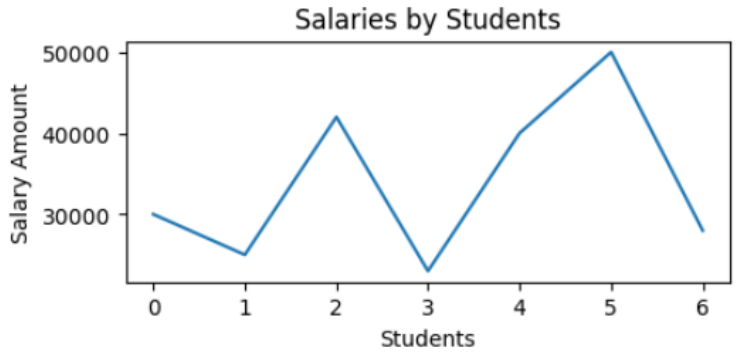
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



NEW TASK→36

```
# A BETTER SALARY VISUALISATION
mdf["Salary"].plot(kind="line", figsize=(5,2), ylabel="Salary Amount", xlabel="Students", title="Salaries by Students")
```

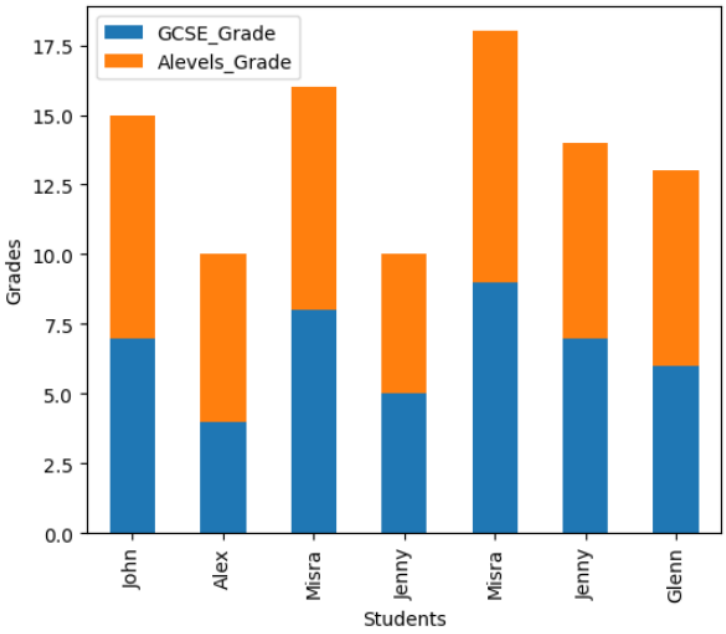
```
<AxesSubplot:title={'center':'Salaries by Students'}, xlabel='Students', ylabel='Salary Amount'>
```



NEW TASK→37

```
# INCLUDE STUDENT NAMES IN X AXIS REPLACING THE STANDARD INDEX NUMBER
mdf.set_index("Name")[["GCSE_Grade","Alevels_Grade"]].plot(kind="bar", figsize=(6,5), ylabel="Grades", xlabel="Students", stacked=True)
```

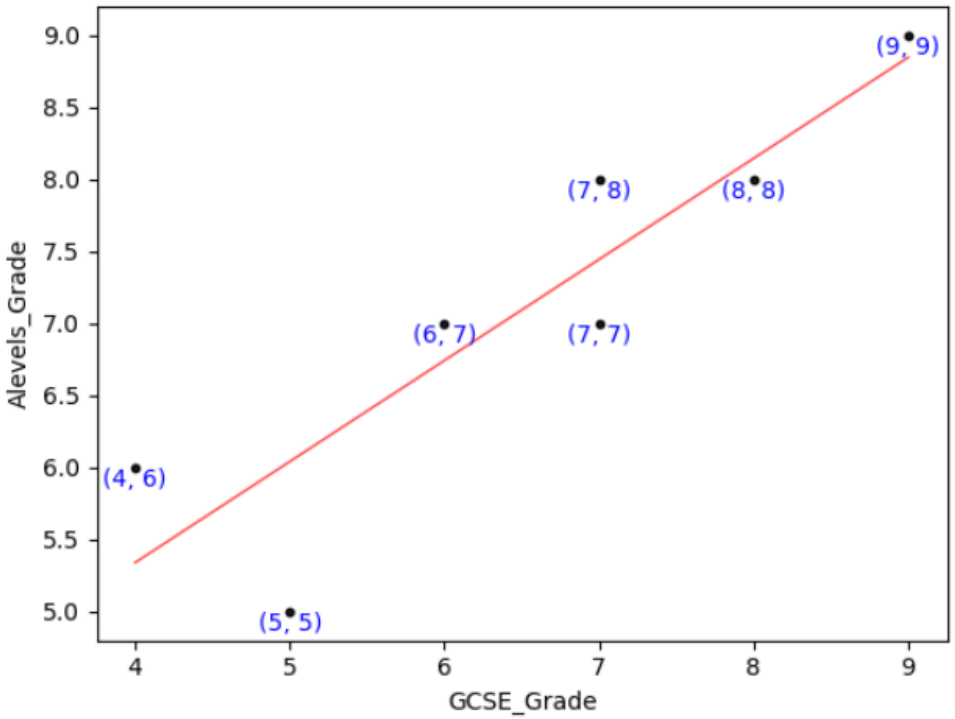
```
<AxesSubplot:xlabel='Students', ylabel='Grades'>
```



NEW TASK→38

```
# TO INSTALL SEABORN LIBRARY BEFORE IMPORTING
%pip install seaborn

# IMPORT LIBRARIES NEEDED
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# CREATE SCATTER PLOT WITH A TRENDLINE
sns.regplot(x="GCSE_Grade", y="Alevels_Grade", data=mdf,
            scatter_kws={"color": "black", "alpha": 0.9, "s": 10}, # Customize scatter plot
            line_kws={"color": "red", "alpha": 0.7, "lw": 1}, # Customize regression line
            marker='o', # Marker style
            ci=0, # Confidence interval
            truncate=True
           )

# ANNOTATING EACH POINT WITH VALUES
for i in range(len(mdf)):
    plt.text(mdf['GCSE_Grade'][i], mdf['Alevels_Grade'][i], f"({mdf['GCSE_Grade'][i]}, {mdf['Alevels_Grade'][i]})", ha='center', va='top', color="blue")

# PRINT THE TRENDLINE EQUATION
print("y=%.6fx+(%.6f)"%(z[0],z[1]))
plt.show()
```

```
y=0.701613x+(2.532258)
```



NEW TASK→ COMPLETED SUCCESSFULLY!

GOOD LUCK!

ADDITIONAL SECTION A:

```
import pandas as pd
data = {
   'Name': ['John', 'Anna', 'Peter', 'Linda', 'James'],
   'Age': [28, 34, 29, 32, 41],
   'City': ['New York', 'Paris', 'Berlin', 'London', 'Tokyo'],
   'Salary': [70000, 72000, 69000, 65000, 98000]
}
df = pd.DataFrame(data)
```

Select rows where the age is greater than 30:

```
filtered_df = df[df['Age'] > 30]
```

Select rows where the age is greater than 30 and the salary is less than 70000:

```
filtered_df = df[(df['Age'] > 30) & (df['Salary'] < 70000)]
```

The `.query()` method allows you to filter rows using a query expression:

```
filtered_df = df.query('Age > 30 & Salary < 70000')
```

Select rows where the city is 'London':

```
filtered_df = df[df['City'] == 'London']
```

Or, using string methods to perform partial matches:

```
filtered_df = df[df['City'].str.contains('Lon')]
```

Select rows where the name is either 'John' or 'James':

```
filtered_df = df[df['Name'].isin(['John', 'James'])]
```

Filter rows where the age is greater than the average age:

```
average_age = df['Age'].mean()
filtered_df = df[df['Age'] > average_age]
```

Or, using a lambda function directly in the filter:

```
filtered_df = df[df['Age'].apply(lambda x: x > average_age)]
```

Select rows where the age is greater than 30, and only the 'Name' and 'Age' columns:

```
filtered_df = df.loc[df['Age'] > 30, ['Name', 'Age']]
```

Select the first three rows and the first two columns:

```
filtered_df = df.iloc[:3, :2]
```

Select rows where the age is NOT greater than 30:

```
filtered_df = df[~(df['Age'] > 30)]
```