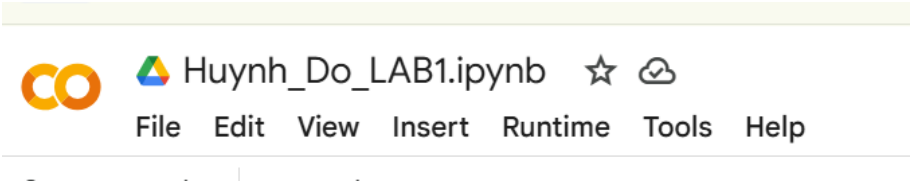
Huynh Do Lab#1:

Objective: To upload data file and to analyze the statistic of the uploaded ceral.csv



1. Import libraries

```
# Import required Python packages
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns  # Optional for better visuals
from google.colab import files
```

This code above imports several Python libraries commonly used in data analysis.

1. **import pandas as pd**
   - A library for data manipulation and analysis (e.g., working with tables, CSVs, etc.).
2. **import numpy as np :**
   - Support for numerical operations and arrays.
3. **from matplotlib import pyplot as plt**
   - Creating static, interactive, and animated plots.
4. **import seaborn as sns**
   - Imports **Seaborn** provides prettier and more informative plots.
5. **from google.colab import files**
   - Used in **Google Colab** to upload/download files from your local system within a Colab notebook.

2. Upload data

```
# Upload 'Cereals.csv' file
uploaded = files.upload()

# Load the dataset
df = pd.read_csv("Cereals.csv")
df.head()
```

The above screen shot is used to upload and load a CSV file named **Cereals.csv** into a Pandas Data Frame in a Google Colab environment.

When upload is done:

- **Cereals.csv**(text/csv) - 5055 bytes, last modified: 4/7/2025 - 100% done

```
Saving Cereals.csv to Cereals.csv
```

	name	mfr	type	calories	protein	fat	sodium	fiber	carbo	sugars	potass	vitamins	shelf	weight	cups	rating
0	100%_Bran	N	C	70	4	1	130	10.0	5.0	6.0	280.0	25	3	1.0	0.33	68.402973
1	100%_Natural_Bran	Q	C	120	3	5	15	2.0	8.0	8.0	135.0	0	3	1.0	1.00	33.983679
2	All-Bran	K	C	70	4	1	260	9.0	7.0	5.0	320.0	25	3	1.0	0.33	59.425505
3	All-Bran_with_Extra_Fiber	K	C	50	4	0	140	14.0	8.0	0.0	330.0	25	3	1.0	0.50	93.704912
4	Almond_Delight	R	C	110	2	2	200	1.0	14.0	8.0	NaN	25	3	1.0	0.75	34.384843

3. Prepare statistic

```
[ ]  # Summary statistics
     summary_stats = df.describe().T
     summary_stats['median'] = df.median(numeric_only=True)
     summary_stats
```

This code generates and displays **summary statistics** for the numeric columns in a Data Frame (df).

summary_stats = df.describe().T

- df.describe() returns **summary statistics** (like count, mean, std, min, max, and quartiles) for each numeric column.
- .T **transposes** the result, so rows become columns and vice versa

summary_stats['median'] = df.median(numeric_only=True)

- Calculates the **median** for each numeric column in df.
- Adds the median as a new column in the summary stats Data Frame.
- numeric_only=True ensures that only numeric columns are accepted.

summary_stats

- Displays the final summary statistics table, now including the **median** along with count, mean, std, min, max, and quartiles.

	count	mean	std	min	25%	50%	75%	max	median
calories	77.0	106.883117	19.484119	50.000000	100.000000	110.000000	110.000000	160.000000	110.000000
protein	77.0	2.545455	1.094790	1.000000	2.000000	3.000000	3.000000	6.000000	3.000000
fat	77.0	1.012987	1.006473	0.000000	0.000000	1.000000	2.000000	5.000000	1.000000
sodium	77.0	159.675325	83.832295	0.000000	130.000000	180.000000	210.000000	320.000000	180.000000
fiber	77.0	2.151948	2.383364	0.000000	1.000000	2.000000	3.000000	14.000000	2.000000
carbo	76.0	14.802632	3.907326	5.000000	12.000000	14.500000	17.000000	23.000000	14.500000
sugars	76.0	7.026316	4.378656	0.000000	3.000000	7.000000	11.000000	15.000000	7.000000
potass	75.0	98.666667	70.410636	15.000000	42.500000	90.000000	120.000000	330.000000	90.000000
vitamins	77.0	28.246753	22.342523	0.000000	25.000000	25.000000	25.000000	100.000000	25.000000
shelf	77.0	2.207792	0.832524	1.000000	1.000000	2.000000	3.000000	3.000000	2.000000
weight	77.0	1.029610	0.150477	0.500000	1.000000	1.000000	1.000000	1.500000	1.000000
cups	77.0	0.821039	0.232716	0.250000	0.670000	0.750000	1.000000	1.500000	0.750000
rating	77.0	42.665705	14.047289	18.042851	33.174094	40.400208	50.828392	93.704912	40.400208

✓ 1m 39s completed at 10:36 AM

The screen shot above is result of using python to generate statistic data.

4. Prepare Correlation matrix

```
# Correlation matrix
quantitative_df = df.select_dtypes(include=[np.number])
correlation_matrix = quantitative_df.corr()
correlation_matrix
```

This code is used to compute and display the **correlation matrix** for the numeric columns.

quantitative_df = df.select_dtypes(include=[np.number])

- This filters the DataFrame df to **keep only numeric columns** (e.g., integers, floats).

correlation_matrix = quantitative_df.corr()

- Calculates the **correlation matrix** for the numeric columns.
- The .corr() method computes **Pearson correlation coefficients** by default.

correlation_matrix

- Displays the correlation matrix.

	calories	protein	fat	sodium	fiber	carbo	sugars	potass	vitamins	shelf	weight	cups	rating
calories	1.000000	0.019066	0.498610	0.300649	-0.293413	0.257638	0.566533	-0.072063	0.265356	0.097234	0.696091	0.087200	-0.689376
protein	0.019066	1.000000	0.208431	-0.054674	0.500330	-0.025012	-0.291853	0.563706	0.007335	0.133865	0.216158	-0.244469	0.470618
fat	0.498610	0.208431	1.000000	-0.005407	0.016719	-0.300003	0.302497	0.200445	-0.031156	0.263691	0.214625	-0.175892	-0.409284
sodium	0.300649	-0.054674	-0.005407	1.000000	-0.070675	0.297687	0.058866	-0.042632	0.361477	-0.069719	0.308576	0.119665	-0.401295
fiber	-0.293413	0.500330	0.016719	-0.070675	1.000000	-0.380357	-0.138760	0.911528	-0.032243	0.297539	0.247226	-0.513061	0.584160
carbo	0.257638	-0.025012	-0.300003	0.297687	-0.380357	1.000000	-0.471184	-0.365003	0.219202	-0.192650	0.138467	0.367460	0.088712
sugars	0.566533	-0.291853	0.302497	0.058866	-0.138760	-0.471184	1.000000	0.001414	0.098231	0.068377	0.455844	-0.048961	-0.763902
potass	-0.072063	0.563706	0.200445	-0.042632	0.911528	-0.365003	0.001414	1.000000	-0.005427	0.385784	0.419933	-0.501607	0.416009
vitamins	0.265356	0.007335	-0.031156	0.361477	-0.032243	0.219202	0.098231	-0.005427	1.000000	0.299262	0.320324	0.128405	-0.240544
shelf	0.097234	0.133865	0.263691	-0.069719	0.297539	-0.192650	0.068377	0.385784	0.299262	1.000000	0.190762	-0.335269	0.025159
weight	0.696091	0.216158	0.214625	0.308576	0.247226	0.138467	0.455844	0.419933	0.320324	0.190762	1.000000	-0.199583	-0.298124
cups	0.087200	-0.244469	-0.175892	0.119665	-0.513061	0.367460	-0.048961	-0.501607	0.128405	-0.335269	-0.199583	1.000000	-0.203160
rating	-0.689376	0.470618	-0.409284	-0.401295	0.584160	0.088712	-0.763902	0.416009	-0.240544	0.025159	-0.298124	-0.203160	1.000000

The table above is the result of calling python **ntitative_df.corr()** library

5. Plot histogram for 'rating'

```
# Plot histogram for 'rating'
plt.figure(figsize=(8,5))
plt.hist(df['rating'], bins=15, edgecolor='black')
plt.title('Histogram of Cereal Ratings')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```

This code creates a **histogram** to visualize the distribution of cereal ratings.

plt.figure(figsize=(8,5))

- Creates a new figure for the plot.
- figsize=(8,5) sets the width to 8 inches and height to 5 inches.

plt.hist(df['rating'], bins=15, edgecolor='black')

- Plots a **histogram** of the rating column from the Data Frame df.
- bins=15 divides the rating range into 15 intervals (bars).

- edgecolor='black' adds a black border around each bar for clarity.

plt.grid(True)

- Adds a grid to the background of the plot for better readability.