CHAPTER 3: DATA MANIPULATION WITH PANDAS

(Week 3-4: Lecture Notes)

WEEK 3: INTRODUCTION TO PANDAS AND READING DATA

3.1. INTRODUCTION: WHY PANDAS? (The Excel Killer)

In engineering, data is usually kept in tables (Rows and Columns).

- **Excel:** Great, but slows down after 100,000 rows, crashes at 1 million.
- **Pandas:** The "Excel" of Python. It processes millions of rows in seconds. *(Name Origin: Derived from "PANel DAta").*

3.2. BASIC STRUCTURES: SERIES AND DATAFRAME

Pandas has two main building blocks:

1. **Series:** One-dimensional data. (A single column in Excel).
2. **DataFrame:** Two-dimensional data. (The entire table).

3.3. CREATING THE FIRST DATAFRAME

(Colab Application)

Step 1: Importing the Library

In [1]:
```
import pandas as pd
```

Step 2: Making a Table from a Dictionary

In [2]:
```
# Construction Site Data
data = {
    "Material": ["Concrete", "Iron", "Cement", "Brick", "Sand"],
    "Quantity": [100, 50, 30, 20, 60],
    "Price": [2000, 15000, 3000, 1500, 500]
}

df = pd.DataFrame(data)
print(df)
```

```
   Material  Quantity  Price
0  Concrete       100   2000
1      Iron        50  15000
2    Cement        30   3000
3     Brick        20   1500
4      Sand        60    500
```

3.4. READING DATA FROM EXTERNAL SOURCES

CSV files are the bread and butter of a data scientist.

In [3]:
```
# Let's pull the Titanic Dataset
url = "https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic
df = pd.read_csv(url)
```

3.5. GETTING TO KNOW THE DATA (Taking an X-Ray)

We loaded the data, but what is inside? Is there any corruption?

In [4]:
```
# 1. First 5 rows
print("--- First 5 Rows ---")
print(df.head())

# 2. Dimensions (Rows, Columns)
print("\n--- Dimensions ---")
print(df.shape)

# 3. General Information (Fill rates and Data types)
print("\n--- General Info ---")
df.info()

# 4. Statistical Summary
print("\n--- Statistical Summary ---")
print(df.describe().T)

# 5. MISSING DATA CHECK (Very Important!)
# How many empty (NaN) cells in which column?
print("\n--- Missing Data Check ---")
print(df.isnull().sum())
```

```
--- First 5 Rows ---
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                 Name     Sex   Age  SibSp  \
0                             Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                              Heikkinen, Miss. Laina  female  26.0      0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                            Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S

--- Dimensions ---
(891, 12)

--- General Info ---
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB

--- Statistical Summary ---
             count        mean         std   min       25%       50%    75%  \
PassengerId  891.0  446.000000  257.353842  1.00  223.5000  446.0000  668.5
Survived     891.0    0.383838    0.486592  0.00    0.0000    0.0000    1.0
Pclass       891.0    2.308642    0.836071  1.00    2.0000    3.0000    3.0
Age          714.0   29.699118   14.526497  0.42   20.1250   28.0000   38.0
SibSp        891.0    0.523008    1.102743  0.00    0.0000    0.0000    1.0
Parch        891.0    0.381594    0.806057  0.00    0.0000    0.0000    0.0
Fare         891.0   32.204208   49.693429  0.00    7.9104   14.4542   31.0

                 max
PassengerId  891.0000
Survived       1.0000
Pclass         3.0000
```

```
Age           80.0000
SibSp          8.0000
Parch          6.0000
Fare         512.3292

--- Missing Data Check ---
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

WEEK 4: CLEANING, FILTERING, AND MANIPULATION

We loaded the data. Now it is time to "sculpt" it.

4.1. COLUMN MANAGEMENT (Selecting and Renaming)

Sometimes column names come very messy (e.g., "Col_XA_12"). We must fix them.

In [5]:
```python
# 1. Selecting Columns
ages = df["Age"]
summary = df[["Name", "Age", "Fare"]]

# 2. Renaming Columns (Rename)
# Let's change "Fare" to "Ticket_Price".
df = df.rename(columns={"Fare": "Ticket_Price", "Sex": "Gender"})

# To make it permanent, we assigned it back to df. Or we could use inplace=True.
print(df.head())
```

```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name  Gender   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket  Ticket_Price Cabin Embarked
0      0         A/5 21171        7.2500   NaN        S
1      0          PC 17599       71.2833   C85        C
2      0  STON/O2. 3101282        7.9250   NaN        S
3      0            113803       53.1000  C123        S
4      0            373450        8.0500   NaN        S
```

4.2. ROW SELECTING: LOC and ILOC

- **.iloc:** Position based (0th row).
- **.loc:** Label based (row named "John").

In [6]:
```
print("--- 0th Row Information ---")
print(df.iloc[0])      # All info of the first row

print("\n--- First 5 Rows ---")
print(df.iloc[0:5])    # First 5 rows
```

```
--- 0th Row Information ---
PassengerId                          1
Survived                             0
Pclass                               3
Name            Braund, Mr. Owen Harris
Gender                            male
Age                               22.0
SibSp                                1
Parch                                0
Ticket                        A/5 21171
Ticket_Price                       7.25
Cabin                               NaN
Embarked                              S
Name: 0, dtype: object

--- First 5 Rows ---
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name  Gender   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket  Ticket_Price Cabin Embarked
0      0         A/5 21171        7.2500   NaN        S
1      0          PC 17599       71.2833   C85        C
2      0  STON/O2. 3101282        7.9250   NaN        S
3      0            113803       53.1000  C123        S
4      0            373450        8.0500   NaN        S
```

4.3. CONDITIONAL FILTERING (Writing Queries)

This is the code version of the filter in Excel. **Scenario:** Find those who are older than 30 AND whose Ticket Price is less than 100.

In [7]:
```python
# Single Condition
adults = df[ df["Age"] > 30 ]

# Multiple Conditions (&: And, |: Or)
my_filter = (df["Age"] > 30) & (df["Ticket_Price"] < 100)
cheap_adults = df[my_filter]

print(f"Shape of filtered data: {cheap_adults.shape}")
```

```
Shape of filtered data: (279, 12)
```

4.4. SORTING - Like in Excel

Sorting data by single or multiple criteria.

In [8]:
```
# 1. Single Criteria: From most expensive to cheapest by Ticket
sorted_df = df.sort_values(by="Ticket_Price", ascending=False)
print(sorted_df.head(3))

# 2. Multiple Criteria (Multi-key Sort)
# Sort first by Class (Pclass) (Small to large),
# Then by Price (Large to small).
multi_sorted_df = df.sort_values(by=["Pclass", "Ticket_Price"], ascending=[True,
print(multi_sorted_df.head(3))
```

```
     PassengerId  Survived  Pclass                                Name  \
258          259         1       1                   Ward, Miss. Anna
737          738         1       1              Lesurer, Mr. Gustave J
679          680         1       1  Cardeza, Mr. Thomas Drake Martinez

     Gender   Age  SibSp  Parch    Ticket  Ticket_Price        Cabin Embarked
258  female  35.0      0      0  PC 17755      512.3292          NaN        C
737    male  35.0      0      0  PC 17755      512.3292         B101        C
679    male  36.0      0      1  PC 17755      512.3292  B51 B53 B55        C
     PassengerId  Survived  Pclass                                Name  \
258          259         1       1                   Ward, Miss. Anna
679          680         1       1  Cardeza, Mr. Thomas Drake Martinez
737          738         1       1              Lesurer, Mr. Gustave J

     Gender   Age  SibSp  Parch    Ticket  Ticket_Price        Cabin Embarked
258  female  35.0      0      0  PC 17755      512.3292          NaN        C
679    male  36.0      0      1  PC 17755      512.3292  B51 B53 B55        C
737    male  35.0      0      0  PC 17755      512.3292         B101        C
```

4.5. MISSING DATA MANAGEMENT (Quick Intervention)

We will do detailed cleaning in week 5, but gaps give errors while analyzing. Let's learn simple intervention for now.

In [9]:
```
# 1. Drop Empty Rows (Dropna)
# Deletes the row with even the slightest gap. (Risk of data loss!)
clean_df = df.dropna()

# 2. Fill Gaps (Fillna) - Safer
# Fill the gaps in the age column with the average age.
average_age = df["Age"].mean()
df["Age"] = df["Age"].fillna(average_age)

# Check
print("Missing values after filling:")
print(df.isnull().sum())
```

```
Missing values after filling:
PassengerId       0
Survived          0
Pclass            0
Name              0
Gender            0
Age               0
SibSp             0
Parch             0
Ticket            0
Ticket_Price      0
Cabin           687
Embarked          2
dtype: int64
```

4.6. NEW COLUMN AND INDEX RESET

As we process the data (filtering, sorting), row numbers (indices) get mixed up (1, 5, 100, 2...). We must fix this.

In [10]:
```python
# 1. Deriving New Column
df["Price_TL"] = df["Ticket_Price"] * 30

# 2. Resetting Index
# Fixes the mixed sequence numbers to (0, 1, 2, 3...).
# If we don't say drop=True, it keeps the old indices as a new column.
df = df.reset_index(drop=True)
print(df.head())
```

```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name  Gender   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket  Ticket_Price Cabin Embarked  Price_TL
0      0         A/5 21171        7.2500   NaN        S   217.500
1      0          PC 17599       71.2833   C85        C  2138.499
2      0  STON/O2. 3101282        7.9250   NaN        S   237.750
3      0            113803       53.1000  C123        S  1593.000
4      0            373450        8.0500   NaN        S   241.500
```

5. STATISTICAL ANALYSIS (Groupby)

It is the logic of Pivot (Summary) table. **Question:** What is the average price of each ticket class (Pclass)?

In [11]:
```python
print( df.groupby("Pclass")["Ticket_Price"].mean() )
```

```
Pclass
1    84.154687
2    20.662183
3    13.675550
Name: Ticket_Price, dtype: float64
```

Lecture Summary

1. **Parenthesis Error:** `isnull()` is a function, if you don't put `()` at the end, it won't work. `sum()` is the same.
2. **Permanence (Inplace):** For your changes (rename, drop, sort) to be permanent, either assign `df = ...` or add `inplace=True` inside the parenthesis.
3. **Index Trap:** If you say `df.iloc[0]` after filtering, it looks for the original index 0 (which might be filtered out), not the one at the top of the list. Therefore, doing `reset_index()` after complex tasks saves lives.