

★ EXAM QUESTION 7 MANDATORY SET

Python & Pandas Lab Solutions Guide

Extracted directly from Dr. Rashed Mazumder's classroom lab notebooks (`data-analy.ipynb` & `test-train.ipynb`).

1. Data Loading & Basic Inspection

How RM Sir initializes Pandas and loads CSV datasets:

```
import pandas as RPD

# 1. Reading from CSV
df = RPD.read_csv('panda_edit.csv')

# 2. Inspecting Rows
print(df.head(10))    # First 10 rows
print(df.tail(10))    # Last 10 rows

# 3. Viewing Specific Rows with .loc[]
print(df.loc[[0, 1]])    # Rows by index 0 and 1
print(df.loc[7:9, 'Duration']) # Rows 7 to 9 of Duration column
```

2. Handling Missing Values (NaN / Empty Cells)

Three core approaches taught in class for dealing with null values:

Method A: Dropping Empty Cells (dropna)

```
# Return a new DataFrame without empty cells (non-destructive)
clean_df = df.dropna()
```

```
# Drop rows with empty cells in-place (modifies original)
df.dropna(inplace=True)
```

Method B: Imputing with a Constant or Column Specific Value (fillna)

```
# Replace ALL empty cells in the dataset with 150
df.fillna(150, inplace=True)

# Replace empty cells in a SPECIFIC column only:
df.fillna({"Calories": 150}, inplace=True)
```

Method C: Imputing with Column Mode / Mean / Median

```
# Find the mode (most frequent value) of a column and fill
mode_val = df["ID"].mode()[1]
df.fillna({"ID": mode_val}, inplace=True)

# Forward fill empty values from preceding rows
df_ffill = df.ffill(axis=0)
```

3. Handling Outliers & Data Errors

RM Sir demonstrated both manual slice correction and automated conditional loops:

Method A: Manual Row-Index Correction

```
# Fix erroneous data in specific rows
df.loc[7:9, 'Duration'] = 29
```

Method B: Conditional Value Replacement (Outlier Clamping)

```
# If Duration is greater than 59, clamp/replace it with 22
for rm in df.index:
    if df.loc[rm, "Duration"] > 59:
        df.loc[rm, "Duration"] = 22
```

Method C: Conditional Row Dropping

```
# If Duration is greater than 59, drop the entire row from the dataset
for rm in df.index:
    if df.loc[rm, "Duration"] > 59:
        df.drop(rm, inplace=True)
```

4. Detecting & Removing Duplicates

How to identify and purge identical rows:

```
# Check for duplicate rows (returns boolean series: True if duplicate)
duplicates_mask = df.duplicated()
print(duplicates_mask.head(40))

# Remove duplicate rows permanently in-place
df.drop_duplicates(inplace=True)
```

5. Data Visualization with Matplotlib

The standard visualization syntax used in class:

```
import matplotlib.pyplot as PLT

# 1. Basic Line Plot of entire DataFrame
df.plot()
PLT.show()

# 2. Scatter Plot between two columns (e.g., Duration vs Calories)
df.plot(kind='scatter', x='Duration', y='Calories')
PLT.title("Duration vs Calories Scatter Plot")
PLT.show()

# 3. Histogram of a single column
df["Calories"].plot(kind='hist')
```

```
PLT.title("Calories Distribution Histogram")  
PLT.show()
```

6. Train / Test Split (from test-train.ipynb)

How the 80/20 split was coded in class:

```
import numpy as np  
  
# Splitting 100 observations into 80% train and 20% test:  
train_x = x[:80]  
train_y = y[:80]  
  
test_x = x[80:]  
test_y = y[80:]  
  
# Scatter plot of training set:  
plt.scatter(train_x, train_y)  
plt.title("Training Set Distribution (80%)")  
plt.show()  
  
# Scatter plot of testing set:  
plt.scatter(test_x, test_y)  
plt.title("Testing Set Distribution (20%)")  
plt.show()
```