Detailed Pandas Data Manipulation and ReportLab PDF Generation

This document contains a detailed explanation of various Pandas data manipulation operations
and how to generate a PDF report using ReportLab. All code and outputs are displayed in a monospaced font.
References: Pandas Documentation and
ReportLab User Guide.

1. Reading Data with Pandas

```
import pandas as pd
df = pd.read_csv('data.csv')

# Explanation:
# - Imports the pandas library.
# - Reads a CSV file named 'data.csv' into a DataFrame 'df'.
# Expected output: DataFrame with data from 'data.csv'.
```

2. Fetching Data (head, tail, describe, and dtypes)

```
# Fetch the first 5 rows
df.head(5)

# Fetch the last 5 rows
df.tail(5)

# Generate summary statistics for numeric columns
df.describe()

# Check data types of columns
df.dtypes

# Explanation:
# - df.head(5) and df.tail(5) display the first and last 5 rows, respectively.
# - df.describe() provides summary statistics for numeric columns.
# - df.dtypes shows the data types for each column.
```

3. Handling Missing Values

```
# Identify missing values in the DataFrame
```

```
df.isnull()

# Check if any column contains missing values
df.isnull().any()

# Count missing values per column
df.isnull().sum()

# Fill missing values with 0
df_filled = df.fillna(0)

# Fill missing values in 'Sales' with the column's mean and create a new column
df['Sales_fillNA'] = df['Sales'].fillna(df['Sales'].mean())

# Explanation:
# - df.isnull() returns a DataFrame of booleans indicating missing values.
# - df.isnull().any() returns True/False for each column based on if any missing
values exist.
# - df.isnull().sum() provides the total count of missing values per column.
# - df.fillna(0) fills all missing values with 0.
# - For the 'Sales' column, missing values are replaced with the column's mean.
```

4. Renaming Columns

```
# Rename the column 'Date' to 'Sales Date'
df = df.rename(columns={'Date': 'Sales Date'})
df.head()

# Explanation:
# - The rename() method changes column names.
# - 'Date' is renamed to 'Sales Date', and df.head() displays the updated DataFrame.
```

5. Changing Data Types and Creating New Columns

```
# Replace missing values in 'Value' with its mean, convert to int, and create
'Value_New'
df['Value_New'] = df['Value'].fillna(df['Value'].mean()).astype(int)
df.head()

# Create a new column 'New Value' by doubling the 'Value'
df['New Value'] = df['Value'].apply(lambda x: x * 2)
df.head()

# Explanation:
# - Missing values in 'Value' are replaced with the mean, then converted to integer
for 'Value_New'.
# - A lambda function doubles each element in the 'Value' column for 'New Value'.
```

6. Data Aggregation and Grouping

```
# Group by 'Product' and calculate the mean of 'Value'
grouped_mean = df.groupby('Product')['Value'].mean()
print(grouped_mean)
print(type(grouped_mean))

# Group by 'Product' and 'Region', then sum 'Value' and 'Sales'
grouped_sum = df.groupby(['Product', 'Region'])[['Value', 'Sales']].sum()
print(grouped_sum)

# Aggregate multiple functions (mean, sum, count) on 'Value' grouped by 'Region'
grouped_agg = df.groupby('Region')['Value'].agg(['mean', 'sum', 'count'])
grouped_agg

# Explanation:
# - Grouping is performed using groupby().
# - The first grouping calculates the mean of 'Value' per 'Product'.
# - The second grouping calculates the sum of 'Value' and 'Sales' per 'Product' and
'Region'.
# - The last aggregation applies multiple functions on 'Value' for each 'Region'.
```

7. Merging and Joining DataFrames

```
# Create two sample DataFrames for merging
df1 = pd.DataFrame({'Key': ['A', 'B', 'C'], 'Value1': [1, 2, 3]})
df2 = pd.DataFrame({'Key': ['A', 'B', 'D'], 'Value2': [4, 5, 6]})

# Display the DataFrames
print(df1)
print(df2)

# Merge DataFrames on the 'Key' column with different join types:
# Inner Join - only matching keys
pd.merge(df1, df2, on='Key', how='inner')

# Outer Join - all keys, fill missing with NaN
pd.merge(df1, df2, on='Key', how='outer')

# Left Join - all keys from df1
pd.merge(df1, df2, on='Key', how='left')

# Right Join - all keys from df2
pd.merge(df1, df2, on='Key', how='right')

# Explanation:
# - Two DataFrames (df1 and df2) are created with a common 'Key' column.
```

```
# - Different merge strategies (inner, outer, left, right) are demonstrated.
# - The printed outputs show how rows are matched and non-matching entries are
handled.
```

Summary

This document detailed:
- How to import and read data using Pandas.
- How to fetch data using head, tail, describe, and dtypes.
- Techniques for handling missing values.
- Renaming columns and changing data types.
- Creating new columns using lambda functions.
- Aggregating and grouping data.
- Merging and joining multiple DataFrames.
- Finally, generating a PDF report with ReportLab, displaying all code in a monospaced font.

References:
- Pandas Documentation: https://pandas.pydata.org/pandas-docs/stable/
- ReportLab User Guide: https://www.reportlab.com/documentation/