

SIT225: Data wrangling

Run each cell to generate output and finally convert this notebook to PDF.

```
# Fill in student ID and name
#
student_id = ""
student_first_last_name = ""
print(student_id, student_first_last_name)
```

Read the Data with Pandas

Pandas has a dedicated function `read_csv()` to read CSV files.

Just in case we have a large number of data, we can just show into only five rows with `head` function. It will show you 5 rows data automatically.

```
import pandas as pd

data_file = "shopping_data.csv"
csv_data = pd.read_csv(data_file)

print(csv_data)

# show into only five rows with head function
print(csv_data.head())
```

Access the Column

Pandas has provided function `.columns` to access the column of the data source.

```
print(csv_data.columns)

# if we want to access just one column, for example "Age"
print("Age:")
print(csv_data["Age"])
```

Access the Row

In addition to accessing data through columns, using pandas can also access using rows. In contrast to access through columns, the function to display data from a row is the `.iloc[i]` function where `[i]` indicates the order of the rows to be displayed where the index starts from 0.

```
# we want to know what line 5 contains
```

```

print(csv_data.iloc[5])

print()

# We can combine both of those function to show row and column we want.
# For the example, we want to show the value in column "Age" at the first row
# (remember that the row starts at 0)
#
print(csv_data["Age"].iloc[1])

```

Show Data Based on Range

After displaying a data set, what if you want to display data from rows 5 to 20 of a dataset? To anticipate this, pandas can also display data within a certain range, both ranges for rows only, only columns, and ranges for rows and columns

```

print("Shows data to 5th to less than 10th in a row:")
print(csv_data.iloc[5:10])

```

Using Numpy to Show the Statistic Information

The describe() function allows to quickly find statistical information from a dataset. Those information such as mean, median, modus, max min, even standard deviation. Don't forget to install Numpy before using describe function.

```

print(csv_data.describe(include="all"))

```

Handling Missing Value

```

# For the first step, we will figure out if there is missing value.
print(csv_data.isnull().values.any())
print()

# We will use another data source with missing values to practice this part.
data_missing = pd.read_csv("shopping_data_missingvalue.csv")
print(data_missing.head())

print()

print("Missing? ", data_missing.isnull().values.any())

```

Ways to deal with missing values.

Follow the tutorial (<https://deepnote.com/app/rickyharyanto14-3390/Data-Wrangling-w-Python-e5d1a23e-33cf-416d-ad27-4c3f7f467442>). It includes -

1. Delete data
 - deleting rows
 - pairwise deletion
 - delete column
2. imputation
 - time series problem
 - Data without trend with seasonality (mean, median, mode, random)
 - Data with trend and without seasonality (linear interpolation)
 - general problem
 - Data categorical (Make NA as multiple imputation)
 - Data numerical or continuous (mean, median, mode, multiple imputation and linear regression)

Filling with Mean Values

The mean is used for data that has a few outliers/noise/anomalies in the distribution of the data and its contents. This value will later fill in the empty value of the dataset that has a missing value case. To fill in an empty value use the `fillna()` function

```
print(data_missing.mean())

"""

Question: This code will generate error. Can you explain why and how
it can be solved?
Move on to the next cell to find one way it can be solved.

Answer: <your answer>

"""

# Genre column contains string values and numerical operation mean
fails.
# Lets drop Genre column since for numerical calculation.
#
data_missing_wo_genre = data_missing.drop(columns=['Genre'])
print(data_missing_wo_genre.head())

print(data_missing_wo_genre.mean())

print("Dataset with empty values! :")
print(data_missing_wo_genre.head(10))

data_filling=data_missing_wo_genre.fillna(data_missing_wo_genre.mean())
```

```

)
print("Dataset that has been processed Handling Missing Values with
Mean :")
print(data_filling.head(10))

# Observe the missing value imputation in corresponding rows.
#

```

Filling with Median

The median is used when the data presented has a high outlier. The median was chosen because it is the middle value, which means it is not the result of calculations involving outlier data. In some cases, outlier data is considered disturbing and often considered noisy because it can affect class distribution and interfere with clustering analysis.

```

print(data_missing_wo_genre.median())
print("Dataset with empty values! :")
print(data_missing_wo_genre.head(10))

data_filling2=data_missing_wo_genre.fillna(data_missing_wo_genre.media
n())
print("Dataset that has been processed Handling Missing Values with
Median :")
print(data_filling2.head(10))

# Observe the missing value imputation in corresponding rows.
#

```