Assignment_01_DMW

August 19, 2025

1 Assignment - 1

Name - Sayali Pawar

PRN - 123B1B229

DIV - D

PROBLEM STATEMENT:

Pre Processing Techniques: Create a dummy dataset or with missing values and duplicate entries or select any data set with missing values (such as Iris dataset, breast cancer dataset) from any repository of data such as SK-Learn, UCI library, Kaggle dataset library etc. Write a program or use a suitable tool to perform the following operations on the selected dataset and display the result. 1. Removal of duplicates 2. Handle missing values 3. Normalizing the data using normalizing technique 4. Apply min-max scalar / Robust scalar / standard scalar to scale the data 5. Use measures of Central Tendency and Dispersion of Data

1.1 1. Importing Libraries and Loading Dataset

[1]:
```
# Importing libraries
import pandas as pd
import numpy as np
from sklearn import preprocessing
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

[2]:
```
# Load the dataset
data = pd.read_csv("/content/drive/MyDrive/TY_DMW/weatherHistory.csv",␣
 ↪delimiter=",")
print("Original shape:", data.shape)
```

```
Original shape: (96453, 12)
```

1.2 2. Initial Data Exploration

```
[3]: data.head()
```

```
[3]:                  Formatted Date        Summary Precip Type  Temperature (C)  \
     0  2006-04-01 00:00:00.000 +0200  Partly Cloudy        rain         9.472222
     1  2006-04-01 01:00:00.000 +0200  Partly Cloudy        rain         9.355556
     2  2006-04-01 02:00:00.000 +0200  Mostly Cloudy        rain         9.377778
     3  2006-04-01 03:00:00.000 +0200  Partly Cloudy        rain         8.288889
     4  2006-04-01 04:00:00.000 +0200  Mostly Cloudy        rain         8.755556

        Apparent Temperature (C)  Humidity  Wind Speed (km/h)  \
     0                  7.388889      0.89            14.1197
     1                  7.227778      0.86            14.2646
     2                  9.377778      0.89             3.9284
     3                  5.944444      0.83            14.1036
     4                  6.977778      0.83            11.0446

        Wind Bearing (degrees)  Visibility (km)  Loud Cover  Pressure (millibars)  \
     0                   251.0          15.8263         0.0               1015.13
     1                   259.0          15.8263         0.0               1015.63
     2                   204.0          14.9569         0.0               1015.94
     3                   269.0          15.8263         0.0               1016.41
     4                   259.0          15.8263         0.0               1016.51

                            Daily Summary
     0  Partly cloudy throughout the day.
     1  Partly cloudy throughout the day.
     2  Partly cloudy throughout the day.
     3  Partly cloudy throughout the day.
     4  Partly cloudy throughout the day.
```

```
[4]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 96453 entries, 0 to 96452
Data columns (total 12 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Formatted Date            96453 non-null  object
 1   Summary                   96453 non-null  object
 2   Precip Type               95936 non-null  object
 3   Temperature (C)           96453 non-null  float64
 4   Apparent Temperature (C)  96453 non-null  float64
 5   Humidity                  96453 non-null  float64
 6   Wind Speed (km/h)         96453 non-null  float64
 7   Wind Bearing (degrees)    96453 non-null  float64
 8   Visibility (km)           96453 non-null  float64
 9   Loud Cover                96453 non-null  float64
 10  Pressure (millibars)      96453 non-null  float64
 11  Daily Summary             96453 non-null  object
dtypes: float64(8), object(4)
memory usage: 8.8+ MB
```

[5]:
```
data.describe()
```

[5]:
```
       Temperature (C)  Apparent Temperature (C)      Humidity  \
count     96453.000000              96453.000000  96453.000000
mean         11.932678                 10.855029      0.734899
std           9.551546                 10.696847      0.195473
min         -21.822222                -27.716667      0.000000
25%           4.688889                  2.311111      0.600000
50%          12.000000                 12.000000      0.780000
75%          18.838889                 18.838889      0.890000
max          39.905556                 39.344444      1.000000

       Wind Speed (km/h)  Wind Bearing (degrees)  Visibility (km)  Loud Cover  \
count       96453.000000            96453.000000     96453.000000     96453.0
mean           10.810640              187.509232        10.347325         0.0
std             6.913571              107.383428         4.192123         0.0
min             0.000000                0.000000         0.000000         0.0
25%             5.828200              116.000000         8.339800         0.0
50%             9.965900              180.000000        10.046400         0.0
75%            14.135800              290.000000        14.812000         0.0
max            63.852600              359.000000        16.100000         0.0

       Pressure (millibars)
count          96453.000000
mean            1003.235956
std              116.969906
min                0.000000
25%             1011.900000
50%             1016.450000
75%             1021.090000
max             1046.380000
```

[6]:
```
data.dtypes
```

[6]:
```
Formatted Date                object
Summary                       object
Precip Type                   object
Temperature (C)              float64
Apparent Temperature (C)     float64
Humidity                     float64
Wind Speed (km/h)            float64
Wind Bearing (degrees)      float64
Visibility (km)             float64
Loud Cover                  float64
Pressure (millibars)        float64
Daily Summary                object
dtype: object
```

1.3 3. Data Cleaning and Normalizing

[7]:

```
# 3.1 Remove Duplicates
duplicate = data[data.duplicated()]
print(duplicate.count())
```

```
Formatted Date              24
Summary                     24
Precip Type                 24
Temperature (C)             24
Apparent Temperature (C)    24
Humidity                    24
Wind Speed (km/h)           24
Wind Bearing (degrees)      24
Visibility (km)             24
Loud Cover                  24
Pressure (millibars)        24
Daily Summary               24
dtype: int64
```

[8]:

```
data.drop_duplicates(keep=False, inplace=True)
data.shape
```

[8]:

```
(96405, 12)
```

[9]:

```
# 3.2 Replacing missing values
data.isnull()
```

[9]:

```
       Formatted Date  Summary  Precip Type  Temperature (C)  \
0               False    False        False            False
1               False    False        False            False
2               False    False        False            False
3               False    False        False            False
4               False    False        False            False
…                 …        …            …               …
96448           False    False        False            False
96449           False    False        False            False
96450           False    False        False            False
96451           False    False        False            False
96452           False    False        False            False
```

```
       Apparent Temperature (C)  Humidity  Wind Speed (km/h)  \
0                         False     False              False
1                         False     False              False
2                         False     False              False
3                         False     False              False
4                         False     False              False
…                         …       …                …
96448                     False     False              False
96449                     False     False              False
96450                     False     False              False
96451                     False     False              False
96452                     False     False              False

       Wind Bearing (degrees)  Visibility (km)  Loud Cover  \
0                       False            False       False
1                       False            False       False
2                       False            False       False
3                       False            False       False
4                       False            False       False
…                       …              …         …
96448                   False            False       False
96449                   False            False       False
96450                   False            False       False
96451                   False            False       False
96452                   False            False       False

       Pressure (millibars)  Daily Summary
0                     False          False
1                     False          False
2                     False          False
3                     False          False
4                     False          False
…                     …            …
96448                 False          False
96449                 False          False
96450                 False          False
96451                 False          False
96452                 False          False

[96405 rows x 12 columns]
```

```
[10]: data.isnull().sum()
```

```
[10]: Formatted Date                0
      Summary                       0
      Precip Type                 517
```

```
Temperature (C)             0
Apparent Temperature (C)    0
Humidity                    0
Wind Speed (km/h)           0
Wind Bearing (degrees)      0
Visibility (km)             0
Loud Cover                  0
Pressure (millibars)        0
Daily Summary               0
dtype: int64
```

[11]:

```
#Separate numeric and categorical columns
num_cols = data.select_dtypes(include=[np.number]).columns
cat_cols = data.select_dtypes(exclude=[np.number]).columns

# Impute missing values
# --- numeric: median ---
from sklearn.impute import SimpleImputer
imputer_median = SimpleImputer(strategy="median")
data[num_cols] = imputer_median.fit_transform(data[num_cols])

# --- categorical: most frequent ---
imputer_freq = SimpleImputer(strategy="most_frequent")
data[cat_cols] = imputer_freq.fit_transform(data[cat_cols])

print("Missing values after imputation:")
print(data.isnull().sum())
```

```
Missing values after imputation:
Formatted Date              0
Summary                     0
Precip Type                 0
Temperature (C)             0
Apparent Temperature (C)    0
Humidity                    0
Wind Speed (km/h)           0
Wind Bearing (degrees)      0
Visibility (km)             0
Loud Cover                  0
Pressure (millibars)        0
Daily Summary               0
dtype: int64
```

[12]:

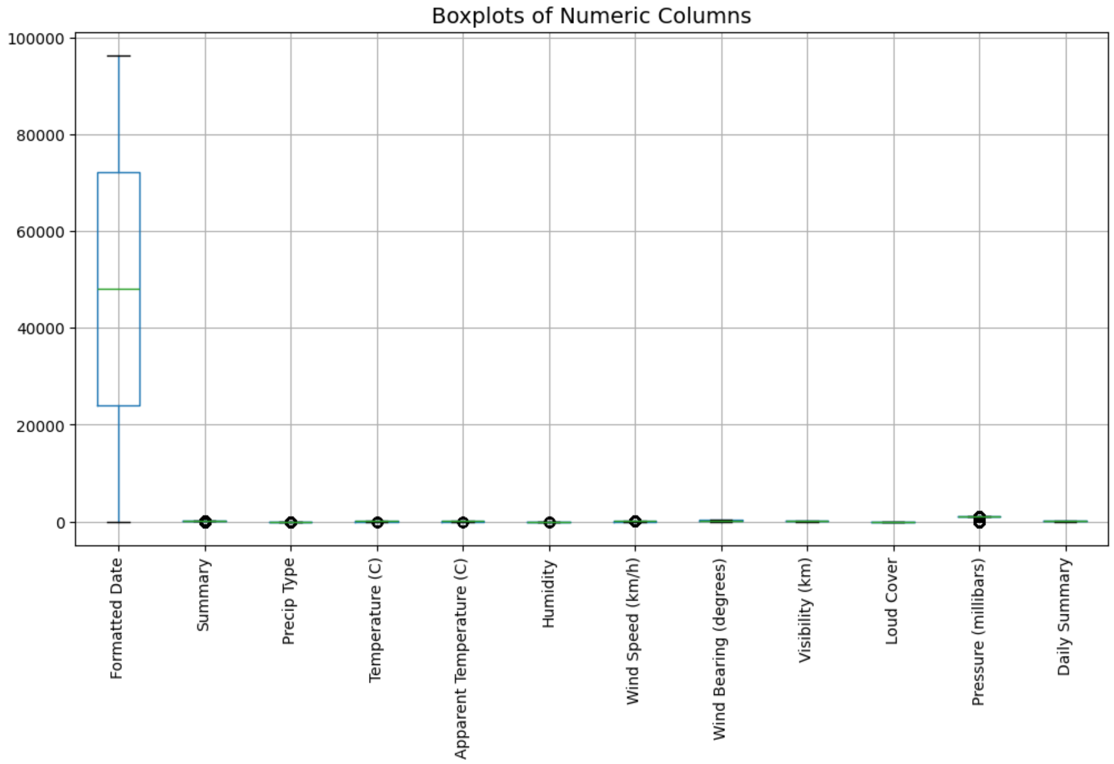
```
# 3.3 Encode categorical columns with LabelEncoder
from sklearn.preprocessing import LabelEncoder
label_encoders = {}
for col in cat_cols:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le   # save encoders in case you need inverse␣
  ↪transform

print("\nFirst 5 rows after Label Encoding:")
print(data.head())
```

```
First 5 rows after Label Encoding:
   Formatted Date  Summary  Precip Type  Temperature (C)  \
0            2159       19            0         9.472222
1            2160       19            0         9.355556
2            2161       17            0         9.377778
3            2162       19            0         8.288889
4            2163       17            0         8.755556

   Apparent Temperature (C)  Humidity  Wind Speed (km/h)  \
0                  7.388889      0.89            14.1197
1                  7.227778      0.86            14.2646
2                  9.377778      0.89             3.9284
3                  5.944444      0.83            14.1036
4                  6.977778      0.83            11.0446

   Wind Bearing (degrees)  Visibility (km)  Loud Cover  Pressure (millibars)  \
0                   251.0          15.8263         0.0               1015.13
1                   259.0          15.8263         0.0               1015.63
2                   204.0          14.9569         0.0               1015.94
3                   269.0          15.8263         0.0               1016.41
4                   259.0          15.8263         0.0               1016.51

   Daily Summary
0            197
1            197
2            197
3            197
4            197
```

1.4 4. Exploratory Data Analysis (EDA)

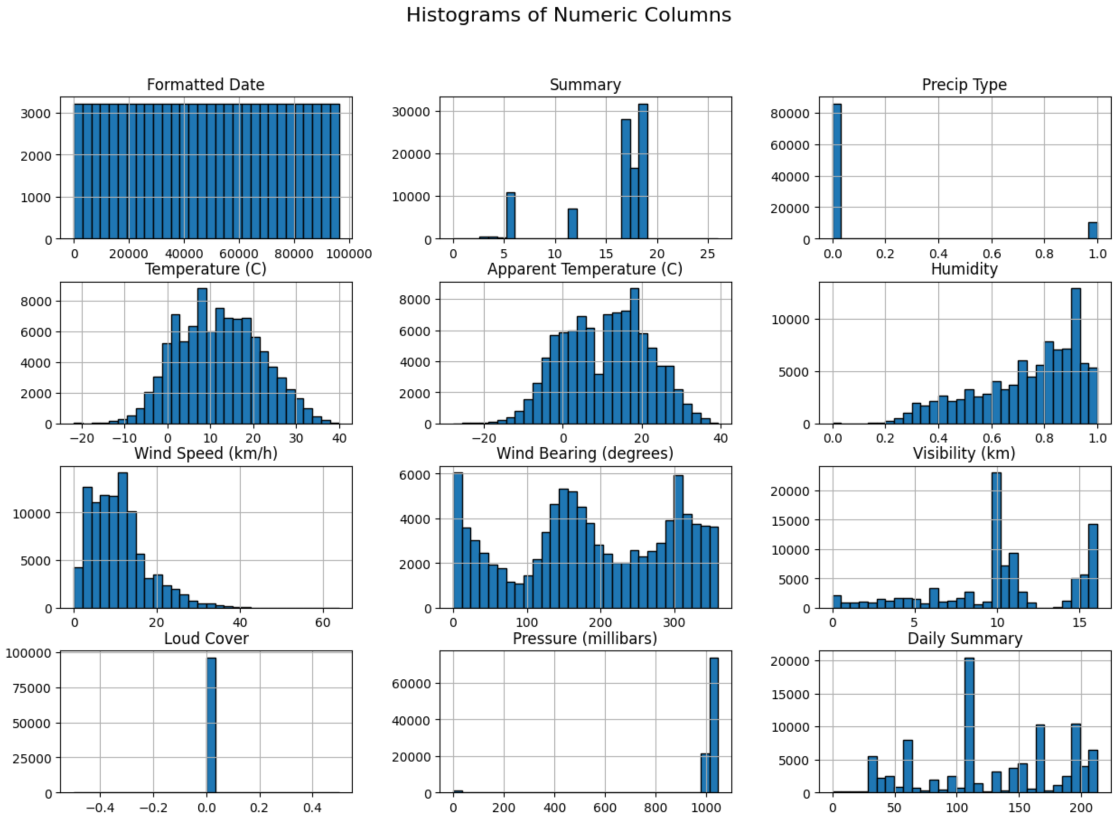
[13]:

```
# 1) Boxplot for numeric columns
# -------------------------------
plt.figure(figsize=(12,6))
data.boxplot(rot=90)
plt.title("Boxplots of Numeric Columns", fontsize=14)
plt.show()
```



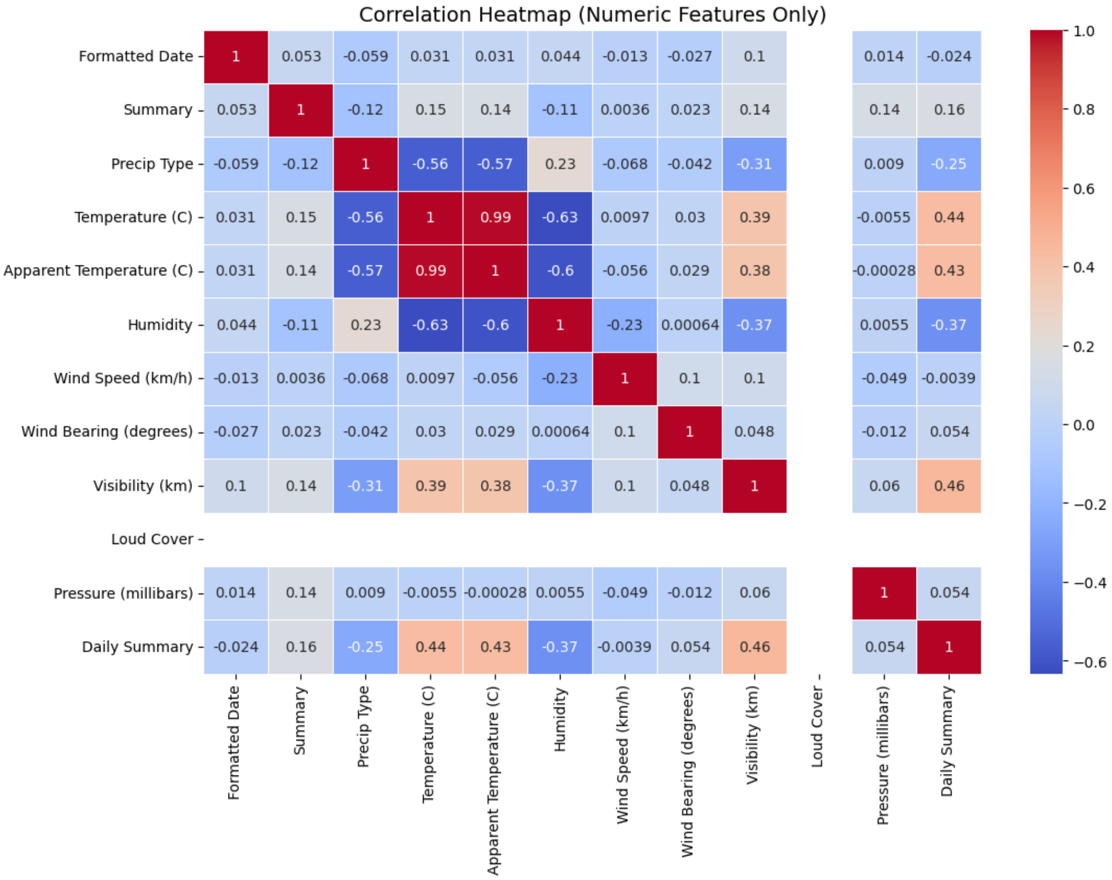
[14]:

```
# 2) Histogram for numeric columns
# -------------------------------
data.hist(figsize=(15,10), bins=30, edgecolor="black")
plt.suptitle("Histograms of Numeric Columns", fontsize=16)
plt.show()
```



[15]:

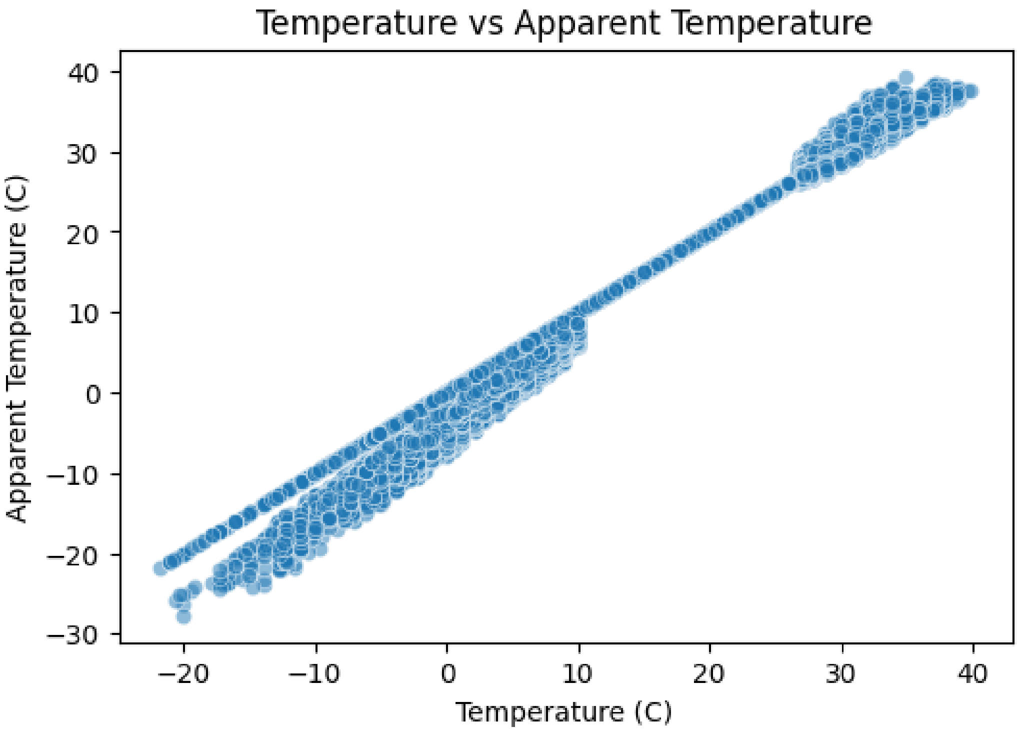
```
# 3) Heatmap of correlations
import matplotlib.pyplot as plt
import seaborn as sns

# Select only numeric columns
numeric_data = data.select_dtypes(include=['int64', 'float64'])

# Compute correlation
corr = numeric_data.corr()

# Plot heatmap
plt.figure(figsize=(12,8))
sns.heatmap(corr, annot=True, cmap="coolwarm", linewidths=0.5)
plt.title("Correlation Heatmap (Numeric Features Only)", fontsize=14)
plt.show()
```



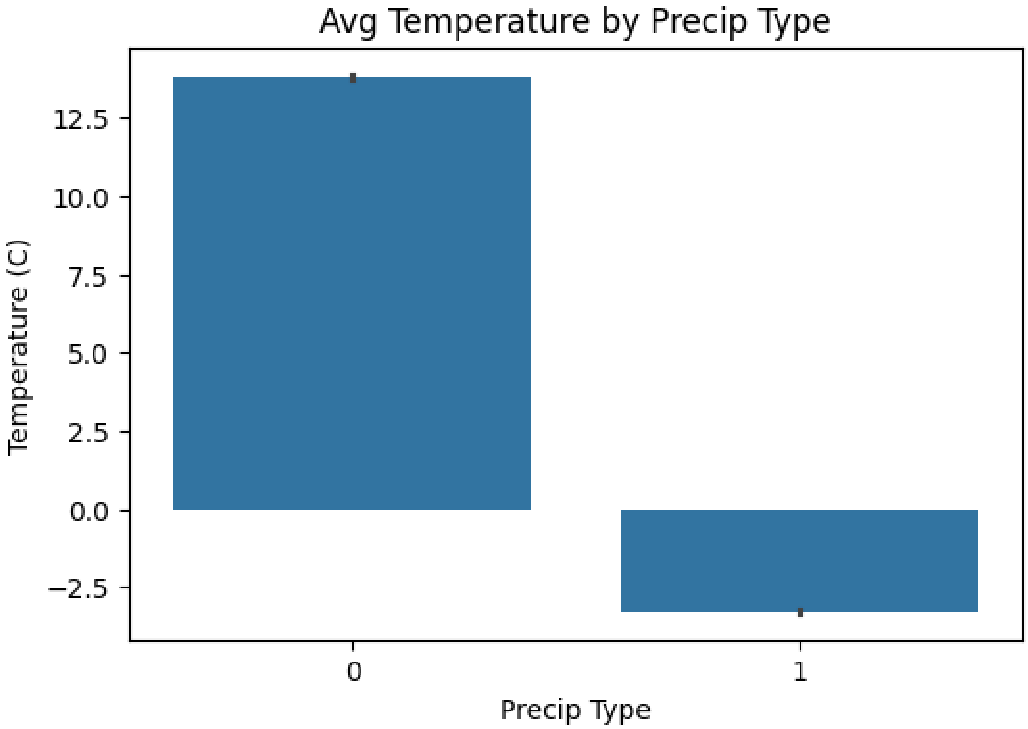
	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
Formatted Date	1	0.053	-0.059	0.031	0.031	0.044	-0.013	-0.027	0.1		0.014	-0.024
Summary	0.053	1	-0.12	0.15	0.14	-0.11	0.0036	0.023	0.14		0.14	0.16
Precip Type	-0.059	-0.12	1	-0.56	-0.57	0.23	-0.068	-0.042	-0.31		0.009	-0.25
Temperature (C)	0.031	0.15	-0.56	1	0.99	-0.63	0.0097	0.03	0.39		-0.0055	0.44
Apparent Temperature (C)	0.031	0.14	-0.57	0.99	1	-0.6	-0.056	0.029	0.38		-0.00028	0.43
Humidity	0.044	-0.11	0.23	-0.63	-0.6	1	-0.23	0.00064	-0.37		0.0055	-0.37
Wind Speed (km/h)	-0.013	0.0036	-0.068	0.0097	-0.056	-0.23	1	0.1	0.1		-0.049	-0.0039
Wind Bearing (degrees)	-0.027	0.023	-0.042	0.03	0.029	0.00064	0.1	1	0.048		-0.012	0.054
Visibility (km)	0.1	0.14	-0.31	0.39	0.38	-0.37	0.1	0.048	1		0.06	0.46
Loud Cover												
Pressure (millibars)	0.014	0.14	0.009	-0.0055	-0.00028	0.0055	-0.049	-0.012	0.06		1	0.054
Daily Summary	-0.024	0.16	-0.25	0.44	0.43	-0.37	-0.0039	0.054	0.46		0.054	1

[16]:

```
# 4) Scatter plot: Temperature vs Apparent Temperature
plt.figure(figsize=(6,4))
sns.scatterplot(x="Temperature (C)", y="Apparent Temperature (C)", data=data,␣
 ↪alpha=0.5)
plt.title("Temperature vs Apparent Temperature")
plt.show()
```



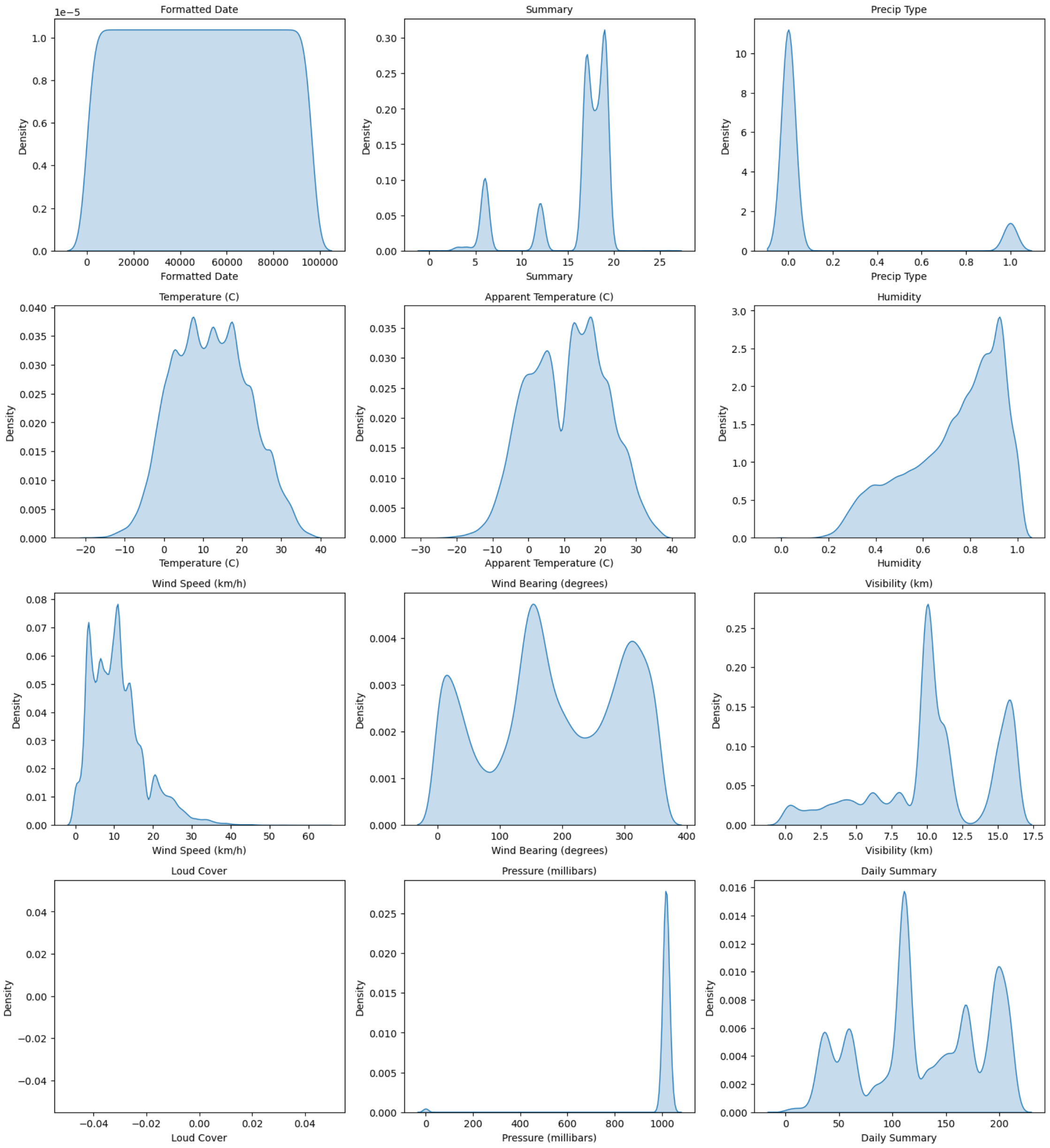
[17]:
```
# 5) Barplot: Average Temperature per Precip Type
plt.figure(figsize=(6,4))
sns.barplot(x="Precip Type", y="Temperature (C)", data=data)
plt.title("Avg Temperature by Precip Type")
plt.show()
```



[18]:

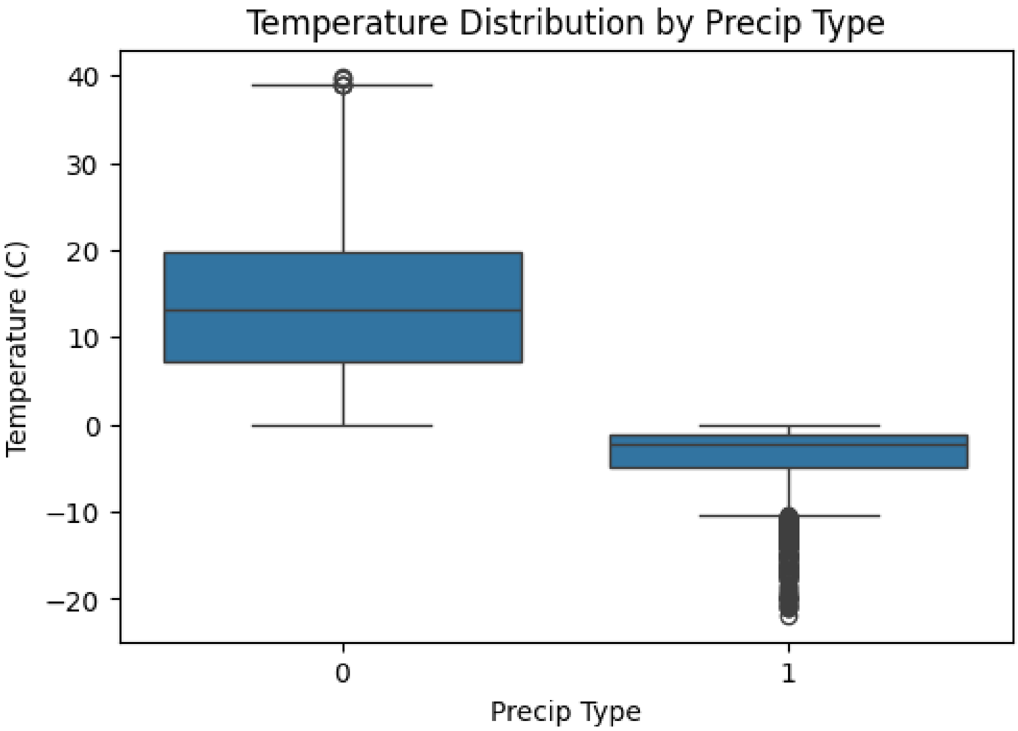
```
# 6) KDE (density) plots for numeric variables
# Select numeric columns
numeric_cols = data.select_dtypes(include=['int64','float64']).columns

# Set up subplot grid
n = len(numeric_cols)
rows = (n // 3) + 1   # 3 plots per row
cols = 3

plt.figure(figsize=(15, 4*rows))

for i, col in enumerate(numeric_cols, 1):
    plt.subplot(rows, cols, i)
    sns.kdeplot(data[col], fill=True)
    plt.title(col, fontsize=10)

plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-2831935248.py:14: UserWarning: Dataset has 0 variance;
skipping density estimate. Pass `warn_singular=False` to disable this warning.
  sns.kdeplot(data[col], fill=True)
```



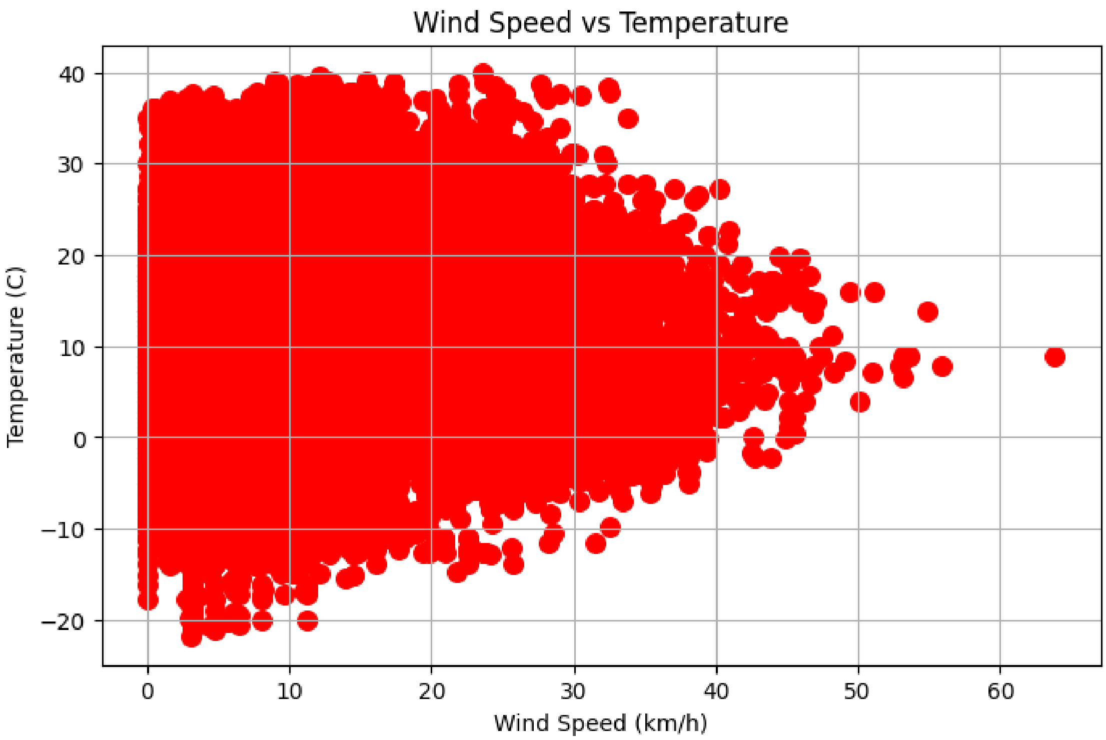
[19]:

```
# 7) Boxplot: Temperature by Precip Type
plt.figure(figsize=(6,4))
sns.boxplot(x="Precip Type", y="Temperature (C)", data=data)
plt.title("Temperature Distribution by Precip Type")
plt.show()
```

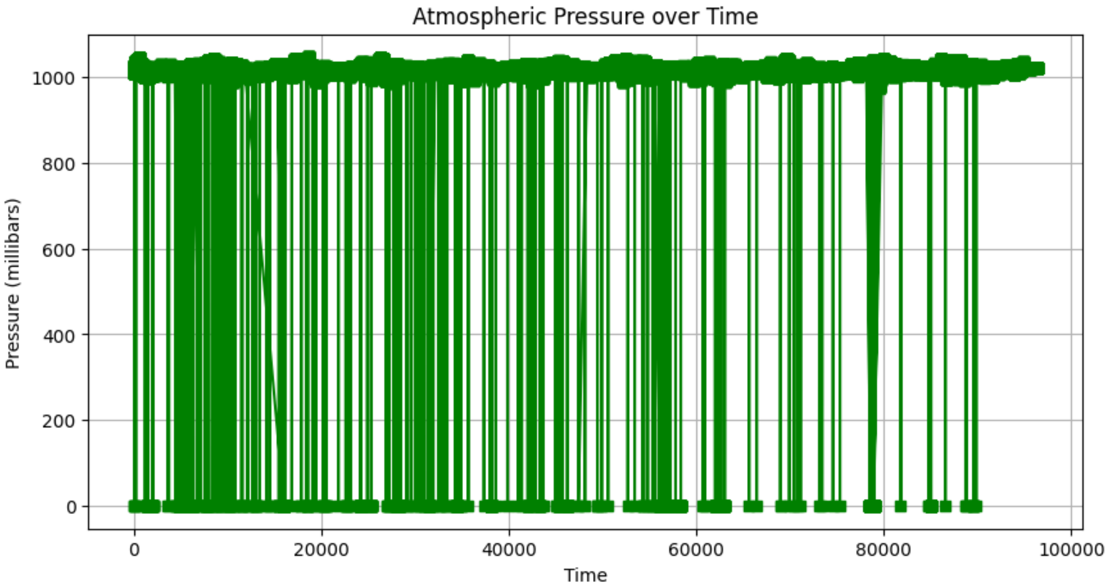


[20]:

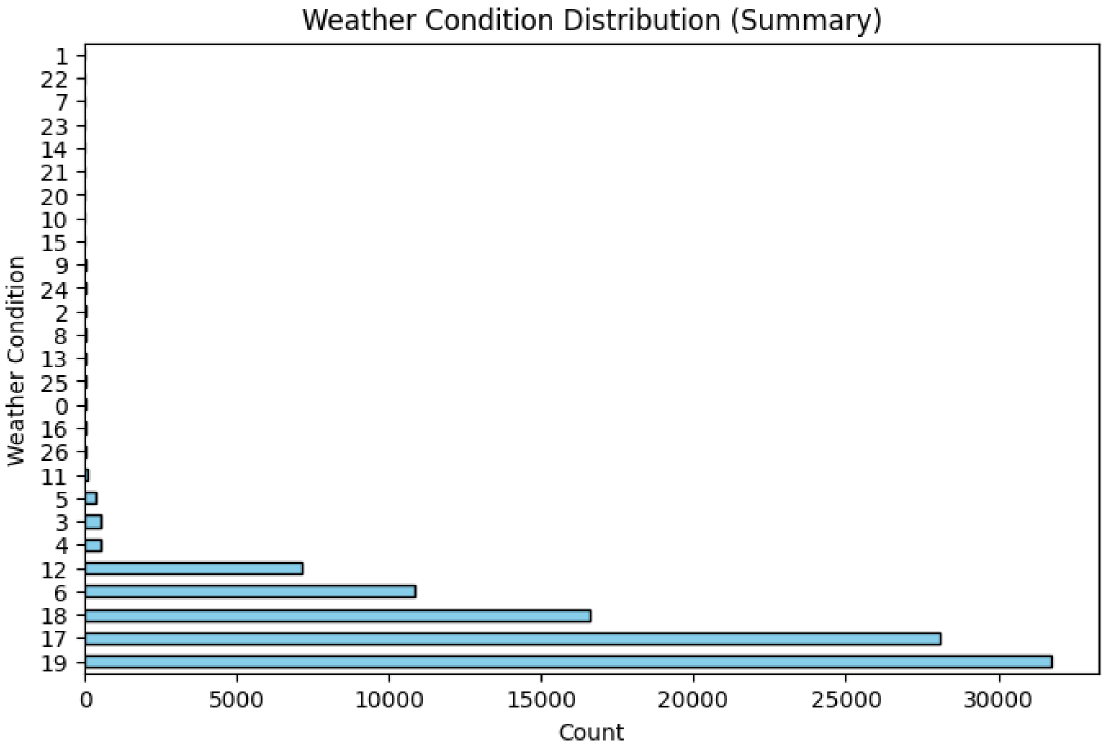
```
# 8) Scatter plot: Wind Speed vs Temperature
plt.figure(figsize=(8, 5))
plt.scatter(data["Wind Speed (km/h)"], data["Temperature (C)"], c="red", s=70)
plt.xlabel("Wind Speed (km/h)")
plt.ylabel("Temperature (C)")
plt.title("Wind Speed vs Temperature")
plt.grid(True)
plt.show()
```



[21]:
```python
# 9) Line chart: Pressure over Time
plt.figure(figsize=(10, 5))
plt.plot(data["Formatted Date"], data["Pressure (millibars)"], marker="s",␣
↪color="green")
plt.xlabel("Time")
plt.ylabel("Pressure (millibars)")
plt.title("Atmospheric Pressure over Time")
plt.grid(True)
plt.show()
```



[22]:
```
# 5. Pie chart: Weather summary distribution
plt.figure(figsize=(8, 5))
data["Summary"].value_counts().plot.barh(color="skyblue", edgecolor="black")
plt.xlabel("Count")
plt.ylabel("Weather Condition")
plt.title("Weather Condition Distribution (Summary)")
plt.show()
```



[23]:
```
# STEP 2: Install required tools: LaTeX + Pandoc
!apt-get install -y texlive-xetex texlive-fonts-recommended␣
 ↪texlive-plain-generic > /dev/null
!apt-get install -y pandoc > /dev/null  # ← This is the missing piece

# STEP 3: Set your notebook path
file_path = "/content/drive/MyDrive/TY_DMW/Assignment_01_DMW"

# STEP 4: Convert the notebook to PDF and output to /content
!jupyter nbconvert --to pdf "{file_path}" --output-dir="/content"
```

```
Extracting templates from packages: 100%
[NbConvertApp] WARNING | pattern
'/content/drive/MyDrive/TY_DMW/Assignment_01_DMW.ipynb' matched no files
This application is used to convert notebook files (*.ipynb)
        to various other formats.

        WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
=======
The options below are convenience aliases to configurable class-options,
as listed in the "Equivalent to" description-line of the aliases.
```

```
To see all configurable class-options for some <cmd>, use:
    <cmd> --help-all

--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
--show-config
    Show the application's configuration (human-readable format)
    Equivalent to: [--Application.show_config=True]
--show-config-json
    Show the application's configuration (json format)
    Equivalent to: [--Application.show_config_json=True]
--generate-config
    generate default config file
    Equivalent to: [--JupyterApp.generate_config=True]
-y
    Answer yes to any questions instead of prompting.
    Equivalent to: [--JupyterApp.answer_yes=True]
--execute
    Execute the notebook prior to export.
    Equivalent to: [--ExecutePreprocessor.enabled=True]
--allow-errors
    Continue notebook execution even if one of the cells throws an error and
include the error message in the cell output (the default behaviour is to abort
conversion). This flag is only relevant if '--execute' was specified, too.
    Equivalent to: [--ExecutePreprocessor.allow_errors=True]
--stdin
    read a single notebook file from stdin. Write the resulting notebook with
default basename 'notebook.*'
    Equivalent to: [--NbConvertApp.from_stdin=True]
--stdout
    Write notebook output to stdout instead of files.
    Equivalent to: [--NbConvertApp.writer_class=StdoutWriter]
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
            relevant when converting to notebook format)
    Equivalent to: [--NbConvertApp.use_output_suffix=False
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=]
--clear-output
    Clear output of current file and save in place,
            overwriting the existing notebook.
    Equivalent to: [--NbConvertApp.use_output_suffix=False
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=
--ClearOutputPreprocessor.enabled=True]
--coalesce-streams
    Coalesce consecutive stdout and stderr outputs into one stream (within each
cell).
    Equivalent to: [--NbConvertApp.use_output_suffix=False
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=
--CoalesceStreamsPreprocessor.enabled=True]
--no-prompt
    Exclude input and output prompts from converted document.
    Equivalent to: [--TemplateExporter.exclude_input_prompt=True
--TemplateExporter.exclude_output_prompt=True]
--no-input
    Exclude input cells and output prompts from converted document.
            This mode is ideal for generating code-free reports.
    Equivalent to: [--TemplateExporter.exclude_output_prompt=True
--TemplateExporter.exclude_input=True
--TemplateExporter.exclude_input_prompt=True]
--allow-chromium-download
    Whether to allow downloading chromium if no suitable version is found on the
system.
    Equivalent to: [--WebPDFExporter.allow_chromium_download=True]
--disable-chromium-sandbox
    Disable chromium security sandbox when converting to PDF..
    Equivalent to: [--WebPDFExporter.disable_sandbox=True]
--show-input
    Shows code input. This flag is only useful for dejavu users.
    Equivalent to: [--TemplateExporter.exclude_input=False]
--embed-images
    Embed the images as base64 dataurls in the output. This flag is only useful
for the HTML/WebPDF/Slides exports.
    Equivalent to: [--HTMLExporter.embed_images=True]
--sanitize-html
    Whether the HTML in Markdown cells and cell outputs should be sanitized..
    Equivalent to: [--HTMLExporter.sanitize_html=True]
--log-level=<Enum>
    Set the log level by value or name.
    Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR',
'CRITICAL']
    Default: 30
    Equivalent to: [--Application.log_level]
--config=<Unicode>
    Full path of a config file.
    Default: ''
    Equivalent to: [--JupyterApp.config_file]
--to=<Unicode>
    The export format to be used, either one of the built-in formats
            ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook',
'pdf', 'python', 'qtpdf', 'qtpng', 'rst', 'script', 'slides', 'webpdf']
            or a dotted object name that represents the import path for an
            ``Exporter`` class
    Default: ''
    Equivalent to: [--NbConvertApp.export_format]
--template=<Unicode>
```

```
    Name of the template to use
    Default: ''
    Equivalent to: [--TemplateExporter.template_name]
--template-file=<Unicode>
    Name of the template file to use
    Default: None
    Equivalent to: [--TemplateExporter.template_file]
--theme=<Unicode>
    Template specific theme(e.g. the name of a JupyterLab CSS theme distributed
    as prebuilt extension for the lab template)
    Default: 'light'
    Equivalent to: [--HTMLExporter.theme]
--sanitize_html=<Bool>
    Whether the HTML in Markdown cells and cell outputs should be sanitized.This
    should be set to True by nbviewer or similar tools.
    Default: False
    Equivalent to: [--HTMLExporter.sanitize_html]
--writer=<DottedObjectName>
    Writer class used to write the
                                        results of the conversion
    Default: 'FilesWriter'
    Equivalent to: [--NbConvertApp.writer_class]
--post=<DottedOrNone>
    PostProcessor class used to write the
                                        results of the conversion
    Default: ''
    Equivalent to: [--NbConvertApp.postprocessor_class]
--output=<Unicode>
    Overwrite base name use for output files.
                Supports pattern replacements '{notebook_name}'.
    Default: '{notebook_name}'
    Equivalent to: [--NbConvertApp.output_base]
--output-dir=<Unicode>
    Directory to write output(s) to. Defaults
                                  to output to the directory of each notebook.
To recover
                                  previous default behaviour (outputting to the
current
                                  working directory) use . as the flag value.
    Default: ''
    Equivalent to: [--FilesWriter.build_directory]
--reveal-prefix=<Unicode>
    The URL prefix for reveal.js (version 3.x).
            This defaults to the reveal CDN, but can be any url pointing to a
copy
            of reveal.js.
            For speaker notes to work, this must be a relative path to a local
            copy of reveal.js: e.g., "reveal.js".
            If a relative path is given, it must be a subdirectory of the
            current directory (from which the server is run).
            See the usage documentation
            (https://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js-
html-slideshow)
            for more details.
    Default: ''
    Equivalent to: [--SlidesExporter.reveal_url_prefix]
--nbformat=<Enum>
    The nbformat version to write.
            Use this to downgrade notebooks.
    Choices: any of [1, 2, 3, 4]
    Default: 4
    Equivalent to: [--NotebookExporter.nbformat_version]

Examples
--------

    The simplest way to use nbconvert is

            > jupyter nbconvert mynotebook.ipynb --to html

            Options include ['asciidoc', 'custom', 'html', 'latex', 'markdown',
'notebook', 'pdf', 'python', 'qtpdf', 'qtpng', 'rst', 'script', 'slides',
'webpdf'].

            > jupyter nbconvert --to latex mynotebook.ipynb

            Both HTML and LaTeX support multiple output templates. LaTeX
includes
            'base', 'article' and 'report'.  HTML includes 'basic', 'lab' and
            'classic'. You can specify the flavor of the format used.

            > jupyter nbconvert --to html --template lab mynotebook.ipynb

            You can also pipe the output to stdout, rather than a file

            > jupyter nbconvert mynotebook.ipynb --stdout

            PDF is generated via latex

            > jupyter nbconvert mynotebook.ipynb --to pdf

            You can get (and serve) a Reveal.js-powered slideshow

            > jupyter nbconvert myslides.ipynb --to slides --post serve

            Multiple notebooks can be given at the command line in a couple of
            different ways:

            > jupyter nbconvert notebook*.ipynb
            > jupyter nbconvert notebook1.ipynb notebook2.ipynb

            or you can specify the notebooks list in a config file, containing::

                c.NbConvertApp.notebooks = ["my_notebook.ipynb"]

            > jupyter nbconvert --config mycfg.py

To see all available configurables, use `--help-all`.
```

[24]:

```
from google.colab import files
import os

# Get the output file name
file_name = os.path.basename(file_path)
pdf_name = file_name.replace(".ipynb", ".pdf")

# Download from /content
files.download(f"/content/{pdf_name}")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
/tmp/ipython-input-2808709972.py in <cell line: 0>()
      7
      8 # Download from /content
----> 9 files.download(f"/content/{pdf_name}")

/usr/local/lib/python3.11/dist-packages/google/colab/files.py in␣
 ↪download(filename)
    231   if not _os.path.exists(filename):
    232     msg = 'Cannot find file: {}'.format(filename)
--> 233     raise FileNotFoundError(msg)  # pylint: disable=undefined-variable
    234
    235   comm_manager = _IPython.get_ipython().kernel.comm_manager

FileNotFoundError: Cannot find file: /content/Assignment_01_DMW.pdf
```