Here's the code analysis and potential viva questions with answers:

---

### \*\*Code Explanation\*\*

1. \*\*Loading Data\*\*

```python

df = pd.read\_csv(r"C:\Users\sumit\OneDrive\Desktop\sample\_weather.txt")

```

- Loads weather data from a text file

- Assumes columns like DATE, TEMP, DEWP (dew point), WDSP (wind speed)

2. \*\*Column Cleaning\*\*

```python

df.columns = df.columns.str.strip()

```

- Removes whitespace from column names (prevents errors in column referencing)

3. \*\*Date Conversion\*\*

```python

df['DATE'] = pd.to\_datetime(df['DATE'])

```

- Converts DATE column to datetime format for time-based operations

4. \*\*Feature Engineering\*\*

```python

df['Year'] = df['DATE'].dt.year

df['Month'] = df['DATE'].dt.month

```

- Extracts year/month from dates for temporal analysis

5. \*\*Aggregation\*\*

```python

year\_avg = df.groupby('Year')[['TEMP','DEWP','WDSP']].mean()

month\_avg = df.groupby('Month')[['TEMP','DEWP','WDSP']].mean()

```

- Calculates annual/monthly averages for key weather metrics

---

### \*\*Viva Questions & Answers\*\*

\*\*Q1: Why use `str.strip()` on column names?\*\*

A1: To handle inconsistent formatting like:

- " TEMP " → "TEMP" (removes leading/trailing spaces)

- Prevents errors when accessing columns like `df[' TEMP ']`

\*\*Q2: What does `pd.to\_datetime()` achieve?\*\*

A2: Converts string/object dates to datetime format, enabling:

- Date arithmetic

- Extraction of year/month/day

- Time-series analysis

\*\*Q3: Why create Year/Month columns?\*\*

A3: To analyze:

- Annual temperature trends (global warming?)

- Seasonal patterns (monthly averages)

- Compare weather across years/months

\*\*Q4: What does `groupby().mean()` do?\*\*

A4: Splits data into groups (years/months) → calculates averages for each group

\*\*Q5: How would you handle missing values?\*\*

A5: Add before aggregation:

```python

df = df.dropna() # Remove rows with missing values

# or

df.fillna(method='ffill', inplace=True) # Forward-fill

```

\*\*Q6: How to calculate median instead of mean?\*\*

A6: Replace `mean()` with `median()`:

```python

year\_median = df.groupby('Year')[['TEMP','DEWP','WDSP']].median()

```

\*\*Q7: What other aggregations are possible?\*\*

A7: Use:

- `sum()`: Total precipitation

- `max()/min()`: Record temperatures

- `std()`: Variability in wind speed

\*\*Q8: How to visualize these results?\*\*

A8: Use Matplotlib/Seaborn:

```python

year\_avg['TEMP'].plot(title='Annual Temperature Trends')

plt.ylabel('Temperature (°C)')

plt.show()

```

\*\*Q9: Why use pandas for this analysis?\*\*

A9: Pandas provides:

- Efficient time-series handling

- Easy grouping/aggregation

- Clean data manipulation syntax

\*\*Q10: What errors might occur here?\*\*

A10: Common issues:

- Date format mismatches

- Missing columns (e.g., no 'WDSP' in data)

- Non-numeric values in TEMP/DEWP/WDSP

---

### \*\*Key Insights\*\*

1. \*\*Yearly Analysis\*\*:

- Track long-term climate patterns

- Identify hottest/coolest years

2. \*\*Monthly Analysis\*\*:

- Reveal seasonal trends (e.g., warmer Julys)

- Compare monsoon/winter wind speeds

3. \*\*Data Quality\*\*:

- Ensure consistent datetime formatting

- Handle missing values appropriately

---

### \*\*Syllabus Connections\*\*

1. \*\*Data Preprocessing\*\*: Cleaning column names, handling dates

2. \*\*Time-Series Analysis\*\*: Year/month extraction

3. \*\*Aggregation\*\*: GroupBy operations

4. \*\*Statistical Analysis\*\*: Mean/median calculations

Would you like me to explain any part in more detail? 😊