Preprocessing Steps (ILO 5):

1. Imported Necessary Libraries
   - Imported essential libraries for data handling (pandas, numpy), visualization (matplotlib, seaborn, missingno), database interaction (psycopg2), and machine learning (sklearn, scipy).

2. Defined Database Connection Parameters
   - Set up parameters required for connecting to the PostgreSQL database, including the host, port, database name, username, and password.

3. Created Database Connection Function
   - Developed a function get_connection() to establish and return a new database connection using the defined parameters to facilitate database operations.

4. Loaded Data from Database
   - Implemented a function load_data() that:
     - Connected to the PostgreSQL database using psycopg2.
     - Executed a SQL query to retrieve all data from the data_lake.safe_driving table.
     - Loaded the retrieved data into a pandas DataFrame.
     - Closed the database connection.

5. Checked for Duplicates
   - Created a function check_duplicates(df) that:
     - Identified duplicate rows in the DataFrame using the duplicated() method.
     - Printed the number of duplicate rows found.

6. Reported Missing Values
   - Developed a function report_missing_values(df) that:
     - Calculated the number of missing values in each column using isnull().sum().
     - Printed a summary of missing values by column.

7. Removed Unnecessary Columns
    - Defined a function remove_unnecessary_columns(df) to clean the DataFrame by:
        - Specifying a list of columns to be removed.
        - Dropping these columns from the DataFrame if they exist.
        - Returning the cleaned DataFrame.

8. Visualized Missing Values
    - Utilized missingno.matrix(df) to create a visual representation of missing values in the DataFrame, helping to identify patterns or concentrations of missing data.

9. Created Box Plots for Outlier Detection
    - Used seaborn to generate box plots for numerical columns (speed_kmh, end_speed_kmh, duration_seconds, maxwaarde). These plots helped in visually detecting outliers based on their distribution and spread.

10. Removed Outliers
    - Calculated z-scores for the numerical columns to quantify the distance of each data point from the mean in terms of standard deviations.
    - Removed rows where any z-score exceeded ±3, as these were considered outliers.

11. Executed SQL Script for Weather Classification
    - Connected to the database and executed a SQL script that:
        - Dropped existing tables with weather data if they existed.
        - Created new tables to store aggregated weather data (wind speed, precipitation).
        - Created a final table integrating weather conditions with event data from safe_driving.

12. Created Views for Weather Data
    - Executed SQL statements to create database views for:
        - Temperature data.
        - Precipitation data.

        - Integrated weather conditions.

13. Fetched Data from View
    - Retrieved data from the vw_weather_conditions view:
        - Executed a SQL query to fetch all records.
        - Loaded the results into a pandas DataFrame for further analysis.

14. Visualized Weather Condition Distribution
    - Plotted the distribution of weather conditions using:
        - A bar plot to show the frequency of each condition.
        - A seaborn count plot to visualize the count of each weather condition.

15. Separated Feature and Target Variables
    - Separated the features (X) from the target variable (y):
        - Dropped the weather_condition column from the DataFrame to use as the target variable.
        - Encoded categorical features using one-hot encoding to prepare them for machine learning algorithms.

16. Split Data into Training and Test Sets
    - Used train_test_split to divide the data into training and test sets:
        - Allocated 80% of the data for training and 20% for testing.
        - Ensured the split was random but reproducible by setting a random seed.

17. Scaled Features
    - Applied RobustScaler to numerical features (wind_speed_kmph, precipitation_secs):
        - Checked for variance in the features.
        - Scaled features to reduce the impact of outliers and improve model performance.

18. Plotted Histogram for Scaled Features
    - Created histograms for scaled features to visualize their distribution post-scaling:
        - Used matplotlib to generate histograms.

        - Assessed the effectiveness of scaling by examining the spread and central tendency.

19. Balanced Classes
    - Balanced the dataset by resampling:
        - Identified the class with the minimum number of samples.
        - Resampled other classes to match the sample size of the smallest class, ensuring equal representation.

20. Applied Log Transformation for Skewed Features
    - Applied a log transformation to wind_speed_kmph to reduce skewness:
        - Shifted the data to ensure all values were positive.
        - Used log1p for the transformation.
        - Plotted the transformed data to verify the reduction in skewness.

21. Calculated and Displayed Skewness
    - Calculated and printed the skewness of the log-transformed feature to confirm the improvement towards a normal distribution.

Conclusion:

The steps described above involved data loading, cleaning, transformation, and preparation for machine learning. Each step was essential for ensuring the data was ready for building accurate and reliable predictive models.