```
BEGIN

  FUNCTION task_A()
    FUNCTION validate_date_input()
      WHILE True
        TRY
          PROMPT user for day (DD)
          IF DD < 1 OR DD > 31 THEN
            PRINT "Out of range - values must be in the range 1 and 31"
            CONTINUE
          ENDIF

          PROMPT user for month (MM)
          IF MM < 1 OR MM > 12 THEN
            PRINT "Out of range - values must be in the range 1 and 12"
            CONTINUE
          ENDIF

          PROMPT user for year (YYYY)
          IF YYYY < 2000 OR YYYY > 2024 THEN
            PRINT "Out of range - values must be in the range 2000 and 2024"
            CONTINUE
          ENDIF

          RETURN (DD, MM, YYYY)
        CATCH ValueError
          PRINT "Integer required"
        ENDTRY
      ENDWHILE
```

```
END FUNCTION

(DD, MM, YYYY) = validate_date_input()

FUNCTION is_leap_year(YYYY)
   RETURN (YYYY MOD 4 == 0)
END FUNCTION

IF (DD, MM, YYYY) are valid THEN
   IF MM in [1, 3, 5, 7, 8, 10, 12] THEN
      max_days = 31
   ELSE IF MM in [4, 6, 9, 11] THEN
      max_days = 30
   ELSE IF MM == 2 THEN
      max_days = 29 IF is_leap_year(YYYY) ELSE 28
   ELSE
      max_days = 0
   ENDIF

   IF DD is in range 1 to max_days THEN
      date = format DD/MM/YYYY
      PRINT "Valid date entered, date is: date"
   ELSE
      PRINT "Invalid data"
   ENDIF
ENDIF

file_name = "traffic_data" + format DD, MM, YYYY
RETURN file_name
```

```
END FUNCTION


FUNCTION validate_continue_input()
    WHILE True
        PROMPT user for input (Y or N)
        IF user_input == "Y" THEN
            PRINT "Loading new dataset..."
            RETURN "y"
        ELSE IF user_input == "N" THEN
            PRINT "End of run"
            RETURN "n"
        ELSE
            PRINT "Invalid input, please enter Y or N"
        ENDIF
    ENDWHILE
END FUNCTION


FUNCTION process_csv_data(file_path)
    TRY
        OPEN file at file_path
        READ data
        SKIP header row
        INITIALIZE counters to 0


        TRY
            total_vehicles = COUNT rows in data
            total_trucks = COUNT rows where vehicle type is "Truck"
            total_electric_vehicles = COUNT rows where electric vehicle is "True"
            total_two_wheeled = COUNT rows where vehicle type is in ["Bicycle", "Motorcycle", "Scooter"]
```

total_busses_leaving_Elm_Avenue_Rabbit_Road_junction_heading_north = COUNT rows where conditions match

total_vehicles_passing_through_both_junctions_without_turning_left_or_right = COUNT rows where conditions match

percentage_of_all_vehicles_recorded_that_are_Trucks = (total_trucks / total_vehicles) * 100

average_of_Bicycles_per_hour = total_bicycles / 24

total_vehicles_recorded_over_the_speed_limit = COUNT rows where speed limit conditions match

total_vehicles_through_only_Elm_Avenue_Rabbit_Road_junction = COUNT rows where junction matches

total_vehicles_through_only_Hanley_Highway_Westway_junction = COUNT rows where junction matches

percentage_of_vehicles_through_Elm_Avenue_Rabbit_Road_Scooters = (scooters_count / total_vehicles_through_only_Elm_Avenue_Rabbit_Road_junction) * 100

CALCULATE rain hours and total_hours_of_rain

CALCULATE peak hour vehicles and time_of_the_peak_traffic_hour_on_Hanley_Highway_Westway

RETURN outcomes as a list of results

```
    CATCH (ValueError, IndexError)
      PRINT "Skipping invalid row Error"
    ENDTRY
  CATCH FileNotFoundError
    PRINT "File does not exist."
  ENDTRY
END FUNCTION

FUNCTION display_outcomes(outcomes)
  FOR each outcome in outcomes
    PRINT outcome
```

```
    ENDFOR
END FUNCTION

FUNCTION save_results_to_file(outcomes, file_name="results.txt ```plaintext
FUNCTION save_results_to_file(outcomes, file_name="results.txt")
    TRY
        OPEN file at file_name in append mode
        FOR each outcome in outcomes
            WRITE outcome to file
        ENDFOR
        WRITE a newline to file
        PRINT "Results successfully saved to file_name"
    CATCH IOError
        PRINT "Error writing to file file_name"
    ENDTRY
END FUNCTION

WHILE True
    file_path = task_A()
    outcomes = process_csv_data(file_path)
    IF outcomes are not empty THEN
        display_outcomes(outcomes)
        save_results_to_file(outcomes)
    ENDIF

    status = validate_continue_input()
    IF status == "n" THEN
        BREAK
    ENDIF
```

```
ENDWHILE


CLASS HistogramApp
    FUNCTION __init__(self, traffic_data, date)
        SET self.traffic_data = traffic_data
        SET self.date = date
        INITIALIZE GUI window
    END FUNCTION

    FUNCTION draw_histogram()
        INITIALIZE hourly_data structure

        FOR each row in traffic_data
            EXTRACT junction and time_of_day
            INCREMENT count for the respective junction and hour
        ENDFOR

        CALCULATE max_count and scale_factor
        DRAW x-axis and y-axis

        FOR each hour from 0 to 23
            DRAW bars for Elm Avenue/Rabbit Road and Hanley Highway/Westway
            ADD labels for each hour
        ENDFOR

        ADD axis labels and histogram title
    END FUNCTION

    FUNCTION run()
```

```
        CALL draw_histogram()
        START GUI main loop
    END FUNCTION
END CLASS

CLASS MultiCSVProcessor
    FUNCTION __init__(self)
        SET self.current_data to None
    END FUNCTION

    FUNCTION load_csv_file(file_path)
        TRY
            OPEN file at file_path
            READ lines and skip header
            SET self.current_data to the read lines
            RETURN True
        CATCH FileNotFoundError
            PRINT "Error: File not found."
            RETURN False
        ENDTRY
    END FUNCTION

    FUNCTION process_data_and_display_histogram(date)
        IF self.current_data is not None THEN
            CREATE HistogramApp instance
            CALL run() method
        ENDIF
    END FUNCTION
```

```
    FUNCTION handle_user_interaction()
        WHILE True
            PROMPT user for date or 'N' to quit
            IF user_input is "N" THEN
                PRINT "Exiting program."
                BREAK
            ENDIF

            file_path = construct file path using date
            IF load_csv_file(file_path) THEN
                process_data_and_display_histogram(date)
            ELSE
                PRINT "Failed to process the file. Please try again."
            ENDIF
        ENDWHILE
    END FUNCTION
END CLASS

IF __name__ == "__main__" THEN
    CREATE MultiCSVProcessor instance
    CALL handle_user_interaction()
ENDIF

END
```