```
# Benjamin Goldberg
# 3501-4910


#Data Loading

import csv

def load_inquiries_data(filename):
# Loads inquiry records from the CSV file into a list of dicts.
# Each row is analyzed with csv.DictReader. This makes a list where every element
has a column/ string value
# Args filename (str | os.PathLike): Path to the CSV file encoded as UTF-8.
# Returns: list[dict[str, str]]: All rows from the CSV in order.
    data = []
    with open(filename, 'r', encoding='utf-8') as file:
            reader = csv.DictReader(file)
            for row in reader:
                data.append (row)
    return data



def print_data_summary(data):
# prints a brief summary of the loaded dataset
# prints the total number of records, number of columns, and the column names
inferred from the first row
# args: data(list[dict[str, str]]): Dataset as returned by `load_inquiries_data`.
# This is a void function so it returns nothing

     total = len(data)
     columns = list(data[0].keys())

     print(f"Total records: {total}")
     print(f"Number of columns: {len(columns)}")
     print(f"Column names: {', '.join(columns)}")
     return


def count_inquiries_by_country(data, country):
#Counts rows where the customer's country matches the target.
#The comparison is an exact, case-sensitive string equality check on the
'customer_country' field.
# Args: data (list[dict[str, str]]): Dataset of inquiry rows. country (str):
Country name/code to match exactly.
# Returns int: Numebr of matching rows

    count = 0
    for row in data:
        if row ['customer_country'] == country:
            count += 1
    return count


def get_budget_statistics(data):
# Compute simple statistics over the buget_value field
# converts each present 'budget value' to a float and computes the total, max, min,
average, and count. If not budget is found it prints a message and returns none
# Args data (list[dict[str, str]]): Dataset of inquiry rows
```

```
# Returns  dict[str, float] | None: A dictionary with keys {'total', 'max', 'min',
'average', 'count'} if data exists; otherwise None.

    costs = [float(row['budget_value']) for row in data if row.get('budget_value',
0)]
    if not costs:
        print("No budget data found.")
        return

    total_budget = sum(costs)
    highest_cost = max(costs)
    lowest_cost = min(costs)
    average_cost = total_budget / len(costs)
    budget_count = len(costs)


    return {
        'total': total_budget,
        'max': highest_cost,
        'min': lowest_cost,
        'average': average_cost,
        'count' : budget_count
    }


def find_popular_travel_months(data):
# Finds most common values on the 'travel_month' field.
# builds frequency table of travel months and returns it sorted by decending count.
# Args data (list[dict[str, str]]): Dataset of inquiry rows.
# Returns list[tuple[str, int]]: List of (month, count) pairs sorted from most to
least frequent.
     travel_month = []
     for row in data:
        travel_month.append(row['travel_month'])

     month_counts = {}
     for month in travel_month:
         if month in month_counts:
             month_counts[month] += 1
         else:
             month_counts[month] = 1

     sorted_months = sorted(month_counts.items(), key=lambda x: x[1], reverse=True)
     return sorted_months

def print_completion_report(data):
# Count TRUE/FALSE values in the 'completed' field.
# Interprets each row's 'completed' value and tallies exact string matches of
'TRUE' and everything else as 'FALSE'.
# Args: data (list[dict[str, str]]): Dataset of inquiry rows.
# Returns: dict[str, int]: A dictionary with keys {'TRUE', 'FALSE'} and counts.
     completion_report = []
     for row in data:
      completion_report.append(row['completed'])

      true_count = 0
      false_count = 0
     completion_count = {}
     for completion in completion_report:
```

```
        if completion == 'TRUE':
            true_count += 1
        else:
            false_count += 1
     return {'TRUE' : true_count, 'FALSE' : false_count}



def categorize_group_sizes(data):
# Categorize inquiries by total group size (adults + children).
# counts the total number of each group size
# args: data (list[dict[str, str | int]]): Dataset where each row may include
num_adults' and 'num_children' (parseable as ints).
# Returns: dict[str, int]: Counts per category keyed by {'solo', 'couple',
'family', 'large_group', 'unknown'}.


    group_categories = {
        'solo' : 0,
        'couple' : 0,
        'family' : 0,
        'large_group' : 0,
        'unknown' : 0
    }

    total_grouplist = []
    for row in data:
        adults = int(row.get('num_adults', 0))
        children = int(row.get('num_children', 0))
        total_group = adults + children
        total_grouplist.append(total_group)


    for total_group in total_grouplist:
     if total_group == 1:
      group_categories['solo'] +=1
     elif total_group == 2:
      group_categories['couple'] +=1
     elif 3 <= total_group <= 5:
      group_categories['family'] +=1
     elif total_group >= 6:
      group_categories['large_group'] += 1
     else:
      group_categories['unknown'] += 1
    return group_categories
```