U.S. Medical Insurance Costs

In this project, I will investigate a csv file with medical insurance costs using Python fundamentals. Through this I will analyze various attributes within insurance.csv to learn more about patient information in this file and gain insight into potential use cases for this dataset.

```
# import csv library
import csv
```

To start, I import any libraries I need. For this project, the only library I need is the csv library to work with insurance.csv data. Considering this, I can move on with just the 'csv' library.

Next, I decide to look through the insurance.csv and understand the data. Things I look for are the following:

- The names of columns and rows
- Any noticeable missing data
- Types of values (numerical vs. categorical)

Understanding this will allow me to make any adjustments that are needed as well as form a plan on how to import the data into my Python file.

```
# I create empty lists for various attributes in insurance.csv
ages = []
sexs = []
bmis = []
children = []
smokers = []
regions = []
charges = []
```

In insurance.csv, I saw the following columns:

- Age
- Sex
- BMI
- Number of Children
- Smoking Status
- Region
- Yearly Cost

I also noted that I did not see any missing data. With this information, I decided to make seven empty lists that would hold each individual column of data.

```
# Open insurance in a context manager
with open("insurance.csv", newline = '') as insurance_docs:
```

```
    # Write data from csv into reader variable
    reader = csv.DictReader(insurance_docs)
    # Iterate through the data and format variables into appropriate
data types
    for row in reader:
        ages.append(int(row['age']))
        sexs.append(row['sex'])
        bmis.append(float(row['bmi']))
        children.append(int(row['children']))
        smokers.append(row['smoker'])
        regions.append(row['region'])
        charges.append(float(row['charges']))
```

In this portion, I appended the data into the appriporate lists. I also decided to change the data types into the types I needed during initialization rather than when I would be computing results to save time and readability.

Now that I have all the data into labeled lists, I am ready for analysis. After reviewing the dataset, I found several interesting aspects to investigate:

- Average Age
- Average Pay
- Average Pay (Male vs Female)
- Total Pay Per Age Group
- Total Pay Per Region
- Quantity Of Male Vs Female
- Number of Children by Gender
- Smokers Per Region

To perform these inspections, I built a class called `Analysis` with contains eight methods:

- `average_age()`
- `average_charge()`
- `male_female_charge_average()`
- `total_children_by_sex()`
- `total_pay_per_region()`
- `male_female_counter()`
- `smokers_per_region()`
- `total_pay_per_age_group()`

Class Structure and Initialization

The `Analysis` class uses five dictionaries that are initialized and populated in the constructor (`__init__`). These dictionaries store aggregated data necessary for the analysis:

- sex_count - Contains the amount of males and females in the data.

- total_charges_per_sex - Contains the total amount of charges seperated by gender.
- total_children_per_sex - Contains the amount of children each gender has.
- total_charges_per_region - Contains the total amount of money seperated by region.
- total_smokers_per_region - Contains the total smokers seperated by region.

Methods

- `average_age()` - Calculates and returns the average age of all patients by summing up the ages and dividing by the number of patients
- `average_charge()` - Calculates and returns the average yearly charges by summing up the charges and dividing by the number of patients.
- `male_female_charge_average()` - Returns a list with the average charges for males and females. It uses `total_charges_per_sex` and `sex_count` dictionaries to compute this.
- `total_children_by_sex()` - Returns a list with the total number of children for males and females, using the `total_children_per_sex` dictionary.
- `total_pay_per_region()` - Returns a dictionary where each region maps to its total charges, rounded to two decimal places. This is computed using the `total_charges_per_region` dictionary
- `male_female_counter()` - Returns the `sex_count` dictionary, showing the number of males and females
- `smokers_per_region()` - Returns the `total_smokers_per_region` dictionary, which contains the count of smokers for each region.
- `total_pay_per_age_group()` - Returns a dictionary tht groups total charges by age ranges (e.g. 19-, 20-29, 30-39, etc.), rounded to two decimal places. This uses the `total_charges_per_age_group` dictionary

By leveraging these methods and dictionaries, the `Analysis class provides a comprehensive view of the data, enabling detailed insight into various aspects of patient information and their insurance charges.

```
class Analysis:
    def __init__(self, ages, sexs, bmis, children, smokers, regions,
charges):
        # Initalize attributes with data lists
        self.ages = ages
        self.sexs = sexs
        self.bmis = bmis
        self.children = children
        self.smokers = smokers
        self.regions = regions
        self.charges = charges

        # Initialize dictionaries to aggregate data
        self.sex_count = {'male': 0, 'female': 0}
        self.total_charges_per_sex = {'male': 0.0, 'female': 0.0}
        self.total_children_per_sex = {'male': 0, 'female': 0}
```

```
        self.total_charges_per_region = {'southwest': 0.0,
'southeast': 0.0, 'northwest': 0.0, 'northeast': 0.0}
        self.total_smokers_per_region = {'southwest': 0, 'southeast':
0, 'northwest': 0, 'northeast': 0}

        # Populate dictionaries with data
        for sex, charge, child, smoker, region in zip(sexs, charges,
children, smokers, regions):
            # Update the count of males and females
            self.sex_count[sex] += 1
            # Update the total charges for the gender
            self.total_charges_per_sex[sex] += charge
            # Updata the total number of children for the gender
            self.total_children_per_sex[sex] += child
            # Updata the total charges for the region
            self.total_charges_per_region[region] += charge
            # If patient is a smoker, increment the smoker count for
the region
            if smoker == 'yes':
                self.total_smokers_per_region[region] += 1

    def average_age(self):
        # Calculate average age of all patients, rounded to two
decimal places
        return round(sum(self.ages) / len(self.ages), 2)

    def average_charge(self):
        # Calculate average charge of all patients, rounded to two
decimal places
        return round(sum(self.charges) / len(self.charges), 2)

    def male_female_charges_average(self):
        # Calculate and return average charges for males and females
        return [
            round(self.total_charges_per_sex['male'] /
self.sex_count['male'], 2),
            round(self.total_charges_per_sex['female'] /
self.sex_count['female'], 2)
        ]

    def total_children_by_sex(self):
        # Return total amount of children for males and females
        return [ self.total_children_per_sex['male'],
self.total_children_per_sex['female']]

    def total_pay_per_region(self):
        # Iterate through total_charges_per_region and returns total
pay rounded in two decimals
        return {region: round(amount, 2) for region, amount in
```

```
self.total_charges_per_region.items()}

    def male_female_counter(self):
        # Returns a dictionary with counts of males and females
        return self.sex_count

    def smokers_per_region(self):
        # Returns a dictionary with counts of smokers per region
        return self.total_smokers_per_region

    def total_pay_per_age_group(self):
        # Initializes a dictionary to group total charges by age
ranges
        age_groups = {'19-': 0, '20-29': 0, '30-39': 0, '40-49': 0,
'50-59': 0, '60+': 0}
        # Iterates over ages and accumulates them into age_groups
dictionary
        for age, charge in zip(self.ages, self.charges):
            if age < 20: age_groups['19-'] += charge
            elif age < 30: age_groups['20-29'] += charge
            elif age < 40: age_groups['30-39'] += charge
            elif age < 50: age_groups['40-49'] += charge
            elif age < 60: age_groups['50-59'] += charge
            else: age_groups['60+'] += charge
        # Returns dictionary with age groups and total charges rounded
to two decimals
        return {group: round(amount, 2) for group, amount in
age_groups.items()}
```

I then created an instance of class `Analysis` with the data that I attained from insurance.csv. Creating this class also calls the constructor which auto inserts data into the analysis variable.

```
analysis = Analysis(ages, sexs, bmis, children, smokers, regions,
charges)
```

Results

```
print("Average Age:", analysis.average_age())
print("Average Charge:", analysis.average_charge())
print("Sex Distribution:", analysis.male_female_counter())
print("Total Children by Sex:", analysis.total_children_by_sex())
```

```
Average Age: 39.21
Average Charge: 13270.42
Sex Distribution: {'male': 676, 'female': 662}
Total Children by Sex: [754, 711]
```

These functions were mainly made to estimate and verify that there was a wide variety of stastics chosen and they came out to a fair outcome.

Considering our ages range from 18-64, 39 is espected.

Considering our charges range from $1121 to $63770, a charge of 13000 seems low however the quantity of charges around that value justify it.

Since the sex distribution and children distribution are fairly close, we can consider this dataset to be fair.

```
print("Average Charges by Sex:",
analysis.male_female_charges_average())
print("Charges by Age Group:", analysis.total_pay_per_age_group())
```

```
Average Charges by Sex: [13956.75, 12569.58]
Charges by Age Group: {'19-': 1151806.85, '20-29': 2677290.29, '30-
39': 3016867.52, '40-49': 4017377.79, '50-59': 4470208.05, '60+':
2422274.49}
```

From viewing these results we can make two claims: Males pay, on average, $1380 more than females and the majority of income is coming from people of the age 40-59.

```
print("Smokers by Region:", analysis.smokers_per_region())
print("Total Charges by Region:", analysis.total_pay_per_region())
```

```
Smokers by Region: {'southwest': 58, 'southeast': 91, 'northwest': 58,
'northeast': 67}
Total Charges by Region: {'southwest': 4012754.65, 'southeast':
5363689.76, 'northwest': 4035712.0, 'northeast': 4343668.58}
```

From comparing these results, we find that all three regions (southwest, northwest, and northeast) that have nearly identical smoker counts, have around the same pay range while southeast which has nearly 25 more smokers has a higher income by ~ 1.2M in costs.