Covid Data Report:

1)Loading data:

When it comes to analyzing data, always the first step is to load our data set and gain information about its characteristics and features.

So typically each data frame has a set of features, and each of these features have a set of possible values. But since the size of almost all of datasets is very large we can't just get all of the columns and possible values for each of them by just observing the data frame. Here, is where come libraries like pandas come and somehow pave the way for us.

So after loading our data, we can use df.columns to get the list of all of the features that exist in our dataset.

```
df.columns
```

```
Index(['iso_code', 'continent', 'location', 'date', 'total_cases', 'new_cases',
       'new_cases_smoothed', 'total_deaths', 'new_deaths',
       'new_deaths_smoothed', 'total_cases_per_million',
       'new_cases_per_million', 'new_cases_smoothed_per_million',
       'total_deaths_per_million', 'new_deaths_per_million',
       'new_deaths_smoothed_per_million', 'reproduction_rate', 'icu_patients',
       'icu_patients_per_million', 'hosp_patients',
       'hosp_patients_per_million', 'weekly_icu_admissions',
       'weekly_icu_admissions_per_million', 'weekly_hosp_admissions',
       'weekly_hosp_admissions_per_million', 'total_tests', 'new_tests',
       'total_tests_per_thousand', 'new_tests_per_thousand',
       'new_tests_smoothed', 'new_tests_smoothed_per_thousand',
       'positive_rate', 'tests_per_case', 'tests_units', 'total_vaccinations',
       'people_vaccinated', 'people_fully_vaccinated', 'total_boosters',
       'new_vaccinations', 'new_vaccinations_smoothed',
       'total_vaccinations_per_hundred', 'people_vaccinated_per_hundred',
       'people_fully_vaccinated_per_hundred', 'total_boosters_per_hundred',
       'new_vaccinations_smoothed_per_million',
       'new_people_vaccinated_smoothed',
       'new_people_vaccinated_smoothed_per_hundred', 'stringency_index',
       'population_density', 'median_age', 'aged_65_older', 'aged_70_older',
       'gdp_per_capita', 'extreme_poverty', 'cardiovasc_death_rate',
       'diabetes_prevalence', 'female_smokers', 'male_smokers',
       'handwashing_facilities', 'hospital_beds_per_thousand',
       'life_expectancy', 'human_development_index', 'population',
       'excess_mortality_cumulative_absolute', 'excess_mortality_cumulative',
       'excess_mortality', 'excess_mortality_cumulative_per_million'],
      dtype='object')
```

so these are the features we have to deal with.

I also got to know the other characteristics of the dataset using other functions.

2) Handling Null Values:

a) Dropping Columns

Passing by null values in datasets is something to expect, especially when the size of the dataset is very large. But sometimes datasets do contain some columns that most their values are filled with null, in this case these columns do not have any useful information for us and therefore we can just drop them so they don't get in our way.

Our covid dataset also is not an exception, and it has some columns with more than 80 percentage of null values.

b) Filling in null values of numerical columns:

There are many methods we can use for filling in the null values of numerical columns. For instance, we can use the mean of all of non-null values to fill in null values, or we can find the median of all values and use that. Due to the fact that outliers may exist in our data, I preferred not to use the mean because of the huge effect that outliers can have on mean. This method will certainly lead to other problems and the analyze of our data would probably be ruined because the values we're putting instead of the null ones are not the right ones.

So, here we use the median method and use the function that we defined to fill the null values. It's a good choice to always check if everything has been done perfectly after the process.

c) Filling in the null values of categorical columns:

This step is a little bit different from the previous one. Before filling in the null values of categorical columns, we have to figure out if any of these categorical columns are related to each other or not (based on their concept and what each feature actually is). In our dataset "iso_code" and "continent" are related to each other, so we can't use the typical method like filling in the null values with the most frequent value of each column. Let's say hypothetically the most frequent continent is Asia and we use this value to fill in all of the null places in our continent column. By doing this we're somehow messing up the data frame, because the continent of some "iso_code"s will be set as Asia(It was null before) while it's not the iso_code does not belong to the continent Asia actually.

So what should we do?

The best solution that came to my mind was to create a mapping using a dictionary. This will be a mapping from iso_code's unique values to continents and later we can use this specific mapping to fill in the null values. So, I got this mapping but then realized that the value of the continent of some iso_codes has been null in every occurrence. Meaning that we actually never had a continent for these iso_codes so even creating a mapping for these two columns wouldn't solve

our problem. Here, I created a mapping on my own to use for filling in the null values.

```
mapping = {
    'OWID_AFR': 'Africa',
    'OWID_ASI': 'Asia',
    'OWID_EUR': 'Europe',
    'OWID_EUN': 'European Union',
    'OWID_HIC': 'High-income countries',
    'OWID_LIC': 'Low-income countries',
    'OWID_LMC': 'Lower-middle-income countries',
    'OWID_NAM': 'North America',
    'OWID_OCE': 'Oceania',
    'OWID_SAM': 'South America',
    'OWID_UMC': 'Upper-middle-income countries',
    'OWID_WRL': 'World'
}
```

3) Removing Outliers:

Outliers are data points that significantly deviate from the rest of the dataset. They may result from measurement errors, data entry mistakes, or rare events.

a) Numerical Outliers:

Many algorithms have been developed for finding the outliers of numerical data, and each for each of them a logical threshold should be set in order to get a good result.

For this dataset I used the IQR (Interquartile Range). We can conclude from the result we got, that removing outliers in this dataset may not be the easiest thing to do. The main problem is that the scale of our numerical features

are very different compared to each other, and as a result of that we can't really set an appropriate threshold for all of them. Even if we scale our columns before applying the algorithm on them, we still won't get a good result because this changes the distribution of our data. If we wanted to train a model on this dataset, scaling (and normalization) would have been a good choice but in this case we want to perform EDA on our data frame it would just screw everything up.

b) Categorical:

When it comes to categorical outliers, we must identify them using a different approach. If the frequency of a category and other categories of a feature is highly different, we can consider that category an outlier. For example, if a category's frequency is 3 while others are at least 80, this one is identified as the outlier.

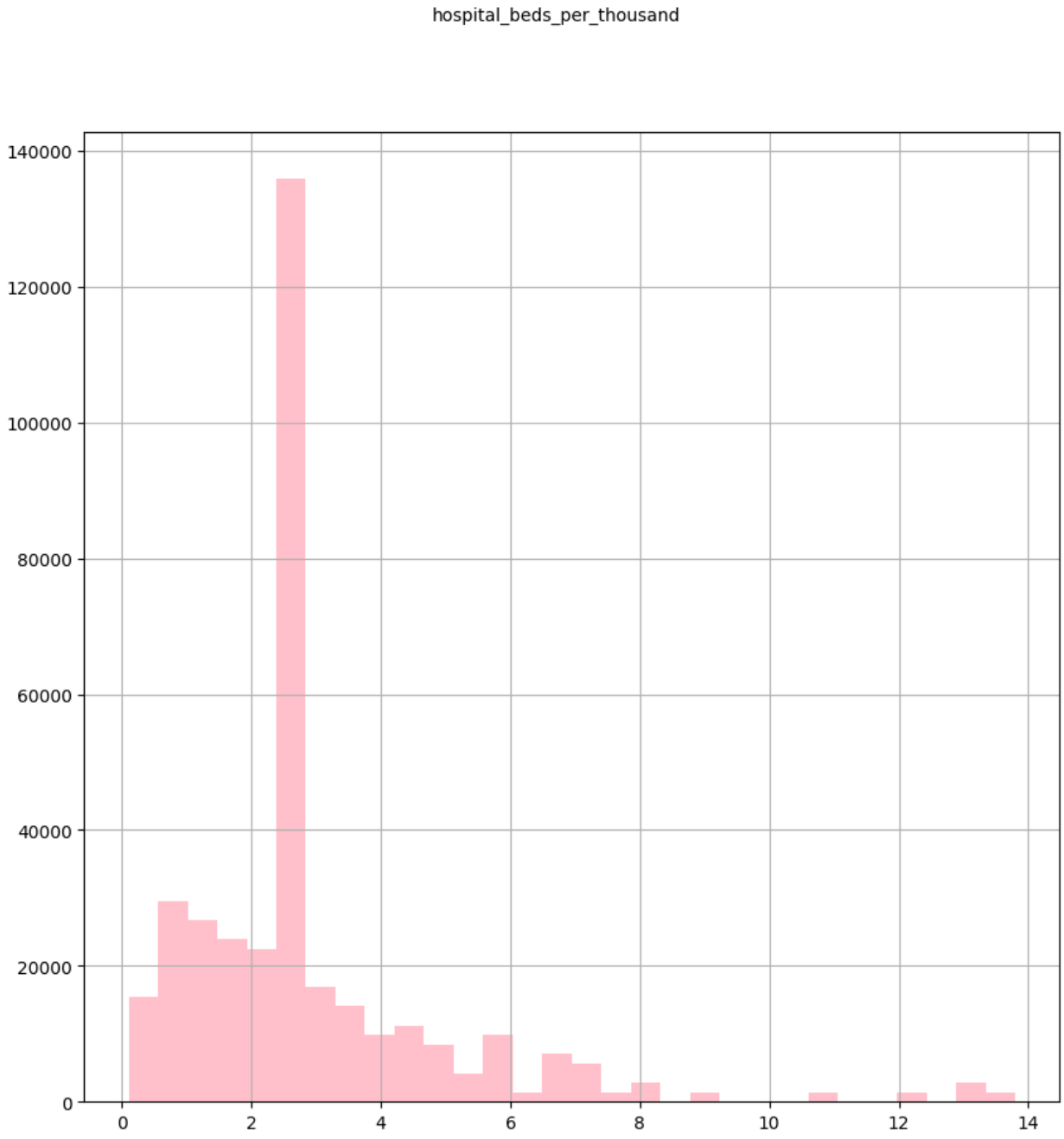
After removing the categorical outliers only 5 rows are eliminated from our data frame.

4) Visualization:

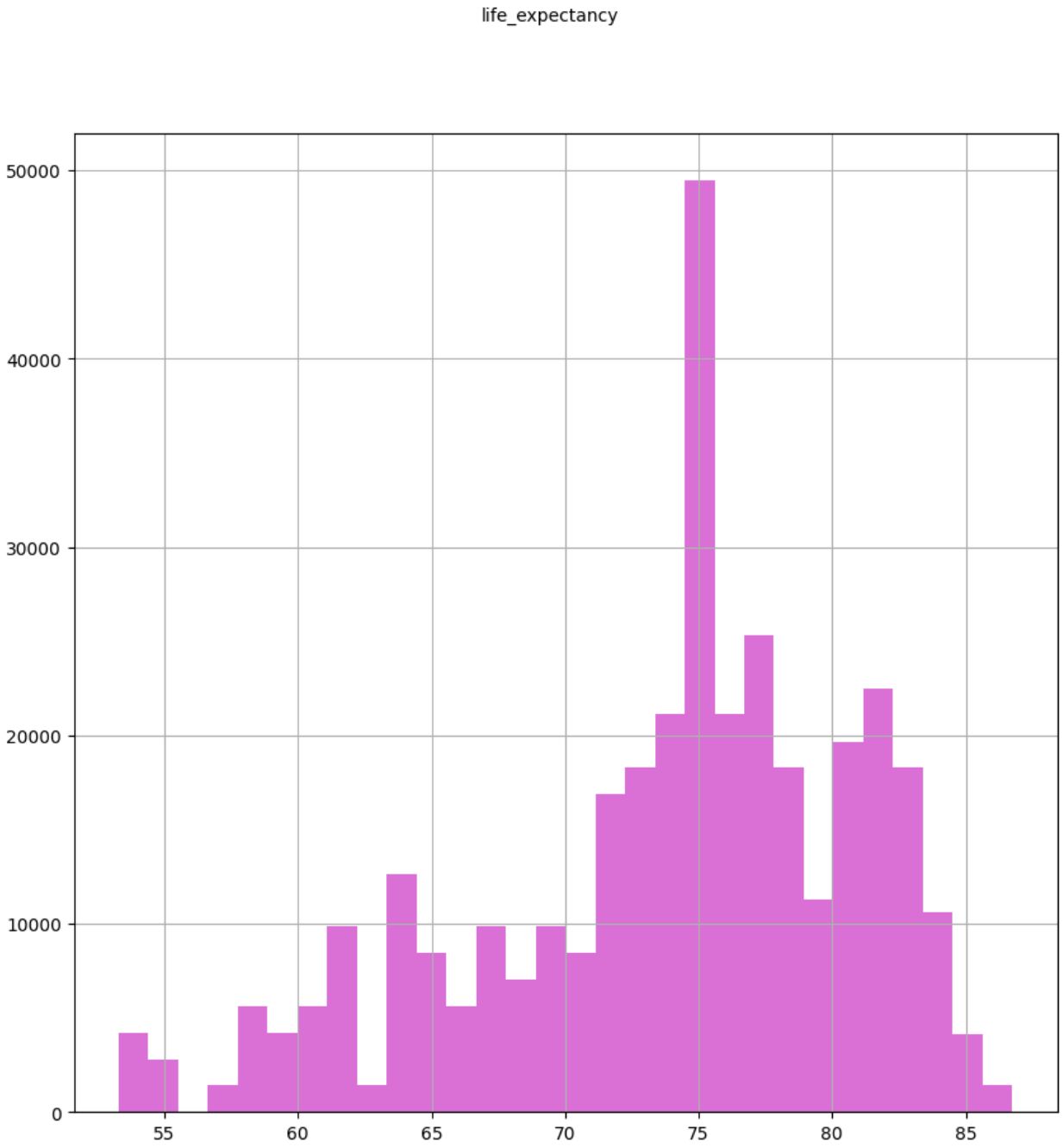
a) Numerical: First, we go for the numerical features.

A histogram is a graphical representation that displays the distribution of a numeric variable. Histograms can show Frequency Distribution, Shape of the Distribution, Central Tendency, Spread and Variability and Outliers.

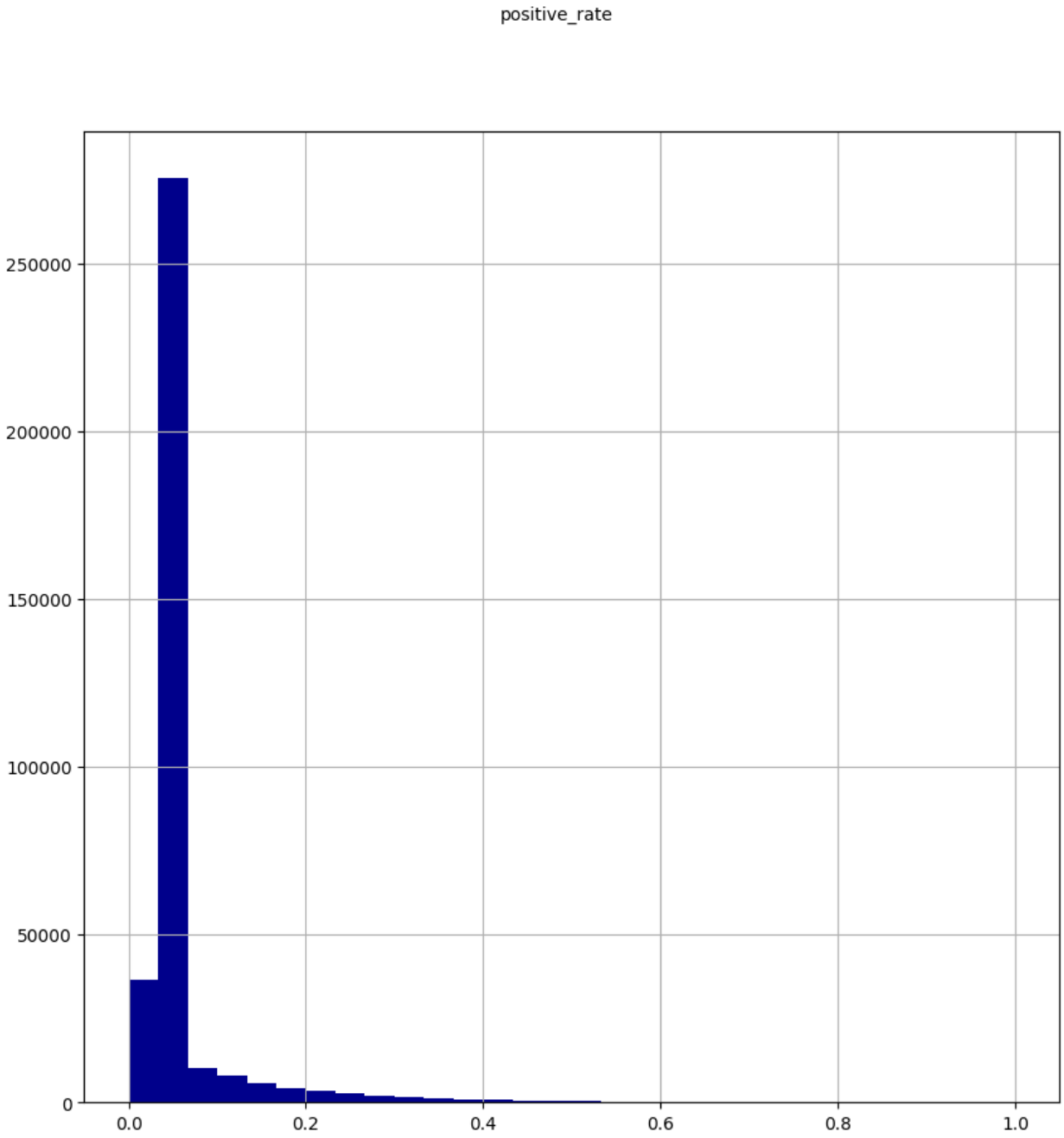
I plotted the histogram using pandas for all of the numeric features.
Here are some of them:



Clearly the distribution of this feature isn't normal. Also the mode of the values is easy to recognize.

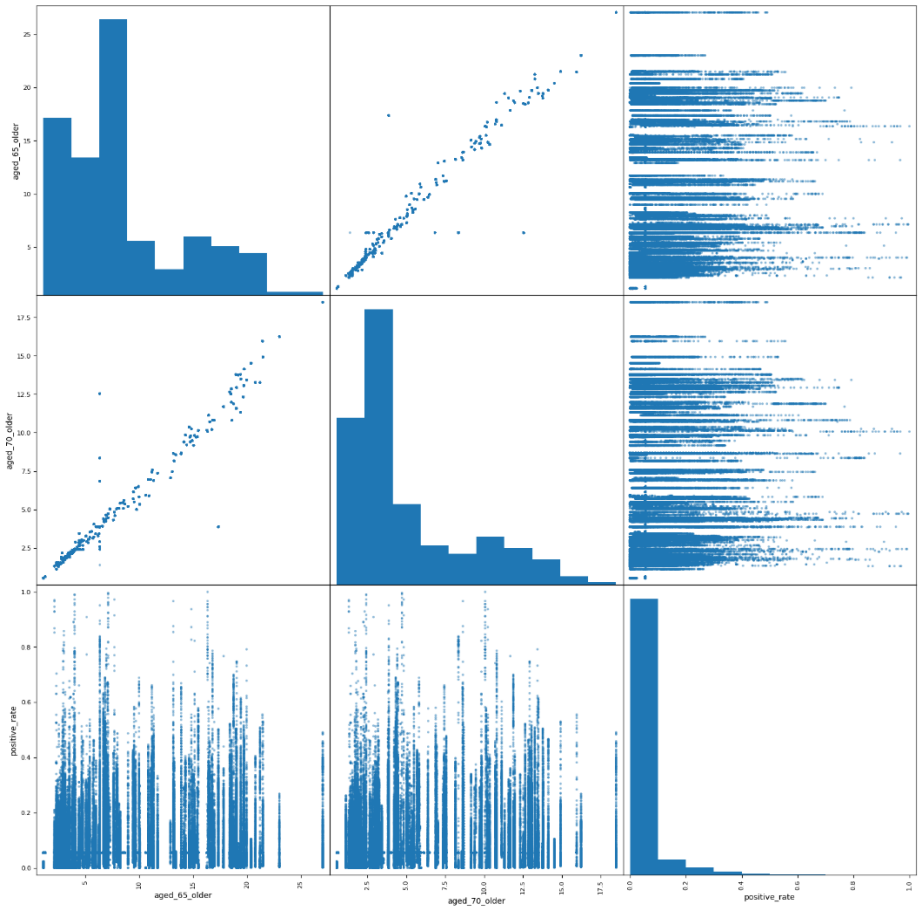


This is the distribution and shape of data for life expectancy. This one isn't normal either.



The distribution of this column of our dataset is skewed, meaning that it has longer tail on one side. In this case, the histogram is right skewed (also known as positive skewed): In this column the mean is greater than the median.

A scatter plot is a visual representation that uses dots to show the relationship between two different numeric variables. Each dot on the plot corresponds to an individual data point, with its position determined by the values of the two variables. By plotting a scatter plot we can get info about correlation and patterns of our data. For instance, this is the scatter plot for columns: {aged_older_70, aged_older_60, positive_rate}



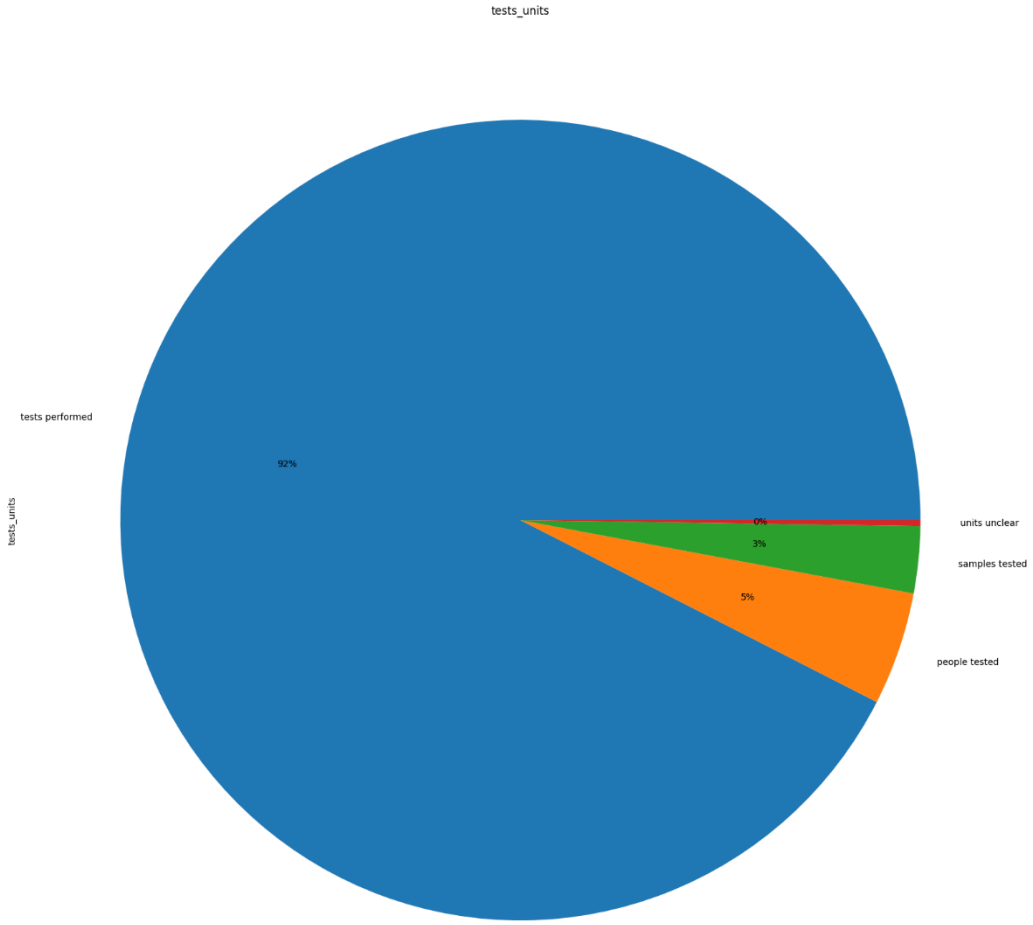
This plot shows us how aged_older_70 and aged_older_65 have a linear relation and how these two are correlated with positive rate.

b) Categorical

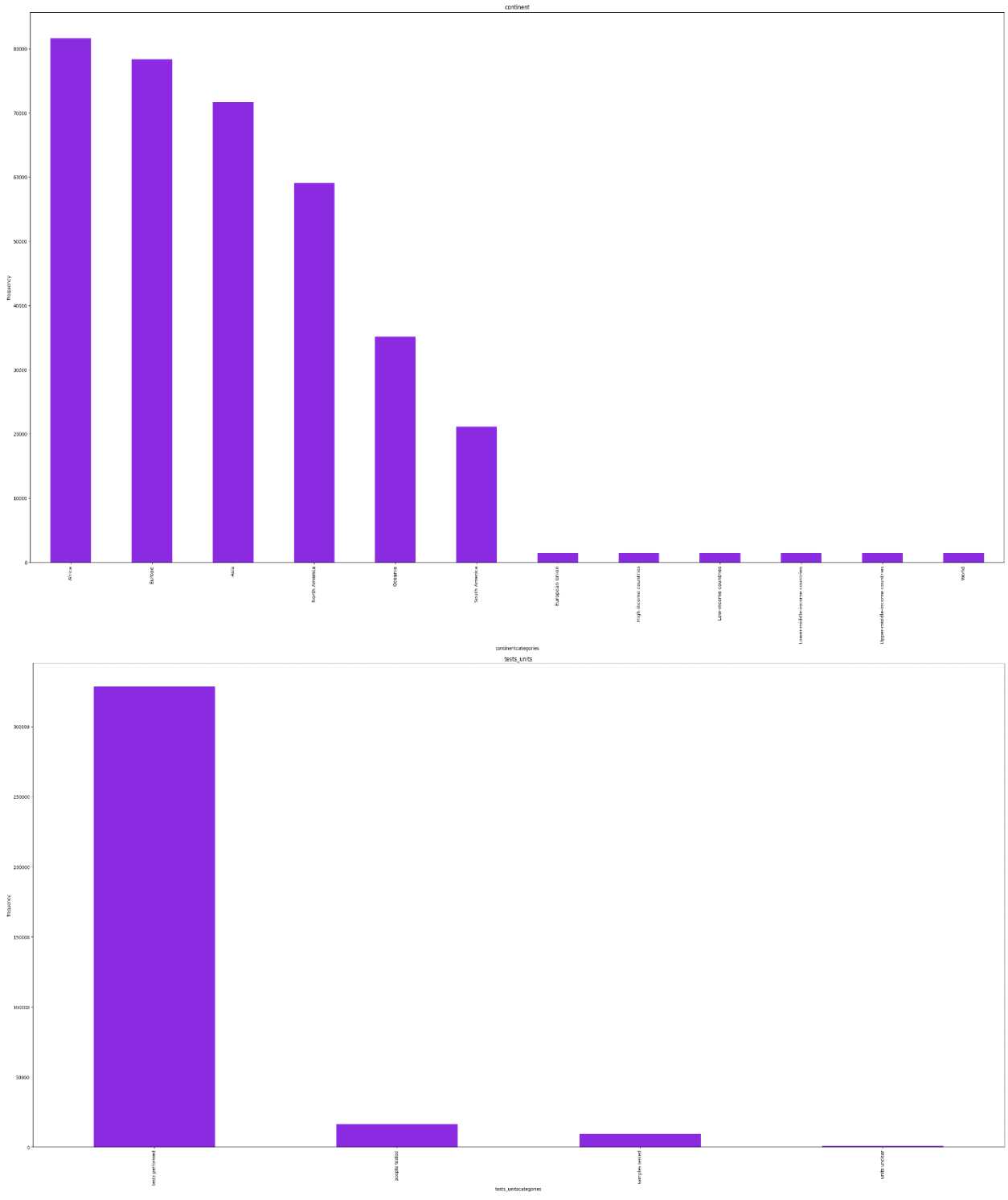
For categorical features we can plot bar charts or pie chart:

A pie chart is a circular graph that visually represents how a total amount is divided into different parts.

Using pie chart, we can get knowledge about: Contributions and Proportional Data



This pie chart shows us the contributions of the categories of test_units (Which is one of our categorical columns).



These bat charts show the frequency of each category, as you can see the distribution of data points between

different categories is not equal. In the second chart, the first category has a high frequency. In the other one, the first few categories have high frequency while the others have lower frequencies.

5)Distribution:

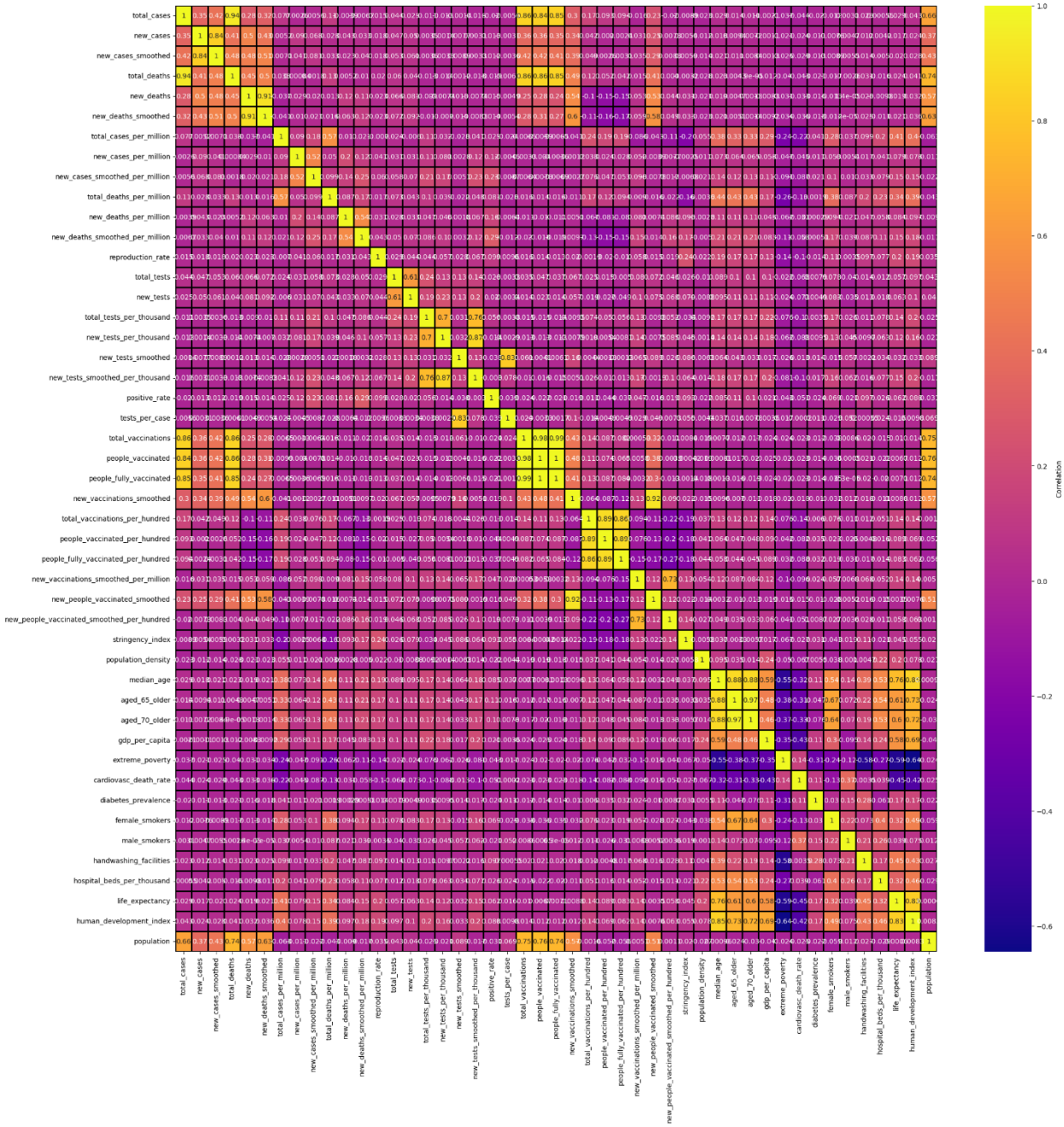
There are some tests that we can use to figure out how data is distributed. Here, we use the Shapiro-Wilk normality test (The Shapiro–Wilk test is a statistical test used to assess whether a given sample of data follows a normal distribution) for all of our numerical columns. The result was that none of our features have normal distribution.

```
[ ]  for col in numerical_columns:
       result = shapiro(dist_df[col])
       if(result.pvalue<0.05):
         print("Not Normal")
       else:
         print("Normal")
```

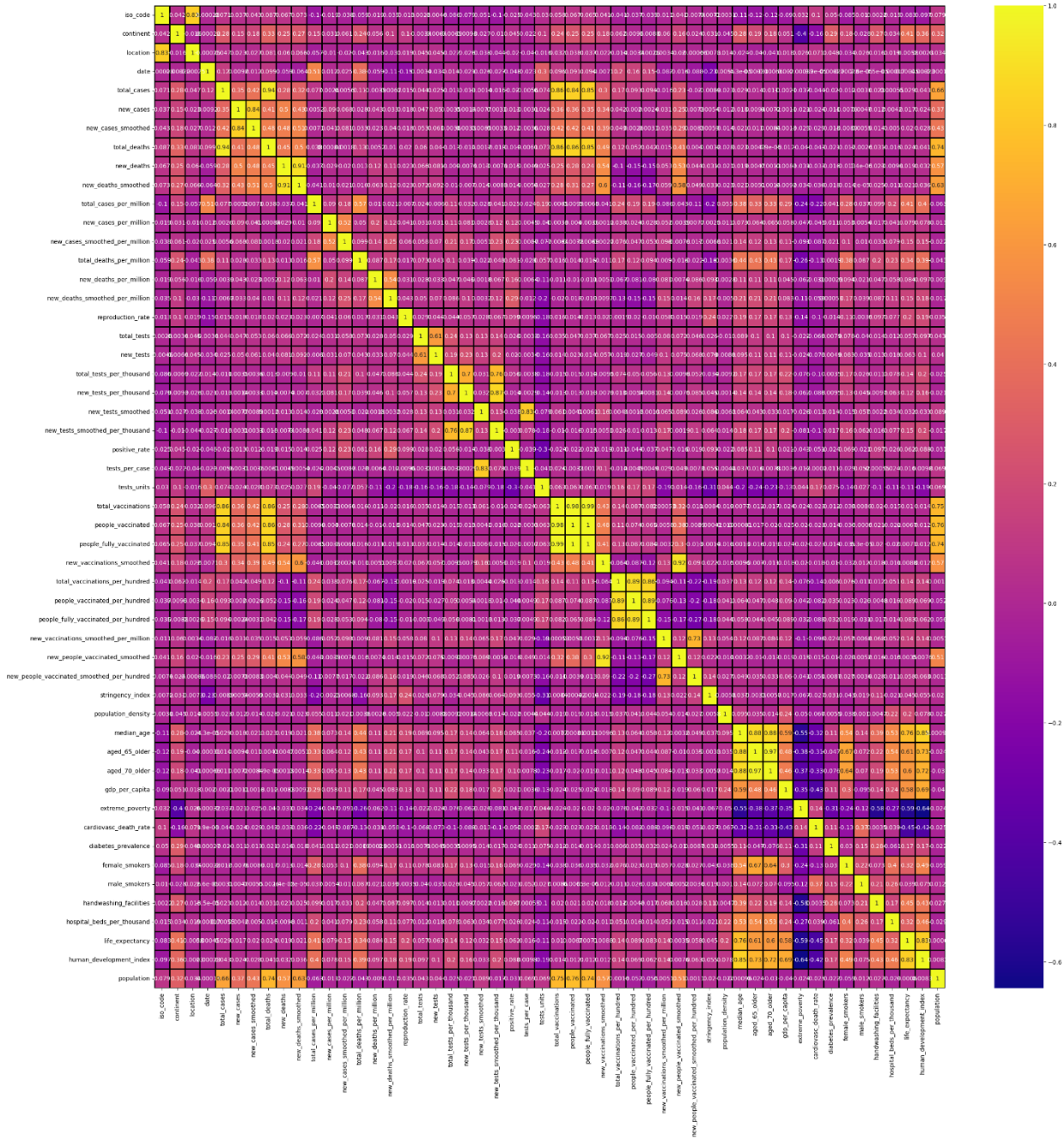
```
/usr/local/lib/python3.10/dist-pacl
  warnings.warn("p-value may not be
Not Normal
Not Normal
Not Normal
Not Normal
Not Normal
Not Normal
Not Normal
Not Normal
Not Normal
Not Normal
Not Normal
Not Normal
Not Normal
Not Normal
Not Normal
Not Normal
Not Normal
Not Normal
Not Normal
Not Normal
Not Normal
Not Normal
Not Normal
Not Normal
Not Normal
Not Normal
```

6) Correlation: Correlation refers to the statistical relationship between two entities. In other words, it's how two variables move in relation to one another.

a) visualization: First, we can plot heat maps to illustrate the correlation between features.



This is the heat map for all of our numerical columns. The number and color of each cell show how strong the relationship between two features are. By looking at the heat map we can easily realize how deeply some features are correlated like: median_age and life_expectancy
And how some pairs are poorly correlated: new_cases and handwashing_facilities.



This is the heat map for all of the columns of our dataset. We labeled the categorical columns, so we'd be able to include them while plotting the heat map.

c) Correlation test:

The Pearson correlation coefficient is a widely used statistical measure that quantifies the linear relationship between two quantitative variables.

I defined a function for applying Pearson's test on a pair of columns, then applied the function on each possible pair of numerical columns using a for loop.

The End