

## **Uncovering Job Market Insights Using Spark**

- MD HINOY RAHMAN [mdhinoy.rahman2@mail.dcu.ie](mailto:mdhinoy.rahman2@mail.dcu.ie) - 15966
- DHANUSH ARAVIND ARUN PRAKASH  
[ghanusharavind.arunprakash2@mail.dcu.ie](mailto:ghanusharavind.arunprakash2@mail.dcu.ie) - 13031
- SHAHUL HAMEED ALI KADER BATCHA  
MOHAMED [shahulhameed.alikaderbatchamohamed2@mail.dcu.ie](mailto:shahulhameed.alikaderbatchamohamed2@mail.dcu.ie) - 15385

### **I. Link for Dataset**

#### **Cleaned Dataset:**

[https://drive.google.com/drive/folders/1DNLuGTWUOBFw-7kN-AFvRAJ71yUXbBl0?usp=drive\\_link](https://drive.google.com/drive/folders/1DNLuGTWUOBFw-7kN-AFvRAJ71yUXbBl0?usp=drive_link)

#### **Uncleaned Dataset:**

[https://drive.google.com/drive/folders/1evZt72YX57UwU8rNm5wTnFcTErgF7\\_BM?usp=drive\\_link](https://drive.google.com/drive/folders/1evZt72YX57UwU8rNm5wTnFcTErgF7_BM?usp=drive_link)

#### **Dataset after Processing:**

[https://drive.google.com/drive/folders/1\\_9oCC7eHpg\\_dNkZi-zjX6HdSz-Pw5SKH?usp=drive\\_link](https://drive.google.com/drive/folders/1_9oCC7eHpg_dNkZi-zjX6HdSz-Pw5SKH?usp=drive_link)

### **II. Visualization WebApp Link**

#### **PC/Laptop:**

[https://public.tableau.com/views/Visualization\\_Cloud\\_Tech/Sheet1?:language=en-US&publish=yes&:sid=&:redirect=auth&:display\\_count=n&:origin=viz\\_share\\_link](https://public.tableau.com/views/Visualization_Cloud_Tech/Sheet1?:language=en-US&publish=yes&:sid=&:redirect=auth&:display_count=n&:origin=viz_share_link)

#### **Mobile:**

[https://public.tableau.com/views/Visualization\\_Cloud\\_Tech\\_Mobile/Dashboard1?:language=en-US&publish=yes&:sid=&:redirect=auth&:display\\_count=n&:origin=viz\\_share\\_link](https://public.tableau.com/views/Visualization_Cloud_Tech_Mobile/Dashboard1?:language=en-US&publish=yes&:sid=&:redirect=auth&:display_count=n&:origin=viz_share_link)

### **III. Link for Source Repository**

<https://github.com/md-hinoy-rahman/Cloud-Technology-Assignment.git>

### **IV. Link for Video Demonstration**

[https://drive.google.com/file/d/1W\\_VCcL-7AkFm\\_5x1ENZPdqZlHO9wqLZ/view?usp=drive\\_link](https://drive.google.com/file/d/1W_VCcL-7AkFm_5x1ENZPdqZlHO9wqLZ/view?usp=drive_link)

### **V. Introduction and Motivation for The App and The Reason for Choosing PySpark**

We set out to create an application using PySpark to harness the power of cloud technology and visualize large datasets. Our motivation stemmed from the immense capabilities of Spark in big data analysis, thanks to its in-memory operations and resilient distributed dataset (RDD) structure, which are optimized for fast processing.

The availability of a Python API in Spark significantly boosts collaboration and integration with other Python-based projects. It opens a wide range of possibilities for utilizing various Python libraries, which simplifies the development process and enhances the functionality of the application. This integration is crucial for collaborative efforts, as it allows developers to work within a familiar environment and utilize the extensive ecosystem of Python libraries for data manipulation, visualization, and machine learning.

Given these advantages, we decided to develop a visualization app to provide crucial insights through graphical representations. We aimed to demonstrate how effective visualizations can be in revealing patterns and correlations within the data. For instance, we wanted to visualize the correlation between two specific columns—mean annual salary and cost of living index—to understand whether there is an influence of one on the other. By plotting these variables, we could intuitively see if higher living costs are associated with higher earnings, thereby providing valuable insights to users. This approach not only makes complex data more accessible but also aids in better decision-making by highlighting important trends and relationships.

Our project began with the acquisition of a large dataset, which we meticulously cleaned using Google Colab and Pandas. This cleaned dataset was then used to extract essential information, including country, continent, job title, mean annual salary, and cost of living index from the existing columns. We grouped the data by country to calculate the average

mean annual salary and the average cost of living index, subsequently sorting the dataset by the average cost of living index.

An interactive scatter plot was then generated, showcasing the relationship between the cost of living index and the average mean annual salary. This visualization allows users to compare multiple countries and explore whether higher living costs correlate with higher earnings. Through this app, we aim to make complex data more accessible and informative for our users.

## **VI. Related works and Comparison with Our System**

The following are the related systems that we have analyzed and used to make comparison with our system.

- **Big Data Analysis of Salary Dataset using Hive:**

[https://globaljournals.org/s3/GJCST\\_Volume16/4-Big-Data-Analysis-of-Salary-Dataset/4-Big-Data-Analysis-of-Salary-Dataset\\_LaTeX.pdf](https://globaljournals.org/s3/GJCST_Volume16/4-Big-Data-Analysis-of-Salary-Dataset/4-Big-Data-Analysis-of-Salary-Dataset_LaTeX.pdf)

This project closely aligns with ours, as it also dealt with big data analysis. The authors aimed to examine government-collected datasets to identify trends and patterns in payroll structures. Their objective was to investigate how payment methods evolved and the factors contributing to these variations, focusing on identifying any potential job sector discrimination.

While their project and ours share similarities, the approaches differ significantly. Our primary goal was to compare the average mean salaries of individuals, regardless of their job type, across different countries. We aimed to answer whether a higher cost of living index correlates with higher salaries. To achieve this, we calculated the ratio between salary and cost of living index to determine which country offers better living conditions. This comparative analysis provided insights into how salaries align with living costs in various regions.

- **Big Data Analysis using MapReduce in Hadoop:**

<https://github.com/Mgosi/Big-Data-Analysis-using-MapReduce-in-Hadoop.git>

This project showcases an intriguing approach to processing and analyzing tweets by creating a word cloud to highlight the most frequently used phrases or words. For its social context, it is a commendable effort, especially considering the complexity and size of Twitter data. The project effectively uses MapReduce for processing Twitter data, which is a smart choice given the large volume and potentially convoluted nature of such data. Since the main objective is to process words, MapReduce serves as a suitable method for achieving this efficiently.

The project highlights the importance of data analysis in understanding and leveraging data effectively. For smaller datasets, processing and deriving insights is relatively straightforward. However, for large companies, analyzing vast datasets is crucial to uncover trends and make informed decisions.

Despite the project's strengths, our project is more complex and comprehensive. While they focused on word processing with MapReduce, we incorporated mathematical analysis, statistical analysis, and visualization in a more interactive and immersive manner. Our project aimed to be more dynamic by addressing broader questions, such as comparing average mean salaries across countries in relation to the cost of living index. This involved calculating the ratio between salary and cost of living index to identify which countries offer better living conditions. Our approach provided a richer and more detailed analysis, leveraging advanced techniques to offer deeper insights.

- **Employee Salary Analysis and Visualization with Python**

<https://github.com/SANJAYSS-SRM-26/Employee-Salary-Analysis-and-Visualization-Python-.git>

This GitHub project demonstrated how to analyze a dataset using Python, highlighting several key features:

1. **Data Display:** Viewing the loaded dataset, showcasing employee names and their corresponding salaries.
2. **Data Analysis:** Analyzing salary data by computing the minimum salary, maximum salary, and the total sum of salaries among employees.
3. **Visualization:** Generating visual representations including line graphs and vertical bar graphs to depict salary distributions based on employee names.

The project, while insightful, is relatively small in scale and relies on Pandas to visualize salaries from the dataset. This approach works well for small datasets but becomes challenging when handling larger datasets due to performance limitations.

In contrast, our project tackled the analysis of a significantly larger dataset, where processing with Pandas would be inefficient. Therefore, we utilized Spark, which is optimized for big data processing. Spark's in-memory operations and distributed computing capabilities make it much more suitable for handling vast amounts of data efficiently.

Additionally, our project is designed to be more formal and comprehensive. We not only visualized the data but also conducted an in-depth analysis to compare average mean salaries across different countries. We sought to answer whether higher cost of living indexes correlates with higher salaries by calculating the ratio between salary and cost of living index, ultimately determining which country offers better living conditions.

Overall, compared to the GitHub project, our project provides a more robust and feasible solution for large-scale data analysis and visualization.

## **VII. Description of the Dataset**

The initial dataset, "glassdoor.csv," was extracted from the "Data Jobs Listings" folder, focusing on relevant attributes such as location, country, latitude, longitude, and salary information. This dataset was then combined with "glassdoor\_salary\_salaries.csv" from the same folder, using an inner join operation based on the "id" and "salary.salaries" columns. This merge allowed for the inclusion of additional salary details, including job title, pay period, and 90th percentile salary.

The merged dataset underwent a series of cleaning steps to ensure data quality and consistency:

- **Missing Values:** Approximately 3% of the rows contained null values in the "country" column, and these rows were removed to maintain data integrity. Similarly, 70,000 rows with null values in the 10th and 90th percentile salary columns were also removed.
- **Country Name Standardization:** To ensure consistency, several country names were standardized. For instance, "Venezuela, Bolivarian Republic of" was replaced with "Venezuela," and "England" and "UK" were replaced with "United Kingdom."
- **Salary Conversion:** Hourly and monthly salaries were converted to annual salaries to facilitate comparisons across different pay periods. Hourly salaries were multiplied by 1880, and monthly salaries were multiplied by 12.

- **Cost of Living Integration:** To address the research question regarding the relationship between cost of living and salary, a "Cost of Living" dataset was incorporated. This dataset was mapped to the main dataset using the "country" column.

The final cleaned dataset, "finalcleaned3.csv," contains the refined and prepared data for subsequent analysis and visualization. This dataset serves as the foundation for exploring the relationship between location, cost of living, and employee salaries, aiming to answer the question of whether higher living costs correlate with higher earnings.

## VIII. Description of Data Processing

After completing the data cleaning and pre-processing, we shifted our focus to leveraging Apache Spark for data processing. Given Spark's robust capabilities and ease of use, we opted to utilize PySpark in Google Colab, which allowed us to harness its full potential effortlessly. Now let us look at the snippet of codes from our Colab Notebook.

First, we installed and imported Spark along with the necessary libraries. The following screenshot shows our activity:

```
!pip install pyspark
import pyspark
from pyspark.sql import SparkSession
```

Later we created a spark session called **spark**.

```
spark = SparkSession.builder.appName("StatisticalAnalysisApp").getOrCreate()
spark
```

Using the **spark** session, we read and loaded our cleaned dataset to **ds** for processing using **spark.read.csv**:

```
ds = spark.read.csv("/content/drive/MyDrive/Cloud Technologies/Notebooks/finalcleaned3 (1).csv",
header=True, inferSchema=True)
```

For better processing, we partitioned our dataset into 4 parts and created a new dataset **ds\_par**.

```
ds_par = ds.repartition(4)
```

Next, we needed to find the mean annual salary, but salaries were in the 10-percentile and 90-percentile format while some were given monthly and hourly. So, first, we had to turn them all in annual format and then put them in new columns titled **Annual-Salary-Percentile-10** and **Annual-Salary-Percentile-90**. Here we used conditional processing using when clause.

```

ds_par_s = ds_par.withColumn(
  "Annual-Salary-Percentile-10",
  when(col("Pay Period") == "MONTHLY", col("Salary-Percentile-10") * 12)
  .when(col("Pay Period") == "HOURLY", col("Salary-Percentile-10") * 52 * 40)
  .otherwise(col("Salary-Percentile-10")) )

ds_par_s = ds_par_s.withColumn(
  "Annual-Salary-Percentile-90",
  when(col("Pay Period") == "MONTHLY", col("Salary-Percentile-90") * 12)
  .when(col("Pay Period") == "HOURLY", col("Salary-Percentile-90") * 52 * 40)
  .otherwise(col("Salary-Percentile-90")) )

```

Here, we assumed employees work 12 months and work 40 hours per week for calculating annual salaries. Later from **Annual-Salary-Percentile-10** and **Annual-Salary-Percentile-90**, we calculated weighted annual salary in a column titled, **Mean-Annual-Salary**:

```

ds_par_ms = ds_par_s.withColumn(
  "Mean-Annual-Salary",
  (col("Annual-Salary-Percentile-10") * 0.1 + col("Annual-Salary-Percentile-90") * 0.9) )

```

Now this Transformed dataset **ds\_par\_ms** was loaded to **Google Drive**:

```

ds_par_ms.write.csv("/content/drive/MyDrive/Cloud Technologies/Notebooks/Transformed Datasets/Dataset
with Mean Salary", mode="overwrite", header=True)

```

From this dataset, we made a new transformed dataset by selecting 4 necessary columns titled **Job Title**, **Cost of Living Index**, **Mean-Annual-Salary**, **Country**, and **Continent** and sorted by Cost of Living Index and named it **ds\_par\_ms\_sli**.

```

ds_par_ms_sli.show()

```

Job Title	Cost of Living Index	Mean-Annual-Salary	Country	Continent
Researcher	18.8	30670.244000000002	Pakistan	Asia
Project Manager	18.8	69043.802000000001	Pakistan	Asia
Vice President Ja...	18.8	102665.5	Pakistan	Asia
Director	18.8	129215.416000000001	Pakistan	Asia

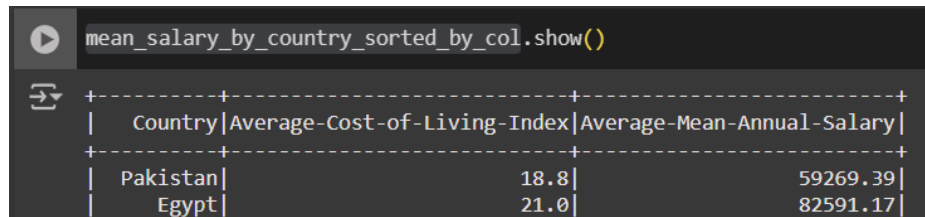
From this new dataset, we realized that there are too many types of jobs numbering in 4040. So, for our visualization purpose and exploratory statistical analysis, we decided to average all the **Mean-Annual-Salary** of different **Job Title** by grouping them by **Country** and creating a new dataset called **grouped\_ds**.

```

grouped_ds = ds_par_ms_sli.groupBy("Country").agg( avg("Mean-Annual-Salary").alias("Average-Mean-Annual-Salary"), avg("Cost of Living Index").alias("Average-Cost-of-LivingIndex"))

```

Next, we created **selected\_columns\_ds** using **Country**, **Cost of Living Index**, and **Average-Mean-Annual-Salary**, and from that, we created our final dataset titled **mean\_salary\_by\_country\_sorted\_by\_col** and loaded to **Google Drive**.



```
mean_salary_by_country_sorted_by_col.show()
```

Country	Average-Cost-of-Living-Index	Average-Mean-Annual-Salary
Pakistan	18.8	59269.39
Egypt	21.0	82591.17

```
mean_salary_by_country_sorted_by_col.write.csv("/content/drive/MyDrive/Cloud  
Technologies/Notebooks/Transformed Datasets/Dataset with Average Mean Annual Salary and Cost of Living  
Index Grouped by Country", mode="overwrite", header=True)
```

## IX. Development of the Pipeline

Our Pipeline utilized ETL mechanisms. The phases are Extraction, Transformation, and Load. Below, we explain the phases and how the pipeline has been constructed.

### Extraction Phase:

1. **Tools:** Google Drive, Google Colab, Python, Pandas Library.
2. **Activity Summary:** The datasets were available on Kaggle, spread across multiple CSV files. They were first extracted from Kaggle and loaded into Google Drive. As Google Drive is easily accessible from Colab, it gives a better opportunity to load and perform various operations. Later, using Pandas and Google Colab, data was merged, cleaned, and pre-processed.
3. **Steps Taken:**
  - a. Datasets loaded into Google Colab using Pandas.
  - b. Merging, Filtering, and Selection Operations were done.
  - c. Depending on needs Excel and Google Sheets were additionally used.

### Transformation Phase:

1. **Tools:** Google Drive, Google Colab, Python, PySpark, Pandas, Plotly.
2. **Activity Summary:** we first loaded the cleaned and pre-processed dataset on Google Colab using PySpark. Next, multiple arithmetic, filtering, sorting, and grouping operations were done using PySpark functions. After the transformation, a transformed dataset with limited columns named **Country**, **Average Cost of Living Index**, and **Average Mean Annual Salary** was formed. The final dataset was loaded into Drive for future Explanatory Visualization. Using Pandas, Plotly



and Plotly Express Visualization was done in the form of a Scattered Plot. The interactive Scattered Plot can be used to compare metrics between two or more countries by using the Legend and to answer the question “Do people in higher cost of living index make more money?.”

### 3. Steps Taken:

- a. PySpark installed on Google Colab.
- b. Spark session created named **StatisticalAnalysisApp**
- c. Cleaned dataset loaded to variable **ds** using Spark from Google Drive.
- d. Dataset **ds** was partitioned into 4 parts using the **repartition** function, and loaded to a new variable **ds\_par**.
- e. Some Columns were renamed to do operations with ease.
- f. For statistical analysis purposes new columns were made named **Annual-Salary-Percentile-10** and **Annual-Salary-Percentile-90** from **Salary-Percentile-10** and **Salary-Percentile-90**.
- g. Using **Annual-Salary-Percentile-10** and **Annual-Salary-Percentile-90**, the **Mean-Annual-Salary** column was constructed.
- h. Our main goal was to compare the **Average-Mean-Annual-Salary** of Countries against the **Average-Cost-of-Living-Index**. Hence both columns were constructed by first grouping countries and averaging the **Cost-of-Living-Index** and **Mean-Annual-Salary** of various **Job Titles**. By doing so, a new dataset named **grouped\_ds** was created.
- i. From **grouped\_ds** new dataset **selected\_columns\_ds** was made using a selected few columns named **Country**, **Average-Cost-of-Living-Index**, **Average-Mean-Annual-Salary**.
- j. Dataset **selected\_ds** was sorted by **Average-Cost-of-Living-Index** and assigned to a new variable named **mean\_salary\_by\_country\_sorted\_by\_col**.
- k. Now this **mean\_salary\_by\_country\_sorted\_by\_col** is the final dataset.

### Analysis Phase:

1. **Tools:** Google Colab, Pandas, PySpark, Plotly.
2. **Activity Summary:** With the final dataset multiple desired statistical analysis were done. From the **mean\_salary\_by\_country\_sorted\_by\_col** dataset, **Salary-CostOfLivingIndex-Ratio** (Salary to Cost of Living Index ratio) was formed and assigned to a new dataset **salary\_col\_ratio**. **Salary-CostOfLivingIndex-Ratio** gave an idea of what a Country's living conditions may look like. Higher value would be an indication that a country provides more salary while the living cost is more affordable. Later using Plotly an interactive scattered graph was made where we can

compare the Salary and Cost of Living Index among multiple companies by clicking on the legend.

### 3. Steps:

- a. Pandas was imported.
- b. **mean\_salary\_by\_country\_sorted\_by\_col** was converted to the pandas data frame.
- c. Using Plotly scattered graph was plotted.

### Loading Phase:

1. **Tools:** Google Colab, CSV, PySpark.
2. **Activity Summary:** Using PySpark, datasets were loaded into Google Drive in various folders. Depending on the size, datasets were partitioned into multiple parts.
3. **Step:**
  - a. The dataset variable was taken on Google Colab and the CSV file was written on Drive by specifying the Google Drive path.

## **X. Data Visualisation Flow**

After processing the dataset, we imported the final cleaned & processed dataset, “**part-00000-54992e1e-ca89-4253-86ca-ff8fbcd2ff60-c000.xlsx**” to Tableau to proceed with visualization. We discussed various visualizations such as choropleth, heatmap, scatterplot & clustered column chart, and proceeded with scatterplot. The scatterplot helped us to address the trends whether it is positive, negative, or there is no correlation and also helps us to see the spread of data points across the two variables. Whereas in the other visualizations, we could not get a clear expected correlation output as most of them would give a cluttered output.

Then, in a blank worksheet in Tableau public, using our dataset, we began to assign values for the scatterplot. We had to create axes, and we assigned “**Average-Cost-of-Living-Index**” to the columns, setting it as the X-axis and “**Average-Mean-Annual-Salary**” to the rows, setting it as the Y-axis. This gave us the data points between the cost of living and salary. We assigned “**Country**” to the colour option under the Marks card, so that each country is represented by a unique colour and instead of a normal circle, we preferred a shaded circle, so that the colour would be evident. Then, we assigned “**Average-Mean-Annual-Salary**” to the size option under the Marks card, to vary the size of the plots based on the salary.

We made sure that the legend was visible, which showed all the countries and their respective colours that were automatically assigned. To make sure that all the countries are plotted, we added all members when prompted. We adjusted the layout to “**Entire View**”, to ensure the scatterplot fit properly and is aligned correctly in the center. Then, we added a title, “**Scatterplot: Cost of living vs Salary**”, and updated the titles of the X-axis and Y-axis as “**Cost of Living Index**” and “**Mean Annual Salary**”, respectively.

Then, we configured the tooltips so that they show the relevant information of each country when we hover over its point.

Country: <Country>

Average-Cost-of-Living-Index: <Average-Cost-of-Living-Index>

Average-Mean-Annual-Salary: <Average-Mean-Annual-Salary>

This visualization shows the relationship between the Cost-of-living Index and Mean Annual Salary across various countries. Each data point represents a country. This scatterplot highlights whether the salaries are either high or low in each country compared to their living costs. The color-coded legend helps to easily identify the countries for comparison.

From this scatterplot, we understand the positive correlation between the Cost-of-living Index and Mean Annual Salary, which indicates that countries with a higher cost of living tend to have higher salaries. However, there are some deviations from this trend, for instance, there are some countries with a relatively high cost of living that have lower salaries, whereas on the other hand, countries with a moderate living cost offer significantly higher wages.

## **XI. Challenges and Lessons Learned**

During our project, we encountered several challenges and learned valuable lessons. Here's a detailed account:

### **1. Handling Big Data:**

- **Challenge:** Managing large datasets is inherently complex. Without appropriate tools, data may not be read or processed correctly.
- **Lesson Learned:** Utilizing powerful frameworks like PySpark is crucial for efficiently handling big data. It ensures that data processing is streamlined and scalable.

### **2. Collaboration on a Project:**

- Challenge: Effective collaboration can be difficult without proper communication and collaboration tools.
- Lesson Learned: Establishing clear communication channels and using collaborative platforms such as GitHub or Google Colab is essential for the smooth execution of team projects.

### 3. Variety in Data Types:

- Challenge: Data can come in various formats, which requires handling and standardizing them appropriately.
- Lesson Learned: Implementing robust data preprocessing steps to ensure uniformity and compatibility across different data types is important for accurate analysis.

### 4. Faulty Values in Data:

- Challenge: Even after initial cleaning, some values in the dataset may still be incorrect or require further processing.
- Lesson Learned: Continuous data validation and cleaning are necessary to maintain data integrity and accuracy throughout the project.

### 5. Writing to Google Drive:

- Challenge: Saving datasets to Google Drive often resulted in the generation of extra meta files.
- Lesson Learned: Being aware of and managing these additional files is important to keep the data storage organized.

### 6. Renaming Datasets:

- Challenge: Renaming datasets directly from notebook cells was not possible; this had to be done manually.
- Lesson Learned: Planning and organizing datasets with clear and meaningful names from the beginning can save time and reduce confusion later in the project.

### 7. Managing Dependencies:

- Challenge: Properly importing and managing dependencies was crucial to ensure the code ran smoothly.
- Lesson Learned: Creating and maintaining a comprehensive list of dependencies and using tools like virtual environments or Docker can help manage these dependencies effectively.

These challenges provided us with significant insights and helped us refine our approaches, ultimately contributing to the success of our project.

## XII. Screenshots

### A. Building a column with Salary to Cost of Living Index ratio:

```
# Calculating the Salary-Cost of Living Ratio
salary_col_ratio = mean_salary_by_country_sorted_by_col.withColumn(["Salary-CostOfLivingIndex-Ratio",
                                                                    col("Average-Mean-Annual-Salary") / col("Average-Cost-of-Living-Index")])

# Function to get ratio by country
def get_ratio_by_country(df, country_name):
    df_filtered = salary_col_ratio.filter(col("Country") == country_name)
    if df_filtered.count() == 0:
        return f"No data found for country: {country_name}"
    ratio = df_filtered.select("Salary-CostOfLivingIndex-Ratio").collect()[0][0]
    return f"The Salary-Cost of Living Index Ratio for {country_name} is: {ratio}"
```

### B. Salary to Cost of Living Index Ratio Column:

```
salary_col_ratio.show(10)
```

Country	Average-Cost-of-Living-Index	Average-Mean-Annual-Salary	Salary-CostOfLivingIndex-Ratio
Pakistan	18.8	59269.39	3152.63
Egypt	21.0	82591.17	3932.91
India	21.2	66207.92	3123.02
Bangladesh	22.5	40081.15	1781.38
Tanzania	23.8	53148.07	2233.11
Syria	24.0	45393.46	1891.39
Paraguay	25.4	71789.42	2826.36
Nepal	25.5	28775.27	1128.44
Ukraine	25.9	57452.1	2218.23
Belarus	26.4	71970.04	2726.14

only showing top 10 rows

### C. Comparing Ration between Countries:

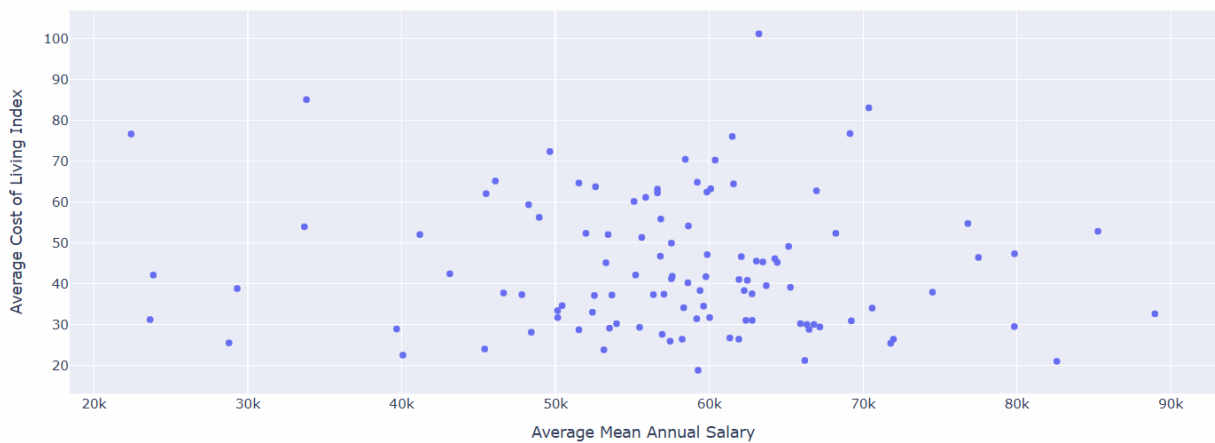
```
[ ] # Try Country Name
country_name1 = "India"
result1 = get_ratio_by_country(mean_salary_by_country_sorted_by_col, country_name1)

country_name2 = "United Kingdom"
result2 = get_ratio_by_country(mean_salary_by_country_sorted_by_col, country_name2)
print(result1)
print(result2)
```

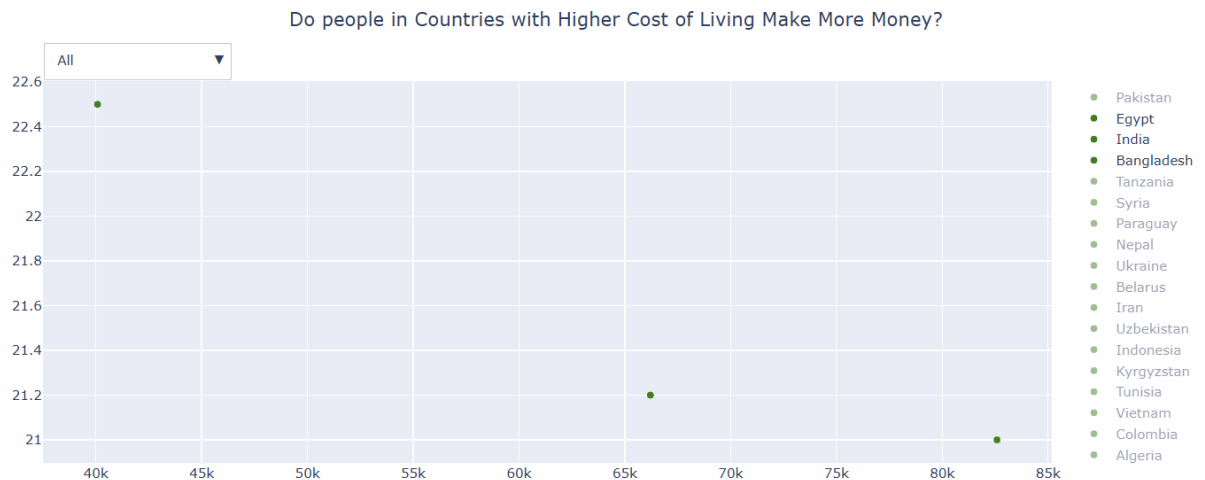
```
➞ The Salary-Cost of Living Index Ratio for India is: 3123.02
The Salary-Cost of Living Index Ratio for United Kingdom is: 733.67
```

#### D. Scattered plot with all the countries:

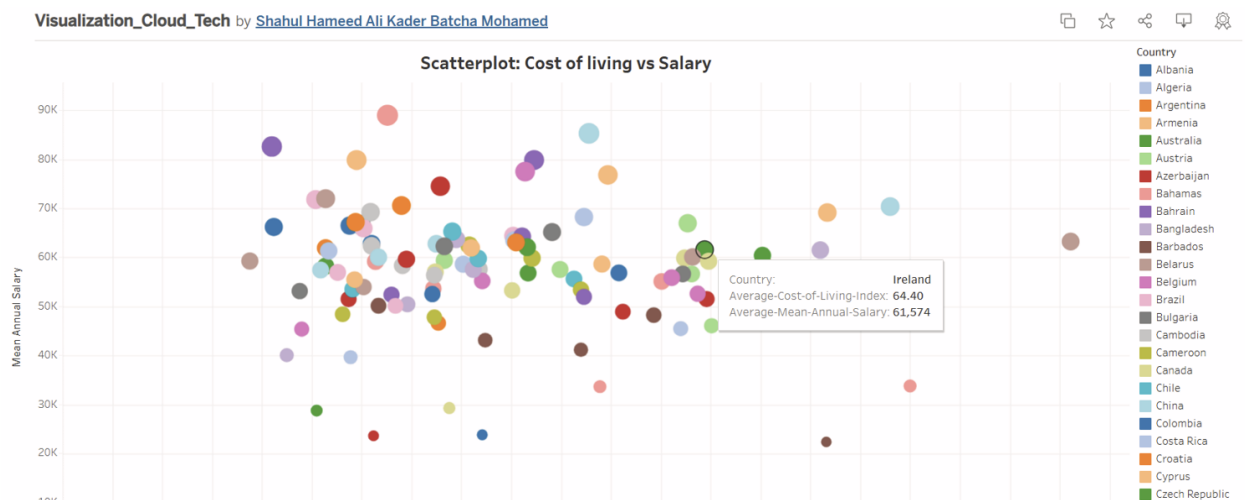
Do people in Countries with Higher Cost of Living Make More Money?



#### E. Main Visualization System, an interactive scattered graph where users can pick and compare countries:



## F. Final Visualization Using Tableau:



## XIII. Contribution

**DHANUSH ARAVIND ARUN PRAKASH:** Data Gathering, Extracting, Pre-processing, and Cleaning.

**MD HINOY RAHMAN:** Data Loading, Transformation, Processing, Statistical Analysis, PySpark Application Creation.

**SHAHUL HAMEED ALI KADER BATCHA MOHAMED:** Data Extracting, Data Exploratory Analysis, Explanatory Analysis, Visualization Webapp.

## XIV. References

[1] G. Kaur and V. Mahajan, "Big Data Analysis of Salary Dataset using Hive," Global Journals, vol. 16, no. 4, pp. 28-34, 2016. [Online]. Available: [https://globaljournals.org/s3/GJCST\\_Volume16/4-Big-Data-Analysis-of-Salary-Dataset/4-Big-Data-Analysis-of-Salary-Dataset\\_LaTeX.pdf](https://globaljournals.org/s3/GJCST_Volume16/4-Big-Data-Analysis-of-Salary-Dataset/4-Big-Data-Analysis-of-Salary-Dataset_LaTeX.pdf). [Accessed: 20-Dec-2024].

[2] M. Gosi, "Big Data Analysis using MapReduce in Hadoop," GitHub repository, [Online]. Available: <https://github.com/Mgosi/Big-Data-Analysis-using-MapReduce-in-Hadoop.git>. [Accessed: 20-Dec-2024].

[3] S. Sanjay, "Employee Salary Analysis and Visualization with Python," GitHub repository, [Online]. Available: <https://github.com/SANJAYSS-SRM-26/Employee-Salary-Analysis-and-Visualization-Python-.git>. [Accessed: 20-Dec-2024].