GROUP-11

Ulaş Yıldız - 24931
Selin Ceydeli - 28877
Berk Kayabaş - 29485
Mehmet Barış Tekdemir - 29068
Canberk Tahıl - 29515

Our GitHub link: https://github.com/berkkayabas/CS306-PROJECT.git

In our Python code, we initially used the MySQL connector as seen in the recitation. However, upon importing the Pandas library, we received an error message saying:
"UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using SQLAlchemy."

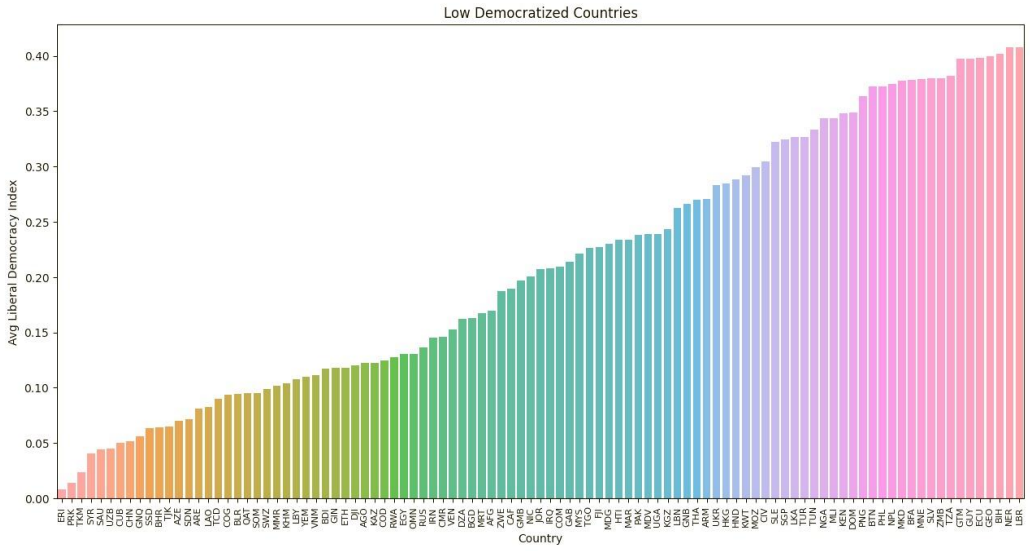
We researched this issue and found that the Pandas library recommends using SQLAlchemy when establishing database connections. That's why, to resolve this issue, we decided to use SQLAlchemy for creating a connection with our MySQL database. We imported the "create_engine" method from sqlalchemy, defined an engine, and established a database connection by applying the connect() function to this engine.

The Python code illustrating our approach to establishing a database connection, as outlined in the steps above, is presented in the below figure:

```python
from sqlalchemy import create_engine

config = {
    'host': 'localhost',
    'user': 'root',
    'password': 'Canberk102002',
    'database': 'cs306_project'
}

db_uri = f"mysql+mysqlconnector://{config['user']}:{config['password']}@{config['host']}/{config['database']}"

# Defining an engine and forming a connection to our MySQL database using this engine
try:
    engine = create_engine(db_uri)
    connection = engine.connect()
except Exception as err:
    print(err)

# Importing the views we created in the previous step of the project
if connection is not None:
    views = [
        "low_democratized_countries",
        "high_schooling_countries",
        "high_development_countries",
        "high_corr_per_countries",
        "high_bribery_countries",
        "high_democratized_countries",
        "low_corr_per_countries"
    ]

    dataframes = {}

    for view in views:
        query = f"SELECT * FROM {view};"
        dataframes[view] = pd.read_sql(query, connection)
```

Explanation of our Visuals:

Visual 1: Column Chart



Python Code of Visual 1:

```
# VISUAL 1 - SECOND VERSION
# Low Democratized Countries View - Visualized with a Vertical Column Chart having "avg_liberal_dem_idx" values ordered

# Sorting the dataframe by 'avg_liberal_dem_idx' column
dataframes["low_democratized_countries"] = dataframes["low_democratized_countries"].sort_values('avg_liberal_dem_idx')

ax = sns.barplot(data=dataframes["low_democratized_countries"], x="country", y="avg_liberal_dem_idx")
ax.set_xlabel("Country")
ax.set_ylabel("Avg Liberal Democracy Index")
ax.set_title("Low Democratized Countries")
plt.xticks(rotation=90, fontsize=8) # The x-axis labels are rotated 90 degrees and
                                    # font size is adjusted to 8 to prevent the overlapping of the labels
plt.show()
```

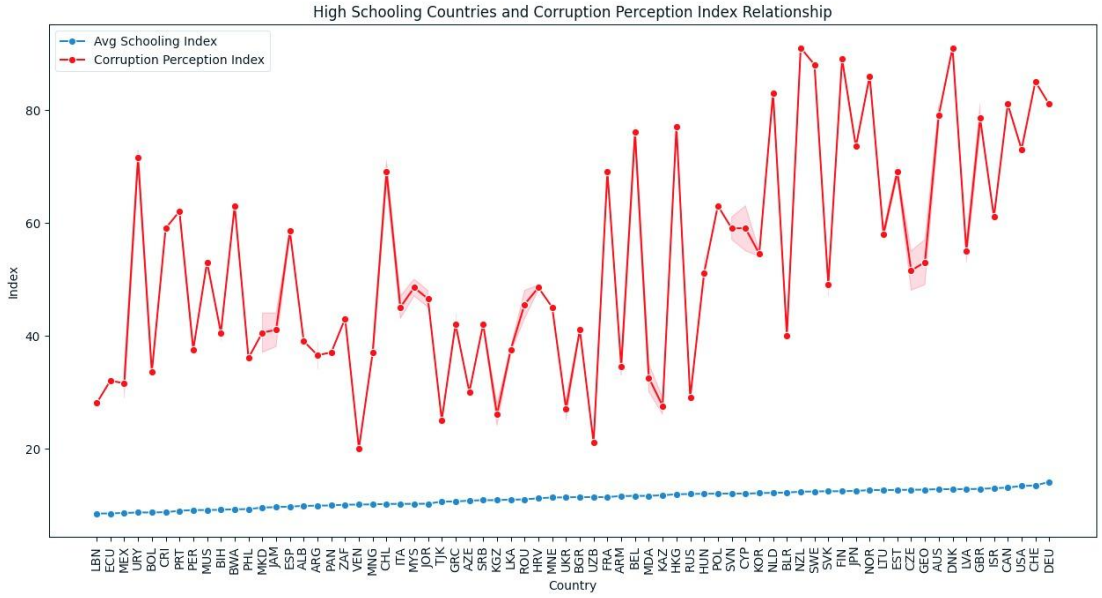
Our Observations about Visual 1:

In visual 1, we desired to compare the average liberal democracy indices of lowly democratized countries (which was a view we created in the previous step of this project). In step 3, the low democratized countries view was created using the “average” aggregate operator, which yielded the average of the liberal democracy indices of the countries for different years. In this step, we plotted the average liberal democracy indices of the countries as a column chart having the “avg_liberal_dem_idx” values ordered so that we can clearly see the countries in increasing order of their liberal democratization indices.

In conclusion, we observed that the range of the average liberal democracy indices of the countries is approximately 0.4. In other words, the country with the lowest liberal democratization index (having an iso_code ERI) has a democracy index of approximately 0.1 whereas the country with the highest liberal democratization index among the lowly democratized countries (having an iso_code LBR) has a democracy index of approximately 0.4.

Thus, it can be concluded from the first visual that even among the lowly democratized countries, democratization amounts vary heavily.

Visual 2: Line Chart



Python Code of Visual 2:

```
# VISUAL 2 – THIRD VERSION IMPROVED
# IMPROVED: High Schooling Countries View & Corruption Perception Describes Table – Visualized with a Line Chart having "avg_schooling_idx" values ordered

# Importing and processing the data from the Corruption Perception Describes Table
corr_perception_df = pd.read_sql_query("SELECT * FROM CorrupPercep_Describes", connection)
merged_df = pd.merge(dataframes["high_schooling_countries"], corr_perception_df, left_on='country', right_on='iso_code', how='inner')

# Sorting the dataframe by 'avg_schooling_idx' column
merged_df = merged_df.sort_values('avg_schooling_idx')

# Plotting the line chart
plt.figure(figsize=(10, 6))
ax = sns.lineplot(data=merged_df, x="country", y="avg_schooling_idx", marker="o", label="Avg Schooling Index")
sns.lineplot(data=merged_df, x="country", y="corr_per_index", marker="o", color="red", label="Corruption Perception Index")
ax.set_xlabel("Country")
ax.set_ylabel("Index")
ax.set_title("High Schooling Countries and Corruption Perception Index Relationship")
plt.xticks(rotation=90)
plt.legend()
plt.show()
```

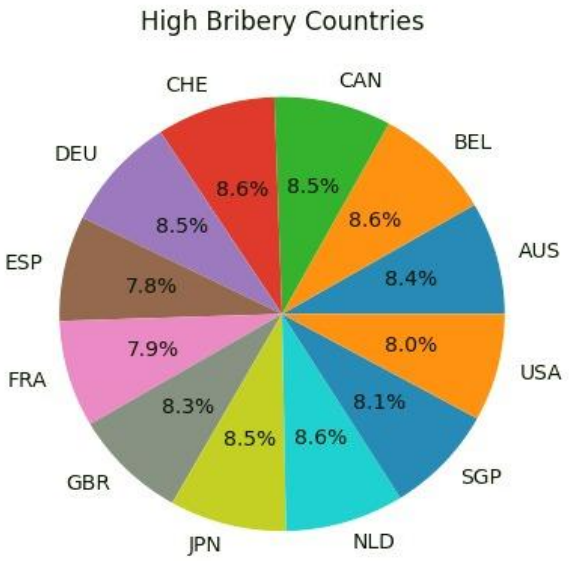
Our Observations about Visual 2:

In the second visual, we aimed to display the relationship between the schooling and corruption perception indices of the countries. Thus, we visualized the high-schooling countries view we created in the previous step of the project, where we used the “average” aggregate operator to take the average of the schooling indices of the countries for different years. In this step, we plotted the schooling indices of the countries as a line chart having the “avg_schooling_idx” values ordered so that we can clearly see the countries in increasing order of their average schooling indices.

As a second step, we also plotted the corruption perception index information of the countries belonging to the high schooling index view so that we can compare the change of the two indices and see if there is a relationship between corruption perception and schooling.

We realized that as the schooling index of the countries increases, the corruption perception does not necessarily decrease. In other words, there is not a clear negative relationship between the two variables since the red line corresponding to the corruption perception index fluctuates without a clear relationship with the increasing schooling index.

Visual 3: Pie Chart



Python Code of Visual 3:

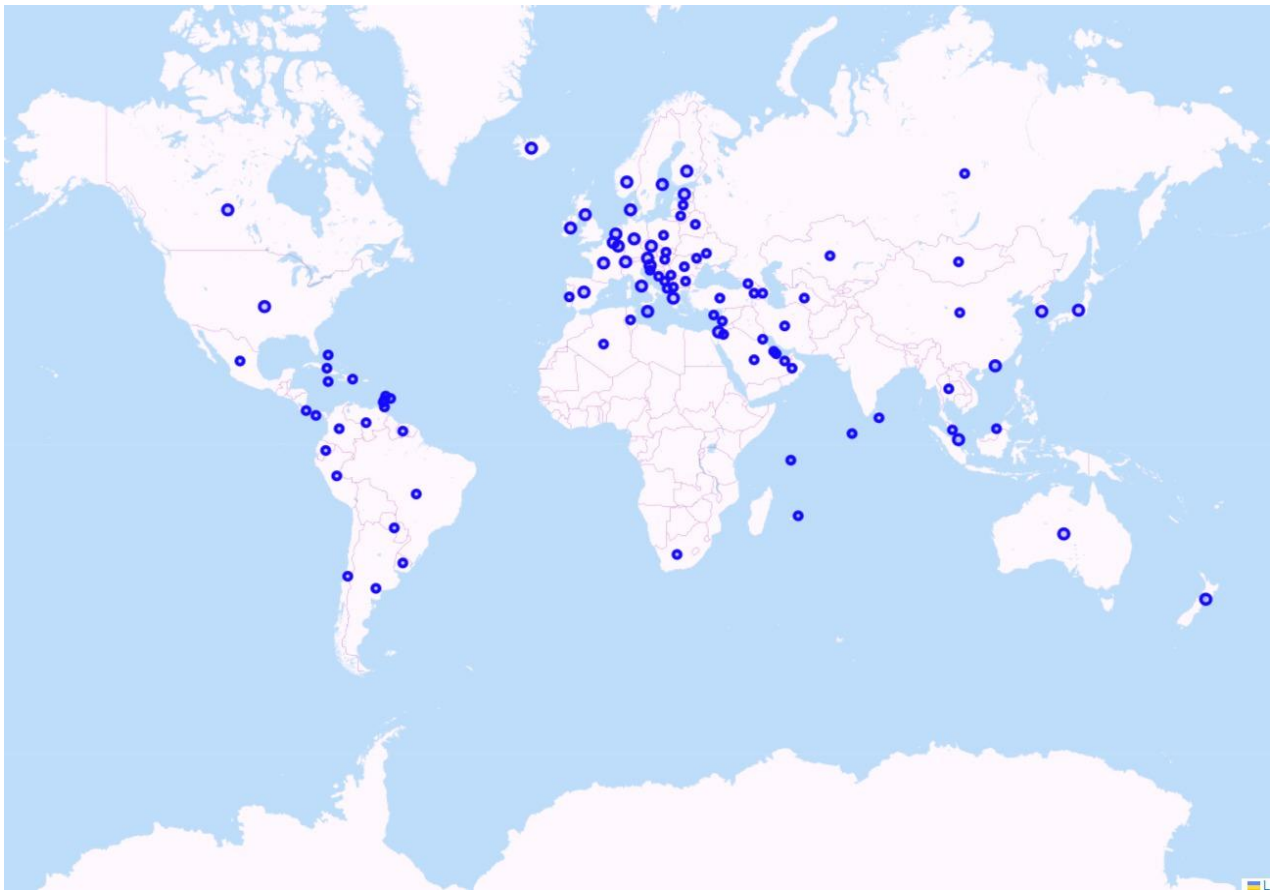
```
# VISUAL 3
# High Bribery Countries View - Visualized with a Pie Chart
plt.pie(dataframes["high_bribery_countries"]["avg_bribe_payers_index"],
        labels=dataframes["high_bribery_countries"]["country"],
        autopct='%1.1f%%')
plt.title("High Bribery Countries")
plt.xticks(rotation=90)
plt.show()
```

Our Observations about Visual 3:

In the third visual, our aim was to analyze the distribution of bribery rates among countries that have high bribery percentages (which was a view we created in the previous step of this project). In step 3, the "high_bribery_countries" view was created using the "average" aggregate operator and ">" operator which led us to the countries which have higher bribery rates than the average rates.

In this step, since there weren't many countries in the view we created in MySQL, we decided to examine the differences in the countries' corresponding bribery rates to see whether there were outlier countries or not. When the pie chart above is evaluated, it can be seen that each country that has high bribery percentages contributes nearly equally to the pie chart, indicating a uniform-like distribution of bribery rates among countries.

Visual 4: Scatter Map



Python Code of Visual 4:

```
# VISUAL 4
# High Development Countries View – Visualized with a Scatter Map / Area Map using the world map as background

# Creating a base map centered around the world
m = folium.Map(location=[0, 0], zoom_start=2)

# Extracting data from the high_development_countries dataframe
data = dataframes["high_development_countries"]

countries={
    "ALB": { "name": "Albania", "latitude": 41.1533, "longitude": 20.1683 },
    "ARE": { "name": "United Arab Emirates", "latitude": 23.4241, "longitude": 53.8478 },
    "ARG": { "name": "Argentina", "latitude": -38.4161, "longitude": -63.6167 },
    "ARM": { "name": "Armenia", "latitude": 40.0691, "longitude": 45.0382 },
    "AUS": { "name": "Australia", "latitude": -25.2744, "longitude": 133.7751 },
    "AUT": { "name": "Austria", "latitude": 47.5162, "longitude": 14.5501 },
    "AZE": { "name": "Azerbaijan", "latitude": 40.1431, "longitude": 47.5769 },
    "BEL": { "name": "Belgium", "latitude": 50.5039, "longitude": 4.4699 },
    "BGR": { "name": "Bulgaria", "latitude": 42.7339, "longitude": 25.4858 },
    "BHR": { "name": "Bahrain", "latitude": 26.0667, "longitude": 50.5577 },
    "BHS": { "name": "Bahamas", "latitude": 25.0343, "longitude": -77.3963 },
    "BIH": { "name": "Bosnia and Herzegovina", "latitude": 43.9159, "longitude": 17.6791 },
    "BLR": { "name": "Belarus", "latitude": 53.7098, "longitude": 27.9534 },
    "BRA": { "name": "Brazil", "latitude": -14.2350, "longitude": -51.9253 },
    "BRB": { "name": "Barbados", "latitude": 13.1939, "longitude": -59.5432 },
    "BRN": { "name": "Brunei", "latitude": 4.5353, "longitude": 114.7277 },
    "CAN": { "name": "Canada", "latitude": 56.1304, "longitude": -106.3468 },
    "CHE": { "name": "Switzerland", "latitude": 46.8182, "longitude": 8.2275 },
    "CHL": { "name": "Chile", "latitude": -35.6751, "longitude": -71.5430 },
    "CHN": { "name": "China", "latitude": 35.8617, "longitude": 104.1954 },
    "COL": { "name": "Colombia", "latitude": 4.5709, "longitude": -74.2973 },
    "CRI": { "name": "Costa Rica", "latitude": 9.7489, "longitude": -83.7534 },
    "CUB": { "name": "Cuba", "latitude": 21.5218, "longitude": -77.7812 },
    "CYP": { "name": "Cyprus", "latitude": 35.1264, "longitude": 33.4299 },
    "CZE": { "name": "Czechia", "latitude": 49.8175, "longitude": 15.4730 },
    "DEU": { "name": "Germany", "latitude": 51.1657, "longitude": 10.4515 },
    "DNK": { "name": "Denmark", "latitude": 56.2639, "longitude": 9.5018 },
    "DOM": { "name": "Dominican Republic", "latitude": 18.7357, "longitude": -70.1627 },
    "DZA": { "name": "Algeria", "latitude": 28.0339, "longitude": 1.6596 },
    "ECU": { "name": "Ecuador", "latitude": -1.8312, "longitude": -78.1834 },
    "ESP": { "name": "Spain", "latitude": 40.4637, "longitude": -3.7492 },
    "EST": { "name": "Estonia", "latitude": 58.5953, "longitude": 25.0136 },
    "FIN": { "name": "Finland", "latitude": 61.9241, "longitude": 25.7482 },
    "FRA": { "name": "France", "latitude": 46.6034, "longitude": 1.8883 },
    "GBR": { "name": "United Kingdom", "latitude": 55.3781, "longitude": -3.4360 },
    "GEO": { "name": "Georgia", "latitude": 42.3154, "longitude": 43.3569 },
```

```
# Adding the data points as a scatter plot to the world map
for index, row in data.iterrows():
    country = row["country"]
    avg_hum_dev_idx = row["avg_hum_dev_idx"]

    c = countries[country]["name"]

    # Getting latitude and longitude information from the above-defined countries dictionary
    lat, lon = countries.get(c, (countries[country]["latitude"], countries[country]["longitude"]))

    # Checking if the coordinates exist for the country
    if lat is not None and lon is not None:
        folium.CircleMarker(
            location=[lat, lon],
            radius=avg_hum_dev_idx * 4,  # Scaling the circle size in the scatter plot according to the value of the avg_hum_dev_idx
            color="blue",
            fill=True,
            fill_color="blue",
            popup=f"{country}: {avg_hum_dev_idx}"
        ).add_to(m)

# Saving the map with an html extension
m.save("high_development_map.html")
```

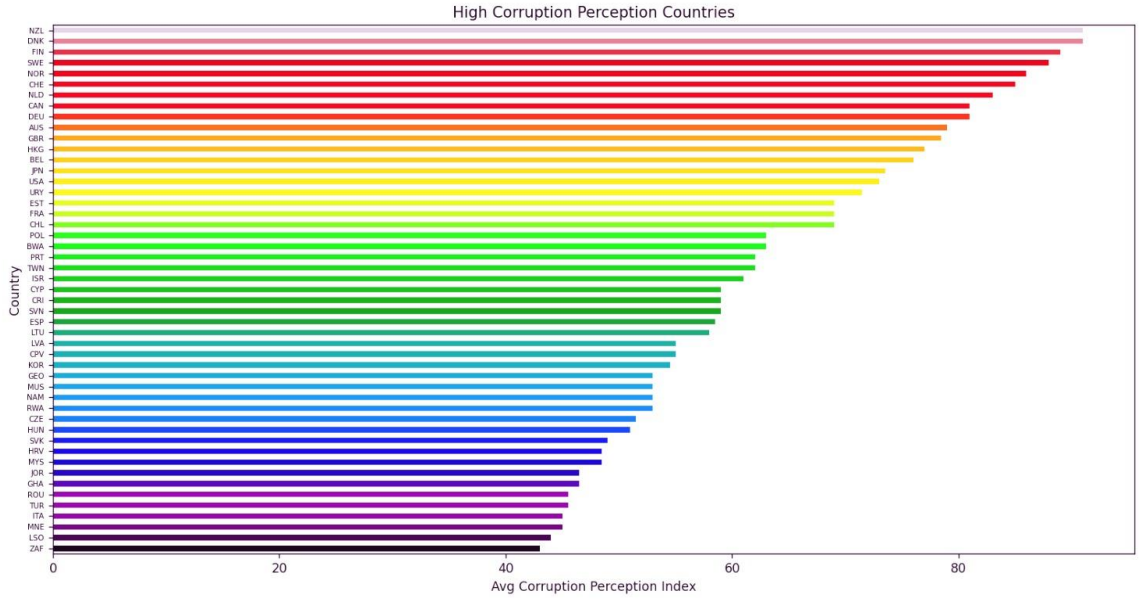
Our Observations about Visual 4:

In the Scatter map we created, we aimed to visualize the high-development countries view we generated in the previous step of this project by taking the average of the human development indices of the countries for different years. We decided to visualize the data on a world map so that we could utilize the location information of the countries to display the distribution of high-development countries in the world.

In the above world map, we scaled the circle size in the scatter plot according to the value of the average human development indices of the countries, which we calculated for each country in the previous step of the project using the "average" aggregate operator, so that when the map is viewed, one can understand which countries have higher human development indices and which have lower as compared to the other countries.

In conclusion, by looking at the circle sizes, we can conclude that each continent has countries with relatively higher human development indices and also has countries with relatively lower human development indices. Therefore, there is no uniformity among the countries in a continent in terms of the countries' development scales. Thus, we can conclude that there is not a clear correlation between the location of the countries and their human development indices.

Visual 5: Horizontal Bar Chart



Python Code of Visual 5:

```
# VISUAL 5
# High Corruption Perception Countries View – Visualized with a Horizontal Bar Plot

# Sorting the dataframe by 'avg_corr_per_idx' in descending order
sorted_df = dataframes["high_corr_per_countries"].sort_values(by="avg_corr_per_idx", ascending=True)

# Creating a color map
color_map = cm.get_cmap('nipy_spectral')
num_of_countries = len(sorted_df)
colors = color_map(np.linspace(0, 1, num_of_countries))

ax = sorted_df.plot.barh(x="country", y="avg_corr_per_idx", legend=False, color=colors, figsize=(15, 15))
ax.set_xlabel("Avg Corruption Perception Index")
ax.set_ylabel("Country")
ax.tick_params(axis='y', labelsize=6) # Decreasing the label font for the y axis to prevent the overlapping
ax.set_title("High Corruption Perception Countries")
plt.show()
```

Our Observations about Visual 5:

In the fifth visual, we visualized our fourth view which selects the countries with above-average corruption perception indices. This bar chart sorts and visualizes the countries with above-average indices which are higher than %42.805 or 0.4, which we acquired using the aggregate operator in step-3. Just like in visual 1, we aimed to plot the indices of the countries while ordering them according to “avg_corr_per_idx” so that we can see the countries in a sorted way in increasing order. The range of this plot is approximately %60 of the countries that are available in our datasets. This implies that 70 countries have above-average corruption perception indices but these above-average corruption perception indices also vary from 0.4 to 0.9.

Thus, it can be concluded from this fifth visual that even among highly corrupted countries, corruption perceptions vary significantly.

Files we have submitted to GitHub:

- **CS306 Project_Step-4 Report_Group-11:** is our step-4 report (pdf file)
- **Visualizations_Python_Code.py:** is the Python code of our visualizations