2023 | Sarah Khalid – 27258 | Urooj Mumtaz Joyo -17594
| Wajeeha Parker – 27505

EcoTrend

Big Data Analytics Project Report

Table of Contents

Introduction

This project is a combination of Machine Learning, Data Analytics and Data Visualization of the World Development Indicators for all the countries of the world. The data is taken from the world bank's databank.

This project aims at finding how different countries compare against each other based on various development indicators. A total of 34 indicators are taken for this project. Majority of the analysis is focused on seeing how the three largest economies in the world-USA, India, and China - fare against each other based on these standards. A special focus is also placed on Pakistan to analyse what factors contribute the most for Pakistan's current socio-economic crisis.

Major factors like access to education, population growth and socio-economic indicators will be analysed along with financial indicators to figure out why certain countries are developed, developing or underdeveloped.

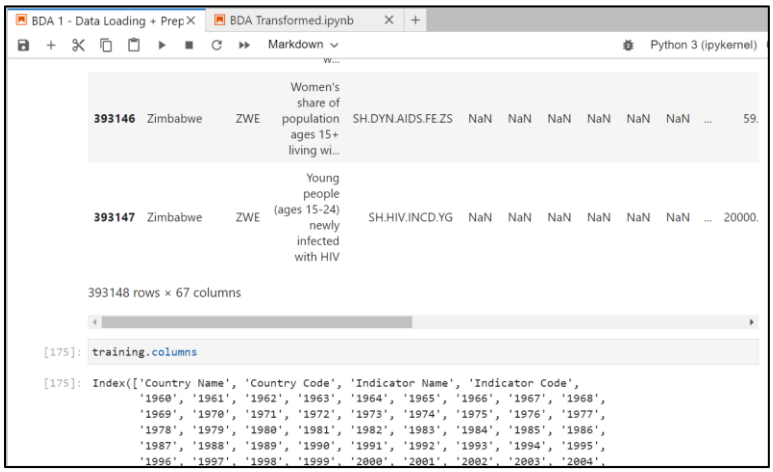
Data Set

Data Preprocessing

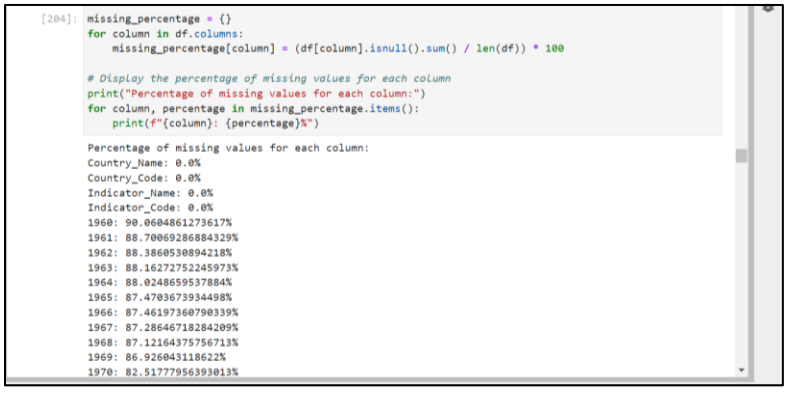
The dataset chosen was from the World Bank – World Development Indicators for 266 countries. The indicators chosen were 1478 which were composed of indicators for health, education, financial, education, socio-economic development, and population growth.

The original data had 393,148 rows and 67 columns

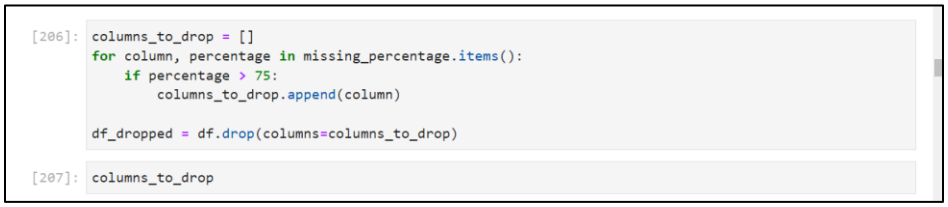
Data was present from 1960 till 2022



Columns with the % of missing values were displayed



Columns with more than 75% missing values were dropped from the dataset



The columns which remained, their missing values were replaced with the mean

```
]: for column in df_dropped.columns:
    if df_dropped[column].isnull().any():
        mean_value = df_dropped[column].mean()
        df_dropped[column].fillna(mean_value, inplace=True)

# Display the DataFrame after filling missing values with mean
print(df_dropped)
```

```
df_dropped.to_csv('C:/Users/HP/Desktop/TestBDA/data/WDI_FINAL.csv', index=False)

filtered_df = df_dropped[(df_dropped['Country_Name'] == 'China' )& (df_dropped['Indicator_Code'] == '

filtered_df
```

	Country_Name	Country_Code	Indicator_Name	Indicator_Code	1983	1984	1
134100	China	CHN	Population, total	SP.POP.TOTL	1.023310e+09	1.036825e+09	1.051040e

1 rows × 43 columns

```
numeric_columns = filtered_df.select_dtypes(include=['int', 'float']).columns
row_sums = filtered_df[numeric_columns].sum(axis=1)

row_sums/1000000

134100    49008.41
dtype: float64

tf=pd.read_csv("C:/Users/HP/Desktop/TestBDA/data/WDI_FINAL.csv", header=0, index_col=None)
```

Since the year columns were placed vertically, the columns had to be transposed into rows for proper querying of data

```
df_dropped.to_csv('C:/Users/HP/Desktop/TestBDA/data/WDI_FINAL.csv', index=False)

filtered_df = df_dropped[(df_dropped['Country_Name'] == 'China' )& (df_dropped['Indicator_Code'] == '

filtered_df
```

	Country_Name	Country_Code	Indicator_Name	Indicator_Code	1983	1984	1
134100	China	CHN	Population, total	SP.POP.TOTL	1.023310e+09	1.036825e+09	1.051040e

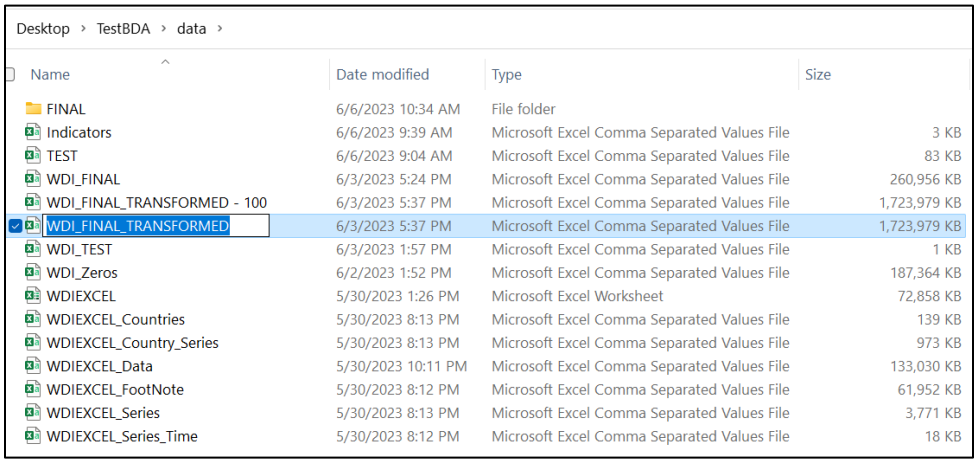
1 rows × 43 columns

```
numeric_columns = filtered_df.select_dtypes(include=['int', 'float']).columns
row_sums = filtered_df[numeric_columns].sum(axis=1)

row_sums/1000000

134100    49008.41
dtype: float64

tf=pd.read_csv("C:/Users/HP/Desktop/TestBDA/data/WDI_FINAL.csv", header=0, index_col=None)
```

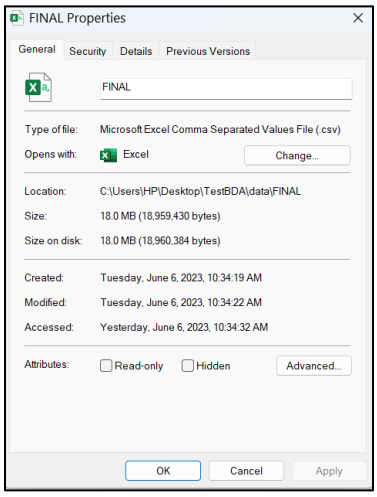
```
tf_pivoted.to_csv('C:/Users/HP/Desktop/TestBDA/data/WDI_FINAL_TRANSFORMED.csv', index=False)

filtered_df = tf_pivoted[(tf_pivoted['Country_Name'] == 'China' )& (tf_pivoted['Indicator_Code'] == '

filtered_df
```

	Country_Name	Country_Code	Indicator_Name	Indicator_Code	Year	Value
134100	China	CHN	Population, total	SP.POP.TOTL	1983	1.023310e+09
527248	China	CHN	Population, total	SP.POP.TOTL	1984	1.036825e+09
920396	China	CHN	Population, total	SP.POP.TOTL	1985	1.051040e+09
1313544	China	CHN	Population, total	SP.POP.TOTL	1986	1.066790e+09
1706692	China	CHN	Population, total	SP.POP.TOTL	1987	1.084035e+09
2099840	China	CHN	Population, total	SP.POP.TOTL	1988	1.101630e+09
2492988	China	CHN	Population, total	SP.POP.TOTL	1989	1.118650e+09
2886136	China	CHN	Population, total	SP.POP.TOTL	1990	1.135185e+09
3279284	China	CHN	Population, total	SP.POP.TOTL	1991	1.150780e+09

Once the dataset was transposed, the size of the dataset became 1.8GB



From this dataset, which was 1.8 GB, the data was filtered based on the following development indicators for all countries. This reduced the size of the dataset to 18MB



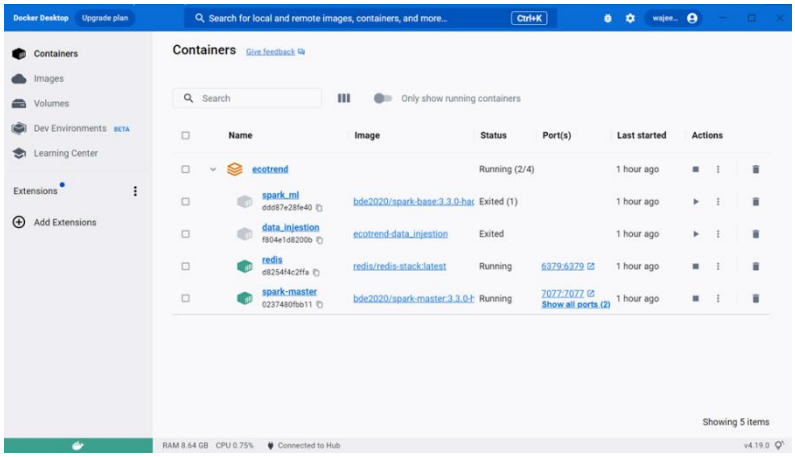
Indicators Used

1. Adjusted net national income (annual % growth)	2. Imports of goods, services and primary income (BoP, current US$)
3. Adjusted savings: education expenditure (current US$)	4. Inflation, consumer prices (annual %)
5. Adolescents out of school (% of lower secondary school age)	6. Interest Rate
7. Birth rate, crude (per 1,000 people)	8. International Liquidity
9. Broad money (% of GDP)	10. Life expectancy at birth, total (years)
11. Charges for the use of intellectual property, payments (BoP, current US$)	12. Literacy rate, adult total (% of people ages 15 and above)
13. Charges for the use of intellectual property, receipts (BoP, current US$)	14. Net primary income (Net income from abroad) (current US$)

15. Children out of school (% of primary school age)	16. Personal remittances, received (current US$)
17. Compulsory education, duration (years)	18. Population growth (annual %)
19. CPIA gender equality rating (1=low to 6=high)	20. Population, female
21. Current account balance (BoP, current US$)	22. Population, male
23. Current health expenditure (% of GDP)	24. Population, total
25. Death rate, crude (per 1,000 people)	26. Revenue, excluding grants (% of GDP)
27. Employment to population ratio, ages 15-24, total (%) (national estimate)	28. Service imports (BoP, current US$)
29. Expense (% of GDP)	30. Tax revenue (% of GDP)
31. Exports of goods, services and primary income (BoP, current US$)	32. Technical cooperation grants (BoP, current US$)
33. External debt stocks, total (DOD, current US$)	34. Total reserves (includes gold, current US$)
35. Foreign direct investment, net inflows (BoP, current US$)	36. Unemployment, total (% of total labor force) (national estimate)
37. Fossil fuel energy consumption (% of total)	38. Unemployment, youth total (% of total labor force ages 15-24) (national estimate)
39. GDP (current US$)	40. Goods, Value of Imports, Cost, Insurance, Freight (CIF), US Dollars
41. GDP growth (annual %)	42. Government expenditure on education, total (% of GDP)
43. GDP per capita (current US$)	44. Grants, excluding technical cooperation (BoP, current US$)
45. Goods, Value of Exports, Free on board (FOB), US Dollars	46. Gross savings (% of GDP)

Data Cleaning & Injestion

Docker-compose.yaml



This docker-compose file creates 4 containers:

CONTAINERS	FUNCTION
REDIS	Redis container serves as a no sql database for the project.
DATA_INJESTION	Data injestion container cleans and inserts data into redis.
SPARK-MASTER	Spark master is the master container for spark_ml.
SPARK_ML	SparkML applys machine learning techniques on the data.

Redis Container

The first container is the Redis container that works as a database container for this project. This docker-compose file creates a Redis container with port mapping, storage, and configuration for periodic data saving to disk. It ensures that only one replica of the Redis container is deployed and sets a restart policy to automatically restart the container in case of failures.

Image:

image: 'redis/redis-stack:latest'

Specifies the prebuilt Redis image that is used.

Port Mapping:

ports:

- '6379:6379'

Mapping port 6379 of Redis to the port 6379 of localhost.

Storage:

volumes:

- ./redisData:/app/local_folder

Mounts the redisData directory from the host machine to the /app/local_folder directory inside the container. This allows persistent storage for Redis data.

Environment:

environment:

- REDIS_ARGS= --save 20 1

This tells redis to save the dataset to the redis container every 20 seconds if atleast one key is changed.

```
services:
  redis:
    image: 'redis/redis-stack:latest'
    container_name: redis
    ports:
      - '6379:6379'
    volumes:
      - ./redisData:/app/local_folder
    environment:
      - REDIS_ARGS= --save 20 1
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure
```

Data_Injestion Container

Data_injestion container in this EcoTrend project is responsible for reading data from .csv files and adding it to the redis database. It utilizes python based approach, uses pandas library for data manipulation and redis library for interaction with Redis. It reads the csv data, cleans it and insert it into redis by converting it to key value pairs. This enables efficient data retrieval and analysis in different containers of this project.

Data_injestion container utilizes the following files:

Docker-compose.yaml

The docker-compose file confiuration for data_injestion container ensure that that container is built using the specified Dockerfile and is dependent on the availability of Redis container. Network mode is set to Redis service for interaction with redis.

```
17  data_injestion:
18    container_name: data_injestion
19    build:
20      context: ./DataInjestion
21      dockerfile: Dockerfile
22    depends_on:
23      - redis
24    network_mode: service:redis
```

Dockerfile

Data_injestion dockerfile contains the basic instructions for building container image.

- It uses Python 3.6 image and sets the working directory to /app.
- It copies requirements.txt file to the app folder of container and install the required dependencies.
- It also copies main.py file along with the 2 csv datafiles.
- Entrypoint.sh file is also included, executed and set as entrypoint for the container.
- Finally, main.py is executed.

DataInjestion > Dockerfile

```
1   FROM python:3.6
2
3   WORKDIR /app
4
5   COPY requirements.txt .
6   RUN pip install --no-cache-dir -r requirements.txt
7
8   COPY main.py .
9   COPY Imports/WDIEXCEL_Data_part1.csv /app/Imports/
10  COPY Imports/WDIEXCEL_Data_part2.csv /app/Imports/
11
12  COPY entrypoint.sh .
13  RUN chmod +x entrypoint.sh
14  ENTRYPOINT ["./entrypoint.sh"]
15
16  CMD ["python", "main.py"]
```

Requirements.txt

Requirements.txt specifies python dependencies required by the data_injestion container. These packages are installed during docker-compose build process and are available for main.py execution.

DataInjestion > requirements.txt

```
1   redis
2   pandas
```

Main.py

Main.py script is the core component of data-injestion container. It read the csv files and populate redis database. It follows the following execution steps:

- Redis Connection: The script establishes a connection to the redis database using redis python library. It initializes the redis object by creates the connection pool using port 6379 and uses the db number zero for storing the data.

- Reading Datafiles: The script reads two csv files and combine them in one dataframe named WDI_data for the ease of querying.

```
pool = redis.ConnectionPool(host='redis', port=6379, db=0)
redis = redis.Redis(connection_pool=pool)

########## fetching data from csv ##########
WDI_data_1 = pd.read_csv('/app/Imports/WDIEXCEL_Data_part1.csv')
WDI_data_2 = pd.read_csv('/app/Imports/WDIEXCEL_Data_part2.csv')
```

- Data Preprocessing: The script then performs preprocessing steps on WDI_data. It drops unnecessary columns – years before 2000, as there was insufficient data for these years and replace null values with the column average.

```
########## dropping unnecessary columns ##########
columns_to_drop = list(range(1960, 2000))
columns_to_drop = [str(year) for year in columns_to_drop]
WDI_data = WDI_data.drop(columns=columns_to_drop)
WDI_data = WDI_data.drop(columns='2022')
```

- Data Insertion in Redis: The data is inserted into redis in three different types of key value pairs:
  - For year-based indicator values:
    Key: “Values:{Country Code}:{year}:{Indicator Code}”
    Value: Value

```
########## inserting year based values in redis (key starting with Indicators) ##
for index, row in WDI_data.iterrows():

    if(row['Indicator Code'] in selected_indicators):
        country_code= row['Country Code']
        indicator_code= row['Indicator Code']

        for year in range(2000, 2021):
            key=f'Values:{country_code}:{year}:{indicator_code}'
            value=str(row[str(year)])
            redis.set(key, value)
```

  - For Indicators:
    Key: “Indicator:{Indicator Code}”
    Value: “{Indicator Name}”

```
########## inserting indicators in redis (key starting with Indicators) ##########
indicator_codes = WDI_data['Indicator Code'].unique()
indicator_names = WDI_data['Indicator Name'].unique()

for code, name in zip(indicator_codes, indicator_names):
    if(code in selected_indicators):
        redis.set(f'Indicator:{code}', name)
```

  - For Countries:
    Key: “Countries:{Country Code}”
    Value: “{Country Name}”

```
########## inserting countries in redis (key starting with Countries) ###
indicator_codes = WDI_data['Country Code'].unique()
indicator_names = WDI_data['Country Name'].unique()

for code, name in zip(indicator_codes, indicator_names):
    redis.set(f'Countries:{code}', name)
```

Spark-master Container

The third container is the spark-master container which act as a control unit for spark cluster. It is a pre-requisite for the final container i.e the spark_ml container.

Image

image: bde2020/spark-master:3.3.0-hadoop3.3

Specifies the pre-built Apache Spark image that includes Hadoop version 3.3. It provides necessary dependencies and configurations to run spark master node.

Port Mapping

ports:

- "8080:8080"

- "7077:7077"

The spark master container exposes two ports 8080 and 7077, which are mapped to the same ports on host machine. Port 8080 is essential for spark master's web UI whereas port 7077 is the default port of spark master to listen for worker registration. Spark workers use this port to connect to spark master.

Environment

environment:

- INIT_DAEMON_STEP=setup_spark

The INIT_DAEMON_STEP variable sets the initialization step for the spark environment. In this case it indicates the initialization of master node of spark's cluster.

```
26  spark-master:
27    image: bde2020/spark-master:3.3.0-hadoop3.3
28    container_name: spark-master
29    ports:
30      - "8080:8080"
31      - "7077:7077"
32    environment:
33      - INIT_DAEMON_STEP=setup_spark
```

Spark_ml container

The final container is the spark_ml container that perform regression analysis using pySpark on the data fetched from redis. It waits for the data_injestion container to complete its working before starting its main process. Finally, it creates a pickle file for the fitted model, which is later used for visualization.

Spark_ml container utilizes following files:

Docker-compose.yml

Docker-compose defines the container using the pre-built image bde2020/spark-base:3.3.0-hadoop3.3. Compose file ensure that that container is built using the specified Dockerfile and is dependent on the availability of spark-master and completion of data_injestion container. Environment variable connects spark_ml container with the default port of spark-master. Network mode is set to Redis service for interaction with redis.

```yaml
spark_ml:
  image: bde2020/spark-base:3.3.0-hadoop3.3
  container_name: spark_ml
  build:
    context: ./SparkML
    dockerfile: Dockerfile
  depends_on:
    - spark-master
    - data_injestion
  environment:
    - "SPARK_MASTER=spark://spark-master:7077"
  volumes:
    - ./SparkML:/app/sparkml
  network_mode: service:redis
```

Dockerfile

Spark_ml dockerfile contains the basic instructions for building container image.

- It uses Python 3.6 image.
- Installs default JAVA runtime environment in the container and set /usr as the JAVA_HOME environment path.
- It sets the working directory to /app.
- It copies requirements.txt file to the app folder of container and install the required dependencies.
- It also copies regression.py file to the container.
- Entrypoint.sh file is also included, executed and set as entrypoint for the container.
- Finally, regression.py is executed.

SparkML > Dockerfile

```dockerfile
FROM python:3.6

# Install Java
RUN apt-get update && apt-get install -y default-jre
#COPY java/ /usr/java/

# Set JAVA_HOME environment variable
ENV JAVA_HOME=/usr

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

RUN pip install scikit-learn

COPY regression.py .

COPY entrypoint.sh .
RUN chmod +x entrypoint.sh
ENTRYPOINT ["./entrypoint.sh"]

CMD ["python", "regression.py"]
```

Entrypoint.sh

Entrypoint.sh specifies the entrypoint for spark_ml container. It waits for data_injestion to complete its working before spark_ml starts its working. It uses while loop and curl command to check the readiness of data_injestion container. Once data_injestion have finished its working, it starts the main process for spark_ml container.

SparkML > $ entrypoint.sh

```bash
#!/bin/bash

# Wait for the data_injestion container to be ready
echo "Waiting for data_injestion container to complete its work..."
while ! curl -s http://data_injestion >/dev/null; do
    sleep 1
done
echo "data_injestion container is ready!"

# Start the main process
exec "$@"
```

Requirements.txt

Requirements.txt specifies python dependencies required by the spark_ml container. These packages are installed during docker-compose build process and are available for regression.py execution.

SparkML > requirements.txt

```
redis
pyspark
numpy
pandas
```

Regression.py

Regression.py script is the core component of spark_ml container. It fetches the data from redis, apply ml techniques to it and creates pickle file for visualization. It follows the following execution steps:

- Redis Connection: The script establishes a connection to the redis database using redis python library. It initializes the redis object by creates the connection pool using port 6379 and uses the db number zero for retrieving data.
- Create Dataframe: The script splits the keys of the fetched data and creates dataframe from it using pandas.

```
for key in keys:
    # Split the key into its components
    components = key.decode().split(':')
    if(components[1] in selected_countries):
        country_codes.append(components[1])
        years.append(components[2])
        indicator_codes.append(components[3])
        country_names.append(redis.get(f'Countries:{components[1]}'))
        indicator_names.append(redis.get(f'Indicator:{components[3]}'))

        value = redis.get(key).decode()
        values.append(value)

########## creating dataframe ##########
data = {
    'Country Name': country_names,
    'Country Code': country_codes,
    'Indicator Name': indicator_names,
    'Indicator Code': indicator_codes,
    'Year': years,
    'Value': values
}
df = pd.DataFrame(data)
```

- Encoding: The script then encodes categorical variables and apply numerical conversions.

```
X = df.drop(columns=['Country Name', 'Indicator Name'])
X['Indicator Code'] = X['Indicator Code'].replace(selected_indicators, range(len(selected_indicators)))
X['Country Code'] = X['Country Code'].replace(selected_countries, range(len(selected_countries)))
X = X.apply(pd.to_numeric, errors='coerce')
```

- Applying Regression: Initialize spark session named Regression. It sets the input and output columns. Applies vector assembler to convert pandas dataframes to spark dataframes and then applys the regression techniques.

```
########## Applying Machine Learning Techniques ##########

spark = SparkSession.builder.appName("Regression").getOrCreate()
inputCols=['Country Code', 'Indicator Code', 'Year']
outputCol='Value'

assembler = VectorAssembler(inputCols=inputCols, outputCol="features")
df_spark = spark.createDataFrame(X)
df_transformed = assembler.transform(df_spark)

trainData, testData = df_transformed.randomSplit([0.7, 0.3], seed=123)

print("Gradient Boosting")
model = GBTRegressor(featuresCol="features", labelCol=outputCol).fit(trainData)
predictions = model.transform(testData)
```

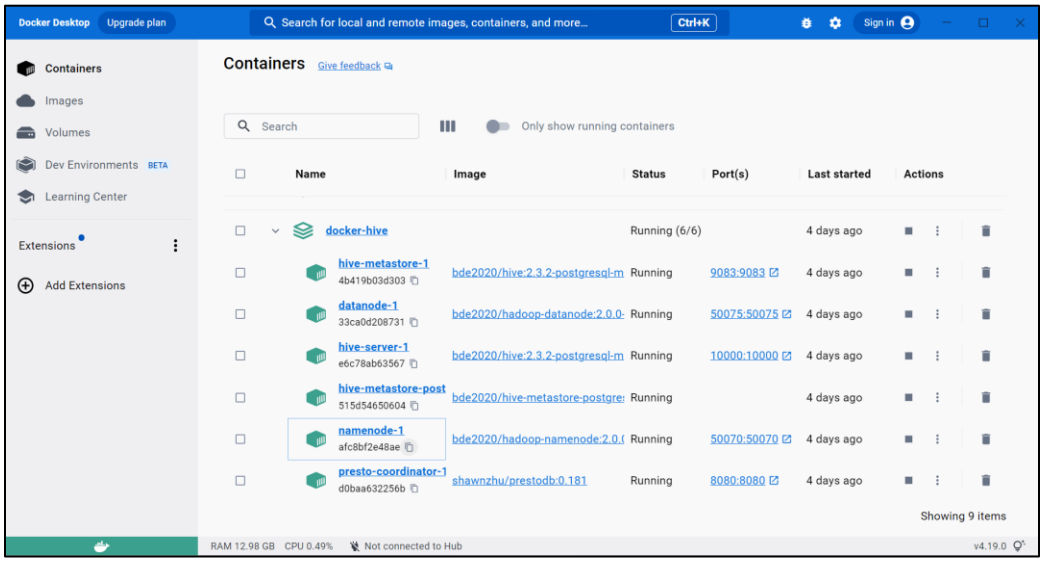
- Create Pickle Object: Finally, it creates a pickle file for the fitted model. The pickle object includes the model and the features for prediction. This pickle file is copied from the container repository to the users's local repository.

```
########## Generating pickle object for streamlit ##########

X = df.drop(columns=['Value'])
reg=GradientBoostingRegressor().fit(X,y)
pickle_model = {
    'model': reg,
    'X':X
}

pickle.dump(pickle_model, open('ecoTrend_model.sav', 'wb'))
local_directory = os.getcwd()
local_file_path = os.path.join(local_directory, 'ecoTrend_model.sav')
os.rename('/app/ecoTrend_model.sav', local_file_path)
```

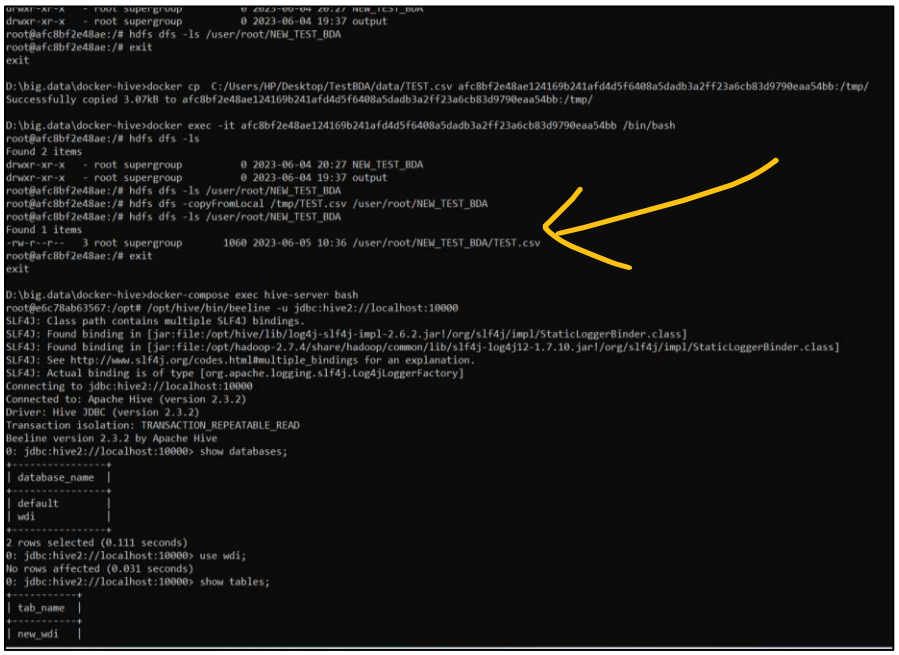
Data Analysis

Data Loading – HDFS

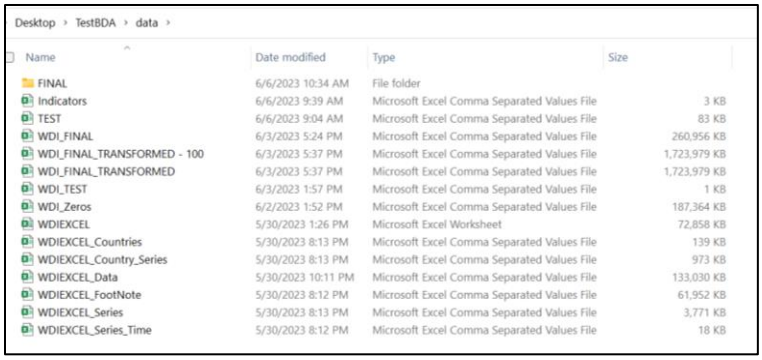
Using a HIVE container;



Several directories were made in the HDFS /user/root/ such as input (to store all input files), output(to store the results of hive queries, and a test directory called NEW_TEST_BDA



The csv files used during the testing were as shown below:



Desktop › TestBDA › data ›

Name	Date modified	Type	Size
FINAL	6/6/2023 10:34 AM	File folder	
Indicators	6/6/2023 9:39 AM	Microsoft Excel Comma Separated Values File	3 KB
TEST	6/6/2023 9:04 AM	Microsoft Excel Comma Separated Values File	83 KB
WDI_FINAL	6/3/2023 5:24 PM	Microsoft Excel Comma Separated Values File	260,956 KB
WDI_FINAL_TRANSFORMED - 100	6/3/2023 5:37 PM	Microsoft Excel Comma Separated Values File	1,723,979 KB
WDI_FINAL_TRANSFORMED	6/3/2023 5:37 PM	Microsoft Excel Comma Separated Values File	1,723,979 KB
WDI_TEST	6/3/2023 1:57 PM	Microsoft Excel Comma Separated Values File	1 KB
WDI_Zeros	6/2/2023 1:52 PM	Microsoft Excel Comma Separated Values File	187,364 KB
WDIEXCEL	5/30/2023 1:26 PM	Microsoft Excel Worksheet	72,858 KB
WDIEXCEL_Countries	5/30/2023 8:13 PM	Microsoft Excel Comma Separated Values File	139 KB
WDIEXCEL_Country_Series	5/30/2023 8:13 PM	Microsoft Excel Comma Separated Values File	973 KB
WDIEXCEL_Data	5/30/2023 10:11 PM	Microsoft Excel Comma Separated Values File	133,030 KB
WDIEXCEL_FootNote	5/30/2023 8:12 PM	Microsoft Excel Comma Separated Values File	61,952 KB
WDIEXCEL_Series	5/30/2023 8:13 PM	Microsoft Excel Comma Separated Values File	3,771 KB
WDIEXCEL_Series_Time	5/30/2023 8:12 PM	Microsoft Excel Comma Separated Values File	18 KB

All files were loaded from the local system to the name node's tmp directory, then copied into the hdfs's NEW_TEST_BDA directory and then used in hive queries.

```
exit

D:\big.data\docker-hive>docker cp  C:/Users/HP/Desktop/TestBDA/data/TEST.csv afc8bf2e48ae124169b241afd4d5f6408a5dadb3a2ff23a6cb83d9790eaa54bb:/tmp/
Successfully copied 3.07kB to afc8bf2e48ae124169b241afd4d5f6408a5dadb3a2ff23a6cb83d9790eaa54bb:/tmp/

D:\big.data\docker-hive>docker exec -it afc8bf2e48ae124169b241afd4d5f6408a5dadb3a2ff23a6cb83d9790eaa54bb /bin/bash
root@afc8bf2e48ae:/# hdfs dfs -ls
Found 2 items
drwxr-xr-x   - root supergroup          0 2023-06-04 20:27 NEW_TEST_BDA
drwxr-xr-x   - root supergroup          0 2023-06-04 19:37 output
root@afc8bf2e48ae:/# hdfs dfs -ls /user/root/NEW_TEST_BDA
root@afc8bf2e48ae:/# hdfs dfs -copyFromLocal /tmp/TEST.csv /user/root/NEW_TEST_BDA
root@afc8bf2e48ae:/# hdfs dfs -ls /user/root/NEW_TEST_BDA
Found 1 items
-rw-r--r--   3 root supergroup       1060 2023-06-05 10:36 /user/root/NEW_TEST_BDA/TEST.csv
root@afc8bf2e48ae:/# exit
exit
```

COMMANDS USED IN THE PROJECT WORKING WITH HDFS AND HIVE

COPY CSV INTO NAMENODE TMP DIRECTORY

docker cp C:/Users/HP/Desktop/TestBDA/data/WDI_TEST.csv e00a03940e2d995f6a8d5805e500f5522155b8d80acb8189912eb28f28ef11e3:/tmp/

BASH INTO NAMENODE

docker exec -it e00a03940e2d995f6a8d5805e500f5522155b8d80acb8189912eb28f28ef11e3 /bin/bash

CREATE A DIRECTORY IN HDFS

hdfs dfs -mkdir -p /user/root/input

CHECK ALL DIRECTORIES IN HDFS

hdfs dfs -ls

GO TO THE DIRECTORY IN HDFS

hdfs dfs -mkdir -p /user/root/NEW_TEST_BDA

COPY FROM NAMENODE TMP DIRECTORY TO THE DIRECTORY IN HDFS

hdfs dfs -copyFromLocal /tmp/WDI_TEST.csv /user/root/NEW_TEST_BDA

CHECK IF THE FILE WAS COPIED

hdfs dfs -ls

hdfs dfs -ls /user/root/NEW_TEST_BDA

BASH INTO THE HIVE SERVER

docker-compose exec hive-server bash

CONNECT TO BEELINE

/opt/hive/bin/beeline -u jdbc:hive2://localhost:10000

CREATE DATABASE

USE DATABASE

CREATE TABLE

create table test2(country_name string, indicator string, amount decimal)ROW FORMAT DELIMITED . . . . . .> FIELDS TERMINATED BY ',' . . . . . .> STORED AS TEXTFILE . . . . . .> LOCATION '/user/root/NEW_TEST_BDA';

create table WDI_DATA(Country_name string, Country_Code string, Indicator String, Indicator_code string , Year BIGINT, value decimal(40,10)) . . . . . .> ROW FORMAT DELIMITED . . . . . .> FIELDS TERMINATED BY ',' . . . . . .> STORED AS TEXTFILE . . . . . .> LOCATION '/user/root/NEW_TEST_BDA';

COPY THE CSV FROM THE HDFS DIRECTORY TO THE TEST TABLE

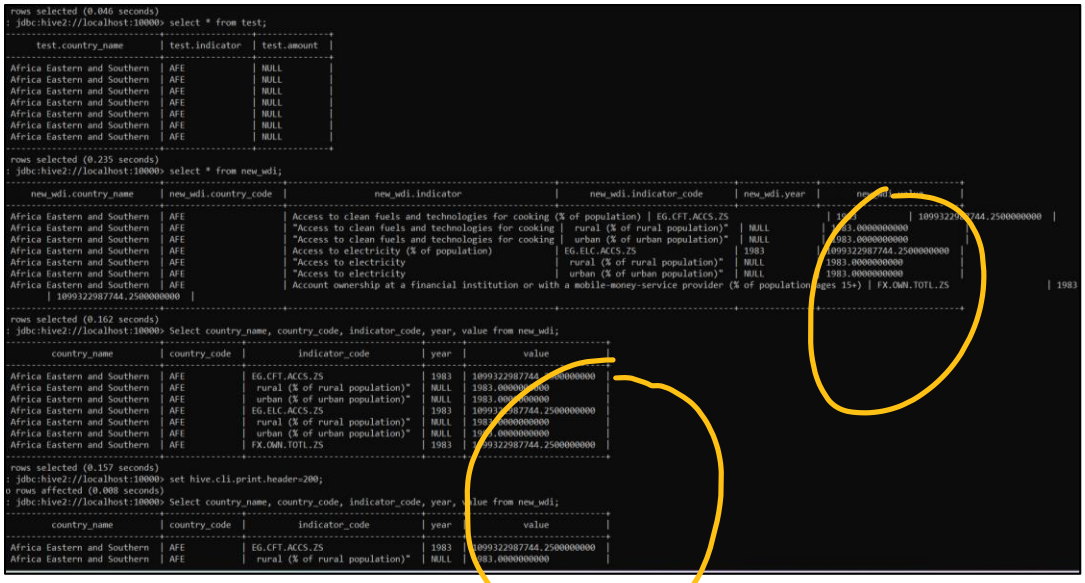
LOAD DATA INPATH '/user/root/NEW_TEST_BDA/WDI_TEST.csv' INTO TABLE TEST;

CHECK IF THE RECORDS WERE INSERTED

SELECT * FROM TEST;

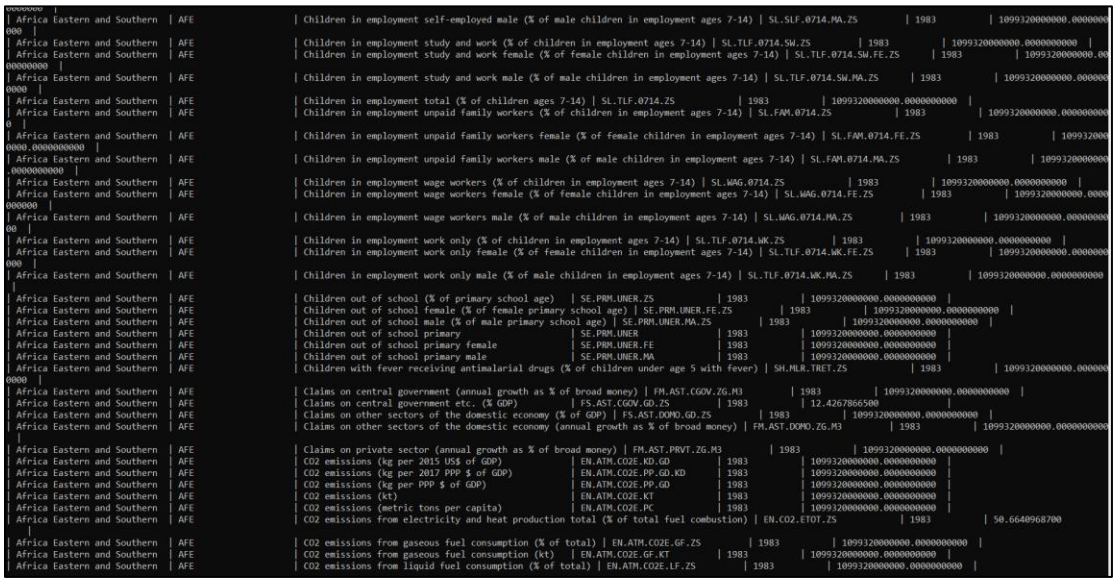
Step 3: Creation of HIVE tables

Once the csvs were loaded into the hdfs directories , we made several test tables to load the data from the hdfs directories. Initially we got a lot of data loading errors and issues as shown below:



So multiple attempts were made to load csvs from the local system to hdfs and then from hdfs to hive tables

1.8GB of data was finally loaded into the hdfs NEW_TEST_BDA directory



Step 4: Loading data into HIVE tables

We created test tables and tables with the names of new_wdi and wdi_final where the main data was loaded and tested several times

34 world development indicators were selected

```
0: jdbc:hive2://localhost:10000> select distinct wdi_final.indicator_code from wdi_final;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
+-----------------------------+
| wdi_final.indicator_code    |
+-----------------------------+
| BHS                         |
| BN.CAB.XOKA.GD.ZS           |
| COD                         |
| COG                         |
| EG.ELC.ACCS.ZS              |
| EG.USE.COMM.FO.ZS           |
| EGY                         |
| FM.LBL.BMNY.GD.ZS           |
| FP.CPI.TOTL.ZG              |
| FSM                         |
| GC.DOD.TOTL.CN              |
| GC.DOD.TOTL.GD.ZS           |
| GC.TAX.TOTL.GD.ZS           |
| GC.XPN.TOTL.CN              |
| GC.XPN.TOTL.GD.ZS           |
| GMB                         |
| HKG                         |
| IQ.CPA.GNDR.XQ              |
| IRN                         |
| Indicator_Code              |
| KOR                         |
| MAC                         |
| NY.ADJ.AEDU.CD              |
| NY.ADJ.NNTY.KD.ZG           |
| NY.GDP.MKTP.KD.ZG           |
| NY.GNS.ICTR.ZS              |
| NY.GSR.NFCY.CD              |
| PRK                         |
| SE.ADT.LITR.ZS              |
| SE.COM.DURS                 |
| SE.PRM.UNER.ZS              |
| SE.SEC.UNER.LO.ZS           |
| SE.TER.CUAT.BA.ZS           |
| SE.XPD.TOTL.GD.ZS           |
| SH.XPD.CHEX.GD.ZS           |
| SL.EMP.1524.SP.NE.ZS        |
| SL.TLF.0714.ZS              |
| SL.UEM.1524.NE.ZS           |
| SL.UEM.TOTL.NE.ZS           |
| SP.DYN.CBRT.IN              |
```

Multiple queries were run to analyse the data, country wise, indicator wise and year wise

Step 5: Queries and data analysis

FINAL TABLES

create table WDI_FINAL(Country_Code string,Indicator_code string , Year BIGINT, value decimal(38,10)) . . . . . .> ROW FORMAT DELIMITED . . . . . .> FIELDS TERMINATED BY ',' . . . . . .> STORED AS TEXTFILE . . . . . .> LOCATION '/user/root/NEW_TEST_BDA';

create table Countries(Country_code string, Country_Name string,) . . . . . .> ROW FORMAT DELIMITED . . . . . .> FIELDS TERMINATED BY ',' . . . . . .> STORED AS TEXTFILE . . . . . .> LOCATION '/user/root/NEW_TEST_BDA';

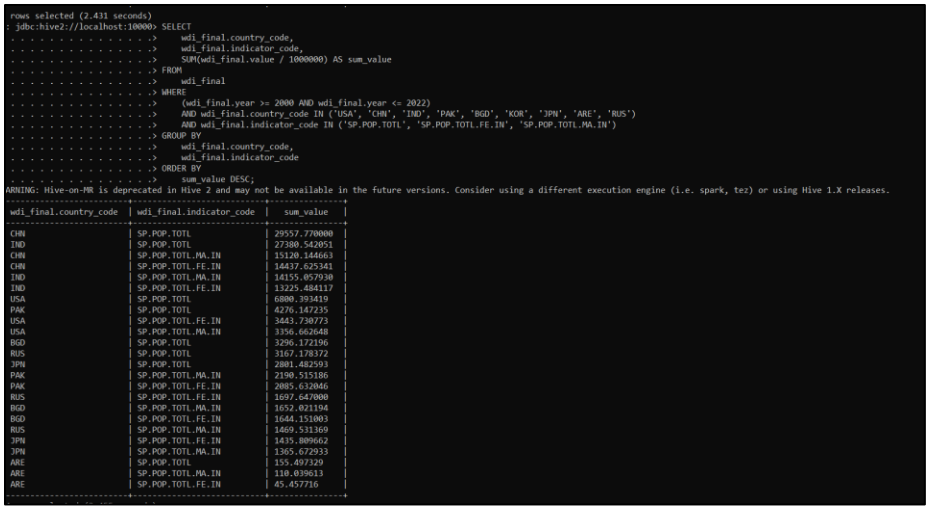
create table Indicators(Indicator_code string, Indicator_name string) . . . . . .> ROW FORMAT DELIMITED . . . . . .> FIELDS TERMINATED BY ',' . . . . . .> STORED AS TEXTFILE . . . . . .> LOCATION '/user/root/NEW_TEST_BDA';

create table Regions(Country_code string, Region string) . . . . . .> ROW FORMAT DELIMITED . . . . . .> FIELDS TERMINATED BY ',' . . . . . .> STORED AS TEXTFILE . . . . . .> LOCATION '/user/root/NEW_TEST_BDA';

Queries

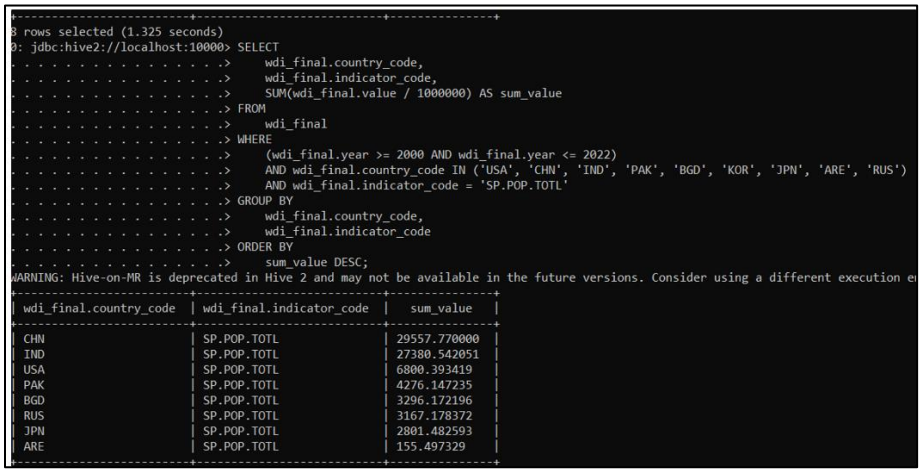
Sample Query

```
22 rows selected (3.918 seconds)
0: jdbc:hive2://localhost:10000> select wdi_final.country_code , wdi_final.indicator_code, wdi_final.year, sum(wdi_final.value/1000000) from wdi_final where (wdi_final.year>=2000 and w
i_final.country_code='USA' and wdi_final.indicator_code='SP.POP.TOTL' group by wdi_final.country_code, wdi_final.indicator_code, wdi_final.year;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
+-------------------------+---------------------------+----------------+-------------+
| wdi_final.country_code  | wdi_final.indicator_code  | wdi_final.year |     _c3     |
+-------------------------+---------------------------+----------------+-------------+
| USA                     | SP.POP.TOTL               | 2000           | 282.162411  |
| USA                     | SP.POP.TOTL               | 2001           | 284.968955  |
| USA                     | SP.POP.TOTL               | 2002           | 287.625193  |
| USA                     | SP.POP.TOTL               | 2003           | 290.107933  |
| USA                     | SP.POP.TOTL               | 2004           | 292.805298  |
| USA                     | SP.POP.TOTL               | 2005           | 295.516599  |
| USA                     | SP.POP.TOTL               | 2006           | 298.379912  |
| USA                     | SP.POP.TOTL               | 2007           | 301.231207  |
| USA                     | SP.POP.TOTL               | 2008           | 304.093966  |
| USA                     | SP.POP.TOTL               | 2009           | 306.771529  |
| USA                     | SP.POP.TOTL               | 2010           | 309.327143  |
| USA                     | SP.POP.TOTL               | 2011           | 311.583481  |
| USA                     | SP.POP.TOTL               | 2012           | 313.877662  |
| USA                     | SP.POP.TOTL               | 2013           | 316.059947  |
| USA                     | SP.POP.TOTL               | 2014           | 318.386329  |
| USA                     | SP.POP.TOTL               | 2015           | 320.738994  |
| USA                     | SP.POP.TOTL               | 2016           | 323.071755  |
| USA                     | SP.POP.TOTL               | 2017           | 325.122128  |
| USA                     | SP.POP.TOTL               | 2018           | 326.838199  |
| USA                     | SP.POP.TOTL               | 2019           | 328.329953  |
| USA                     | SP.POP.TOTL               | 2020           | 331.501080  |
| USA                     | SP.POP.TOTL               | 2021           | 331.893745  |
+-------------------------+---------------------------+----------------+-------------+
22 rows selected (1.377 seconds)
```



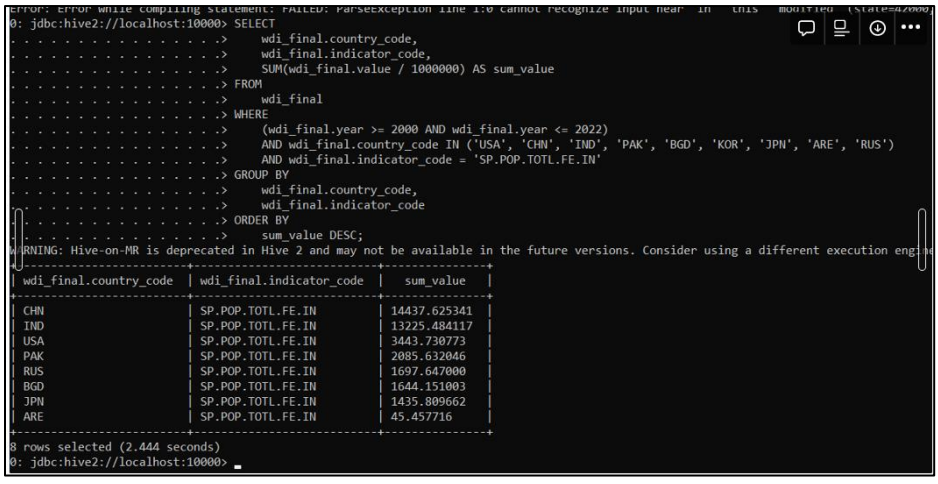
```
rows selected (2.431 seconds)
: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value / 1000000) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code IN ('SP.POP.TOTL', 'SP.POP.TOTL.FE.IN', 'SP.POP.TOTL.MA.IN')
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
ARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
+-------------------------+--------------------------+--------------+
| wdi_final.country_code  | wdi_final.indicator_code |   sum_value  |
+-------------------------+--------------------------+--------------+
| CHN                     | SP.POP.TOTL              | 29557.770000 |
| IND                     | SP.POP.TOTL              | 27380.542051 |
| CHN                     | SP.POP.TOTL.MA.IN        | 15120.144663 |
| CHN                     | SP.POP.TOTL.FE.IN        | 14437.625341 |
| IND                     | SP.POP.TOTL.MA.IN        | 14155.057930 |
| IND                     | SP.POP.TOTL.FE.IN        | 13225.484117 |
| USA                     | SP.POP.TOTL              | 6800.393419  |
| PAK                     | SP.POP.TOTL              | 4276.147235  |
| USA                     | SP.POP.TOTL.FE.IN        | 3443.730773  |
| USA                     | SP.POP.TOTL.MA.IN        | 3356.662648  |
| BGD                     | SP.POP.TOTL              | 3296.172196  |
| RUS                     | SP.POP.TOTL              | 3167.178372  |
| JPN                     | SP.POP.TOTL              | 2801.482593  |
| PAK                     | SP.POP.TOTL.MA.IN        | 2190.515186  |
| PAK                     | SP.POP.TOTL.FE.IN        | 2085.632046  |
| RUS                     | SP.POP.TOTL.FE.IN        | 1697.647000  |
| BGD                     | SP.POP.TOTL.MA.IN        | 1652.021194  |
| BGD                     | SP.POP.TOTL.FE.IN        | 1644.151003  |
| RUS                     | SP.POP.TOTL.MA.IN        | 1469.531369  |
| JPN                     | SP.POP.TOTL.FE.IN        | 1435.809662  |
| JPN                     | SP.POP.TOTL.MA.IN        | 1365.672933  |
| ARE                     | SP.POP.TOTL              | 155.497329   |
| ARE                     | SP.POP.TOTL.MA.IN        | 110.039613   |
| ARE                     | SP.POP.TOTL.FE.IN        | 45.457716    |
+-------------------------+--------------------------+--------------+
```

Query: Population, total SP.POP.TOTL



```
+-------------------------+--------------------------+--------------+
8 rows selected (1.325 seconds)
0: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value / 1000000) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'SP.POP.TOTL'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution e
+-------------------------+--------------------------+--------------+
| wdi_final.country_code  | wdi_final.indicator_code |   sum_value  |
+-------------------------+--------------------------+--------------+
| CHN                     | SP.POP.TOTL              | 29557.770000 |
| IND                     | SP.POP.TOTL              | 27380.542051 |
| USA                     | SP.POP.TOTL              | 6800.393419  |
| PAK                     | SP.POP.TOTL              | 4276.147235  |
| BGD                     | SP.POP.TOTL              | 3296.172196  |
| RUS                     | SP.POP.TOTL              | 3167.178372  |
| JPN                     | SP.POP.TOTL              | 2801.482593  |
| ARE                     | SP.POP.TOTL              | 155.497329   |
+-------------------------+--------------------------+--------------+
```

Query: Population, female SP.POP.TOTL.FE.IN



```
Error: Error while compiling statement: FAILED: ParseException line 1:0 cannot recognize input near 'in' 'this' 'modified' (state=42000
0: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value / 1000000) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'SP.POP.TOTL.FE.IN'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine
+-------------------------+--------------------------+--------------+
| wdi_final.country_code  | wdi_final.indicator_code |   sum_value  |
+-------------------------+--------------------------+--------------+
| CHN                     | SP.POP.TOTL.FE.IN        | 14437.625341 |
| IND                     | SP.POP.TOTL.FE.IN        | 13225.484117 |
| USA                     | SP.POP.TOTL.FE.IN        | 3443.730773  |
| PAK                     | SP.POP.TOTL.FE.IN        | 2085.632046  |
| RUS                     | SP.POP.TOTL.FE.IN        | 1697.647000  |
| BGD                     | SP.POP.TOTL.FE.IN        | 1644.151003  |
| JPN                     | SP.POP.TOTL.FE.IN        | 1435.809662  |
| ARE                     | SP.POP.TOTL.FE.IN        | 45.457716    |
+-------------------------+--------------------------+--------------+
8 rows selected (2.444 seconds)
0: jdbc:hive2://localhost:10000>
```

Query : Population, male SP.POP.TOTL.MA.IN

```
8 rows selected (2.444 seconds)
0: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value / 1000000) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'SP.POP.TOTL.MA.IN'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engi
+-------------------------+--------------------------+--------------+
| wdi_final.country_code  | wdi_final.indicator_code |   sum_value  |
+-------------------------+--------------------------+--------------+
| CHN                     | SP.POP.TOTL.MA.IN        | 15120.144663 |
| IND                     | SP.POP.TOTL.MA.IN        | 14155.057930 |
| USA                     | SP.POP.TOTL.MA.IN        | 3356.662648  |
| PAK                     | SP.POP.TOTL.MA.IN        | 2190.515186  |
| BGD                     | SP.POP.TOTL.MA.IN        | 1652.021194  |
| RUS                     | SP.POP.TOTL.MA.IN        | 1469.531369  |
| JPN                     | SP.POP.TOTL.MA.IN        | 1365.672933  |
| ARE                     | SP.POP.TOTL.MA.IN        | 110.039613   |
+-------------------------+--------------------------+--------------+
8 rows selected (2.431 seconds)
```

Query : Birth rate, crude (per 1,000 people) SP.DYN.CBRT.IN

```
24 rows selected (1.257 seconds)
0: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value / 1000000) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'SP.DYN.CBRT.IN'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution e
+-------------------------+--------------------------+------------+
| wdi_final.country_code  | wdi_final.indicator_code | sum_value  |
+-------------------------+--------------------------+------------+
| PAK                     | SP.DYN.CBRT.IN           | 0.000694   |
| BGD                     | SP.DYN.CBRT.IN           | 0.000489   |
| IND                     | SP.DYN.CBRT.IN           | 0.000469   |
| ARE                     | SP.DYN.CBRT.IN           | 0.000291   |
| USA                     | SP.DYN.CBRT.IN           | 0.000285   |
| CHN                     | SP.DYN.CBRT.IN           | 0.000268   |
| RUS                     | SP.DYN.CBRT.IN           | 0.000247   |
| JPN                     | SP.DYN.CBRT.IN           | 0.000183   |
+-------------------------+--------------------------+------------+
```

Query : Death rate, crude (per 1,000 people) SP.DYN.CDRT.IN

```
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value / 1000000) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'SP.DYN.CDRT.IN'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engi
+-------------------------+--------------------------+------------+
| wdi_final.country_code  | wdi_final.indicator_code | sum_value  |
+-------------------------+--------------------------+------------+
| RUS                     | SP.DYN.CDRT.IN           | 0.000316   |
| JPN                     | SP.DYN.CDRT.IN           | 0.000212   |
| USA                     | SP.DYN.CDRT.IN           | 0.000186   |
| PAK                     | SP.DYN.CDRT.IN           | 0.000168   |
| IND                     | SP.DYN.CDRT.IN           | 0.000167   |
| CHN                     | SP.DYN.CDRT.IN           | 0.000149   |
| BGD                     | SP.DYN.CDRT.IN           | 0.000136   |
| ARE                     | SP.DYN.CDRT.IN           | 0.000030   |
+-------------------------+--------------------------+------------+
```

Query : Life expectancy at birth, total (years) SP.DYN.LE00.IN

```
8 rows selected (2.683 seconds)
0: jdbc:hive2://localhost:10000> SELECT
. . . . . .> wdi_final.country_code,
. . . . . .> wdi_final.indicator_code,
. . . . . .> SUM(wdi_final.value) AS sum_value
. . . . . .> FROM
. . . . . .> wdi_final
. . . . . .> WHERE
. . . . . .> (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .> AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .> AND wdi_final.indicator_code = 'SP.DYN.LE00.IN'
. . . . . .> GROUP BY
. . . . . .> wdi_final.country_code,
. . . . . .> wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .> sum_value DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution
+-------------------------+--------------------------+-------------+
| wdi_final.country_code  | wdi_final.indicator_code |  sum_value  |
+-------------------------+--------------------------+-------------+
| JPN                     | SP.DYN.LE00.IN           | 1825.17779  |
| USA                     | SP.DYN.LE00.IN           | 1714.53898  |
| ARE                     | SP.DYN.LE00.IN           | 1711.33900  |
| CHN                     | SP.DYN.LE00.IN           | 1662.48900  |
| BGD                     | SP.DYN.LE00.IN           | 1517.88800  |
| RUS                     | SP.DYN.LE00.IN           | 1514.62439  |
| IND                     | SP.DYN.LE00.IN           | 1475.77100  |
| PAK                     | SP.DYN.LE00.IN           | 1419.04800  |
+-------------------------+--------------------------+-------------+
```

Query : Population growth (annual %) SP.POP.GROW

```
0: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'SP.POP.GROW'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution en
+-------------------------+--------------------------+------------+
| wdi_final.country_code  | wdi_final.indicator_code | sum_value  |
+-------------------------+--------------------------+------------+
| ARE                     | SP.POP.GROW              | 110.63791  |
| PAK                     | SP.POP.GROW              | 43.55606   |
| IND                     | SP.POP.GROW              | 30.21591   |
| BGD                     | SP.POP.GROW              | 28.97498   |
| USA                     | SP.POP.GROW              | 17.34598   |
| CHN                     | SP.POP.GROW              | 11.99329   |
| JPN                     | SP.POP.GROW              | -0.75254   |
| RUS                     | SP.POP.GROW              | -2.59108   |
+-------------------------+--------------------------+------------+
8 rows selected (2.659 seconds)
```

Query : CPIA gender equality rating (1=low to 6=high) IQ.CPA.GNDR.XQ

```
8 rows selected (2.659 seconds)
0: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'IQ.CPA.GNDR.XQ'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engin
+-------------------------+--------------------------+----------------------+
| wdi_final.country_code  | wdi_final.indicator_code |      sum_value       |
+-------------------------+--------------------------+----------------------+
| USA                     | IQ.CPA.GNDR.XQ           | 65203565949323.21190 |
| RUS                     | IQ.CPA.GNDR.XQ           | 65203565949323.21190 |
| JPN                     | IQ.CPA.GNDR.XQ           | 65203565949323.21190 |
| CHN                     | IQ.CPA.GNDR.XQ           | 65203565949323.21190 |
| ARE                     | IQ.CPA.GNDR.XQ           | 65203565949323.21190 |
| IND                     | IQ.CPA.GNDR.XQ           | 46239437514208.72510 |
| BGD                     | IQ.CPA.GNDR.XQ           | 5645750368614.35990  |
| PAK                     | IQ.CPA.GNDR.XQ           | 5645750368593.85990  |
+-------------------------+--------------------------+----------------------+
8 rows selected (2.538 seconds)
```

Query : Current health expenditure (% of GDP) SH.XPD.CHEX.GD.ZS

```
0: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'SH.XPD.CHEX.GD.ZS'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine
+-----------------------+-------------------------+----------------------+
| wdi_final.country_code | wdi_final.indicator_code |      sum_value       |
+-----------------------+-------------------------+----------------------+
| USA                   | SH.XPD.CHEX.GD.ZS       | 9594000591930.60251  |
| JPN                   | SH.XPD.CHEX.GD.ZS       | 9594000591796.42296  |
| RUS                   | SH.XPD.CHEX.GD.ZS       | 9594000591713.56243  |
| CHN                   | SH.XPD.CHEX.GD.ZS       | 9594000591699.14112  |
| IND                   | SH.XPD.CHEX.GD.ZS       | 9594000591677.97126  |
| ARE                   | SH.XPD.CHEX.GD.ZS       | 9594000591673.80759  |
| BGD                   | SH.XPD.CHEX.GD.ZS       | 9594000591657.67769  |
| PAK                   | SH.XPD.CHEX.GD.ZS       | 9594000591657.63435  |
```

Query : Employment to population ratio, ages 15-24, total (%) (national estimate) SL.EMP.1524.SP.NE.ZS

```
8 rows selected (2.665 seconds)
0: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'SL.EMP.1524.SP.NE.ZS'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine
+-----------------------+-------------------------+----------------------+
| wdi_final.country_code | wdi_final.indicator_code |       sum_value      |
+-----------------------+-------------------------+----------------------+
| CHN                   | SL.EMP.1524.SP.NE.ZS    | 65203565949323.21190 |
| BGD                   | SL.EMP.1524.SP.NE.ZS    | 51045850705256.66430 |
| ARE                   | SL.EMP.1524.SP.NE.ZS    | 34537423824301.60520 |
| IND                   | SL.EMP.1524.SP.NE.ZS    | 31244240417998.51830 |
| PAK                   | SL.EMP.1524.SP.NE.ZS    | 22869154866385.98120 |
| USA                   | SL.EMP.1524.SP.NE.ZS    | 1116.78000           |
| JPN                   | SL.EMP.1524.SP.NE.ZS    | 916.15000            |
| RUS                   | SL.EMP.1524.SP.NE.ZS    | 709.89000            |
+-----------------------+-------------------------+----------------------+
8 rows selected (2.516 seconds)
```

Query : Literacy rate, adult total (% of people ages 15 and above) SE.ADT.LITR.ZS

```
0: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'SE.ADT.LITR.ZS'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i
+-----------------------+-------------------------+----------------------+
| wdi_final.country_code | wdi_final.indicator_code |       sum_value      |
+-----------------------+-------------------------+----------------------+
| USA                   | SE.ADT.LITR.ZS          | 65203565949323.21190 |
| JPN                   | SE.ADT.LITR.ZS          | 65203565949323.21190 |
| IND                   | SE.ADT.LITR.ZS          | 55805407590522.09718 |
| CHN                   | SE.ADT.LITR.ZS          | 51950835734446.75848 |
| RUS                   | SE.ADT.LITR.ZS          | 51677202351565.19760 |
| ARE                   | SE.ADT.LITR.ZS          | 47145817084815.41999 |
| PAK                   | SE.ADT.LITR.ZS          | 27816622914549.69876 |
| BGD                   | SE.ADT.LITR.ZS          | 23226820150746.98275 |
+-----------------------+-------------------------+----------------------+
8 rows selected (2.636 seconds)
```

Query : Access to electricity (% of population) EG.ELC.ACCS.ZS

```
0: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'EG.ELC.ACCS.ZS'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution en
+-------------------------+---------------------------+-----------------------+
| wdi_final.country_code  | wdi_final.indicator_code  |       sum_value       |
+-------------------------+---------------------------+-----------------------+
| USA                     | EG.ELC.ACCS.ZS            | 9594000593703.51000   |
| JPN                     | EG.ELC.ACCS.ZS            | 9594000593703.51000   |
| ARE                     | EG.ELC.ACCS.ZS            | 9594000593703.51000   |
| RUS                     | EG.ELC.ACCS.ZS            | 9594000593699.01000   |
| CHN                     | EG.ELC.ACCS.ZS            | 9594000593684.11909   |
| IND                     | EG.ELC.ACCS.ZS            | 9594000593214.30087   |
| PAK                     | EG.ELC.ACCS.ZS            | 9594000593099.39778   |
| BGD                     | EG.ELC.ACCS.ZS            | 9594000592867.15676   |
+-------------------------+---------------------------+-----------------------+
8 rows selected (2.488 seconds)
```

Query : Unemployment, total (% of total labor force) (national estimate) SL.UEM.TOTL.NE.ZS

```
0: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'SL.UEM.TOTL.NE.ZS'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine
+-------------------------+---------------------------+-----------------------+
| wdi_final.country_code  | wdi_final.indicator_code  |       sum_value       |
+-------------------------+---------------------------+-----------------------+
| BGD                     | SL.UEM.TOTL.NE.ZS         | 45606979217361.95130  |
| IND                     | SL.UEM.TOTL.NE.ZS         | 31244240417818.45830  |
| ARE                     | SL.UEM.TOTL.NE.ZS         | 28681017481662.35860  |
| PAK                     | SL.UEM.TOTL.NE.ZS         | 15874415470369.47670  |
| CHN                     | SL.UEM.TOTL.NE.ZS         | 6798114323204.98200   |
| RUS                     | SL.UEM.TOTL.NE.ZS         | 144.07000             |
| USA                     | SL.UEM.TOTL.NE.ZS         | 131.04000             |
| JPN                     | SL.UEM.TOTL.NE.ZS         | 88.18000              |
+-------------------------+---------------------------+-----------------------+
8 rows selected (2.545 seconds)
0: jdbc:hive2://localhost:10000>
```

Query : Unemployment, youth total (% of total labor force ages 15-24) (national estimate) SL.UEM.1524.NE.ZS

```
8 rows selected (2.545 seconds)
0: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'SL.UEM.1524.NE.ZS'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.
+-------------------------+---------------------------+-----------------------+
| wdi_final.country_code  | wdi_final.indicator_code  |       sum_value       |
+-------------------------+---------------------------+-----------------------+
| BGD                     | SL.UEM.1524.NE.ZS         | 47607285268225.05570  |
| CHN                     | SL.UEM.1524.NE.ZS         | 38399942916186.23090  |
| ARE                     | SL.UEM.1524.NE.ZS         | 33188434797074.49060  |
| IND                     | SL.UEM.1524.NE.ZS         | 31244240417908.96830  |
| PAK                     | SL.UEM.1524.NE.ZS         | 19427460292256.15010  |
| RUS                     | SL.UEM.1524.NE.ZS         | 357.17000             |
| USA                     | SL.UEM.1524.NE.ZS         | 272.82000             |
| JPN                     | SL.UEM.1524.NE.ZS         | 160.91000             |
+-------------------------+---------------------------+-----------------------+
8 rows selected (2.557 seconds)
```

Query : Adjusted net national income (annual % growth) NY.ADJ.NNTY.KD.ZG

```
0: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'NY.ADJ.NNTY.KD.ZG'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution eng
+-------------------------+--------------------------+----------------------+
| wdi_final.country_code  | wdi_final.indicator_code |       sum_value      |
+-------------------------+--------------------------+----------------------+
| ARE                     | NY.ADJ.NNTY.KD.ZG        | 11554730015007.08638 |
| CHN                     | NY.ADJ.NNTY.KD.ZG        | 9594000591790.11317  |
| IND                     | NY.ADJ.NNTY.KD.ZG        | 9594000591733.29884  |
| BGD                     | NY.ADJ.NNTY.KD.ZG        | 9594000591721.73077  |
| RUS                     | NY.ADJ.NNTY.KD.ZG        | 9594000591699.94028  |
| PAK                     | NY.ADJ.NNTY.KD.ZG        | 9594000591672.64126  |
| USA                     | NY.ADJ.NNTY.KD.ZG        | 9594000591641.62392  |
| JPN                     | NY.ADJ.NNTY.KD.ZG        | 9594000591613.98828  |
+-------------------------+--------------------------+----------------------+
```

Query : Broad money (% of GDP) FM.LBL.BMNY.GD.ZS

```
8 rows selected (2.716 seconds)
0: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value/1000) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'FM.LBL.BMNY.GD.ZS'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine
+-------------------------+--------------------------+---------------------+
| wdi_final.country_code  | wdi_final.indicator_code |      sum_value      |
+-------------------------+--------------------------+---------------------+
| RUS                     | FM.LBL.BMNY.GD.ZS        | 10517125611.727054  |
| JPN                     | FM.LBL.BMNY.GD.ZS        | 4.926013            |
| CHN                     | FM.LBL.BMNY.GD.ZS        | 3.837134            |
| USA                     | FM.LBL.BMNY.GD.ZS        | 1.875283            |
| IND                     | FM.LBL.BMNY.GD.ZS        | 1.602345            |
| ARE                     | FM.LBL.BMNY.GD.ZS        | 1.496495            |
| BGD                     | FM.LBL.BMNY.GD.ZS        | 1.160877            |
| PAK                     | FM.LBL.BMNY.GD.ZS        | 1.086921            |
+-------------------------+--------------------------+---------------------+
8 rows selected (2.596 seconds)
```

Query : Central government debt, total (% of GDP) GC.DOD.TOTL.GD.ZS

```
+-------------------------+--------------------------+---------------------+
8 rows selected (2.596 seconds)
0: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value/1000) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'GC.DOD.TOTL.GD.ZS'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine
+-------------------------+--------------------------+---------------------+
| wdi_final.country_code  | wdi_final.indicator_code |      sum_value      |
+-------------------------+--------------------------+---------------------+
| CHN                     | GC.DOD.TOTL.GD.ZS        | 65203565949.323214  |
| PAK                     | GC.DOD.TOTL.GD.ZS        | 64280440930.201145  |
| ARE                     | GC.DOD.TOTL.GD.ZS        | 62225945603.470754  |
| BGD                     | GC.DOD.TOTL.GD.ZS        | 61802751752.523122  |
| IND                     | GC.DOD.TOTL.GD.ZS        | 22487517469.320833  |
| RUS                     | GC.DOD.TOTL.GD.ZS        | 2488262543.496102   |
| JPN                     | GC.DOD.TOTL.GD.ZS        | 3.559619            |
| USA                     | GC.DOD.TOTL.GD.ZS        | 1.758366            |
+-------------------------+--------------------------+---------------------+
```

Query : Central government debt, total (current LCU) GC.DOD.TOTL.CN

```
: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value/1000) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'GC.DOD.TOTL.CN'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
ARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spar
+-------------------------+--------------------------+-----------------------+
| wdi_final.country_code  | wdi_final.indicator_code |       sum_value       |
+-------------------------+--------------------------+-----------------------+
| JPN                     | GC.DOD.TOTL.CN           | 18849000000000.000000 |
| IND                     | GC.DOD.TOTL.CN           | 808687517468.281789   |
| USA                     | GC.DOD.TOTL.CN           | 298230000000.000000   |
| RUS                     | GC.DOD.TOTL.CN           | 188578262543.125929   |
| PAK                     | GC.DOD.TOTL.CN           | 67110440930.134392    |
| CHN                     | GC.DOD.TOTL.CN           | 65203565949.323214    |
| BGD                     | GC.DOD.TOTL.CN           | 64809751752.428592    |
| ARE                     | GC.DOD.TOTL.CN           | 62253073530.684908    |
+-------------------------+--------------------------+-----------------------+
 rows selected (2.544 seconds)
```

Query : Current account balance (% of GDP) BN.CAB.XOKA.GD.ZS

```
 rows selected (2.544 seconds)
0: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value/1000) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'BN.CAB.XOKA.GD.ZS'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
URNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine
+-------------------------+--------------------------+---------------------+
| wdi_final.country_code  | wdi_final.indicator_code |      sum_value      |
+-------------------------+--------------------------+---------------------+
| ARE                     | BN.CAB.XOKA.GD.ZS        | 65203565949.323214  |
| RUS                     | BN.CAB.XOKA.GD.ZS        | 0.135484            |
| CHN                     | BN.CAB.XOKA.GD.ZS        | 0.071904            |
| JPN                     | BN.CAB.XOKA.GD.ZS        | 0.065017            |
| BGD                     | BN.CAB.XOKA.GD.ZS        | 0.005110            |
| IND                     | BN.CAB.XOKA.GD.ZS        | -0.027219           |
| PAK                     | BN.CAB.XOKA.GD.ZS        | -0.040775           |
| USA                     | BN.CAB.XOKA.GD.ZS        | -0.075168           |
+-------------------------+--------------------------+---------------------+
```

Query : Expense (% of GDP) GC.XPN.TOTL.GD.ZS

```
: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value/1000) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'GC.XPN.TOTL.GD.ZS'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
ARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine
+-------------------------+--------------------------+---------------------+
| wdi_final.country_code  | wdi_final.indicator_code |      sum_value      |
+-------------------------+--------------------------+---------------------+
| CHN                     | GC.XPN.TOTL.GD.ZS        | 65203565949.323214  |
| PAK                     | GC.XPN.TOTL.GD.ZS        | 64280440930.149889  |
| IND                     | GC.XPN.TOTL.GD.ZS        | 22487517468.579489  |
| ARE                     | GC.XPN.TOTL.GD.ZS        | 16437505125.935863  |
| BGD                     | GC.XPN.TOTL.GD.ZS        | 923125019.368952    |
| RUS                     | GC.XPN.TOTL.GD.ZS        | 0.571152            |
| USA                     | GC.XPN.TOTL.GD.ZS        | 0.506122            |
| JPN                     | GC.XPN.TOTL.GD.ZS        | 0.375335            |
+-------------------------+--------------------------+---------------------+
 rows selected (2.584 seconds)
```

Query : Expense (current LCU) GC.XPN.TOTL.CN

```
8 rows selected (2.584 seconds)
0: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'GC.XPN.TOTL.CN'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution e
+-------------------------+--------------------------+-------------------------+
| wdi_final.country_code  | wdi_final.indicator_code |        sum_value        |
+-------------------------+--------------------------+-------------------------+
| JPN                     | GC.XPN.TOTL.CN           | 1983100000000000.00000  |
| RUS                     | GC.XPN.TOTL.CN           | 355840000000000.00000   |
| IND                     | GC.XPN.TOTL.CN           | 261397517468281.78900   |
| USA                     | GC.XPN.TOTL.CN           | 82420000000000.00000    |
| CHN                     | GC.XPN.TOTL.CN           | 65203565949323.21190    |
| PAK                     | GC.XPN.TOTL.CN           | 64938440930134.38990    |
| BGD                     | GC.XPN.TOTL.CN           | 24280125019188.82200    |
| ARE                     | GC.XPN.TOTL.CN           | 17097223051949.64070    |
+-------------------------+--------------------------+-------------------------+
```

Query : Fossil fuel energy consumption (% of total) EG.USE.COMM.FO.ZS

```
8 rows selected (2.654 seconds)
0: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'EG.USE.COMM.FO.ZS'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution er
+-------------------------+--------------------------+-----------------------+
| wdi_final.country_code  | wdi_final.indicator_code |       sum_value       |
+-------------------------+--------------------------+-----------------------+
| ARE                     | EG.USE.COMM.FO.ZS        | 41699304501057.25450  |
| RUS                     | EG.USE.COMM.FO.ZS        | 37464256449722.51834  |
| CHN                     | EG.USE.COMM.FO.ZS        | 37464256449646.93512  |
| IND                     | EG.USE.COMM.FO.ZS        | 37464256449385.01171  |
| BGD                     | EG.USE.COMM.FO.ZS        | 37464256449359.41728  |
| PAK                     | EG.USE.COMM.FO.ZS        | 37464256449259.88615  |
| JPN                     | EG.USE.COMM.FO.ZS        | 34250434422800.43702  |
| USA                     | EG.USE.COMM.FO.ZS        | 34250434422788.97668  |
+-------------------------+--------------------------+-----------------------+
8 rows selected (2.477 seconds)
```

Query : GDP growth (annual %) NY.GDP.MKTP.KD.ZG

```
8 rows selected (2.477 seconds)
0: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'NY.GDP.MKTP.KD.ZG'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine
+-------------------------+--------------------------+------------+
| wdi_final.country_code  | wdi_final.indicator_code | sum_value  |
+-------------------------+--------------------------+------------+
| CHN                     | NY.GDP.MKTP.KD.ZG        | 190.63632  |
| BGD                     | NY.GDP.MKTP.KD.ZG        | 131.98485  |
| IND                     | NY.GDP.MKTP.KD.ZG        | 131.16232  |
| PAK                     | NY.GDP.MKTP.KD.ZG        | 90.91655   |
| ARE                     | NY.GDP.MKTP.KD.ZG        | 85.87208   |
| RUS                     | NY.GDP.MKTP.KD.ZG        | 77.25373   |
| USA                     | NY.GDP.MKTP.KD.ZG        | 44.87740   |
| JPN                     | NY.GDP.MKTP.KD.ZG        | 13.93812   |
+-------------------------+--------------------------+------------+
8 rows selected (2.504 seconds)
0: jdbc:hive2://localhost:10000>
```

Query : Tax revenue (% of GDP) GC.TAX.TOTL.GD.ZS

```
8 rows selected (2.504 seconds)
0: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'GC.TAX.TOTL.GD.ZS'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine
+-------------------------+---------------------------+-----------------------+
| wdi_final.country_code  | wdi_final.indicator_code  |       sum_value       |
+-------------------------+---------------------------+-----------------------+
| JPN                     | GC.TAX.TOTL.GD.ZS         | 65203565949323.21190  |
| PAK                     | GC.TAX.TOTL.GD.ZS         | 64280440930143.48677  |
| IND                     | GC.TAX.TOTL.GD.ZS         | 22487517468477.65008  |
| ARE                     | GC.TAX.TOTL.GD.ZS         | 16437500125893.43575  |
| CHN                     | GC.TAX.TOTL.GD.ZS         | 15239750960309.60197  |
| BGD                     | GC.TAX.TOTL.GD.ZS         | 923125019348.35624    |
| RUS                     | GC.TAX.TOTL.GD.ZS         | 290.52510             |
| USA                     | GC.TAX.TOTL.GD.ZS         | 229.42049             |
+-------------------------+---------------------------+-----------------------+
8 rows selected (2.519 seconds)
```

Query : Gross savings (% of GDP) NY.GNS.ICTR.ZS

```
0: jdbc:hive2://localhost:10000> SELECT
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code,
. . . . . .>     SUM(wdi_final.value/1000) AS sum_value
. . . . . .> FROM
. . . . . .>     wdi_final
. . . . . .> WHERE
. . . . . .>     (wdi_final.year >= 2000 AND wdi_final.year <= 2022)
. . . . . .>     AND wdi_final.country_code IN ('USA', 'CHN', 'IND', 'PAK', 'BGD', 'KOR', 'JPN', 'ARE', 'RUS')
. . . . . .>     AND wdi_final.indicator_code = 'NY.GNS.ICTR.ZS'
. . . . . .> GROUP BY
. . . . . .>     wdi_final.country_code,
. . . . . .>     wdi_final.indicator_code
. . . . . .> ORDER BY
. . . . . .>     sum_value DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine
+-------------------------+---------------------------+---------------------+
| wdi_final.country_code  | wdi_final.indicator_code  |      sum_value      |
+-------------------------+---------------------------+---------------------+
| ARE                     | NY.GNS.ICTR.ZS            | 65203565949.323214  |
| CHN                     | NY.GNS.ICTR.ZS            | 1.002627            |
| BGD                     | NY.GNS.ICTR.ZS            | 0.778399            |
| IND                     | NY.GNS.ICTR.ZS            | 0.713278            |
| JPN                     | NY.GNS.ICTR.ZS            | 0.619713            |
| RUS                     | NY.GNS.ICTR.ZS            | 0.618715            |
| USA                     | NY.GNS.ICTR.ZS            | 0.402552            |
| PAK                     | NY.GNS.ICTR.ZS            | 0.357279            |
+-------------------------+---------------------------+---------------------+
8 rows selected (2.512 seconds)
```

*These were just some of the queries that were run. The aggregate technique is not ideal but we also ran queries **without** the aggregate, it was just taking up too much screen space.*

Queries were then also run on the following indicators:

Query	Gross savings (% of GDP) NY.GNS.ICTR.ZS
Query	Inflation, consumer prices (annual %) FP.CPI.TOTL.ZG
Query	Net primary income (Net income from abroad) (current US$) Y.GSR.NFCY.CD
Query	Adjusted savings: education expenditure (current US$) NY.ADJ.AEDU.CD
Query	Adolescents out of school (% of lower secondary school age) SE.SEC.UNER.LO.ZS
Query	Children in employment, total (% of children ages 7-14) SL.TLF.0714.ZS
Query	Children out of school (% of primary school age) SE.PRM.UNER.ZS
Query	Compulsory education, duration (years) SE.COM.DURS
Query	Educational attainment, at least Bachelor's or equivalent, population 25+, total (%) (cumulative) SE.TER.CUAT.BA.ZS
Query	Government expenditure on education, total (% of GDP) .XPD.TOTL.GD.ZS

Dashboard Visualization

Data Set

The data used in the creation of the visualization on the dashboard was derived from the World Bank *(Indicators | Data, 2023)* *(World Development Indicators | DataBank, 2023)*. The complete data was too heavy to run hence for the visualisation, specific financial, educational, and demographic statistics/indicators were chosen which were compiled into a csv. The dataset covers the years 2000 through 2022 and offers a wide range of detailed information. This data set is an invaluable tool for understanding the complexities of the economy since it gives us access to a variety of specific information. As a result of the dataset's foundational role in the creation of prediction models, we are also able to generate accurate predictions based on the extensive data it contains.

```
file_path = r"F:\Urooj\Masters\IBA\BDA\project\combine.csv"
if os.path.isfile(file_path):
    df = pd.read_csv(file_path)
else:
    st.error("File not found. Please check the file path.")
```

Data Preprocessing

The first step is to define a list called "columns_with_missing_values," which comprises the names of the columns that could have missing data. The code then substitutes all instances of '-' in the given columns with NaN (Not a Number), which is a standard way to indicate missing data. This process enables uniform treatment of missing data throughout the whole dataset. The columns with missing values are then converted to a numeric data type in the following lines of code to ensure compatibility with further computations or analysis.

The "columns_with_missing_values" list of columns is iterated over in a loop to handle the missing values. The Pandas library's.mean() method is used to calculate the mean value for each column. The.fillna() function, which alters the original DataFrame, is then used to fill in the missing values in the corresponding column using this mean value.

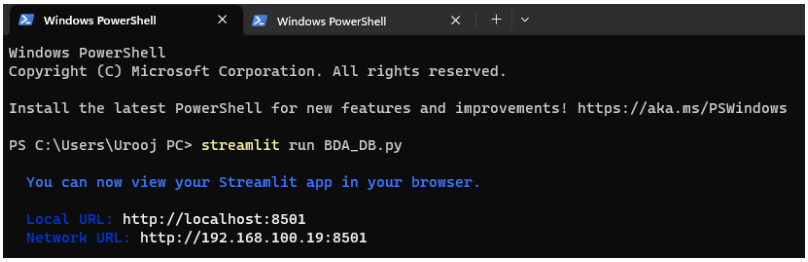
Column Year 2022 was dropped due to insufficient data.

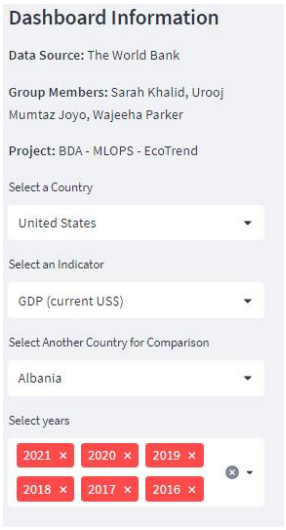
```
columns_with_missing_values = ["2000", "2001", "2002", "2003", "2004", "2005", "2006", "2007", "2008", "2009",
                               "2010", "2011", "2012", "2013", "2014", "2015", "2016", "2017", "2018", "2019",
                               "2020", "2021", "2022"]  # Replace with your column names

# Replace '-' values with NaN
df[columns_with_missing_values] = df[columns_with_missing_values].replace('-', np.nan)

# Convert the columns with missing values to numeric type
df[columns_with_missing_values] = df[columns_with_missing_values].astype(float)

# Fill missing values with the respective mean values
for column in columns_with_missing_values:
    mean_value = df[column].mean()
    df[column].fillna(mean_value, inplace=True)

df = df.drop(columns='2022')
```

StreamLit

A user-friendly interface for studying and comparing the development and economic indicators of various nations using World Bank data is offered by streamlit-based dashboards. To acquire an understanding of these indicators, users can make use of two visualisation options: a bar chart and a line graph.

Streamlit app was run using powershell





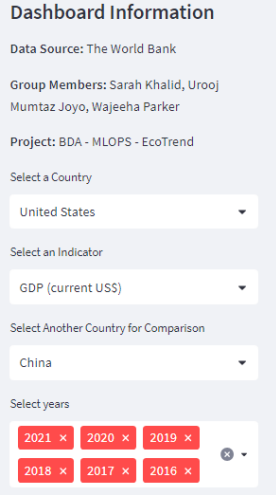
World Development Analysis

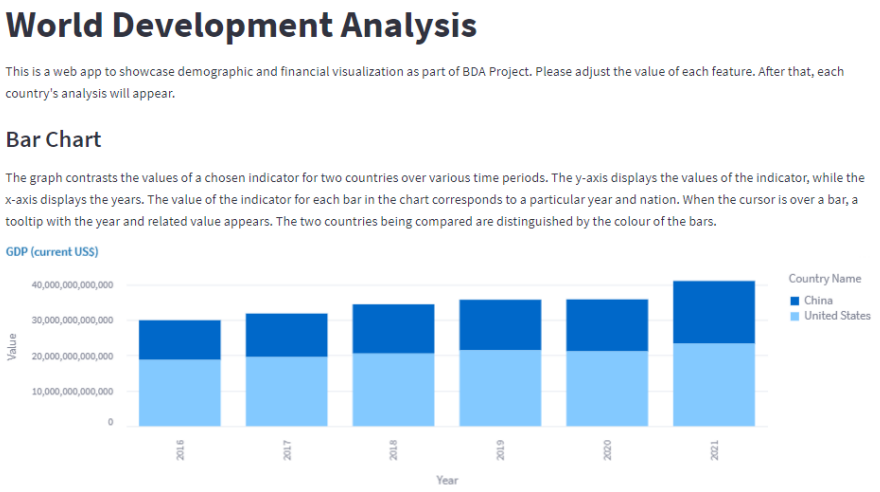
This is a web app to showcase demographic and financial visualization as part of BDA Project. Please adjust the value of each feature. After that, each country's analysis will appear.

Bar Chart

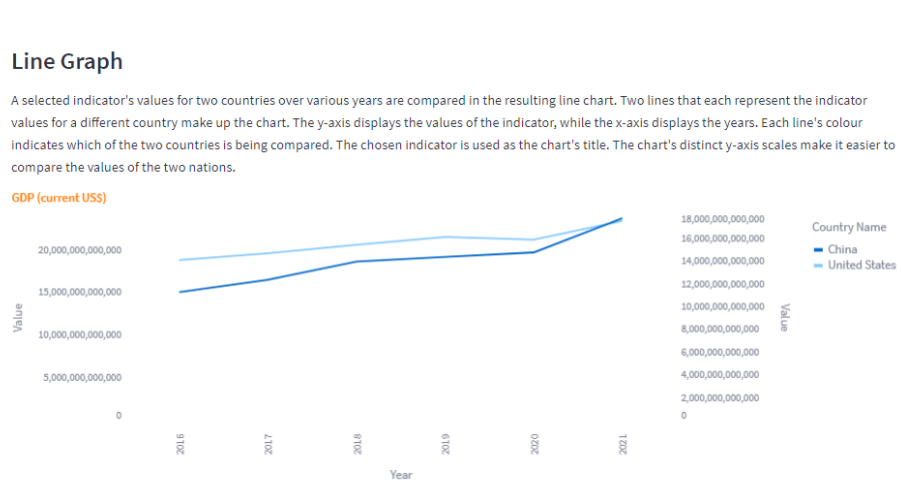
The graph contrasts the values of a chosen indicator for two countries over various time periods. The y-axis displays the values of the indicator, while the x-axis displays the years. The value of the indicator for each bar in the chart corresponds to a particular year and nation. When the cursor is over a bar, a tooltip with the year and related value appears. The two countries being compared are distinguished by the colour of the bars.

The Streamlit and Altair libraries were used to create interactive visualization. Here, users will be able to choose a particular country, indicator, and years to compare and visualize data from a particular dataset. There is an option to select countries and indicators in the sidebar on the left. Users may pick a primary nation for study, an indicator to evaluate it and can even choose to select a secondary country for comparison purposes. The selections are then used to filter the dataset using the code. Two charts—a bar chart and a line chart—make up the visualisations.



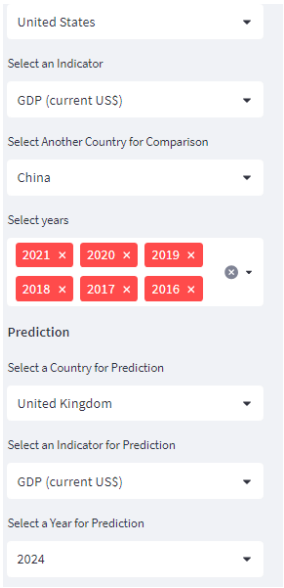


The selected indicator's values for the specified countries across various years are shown in the bar chart. Each bar refers to a particular year, and the height of the bar reflects the indicator's value. The bars' colours distinguish between the two contrasting nations.



The line graph shows how the indicator values have changed over time in the two countries. The y-axis shows the indicator values, the x-axis shows the years, and each line represents a nation. Each line's distinctive colour helps to identify between the two nations. The graphic makes it easy to compare the indicator values between the two nations by using distinct y-axis (secondary axis) scales for each line.

Example: The bar chart allows for a clear comparison of the GDP values for the specified time period between the United States and China, focusing on the GDP (current $) trend of both countries during the last 6 years. The amazing increasing trend of China's GDP, which has overtaken that of the United States, is one striking finding from the visualisations. This pattern demonstrates how rapidly China's economy has expanded in recent years.




```
features = {
    'Country Code': LabelEncoder().fit_transform([selected_country])[0],
    'Indicator Code': LabelEncoder().fit_transform([selected_indicator])[0],
    'Year': int(selected_year)
}

st.write("Selected Features:")
st.write(pd.DataFrame([features]))

col1, col2 = st.columns((1, 2))

with col1:
    prButton = st.button('Predict')

with col2:
    if prButton:
        features_df = pd.DataFrame([features])
        prediction = reg.predict(features_df)
        st.write('Based on the selected features, the predicted value is: {}'.format(int(prediction)))
```

Regression models that are able to forecast indicators for six years were included to further improve the functioning of our display. Users can, for instance, predict the GDP of the United Kingdom for 2024. The regression model, powered by encoded parameters like country code and indicator code, produces precise predictions when the desired nation, indicator, and desired year are selected. When the 'Predict' button is clicked, the model receives the specified features and makes a forecast based on the historical data. Based on anticipated indicator values, these forecasts enable users to make well-informed judgements and projections.

Docker

A docker file and requirement text file was made to build the image and then container. However, multiple times an error was surfaced due to which the container for streamlit app on docker could not be made.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Urooj PC> docker build -t streamlit-app -f "C:\Users\Urooj PC\Dockerfile.txt" .
[+] Building 60.2s (6/8)
 => [internal] load .dockerignore                                                0.0s
 => => transferring context: 2B                                                  0.0s
 => [internal] load build definition from Dockerfile.txt                         0.1s
 => => transferring dockerfile: 360B                                             0.0s
 => [internal] load metadata for docker.io/library/python:3.10.6                 0.0s
 => [1/4] FROM docker.io/library/python:3.10.6                                   0.0s
 => CANCELED [internal] load build context                                      58.7s
 => => transferring context: 467.55MB                                           58.6s
 => CACHED [2/4] WORKDIR /app                                                    0.0s
ERROR: failed to solve: error from sender: context canceled
PS C:\Users\Urooj PC>
```

Dockerfile

File Edit View

```
FROM python:3.10.6

WORKDIR /app

COPY . .

RUN pip install --no-cache-dir -r requirements.txt

EXPOSE 8501

HEALTHCHECK --interval=30s --timeout=3s CMD curl --fail http://localhost:8501/_stcore/health || exit 1

ENTRYPOINT ["streamlit", "run", "BDA2.py", "--server.port=8501", "--server.address=0.0.0.0"]
```

requirements

File Edit View

```
streamlit
pandas
altair
```

Stream lit Code.

```python
import pandas as pd
import matplotlib.pyplot as plt
import time

from sklearn.preprocessing import LabelEncoder
import streamlit as st
from sklearn.ensemble import GradientBoostingRegressor, VotingRegressor
from xgboost import XGBRegressor
import lightgbm as lgb
import warnings
warnings.filterwarnings('ignore')

from numpy import mean
import numpy as np
from mpl_toolkits.mplot3d import Axes3D


import streamlit as st
import pickle

import os
from PIL import Image
import streamlit.components.v1 as components
import io
import altair as alt
```

```
file_path = r"F:\Urooj\Masters\IBA\BDA\project\combine.csv"
if os.path.isfile(file_path):
    df = pd.read_csv(file_path)
else:
    st.error("File not found. Please check the file path.")

columns_with_missing_values = ["2000", "2001", "2002", "2003", "2004", "2005", "2006", "2007", "2008", "2009",
                               "2010", "2011", "2012", "2013", "2014", "2015", "2016", "2017", "2018", "2019",
                               "2020", "2021", "2022"]  # Replace with your column names

# Replace '-' values with NaN
df[columns_with_missing_values] = df[columns_with_missing_values].replace('-', np.nan)

# Convert the columns with missing values to numeric type
df[columns_with_missing_values] = df[columns_with_missing_values].astype(float)

# Fill missing values with the respective mean values
for column in columns_with_missing_values:
    mean_value = df[column].mean()
    df[column].fillna(mean_value, inplace=True)

df = df.drop(columns='2022')

st.set_page_config(layout="wide")
image = Image.open(r"F:\Urooj\Masters\IBA\BDA\project\1601738239.jpg")

st.image(image)
st.title('World Development Analysis')
st.write('This is a web app to showcase demographic and financial visualization as part of BDA Project. Please adjust the val

df.columns = df.columns.str.strip()

st.sidebar.title("Dashboard Information")
#st.sidebar.markdown("**Created by:** Urooj Mumtaz Joyo")
st.sidebar.markdown("**Data Source:** The World Bank")
st.sidebar.markdown("**Group Members:** Sarah Khalid, Urooj Mumtaz Joyo, Wajeeha Parker")
st.sidebar.markdown("**Project:** BDA - MLOPS - EcoTrend")
```

```
with st.sidebar:
    # Create a list of country names
    countries = df['Country Name'].unique().tolist()
    indicators = df['Indicator Name'].unique().tolist()
    # Country selection
    selected_country = st.selectbox('Select a Country', countries)
    selected_indicator = st.selectbox('Select an Indicator', indicators)
    selected_country2 = st.selectbox('Select Another Country for Comparison', countries)
    # Filter the data based on the selected country and indicator column
    filtered_df = df[
        ((df['Country Name'] == selected_country) |
        (df['Country Name'] == selected_country2)) &
        (df['Indicator Name'] == selected_indicator)]
    # Years column start from column 5 according to python 4
    years = filtered_df.columns[4:].tolist()
    # Year selection (multiple)
    selected_years = st.multiselect('Select years', years)
    # Filter the data based on the selected years
    filtered_df = filtered_df[['Country Name'] + selected_years]

# Melt the dataframe to reshape it
melted_df = pd.melt(filtered_df, id_vars='Country Name', var_name='Year', value_name='Value')

# Bar Chart
bar_chart = alt.Chart(melted_df).mark_bar().encode(
    x='Year',
    y='Value',
    tooltip=['Year', 'Value'],
    color=alt.Color('Country Name:N')
).properties(
    width=alt.Step(20)  # Set the width of the bars
).configure_title(
    fontSize=14,
    anchor='start',
    color='#1f77b4',
    offset=10
).properties(
    title=selected_indicator,
    width=500,
    height=300
)
```

```
filtered_df_primary = melted_df[melted_df['Country Name'] == selected_country]
filtered_df_secondary = melted_df[melted_df['Country Name'] == selected_country2]
# Primary line chart
line_chart_primary = alt.Chart(filtered_df_primary).mark_line().encode(
    X='Year',
    y='Value',
    color=alt.Color('Country Name:N')  # Set the color of the primary line
)

# Filter the data for the secondary line chart
#filtered_df2 = melted_df[melted_df['Country Name'] == selected_country2]

# Secondary line chart
line_chart_secondary = alt.Chart(filtered_df_secondary).mark_line().encode(
    X='Year',
    y='Value',
    color=alt.Color('Country Name:N')  # Set the color of the secondary line
)

# Combine the line charts
line_chart_combined = alt.layer(line_chart_primary, line_chart_secondary).resolve_scale(
    y='independent'  # Use independent scales for the y-axis
)

# Configure the chart properties
line_chart_combined = line_chart_combined.properties(
    width=alt.Step(20),  # Set the width of the line
    title=selected_indicator
).configure_title(
    fontSize=14,
    anchor='start',
    color='#ff7f0e',
    offset=10
)

st.subheader('Bar Chart')
st.write('The graph contrasts the values of a chosen indicator for two countries over various time periods. The y-axis displa
st.altair_chart(bar_chart, use_container_width=True)

st.subheader('Line Graph')
st.write('A selected indicator\'s values for two countries over various years are compared in the resulting line chart. Two l
st.altair_chart(line_chart_combined, use_container_width=True)
```

```
st.title('Prediction')

st.sidebar.markdown('**Prediction**')
st.write('Please select the country, indicator and year for which you want to predict.')

with st.sidebar:
    # Create a list of country codes and indicator codes
    countries = df['Country Code'].unique().tolist()
    indicators = df['Indicator Code'].unique().tolist()

    # Country code to name mapping
    country_names = df[['Country Code', 'Country Name']].drop_duplicates().set_index('Country Code')['Country Name'].to_dict(

    # Indicator code to name mapping
    indicator_names = df[['Indicator Code', 'Indicator Name']].drop_duplicates().set_index('Indicator Code')['Indicator Name'

    years = ['2022', '2023', '2024', '2025', '2026', '2027']
    # Country selection
    selected_country = st.selectbox('Select a Country for Prediction', countries, format_func=lambda x: country_names[x])
    selected_indicator = st.selectbox('Select an Indicator for Prediction', indicators, format_func=lambda x: indicator_names
    selected_year = st.selectbox('Select a Year for Prediction', years)

df = df[df["Indicator Code"].isin([selected_indicator])]  # Changed from 'Indicator Name'
id_columns = ['Country Name', 'Country Code', 'Indicator Name', 'Indicator Code']  # Changed from 'Indicator Name'
df = pd.melt(df, id_vars=id_columns, var_name='Year', value_name='Value')
df = df.sort_values(['Country Name', 'Year'])
df = df.reset_index(drop=True)
df = df[df["Country Code"].isin([selected_country])]

X = df.drop(columns=['Country Name', 'Indicator Name', 'Value'])  # Changed from 'Indicator Name'
X['Indicator Code'] = LabelEncoder().fit_transform(X['Indicator Code'])  # Encode the Indicator Code
X['Country Code'] = LabelEncoder().fit_transform(X['Country Code'])
X = X.astype(int)
y = df[['Value']]

Xt = X.copy()
Xt['Year'] = int(selected_year)

gb = GradientBoostingRegressor(max_depth=5, max_features=4, n_estimators=2300, learning_rate=0.056, random_state=0)
xgb = XGBRegressor(n_estimators=1300, learning_rate=0.04, n_jobs=-1)
lgbm = lgb.LGBMRegressor(learning_rate=0.09, n_estimators=5800, max_depth=4)
vot = VotingRegressor([('gb', gb), ('xgb', xgb), ('lgb', lgbm)])

reg = vot.fit(X, y)
```

```
features = {
    'Country Code': LabelEncoder().fit_transform([selected_country])[0],
    'Indicator Code': LabelEncoder().fit_transform([selected_indicator])[0],
    'Year': int(selected_year)
}

st.write("Selected Features:")
st.write(pd.DataFrame([features]))

col1, col2 = st.columns((1, 2))

with col1:
    prButton = st.button('Predict')

with col2:
    if prButton:
        features_df = pd.DataFrame([features])
        prediction = reg.predict(features_df)
        st.write('Based on the selected features, the predicted value is: {}'.format(int(prediction)))

# Line Chart
st.write("Line Chart:")
selected_country_data = df[df['Country Code'] == selected_country]
selected_country_data = selected_country_data[selected_country_data['Indicator Code'] == selected_indicator]  # Changed from
selected_country_data = selected_country_data[selected_country_data['Year'].astype(int) <= int(selected_year)]
line_chart = pd.DataFrame(selected_country_data[['Year', 'Value']])
line_chart = line_chart.set_index('Year')
st.line_chart(line_chart)
```

Conclusion

The aim of Ecotrend project is to develop a data analysis and machine learning pipeline using PySpark to perform regression techniques on economic and financial data. The project involved four key components: Redis, data_injestion, spark-master, and spark_ml containers.

The Redis container acts as the database and provides a convenient solution for accessing the datasets. The data_injestion container is responsible for fetching data from csv files, processing it, and storing it in Redis. The spark-master container acts as the control unit for the Spark cluster. Whereas the spark_ml container applies machine learning techniques to the data which involves data preprocessing, feature encoding, model training using Spark's Gradient Boosting Regressor, and model evaluation. Additionally, this trained model was saved using pickle for future visualization.

Moreover, the provided visualisations provide an easy approach to examine and contrast data from various nations, indicators, and time periods. The line chart depicts trends and changes in indicator values over time for the chosen nations, while the bar chart shows a quick comparison of indicator values for certain years.

In conclusion, the Ecotrend project develops a scalable and efficient data analysis and machine learning pipeline. The project uses Spark that enables the processing of large-scale datasets and facilitated the regression models for predicting economic trends.

Links

Project Link: BDA Project 2023

Video Link: EcoTrend - Sarah, Urooj, Wajeeha.mp4

Data Link: Data Set

Data Cleaning & Spark: Data Cleaning and SparkML

Data Analysis: Data EDA + Preprocessing + HIVE Analytics (Using HDFS)

Data Visualization: Data Visualization

References

Indicators | *Data*. (2023). Retrieved from World Bank Open Data | Data: https://data.worldbank.org/indicator

World Development Indicators | *DataBank*. (2023). Retrieved from World Bank Group - International Development, Poverty, & Sustainability: https://databank.worldbank.org/source/world-development-indicators