



# ETL Workshop 003 - Machine Learning (Airflow + Kafka)

by [DCajiao](#)



[Code Repository](#)

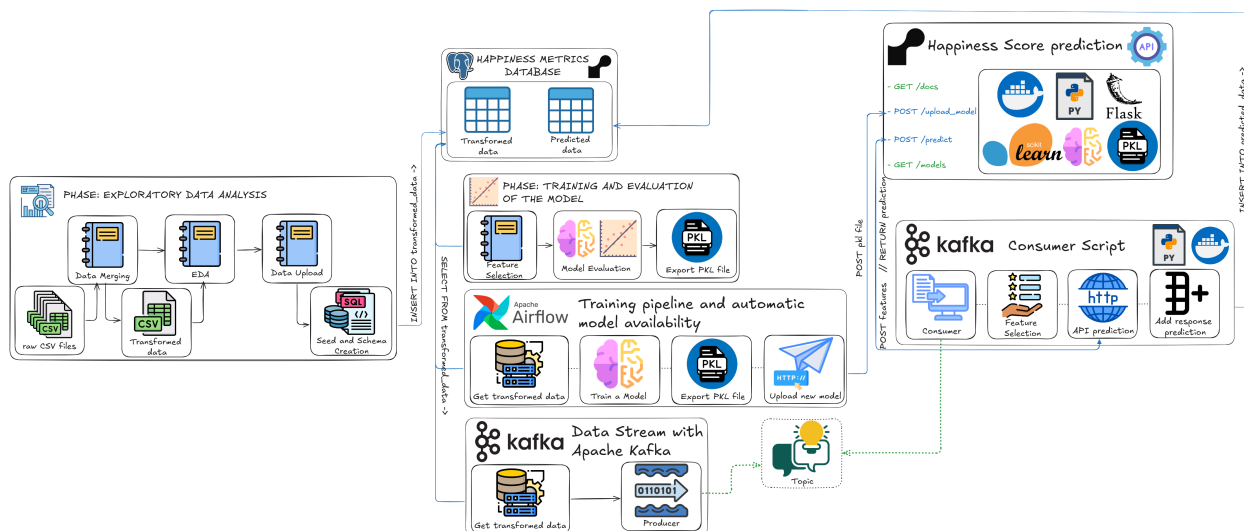


[Canva Presentation](#)

## Overview

This project focuses on building a machine learning pipeline to predict happiness scores for various countries using data from multiple CSV files. It involves conducting exploratory data analysis (EDA) and feature engineering, training a regression model with a 70-30 training-testing split, and integrating data streaming using Apache Kafka. The streaming system processes transformed data, which is consumed to make predictions with the trained model. The predictions, along with their corresponding inputs, are stored in a database. The project's performance is evaluated using metrics derived from testing data and predicted outcomes. Key technologies utilized include Python, Jupyter Notebook, Scikit-learn, Kafka, and database integration.

## Diagram of this project



## PHASE: Exploratory Data Analysis

## Merge

	Year 2015	Year 2016	Year 2017	Year 2018	Year 2019
0	Country	Country	Country	Overall rank	Overall rank
1	Region	Region	Happiness.Rank	Country or region	Country or region
2	Happiness Rank	Happiness Rank	Happiness.Score	Score	Score
3	Happiness Score	Happiness Score	Whisker.high	GDP per capita	GDP per capita
4	Standard Error	Lower Confidence Interval	Whisker.low	Social support	Social support
5	Economy (GDP per Capita)	Upper Confidence Interval	Economy..GDP.per.Capita.	Healthy life expectancy	Healthy life expectancy
6	Family	Economy (GDP per Capita)	Family	Freedom to make life choices	Freedom to make life choices
7	Health (Life Expectancy)	Family	Health..Life.Expectancy.	Generosity	Generosity
8	Freedom	Health (Life Expectancy)	Freedom	Perceptions of corruption	Perceptions of corruption
9	Trust (Government Corruption)	Freedom	Generosity	NaN	NaN
10	Generosity	Trust (Government Corruption)	Trust..Government.Corruption.	NaN	NaN
11	Dystopia Residual	Generosity	Dystopia.Residual	NaN	NaN
12	NaN	Dystopia Residual	NaN	NaN	NaN

As we can see, we did not have the same columns in all the datasets, so it was necessary to standardize those that were common, and eliminate those that were not present in all the records.

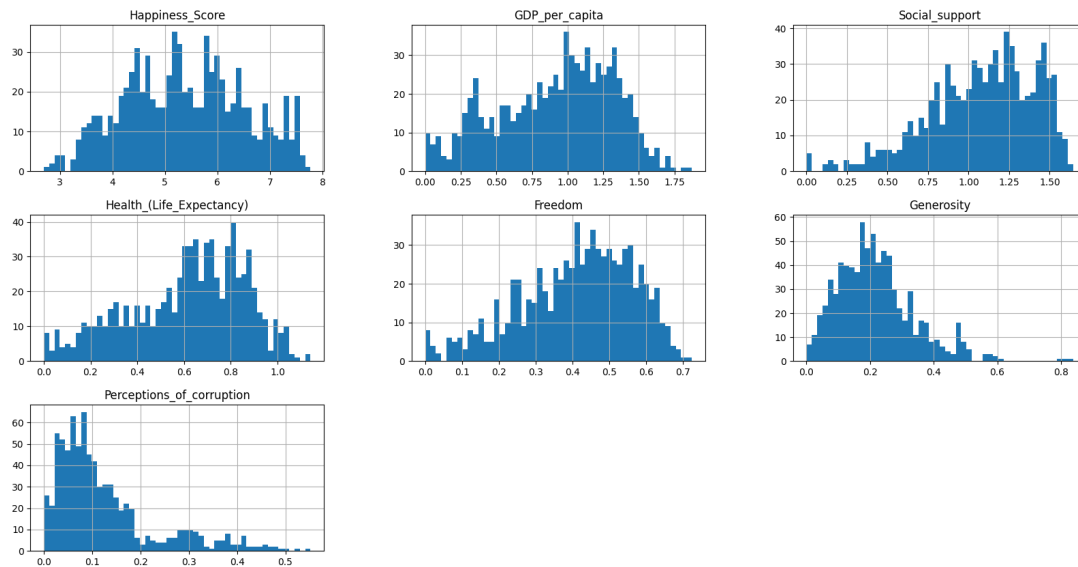
	Year 2015	Year 2016	Year 2017	Year 2018	Year 2019
0	Happiness_Rank	Happiness_Rank	Happiness_Rank	Happiness_Rank	Happiness_Rank
1	Country	Country	Country	Country	Country
2	Happiness_Score	Happiness_Score	Happiness_Score	Happiness_Score	Happiness_Score
3	GDP_per_capita	GDP_per_capita	GDP_per_capita	GDP_per_capita	GDP_per_capita
4	Social_support	Social_support	Social_support	Social_support	Social_support
5	Health_(Life_Expectancy)	Health_(Life_Expectancy)	Health_(Life_Expectancy)	Health_(Life_Expectancy)	Health_(Life_Expectancy)
6	Freedom	Freedom	Freedom	Freedom	Freedom
7	Generosity	Generosity	Generosity	Generosity	Generosity
8	Perceptions_of_corruption	Perceptions_of_corruption	Perceptions_of_corruption	Perceptions_of_corruption	Perceptions_of_corruption
9	Year	Year	Year	Year	Year

At this point I kept these standardized columns in order to proceed with a concatenation of the datasets after the aggregation of the Year column.

	Column	Data Type	Missing Values	Unique Values	Duplicates	Missing Values (%)
0	Happiness_Rank	int64	0	158	623	0.0
1	Country	object	0	170	611	0.0
2	Happiness_Score	float64	0	715	66	0.0
3	GDP_per_capita	float64	0	741	40	0.0
4	Social_support	float64	0	731	50	0.0
5	Health_(Life_Expectancy)	float64	0	704	77	0.0
6	Freedom	float64	0	696	85	0.0
7	Generosity	float64	0	663	118	0.0
8	Perceptions_of_corruption	float64	0	635	146	0.0
9	Year	int64	0	5	776	0.0

Finally, these are the characteristics of the merged dataset.

## EDA



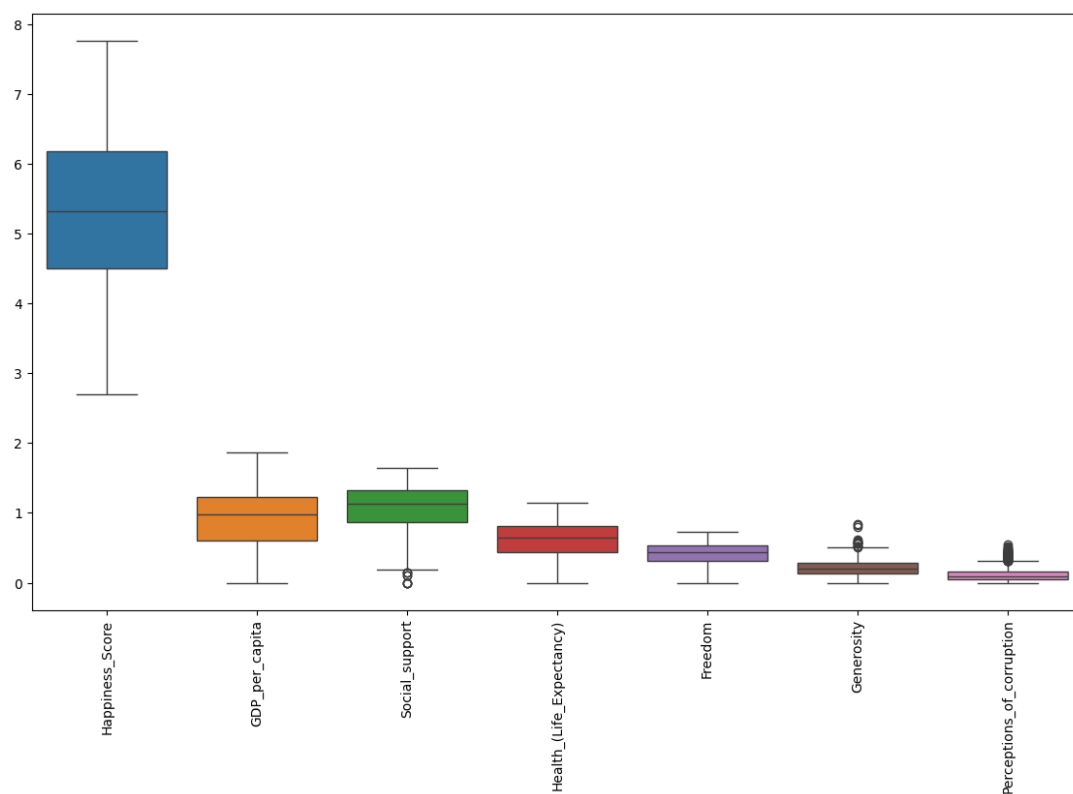
- **Happiness\_Score:**
  - Most countries have happiness scores between 4 and 7.
  - There are few countries with very low (<4) or very high (>7) scores, suggesting that happiness is distributed in the medium-high range for most countries.
- **GDP\_per\_capita:**
  - The distribution is skewed to the right, with most countries having GDP per capita between 0.5 and 1.5.
  - There are few countries with extremely low or extremely high GDP per capita, indicating that most have moderate per capita income.
- **Social\_support:**
  - Most countries have a high level of social support (between 1.0 and 1.5).
  - It is less common for countries to have low levels of social support (<0.5), suggesting that social support is a common and high characteristic in most regions.
- **Health\_(Life\_Expectancy):**
  - Health-adjusted life expectancy shows a peak around 0.8, with a wider range at moderate values (0.4-0.9).
  - Few countries have extremely low values (<0.2) or close to 1.
- **Freedom:**
  - Perceived freedom to make personal decisions is centered around 0.4-0.6.
  - There are relatively few countries with very low (<0.2) or very high (>0.6) levels, indicating a trend toward moderate levels.

- **Generosity:**

- Generosity has a concentrated distribution toward lower values, with a peak around 0.2.
- This suggests that most countries have low or moderate levels of generosity, with high generosity ( $>0.4$ ) being rare.

- **Perceptions\_of\_corruption:**

- Perceptions of corruption peaks at low values (around 0.1), indicating that in many countries, perceptions of corruption are high (low values indicate high perceived corruption).
- Few countries have high values for perceived low corruption ( $>0.4$ ), suggesting that corruption is a widespread problem.



1. **Happiness\_Score:**

- The median happiness score is around 5, with most countries falling between 4 and 6 (interquartile range).
- There are no extreme outliers, and the range goes from approximately 3 to 8.

2. **GDP\_per\_capita:**

- GDP per capita has a median around 1, with most values between 0.75 and 1.25.
- The range extends from near 0 to about 2, indicating some high-income outliers.

3. **Social\_support:**

- Social support is relatively high, with a median slightly above 1.
- Most countries are between 0.75 and 1.25, but a few outliers fall below 0.75.

#### 4. **Health\_(Life\_Expectancy):**

- Health, measured through life expectancy, has a median of about 0.75.
- The interquartile range spans 0.5 to 0.9, with no notable extreme values.

#### 5. **Freedom:**

- Freedom has a median of approximately 0.45, with most countries within the 0.3–0.6 range.
- The distribution is relatively tight, with no significant outliers.

#### 6. **Generosity:**

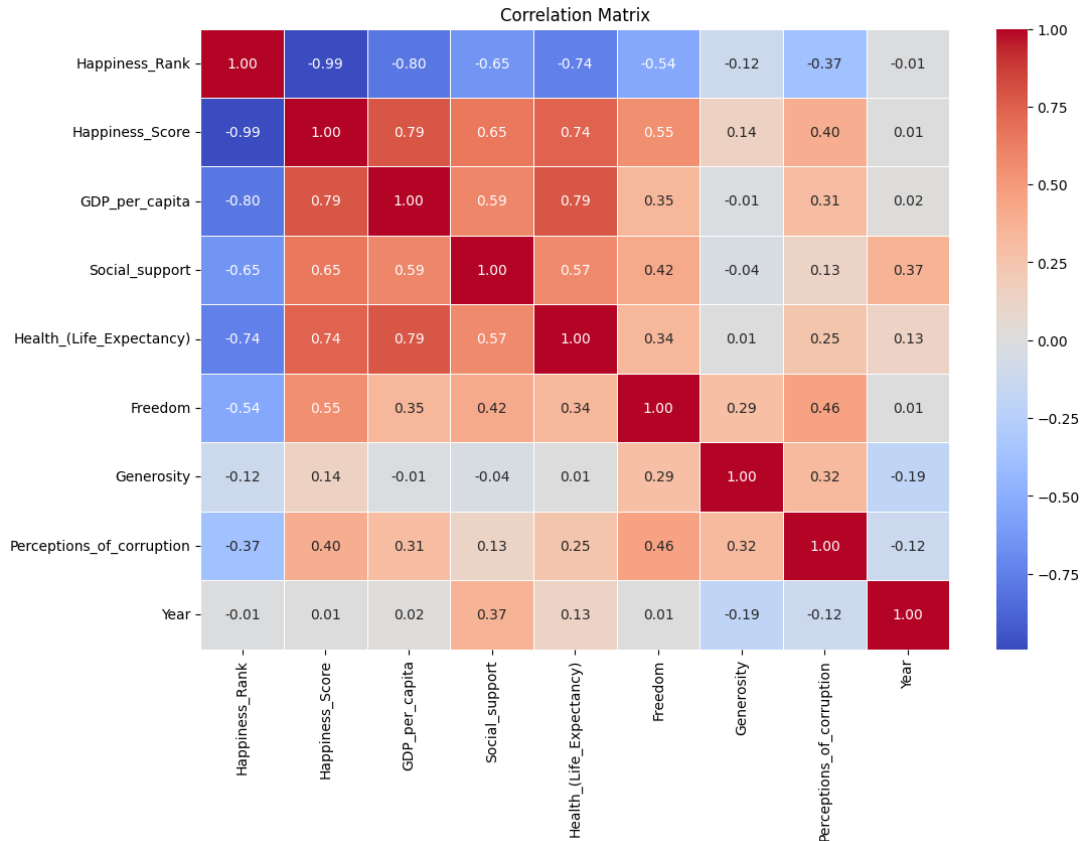
- Generosity has a smaller range, with a median close to 0.2.
- There are several outliers, indicating that some countries exhibit higher levels of generosity compared to the majority.

#### 7. **Perceptions\_of\_corruption:**

- The median perception of corruption is very low, around 0.1, indicating high perceived corruption in most countries.
- There are a few outliers where corruption is perceived to be lower.

### Key Insights:

- **Happiness\_Score** has the largest scale, emphasizing its higher variance compared to other variables.
  - **GDP\_per\_capita**, **Social\_support**, and **Health** exhibit relatively similar interquartile spreads, suggesting they may play a significant role in determining happiness.
  - Variables like **Freedom**, **Generosity**, and **Perceptions\_of\_corruption** have tighter distributions, indicating less variability across countries.
  - The presence of outliers in **Social\_support**, **Generosity**, and **Perceptions\_of\_corruption** reflects some countries with unusual patterns in these areas.
-



## Key Observations:

### 1. Happiness\_Rank and Happiness\_Score:

- **Strong negative correlation (-0.99):** Since rank is inversely related to score (lower rank indicates higher happiness), this strong negative relationship is expected.

### 2. GDP\_per\_capita:

- **Positively correlated with Happiness\_Score (0.79):** Countries with higher GDP per capita tend to have higher happiness scores.
- **Moderate correlation with Health (0.79) and Social\_support (0.59):** Wealthier countries often have better health systems and social safety nets.

### 3. Social\_support:

- **Positive correlation with Happiness\_Score (0.65):** Strong social networks contribute significantly to happiness.
- **Moderate correlation with GDP\_per\_capita (0.59):** Social support systems are stronger in wealthier nations.

### 4. Health (Life\_Expectancy):

- **Strong correlation with Happiness\_Score (0.74):** Better health outcomes are strongly associated with greater happiness.

- **Positive correlation with GDP\_per\_capita (0.79):** Wealthier countries tend to have better health outcomes.

#### 5. Freedom:

- **Moderate correlation with Happiness\_Score (0.55):** A sense of freedom in making life choices moderately contributes to happiness.
- **Weaker correlations with GDP\_per\_capita (0.35) and Social\_support (0.42).**

#### 6. Generosity:

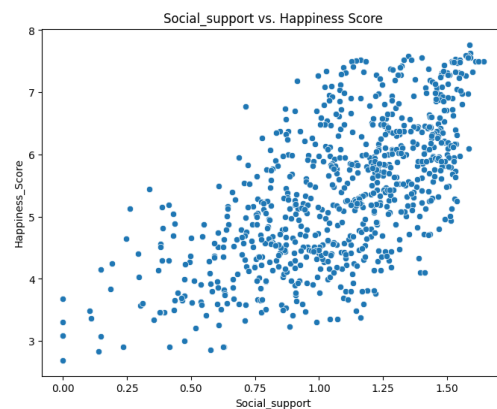
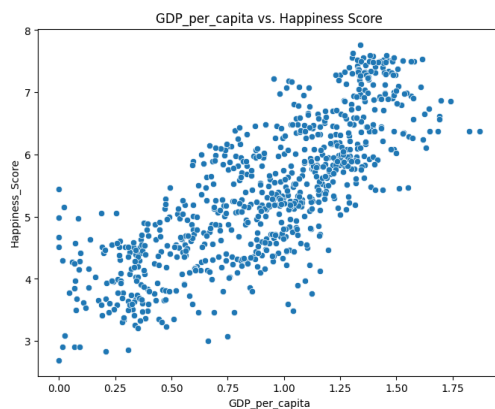
- **Weak correlation with Happiness\_Score (0.14):** Generosity has a limited impact on overall happiness compared to other factors.
- **Almost no correlation with GDP\_per\_capita (-0.01):** Wealth and generosity do not seem directly linked.

#### 7. Perceptions\_of\_corruption:

- **Moderate positive correlation with Happiness\_Score (0.40):** Lower corruption perception is associated with higher happiness.
- **Weaker correlations with GDP\_per\_capita (0.31) and Freedom (0.46).**

#### 8. Year:

- **Very weak or negligible correlations:** The variable "Year" does not significantly correlate with other factors, suggesting that the dataset does not show major time-based trends.





### 1. GDP\_per\_capita vs. Happiness\_Score

- There is a clear positive correlation between GDP per capita and the Happiness Score.
- Countries with higher GDP per capita tend to have higher Happiness Scores.
- The trend shows diminishing returns: as GDP increases significantly, the incremental increase in happiness appears to slow.

### 2. Social\_support vs. Happiness\_Score

- A strong positive correlation is evident between Social Support and Happiness Score.
- Countries with better social support systems generally report higher happiness.
- The points form a dense upward trend, indicating social support is a consistent and significant predictor of happiness.

### 3. Health (Life\_Expectancy) vs. Happiness\_Score

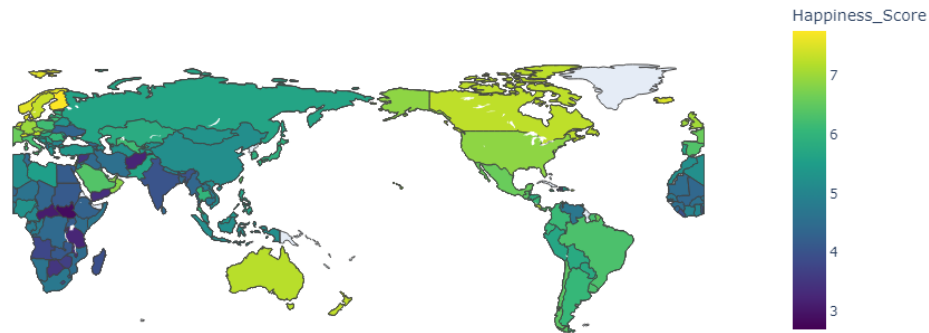
- A positive correlation exists between Health (Life Expectancy) and Happiness Score.
- Countries with higher life expectancy tend to report greater happiness.
- The relationship is strong, but like GDP, there seems to be a limit to how much increased life expectancy can improve happiness.

#### Summary of Insights:

- All three factors show significant positive relationships with happiness.
- **Social Support** and **Health (Life Expectancy)** have slightly more consistent trends compared to GDP per capita, which shows diminishing returns at higher levels.
- These visualizations reinforce the idea that wealth, health, and social support are key contributors to national happiness levels.



World Happiness Score by Country



#### 1. High Happiness Scores:

- Countries in **Northern Europe** (e.g., Finland, Denmark, Sweden, Norway) display the highest happiness scores (yellow regions, scores around 7).
- Some countries in **North America** (e.g., Canada, the United States) and **Oceania** (e.g., Australia, New Zealand) also rank highly.

#### 2. Moderate Happiness Scores:

- Countries in **Eastern Europe**, **South America**, and parts of **Asia** (e.g., China, Japan) show mid-range scores (green regions, scores between 4 and 6).

#### 3. Low Happiness Scores:

- Countries in **Sub-Saharan Africa** and some parts of the **Middle East** tend to have the lowest happiness scores (dark blue regions, scores below 4).
- Political instability, poverty, and other socio-economic challenges likely contribute to these lower scores.

#### 4. Regional Patterns:

- **Developed nations** tend to cluster towards higher scores, reflecting the influence of wealth, health, and social support.
- **Developing nations** and regions with significant challenges (e.g., conflict, inequality) often display lower scores.

---

▼ Data Upload

```
generator = SQLSchemaGenerator(table_name='transformed_data')

generator.generate_schema(merged_data_df, '../sql/transformed_data_schema.sql')
generator.generate_seed_data(merged_data_df, '../sql/transformed_data_seed_data.sql')
```

```
INFO:root:Generating schema for transformed_data
INFO:root:Inferring SQL type for int64
INFO:root:Inferring SQL type for object
INFO:root:Inferring SQL type for float64
INFO:root:Inferring SQL type for float64
INFO:root:Inferring SQL type for float64
INFO:root:Inferring SQL type for float64
INFO:root:Inferring SQL type for float64
INFO:root:Inferring SQL type for float64
INFO:root:Inferring SQL type for float64
INFO:root:Inferring SQL type for int64
INFO:root:Query written to ../sql/transformed_data_schema.sql
INFO:root:Generating seed data for transformed_data
INFO:root:Query written to ../sql/transformed_data_seed_data.sql
```

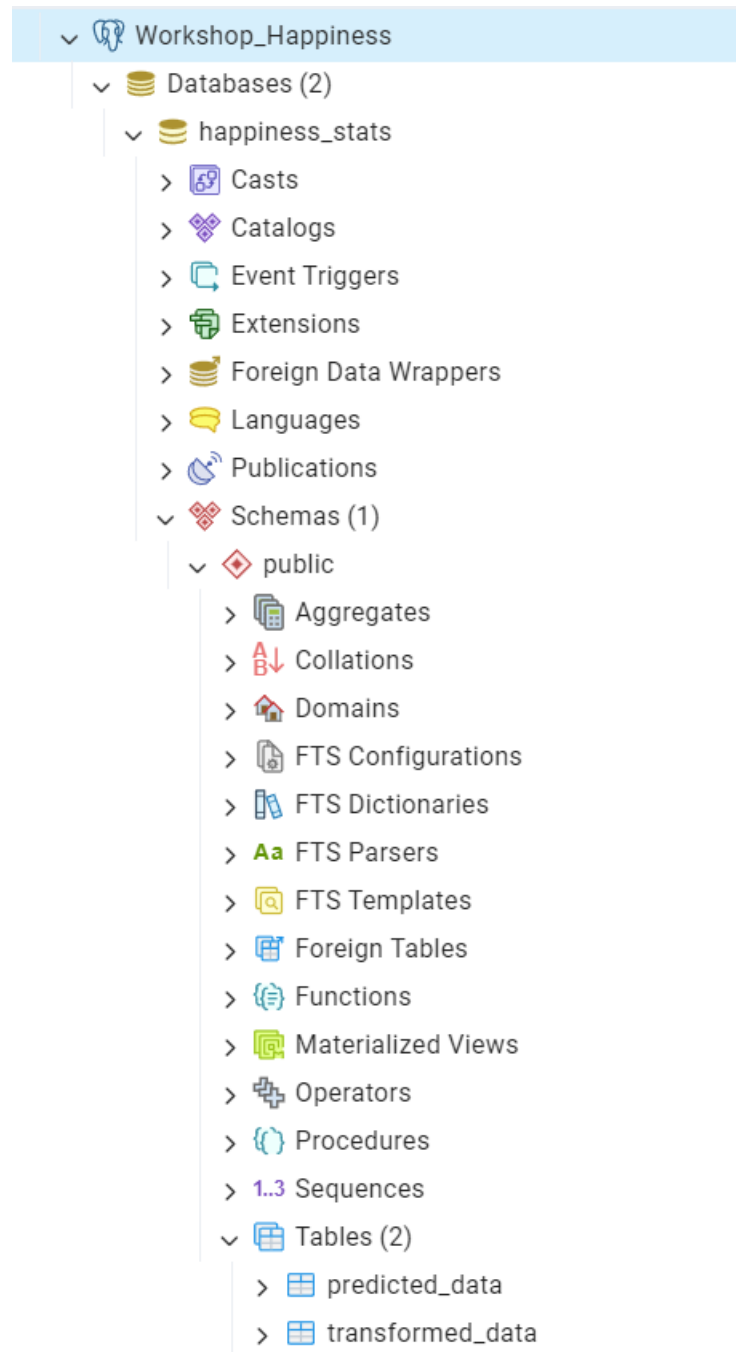
```
# Create schema
db.execute_query_file("../sql/transformed_data_schema.sql", fetch_results=False)
```

```
INFO:root:✓ Connected to database
INFO:root:✓ Query executed
INFO:root:✓ Cursor closed
INFO:root:✓ Connection closed
```

```
# Seed data
db.execute_query_file("../sql/transformed_data_seed_data.sql", fetch_results=False)
```

```
INFO:root:✓ Connected to database
INFO:root:✓ Query executed
INFO:root:✓ Cursor closed
INFO:root:✓ Connection closed
```

Since it was necessary for the pipeline design that the transformed/merged data be available, at this point I decided to upload them to Render using the created automation library



## DB: Happiness Metrics Database

**New PostgreSQL** View docs

**Name**  
A unique name for your PostgreSQL instance.  
workshop-003-machine-learning-kafka

**Project** Optional  
Add this database to a **project** once it's created.

**Create a new project to add this to?**

You don't have any projects in this workspace. **Projects** allow you to group resources into environments so you can better manage related resources.

[+ Create a project](#)

**Database** Optional  
The PostgreSQL `dbname`  
dcajiao

**User** Optional  
randomly generated unless specified

**Region**  
Your services in the same **region** can communicate over a [private network](#). You currently have services running in **Oregon**.

☒ **Oregon (US West)** 8 existing services

[Deploy in a new region](#) +

**Plan Options**

**Instance Type**  
Set your database's RAM and CPU. You can change your instance type later.

**New**  
You can now set your database's storage separately from its instance type. [Learn more](#)

☒ **Free**  
For testing out PostgreSQL on Render.

**Free** 256 MB (RAM)

**\$0 / month** 0.1 CPU

1 GB (Storage)

☐ **Basic**  
Reliability and performance for hobby projects. Starting at \$6 / month plus storage.

☐ **Pro**  
Perfect for production use cases at any scale. Starting at \$55 / month plus storage.

☐ **Accelerated**  
Memory-optimized plans offer significant performance improvements. Starting at \$160 / month plus storage.

Need a [custom instance type](#)? We support up to 1024 GB RAM, 128 CPUs, and 5 TB storage.

**Storage**  
Your database's capacity, in GB. You can increase storage at any time, but you can't decrease it. Specify 1 GB or any multiple of 5 GB.

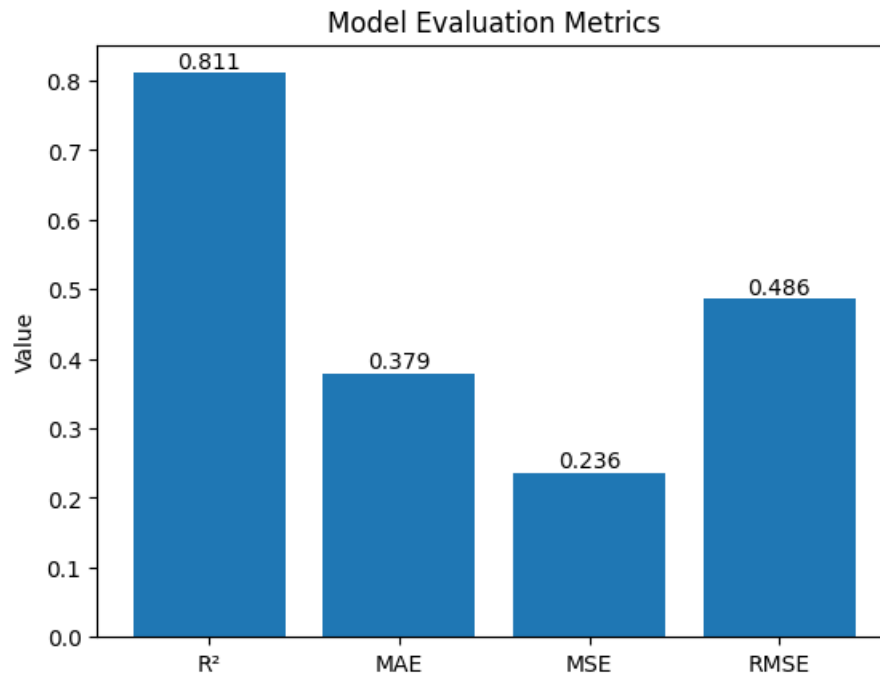
1 GB \$0 / month

This database was created in Render with the free service type and the login credentials were exported for later connection from both the Airflow pipeline and the Kafka consumer, as well as the notebooks from the workspace.

## PHASE: Training and evaluation of the model

```
# Separating the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=42)

# Create a pipeline that includes scaling and the model
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('model', RandomForestRegressor())
])
```



```
# Export the model as a pickle file
output_paths = ['./models/00_happiness_score_prediction_model.pkl',
                 './api/src/models/00_happiness_score_prediction_model.pkl']

for path in output_paths:
    with open(path, 'wb') as file:
        pickle.dump(pipeline, file)

✓ 0.0s
```

Finally, at the model stage I chose an initial regression model with an R2 of 0.811. While these are not the best metrics and a better model surely exists, that is exactly why the pipeline was created in Airflow, to make testing new models and deploying them a simpler task.

## PIPELINE: Training and automatic model availability

Airflow

DAGs

Cluster Activity

Datasets

Security

Browse

Admin

Docs

01:46 UTC

AA

Triggered Training\_pipeline\_and\_automatic\_model\_availability with new Run ID manual\_\_2024-11-16T01:45:30.023100+00:00, it should start any moment now.

DAG: Training\_pipeline\_and\_automatic\_model\_availability

DAG to train a model and make it available for predictions in the API

Schedule: 1 day, 0:00:00

Next Run ID: 2024-11-16, 00:00:00 UTC

16/11/2024

01:45:30 a.m.

All Run Types

All Run States

Clear Filters

Auto-refresh

25

Press **shift** + **f** for Shortcuts

deferred

failed

queued

removed

restarting

running

scheduled

shutdown

skipped

success

up\_for\_reschedule

up\_for\_retry

upstream\_failed

no\_status

DAG

Run

Task

Training\_pipeline\_and\_automatic\_model\_availability

2024-11-15, 01:45:30 UTC

upload\_model

Clear task

Mark state as...

Filter DAG by task

Details

Graph

Gantt

Code

Event Log

Logs

XCom

Task Duration

All Levels

All File Sources

Wrap

Download

See More

```

00991ce37679
*** Found local files:
***   /opt/airflow/logs/dag_id=Training_pipeline_and_automatic_model_availability/run_id=manual__2024-11-16T01:45:30.023100+00:00/task_id=upload_model/attempt-1.log
[2024-11-16, 01:45:39 UTC] [local_task_job_runner.py:123] ▶ Pre task execution logs
[2024-11-16, 01:46:39 UTC] [load_function.py:26] INFO - uploading model file: models/model.pkl
[2024-11-16, 01:46:13 UTC] [load_function.py:193] INFO - Model uploaded successfully.
[2024-11-16, 01:46:13 UTC] (python.py:240) INFO - Done. Returned value was: {'message': 'New model uploaded and loaded successfully.', 'model_path': './models/02_happiness_score_prediction_model.pkl'}
[2024-11-16, 01:46:13 UTC] [taskinstance.py:340] ▶ Post task execution logs

```

Duration

00:00:43

00:00:21

00:00:08

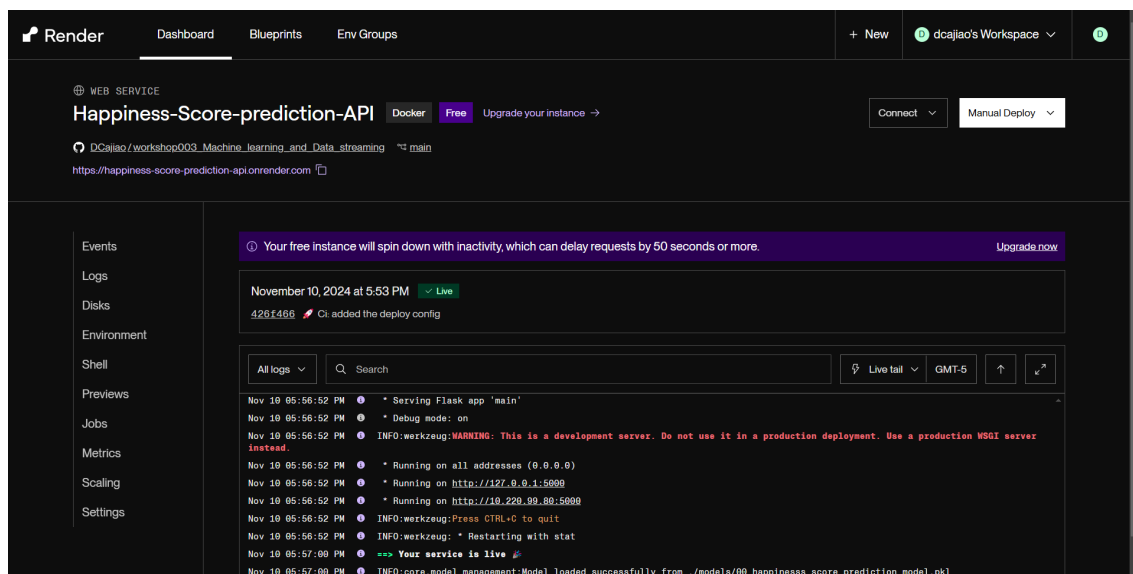
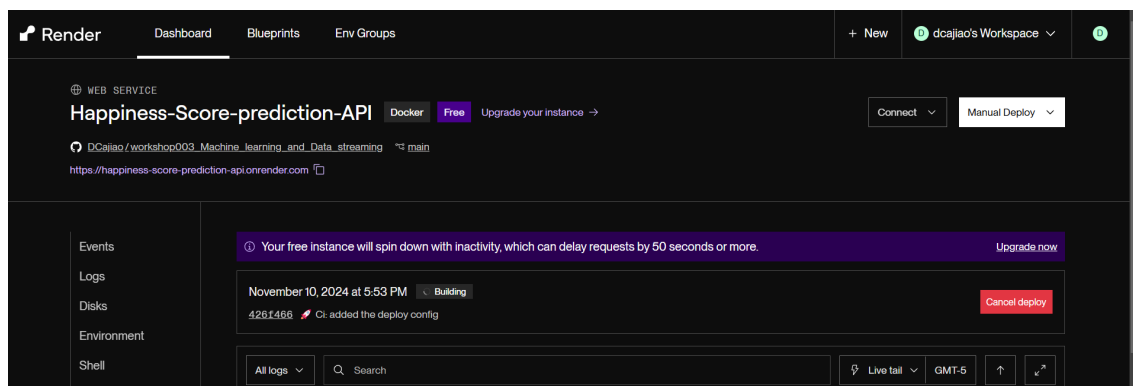
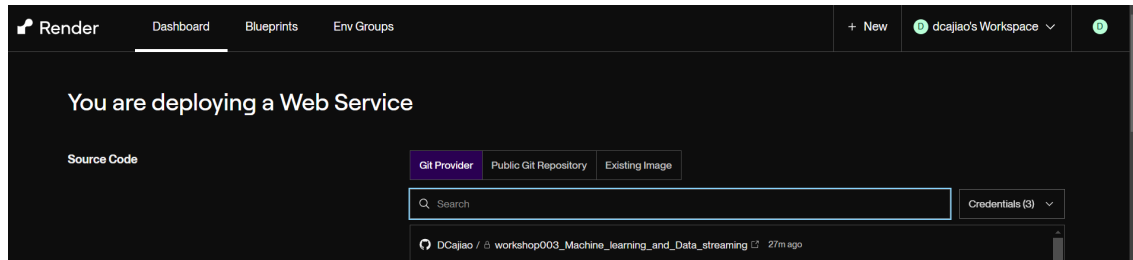
get\_db\_data

train\_model

upload\_model

As we could observe, this is the complete operation and integration of both the Airflow training and availability pipeline, as well as the arrival to the model from the logs in the Render terminal.

## API: Happiness Score Prediction API



Regarding the deployment of the API, we made use of a free Render service but in this case of Web Service type, with which we configured a continuous deployment through a connection to my Github account. This allows us to automate the redeployments under the new commits in the main branch.

HTTP Happiness Prediction API - Methods / **Predict**

**POST** ▼ <https://happiness-score-prediction-api.onrender.com/predict>

Params Authorization Headers (9) **Body** • Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼

```

1  {
2      "GDP_per_capita": 1.30232,
3      "Social_support": 1.34951,
4      "Health_(Life_Expectancy)": 0.94143,
5      "Freedom": 0.66557,
6      "Generosity": 0.29678,
7      "Perceptions_of_corruption": 0.41978,
8      "Year_2015": true,
9      "Year_2016": false,
10     "Year_2017": false,
11     "Year_2018": false,
12     "Year_2019": false
13 }

```

**Body** Cookies Headers (12) Test Results 200 OK

Pretty Raw Preview Visualize **JSON** ▼

```

1  {
2      "prediction": 7.36241000086466
3  }

```

HTTP Happiness Prediction API - Methods / **Upload Model**

**POST** ▼ [https://happiness-score-prediction-api.onrender.com/upload\\_model](https://happiness-score-prediction-api.onrender.com/upload_model)

Params Authorization Headers (9) **Body** • Scripts Settings

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

	Key		Value	Description
<input checked="" type="checkbox"/>	model	File ▼	00_happiness_score_prediction_model.pkl	
	Key	Text ▼	Value	Description

**Body** Cookies Headers (12) Test Results 200 OK • 1

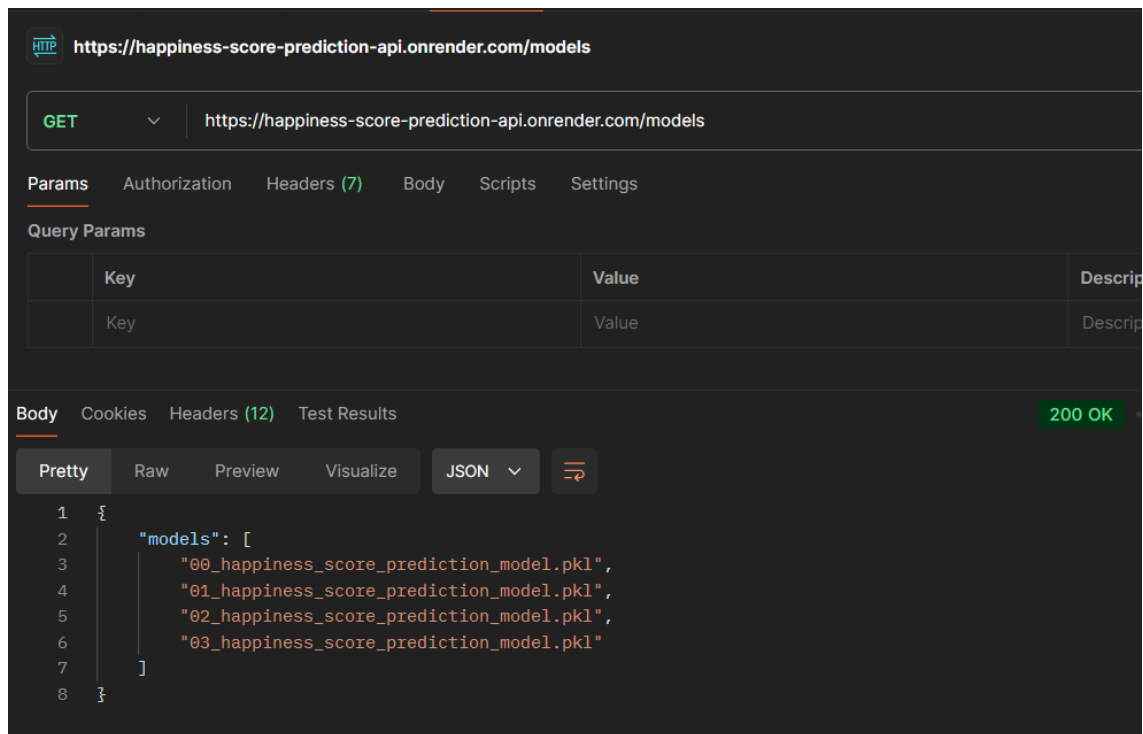
Pretty Raw Preview Visualize **JSON** ▼

```

1  {
2      "message": "New model uploaded and loaded successfully.",
3      "model_path": "./models/03_happiness_score_prediction_model.pkl"
4  }

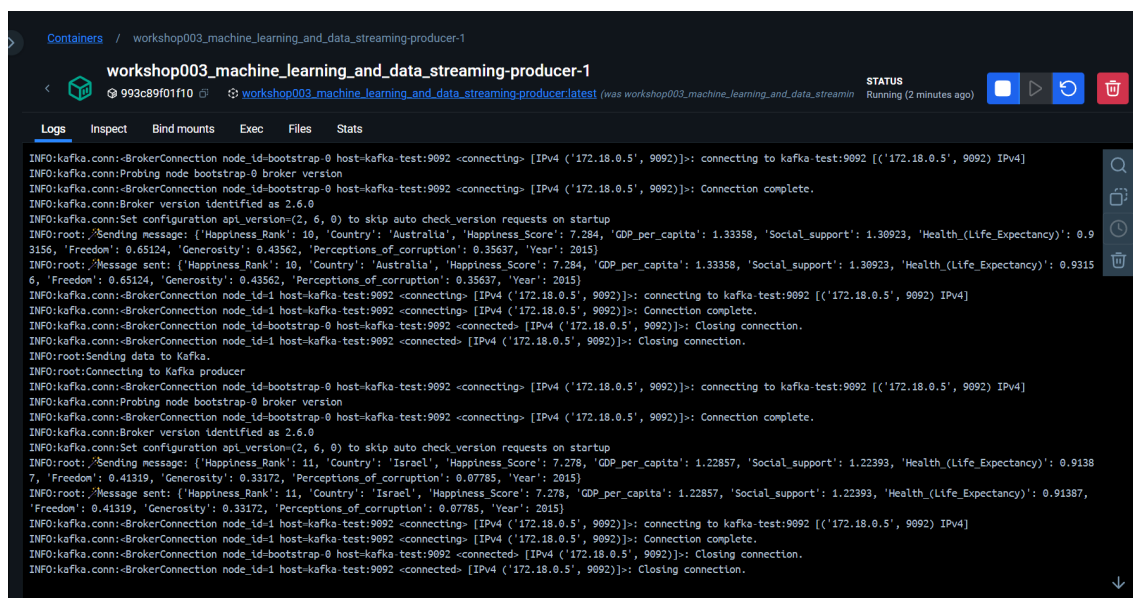
```





Additionally, we can observe the operation of the API methods consumed from Postman.

## STREAM: Data stream with Apache Kafka



Regarding the producer, in this case it was done with a script and a standalone Docker container based on a Python image, which creates the topic, queries the data and starts streaming one by one.

```
Containers / workshop003_machine_learning_and_data_streaming-consumer-1
workshop003_machine_learning_and_data_streaming-consumer-1
ca72c39ad185 workshop003_machine_learning_and_data_streaming-consumer:latest (was workshop003_machine_learning_and_data_streaming-consumer:latest) Running (1 minute ago)
Logs Inspect Bind mounts Exec Files Stats
INFO:root:Query executed
INFO:root:Cursor closed
INFO:root:Connection closed
INFO:root:Data inserted into the predicted_data table
INFO:root:Received message: {'Happiness_Rank': 6, 'Country': 'Finland', 'Happiness_Score': 7.406, 'GDP_per_capita': 1.29025, 'Social_support': 1.31826, 'Health_Life_Expectancy': 0.8891, 'Freedom': 0.64169, 'Generosity': 0.23351, 'Perceptions_of_corruption': 0.41372, 'Year': 2015}
INFO:main:Making a POST request to the API prediction
INFO:root:Connected to database
INFO:root:Query executed
INFO:root:Cursor closed
INFO:root:Connection closed
INFO:root:Data inserted into the predicted_data table
INFO:root:Received message: {'Happiness_Rank': 7, 'Country': 'Netherlands', 'Happiness_Score': 7.378, 'GDP_per_capita': 1.32944, 'Social_support': 1.28017, 'Health_Life_Expectancy': 0.89284, 'Freedom': 0.61576, 'Generosity': 0.4761, 'Perceptions_of_corruption': 0.31814, 'Year': 2015}
INFO:main:Making a POST request to the API prediction
INFO:root:Connected to database
INFO:root:Query executed
INFO:root:Cursor closed
INFO:root:Connection closed
INFO:root:Data inserted into the predicted_data table
INFO:root:Received message: {'Happiness_Rank': 8, 'Country': 'Sweden', 'Happiness_Score': 7.364, 'GDP_per_capita': 1.33171, 'Social_support': 1.28907, 'Health_Life_Expectancy': 0.91087, 'Freedom': 0.6599, 'Generosity': 0.36262, 'Perceptions_of_corruption': 0.43844, 'Year': 2015}
INFO:main:Making a POST request to the API prediction
INFO:root:Connected to database
INFO:root:Query executed
INFO:root:Cursor closed
INFO:root:Connection closed
INFO:root:Data inserted into the predicted_data table
```

On the other side, the consumer is also a Python script but packaged in a different container. It is in charge of processing the message data in the topic, generating the payload to send to the API and once the prediction is obtained, it generates the insert in the prediction database.

extensions

Foreign Data Wrappers

Languages

Publications

Schemas (1)

public

Aggregates

Collations

Domains

FTS Configurations

FTS Dictionaries

FTS Parsers

FTS Templates

Foreign Tables

Materialized Views

Functions

Operators

Procedures

Sequences

Tables (2)

predicted\_data

transformed\_data

Trigger Functions

Types

Views

Subscriptions

postgres

Login/Group Roles

Tablespaces

kafkadb

workshop002

happiness\_stats/dcjiao@happiness

Query Query History


Scratch Pad

1 SELECT \* FROM predicted\_data

Data Output Messages Notifications

	GDP_per_capita double precision	Social_support double precision	Health_(Life_Expectancy) double precision	Freedom double precision	Generosity double precision	Perceptions_of_corruption double precision	Year smallint	Predict_Happiness_Score double precision
1	1.30232	1.40223	0.94784	0.62877	0.4363	0.14145	2015	7.18839997558596
2	1.32548	1.36058	0.87464	0.64938	0.34139	0.48357	2015	6.991920000476837
3	1.459	1.33095	0.88521	0.66973	0.34699	0.36503	2015	7.453379998359687
4	1.32629	1.32261	0.90563	0.63297	0.45811	0.32957	2015	7.353420000000009
5	1.29025	1.31826	0.88911	0.64169	0.23351	0.41372	2015	7.314269999923698
6	1.32944	1.28017	0.89284	0.61576	0.4761	0.31814	2015	7.144129998321538
7	1.33171	1.28907	0.91087	0.6598	0.36262	0.43844	2015	7.049360000801082
8	1.25018	1.31967	0.90837	0.63938	0.47501	0.42922	2015	7.169630001296988
9	1.33358	1.30923	0.93156	0.65124	0.43562	0.35637	2015	7.290470000864661
10	1.22857	1.22393	0.91387	0.41319	0.33172	0.07785	2015	6.43066999511719
11	0.95578	1.23788	0.86027	0.63376	0.25497	0.10583	2015	6.711099997825622
12	1.33723	1.29704	0.89042	0.62433	0.33088	0.18676	2015	7.111069998931891
13	1.02054	0.91451	0.81444	0.48181	0.14074	0.21312	2015	6.632320002288824
14	1.39451	1.24711	0.86179	0.54604	0.40105	0.1589	2015	7.069089997138989
15	0.98124	1.23287	0.69702	0.49049	0.14574	0.17521	2015	6.510029991188044

Total rows: 20 of 20 Query complete 00:00:01.083 Ln 1, Col 29

 You will be able to find evidence of execution uploaded in the repository.