Zbieranie danych o meczach piłkarskich

Raport końcowy

Zespół D.A.T.A

Spis treści:

Uzasadnienie biznesowe

Opis danych

Architektura

Przepływ danych w NiFi i testy

Transformacja danych i testy

Wizualizacje

Opis projektu i uzasadnienie biznesowe:

Celem projektu jest stworzenie kompleksowego systemu Big Data, który pozwoli na zbieranie i analizowanie danych z meczów piłkarskich. System ten będzie zarówno oferował dostęp do danych w czasie rzeczywistym z aktualnie rozgrywanych meczów jak i gromadził dane historyczne do późniejszej analizy. W projekcie zostaną wykorzystane powszechnie stosowane narzędzia z obszaru Big Data.

Piłka nożna jest najpopularniejszym sportem na świecie. Największe ligi takie jak angielska Premier League czy hiszpańska Primera Division gromadzą co tydzień przed telewizorami miliony kibiców, którzy na bieżąco śledzą statystyki ze świata futbolu. Popularność piłki powoduje, że staje się ona także opłacalnym biznesem a rynek analizy danych piłkarskich rośnie w bardzo szybkim tempie. Trenerzy i analitycy piłkarscy obmyślają na podstawie danych strategie na kolejne mecze, media prześcigają się w opublikowanie najświeższych statystyk w celu przyciągnięcia uwagi internautów a bukmacherzy tworzą na podstawie danych odpowiednie modele tak aby nie ponieść strat. Nasz projekt pomoże więc zarówno pasjonatom interesującym się piłką nożną z miłości do tego pięknego sportu jak i osobom zajmującym się piłką nożną zawodowo.

Opis danych:

Dane meczowe:

Statystyki drużynowe i indywidualne oraz informacje meczowe będą pobierane są z platformy FBRef, która oferuje bardzo szczegółowe dane o przebiegu meczów. Dane te będą pozyskiwane przy użyciu webscrapingu z wykorzystaniem biblioteki Beautiful Soup, co umożliwia automatyczne gromadzenie i regularne aktualizowanie danych. Planowana częstotliwość odświeżania danych: po każdym meczu lub symulowanie co minutę w trakcie meczu.

Link: https://fbref.com/en

Dane zawierają:

- informacje ogólne o meczu (sezon, runda, data i godzina, frekwencja, sędzia)

- statystyki meczowe każdej z drużyn (np. liczba podań, strzały na bramkę, xG, kartki)

Dane pogodowe:

Informacje o warunkach pogodowych podczas meczów będzie pozyskiwana za pomocą otwartego API Open-Meteo. Platforma oferuje dostęp do aktualizowanych co 30 minut danych pogodowych Będziemy zbierać dane na podstawie daty rozgrywania meczu i współrzędnych geograficznych stadionu. Planowana częstotliwość odświeżania danych: co 30 min w trakcie meczu.

Link: https://open-meteo.com/

Dane zawierają:

- zmienne atmosferyczne (np. temperatura, wilgotność, opady, prędkość wiatru)

- informację o współrzędnych geograficznych oraz dacie i godzinie, na podstawie czego można je przypisać do odpowiedniego meczu

Dane dotyczące zawodników:

Charakterystyki zawodników, w tym ich umiejętności i parametry fizyczne, pobierane będą z serwisu Sofifa, który oferuje bogaty zestaw danych o zawodnikach z różnych edycji gry FIFA od EA Sports. Planowana częstotliwość odświeżania danych: po każdym meczu lub symulowanie co minutę w trakcie meczu.

Link: https://sofifa.com/

Dane zawierają:

- podstawowe informacje o zawodnikach (imię, nazwisko, wiek, drużyna), na bazie któych można przypisać zawodnika do meczu

- szczegółowe cechy indywidualne zawodników (np. zdolności ofensywne, umiejętności techniczne, siła fizyczna)

Dane meczowe aktualizowane live:

Za pomocą otwartego Api-Football będziemy mogli pobierać dane meczowe z aktualnie rozgrywanego meczu. Niestety większość tego typu API ma limity zapytań. W tym wypadku jest to 100 zapytań na godzinę. Planowana częstotliwość odświeżania danych: co minutę w trakcie meczu.

Link: https://www.api-football.com/

Dane zawierają:

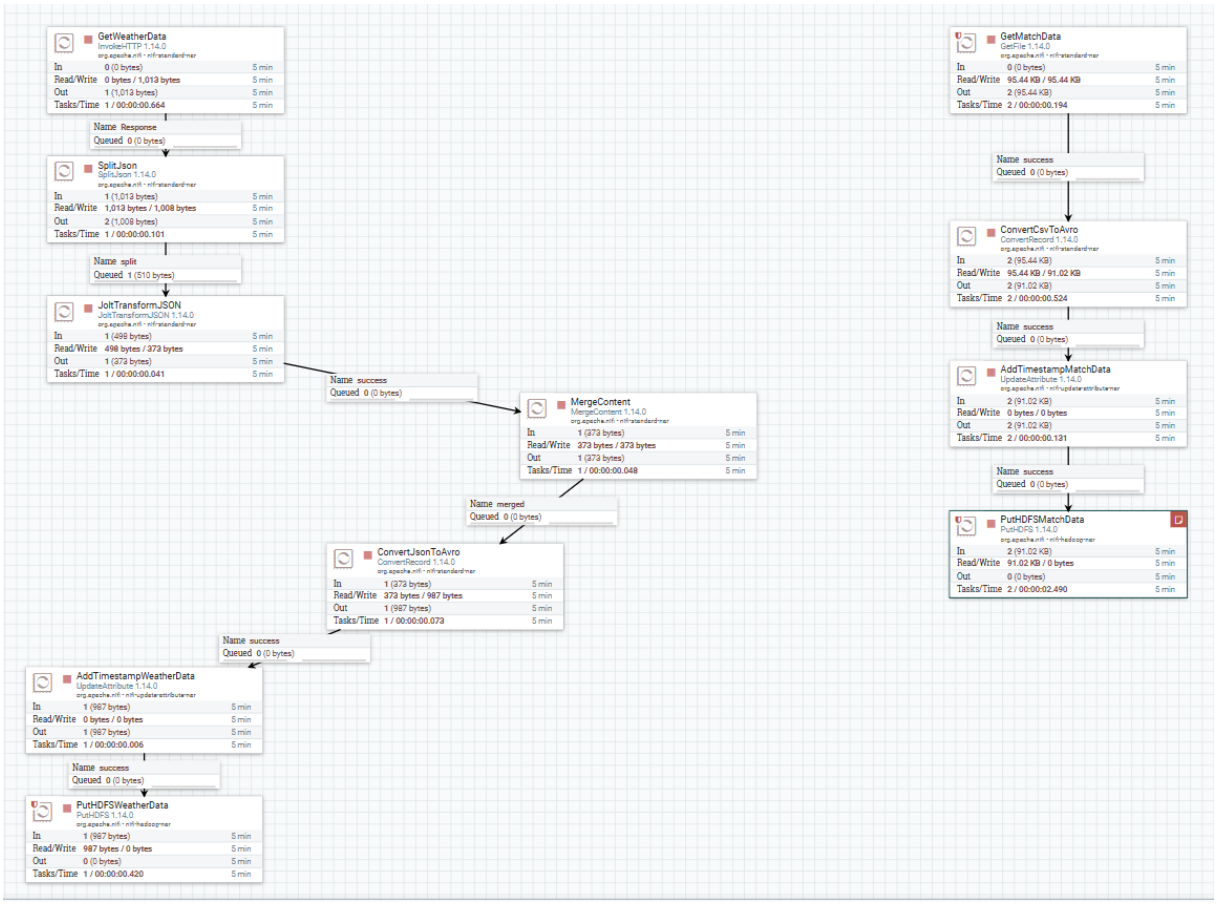
- wyniki I statystyki meczu aktualizowane w czasie rzeczywistym

- historyczne dane meczowe

Architektura

Automatyzacja przepływu danych została zaprojektowana przy pomocy narzędzia Apache NiFi. Do składowania danych wykorzystaliśmy zarówno Apache Hadoop i Apache Hive. W naszym rozwiązaniu ładujemy kolejne porcje danych wraz z kolejnymi rozgrywanymi kolejkami piłkarskimi.

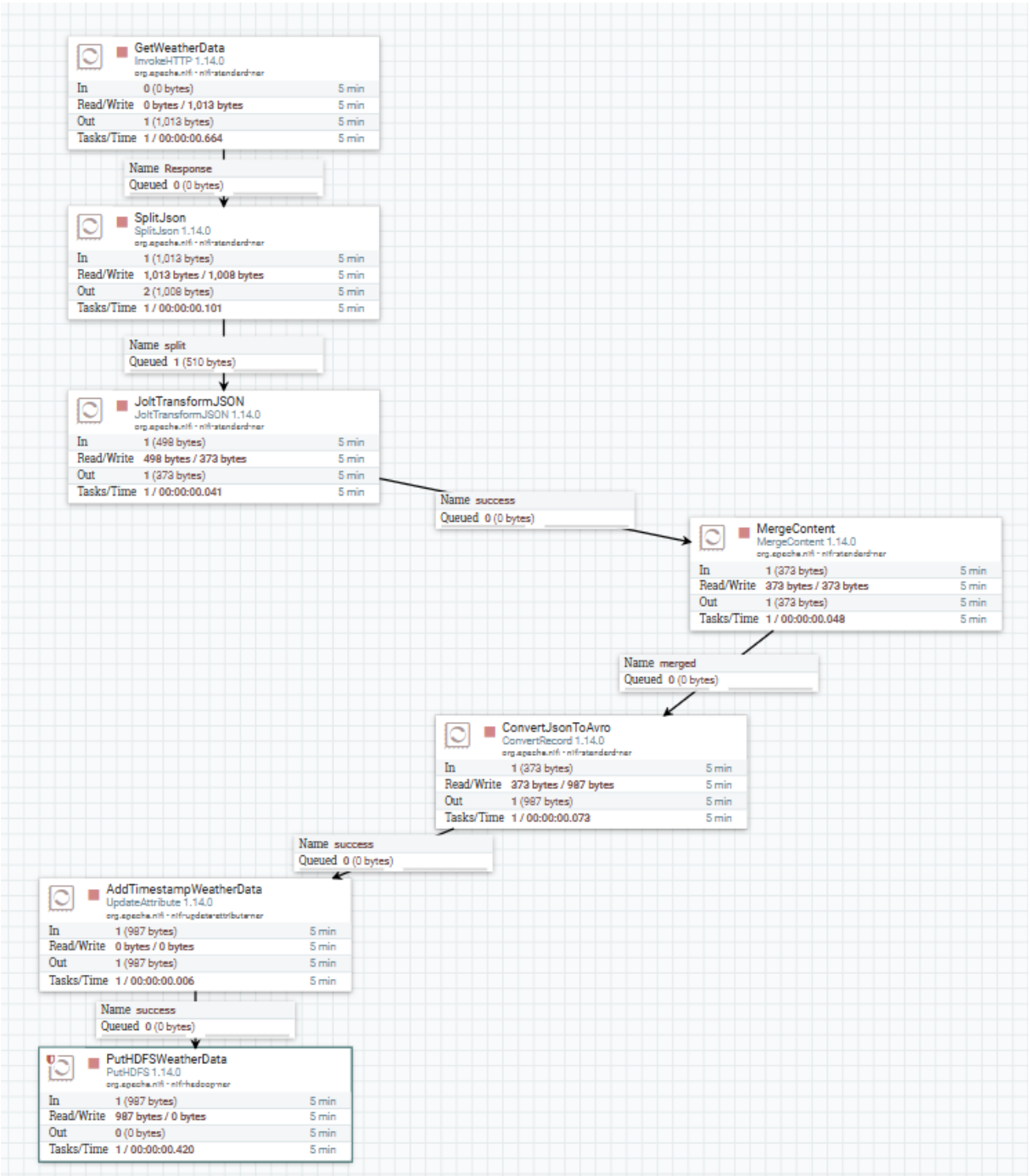
Przepływ danych w NiFi:



Składowanie danych pogodowych:

Dane pogodowe pobierane są za pomocą OPEN-METEO API, które umożliwia pobieranie informacji o aktualnej pogodzie (temepratura powietrza, ilość opadów, siła wiatru, wilgotność powietrza, zachmurzenie) dla danych lokalizacji.

Procesory NiFi:

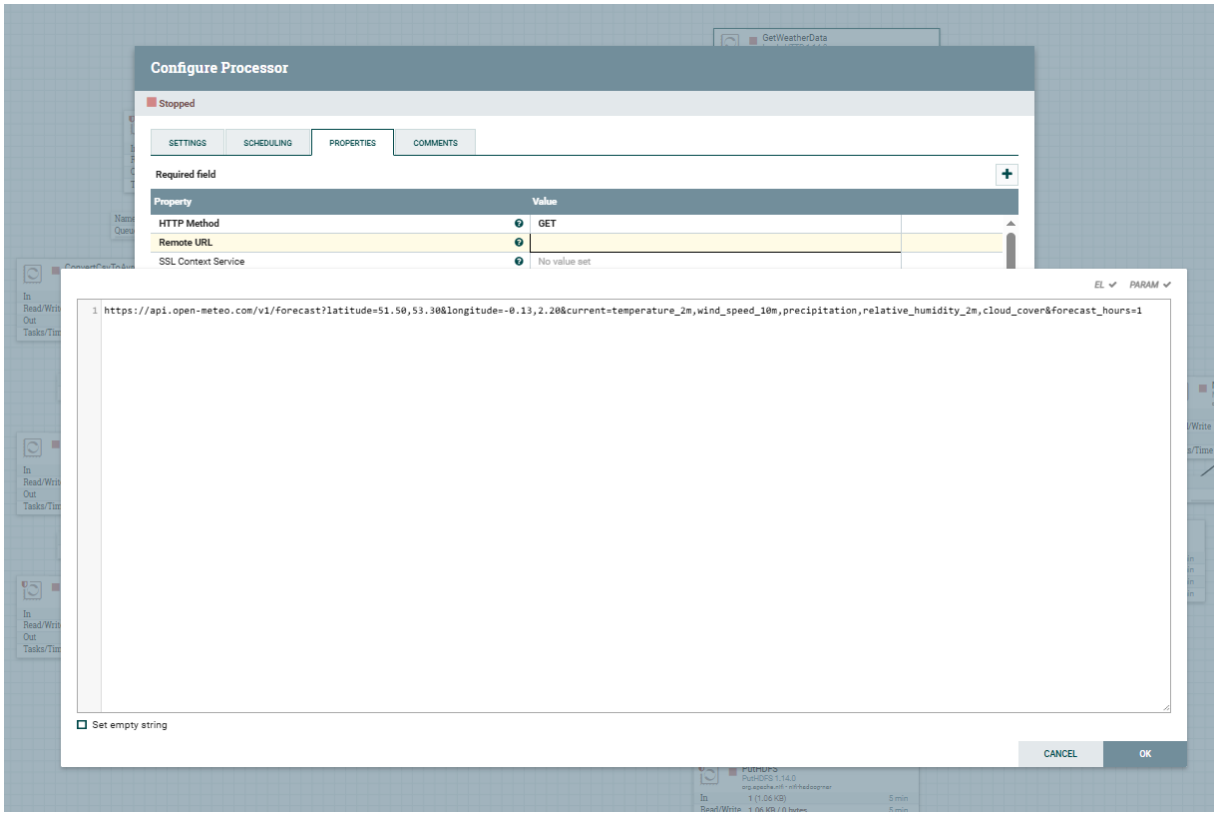


1. InvokeHTTP

- Korzystamy z metody GET, aby pobrać dane. Są one zwracane w postaci JSON.

- W polu Remote URL wpisujemy interesujące nas zapytanie. API to pozwala na wpisanie po przecinku kilku longitude i latitude, dzięki czemu możemy pobierać naraz dane pogodowe dla różnych lokalizacji, co jest przydatne, ponieważ często rozgrywanych jest kilka meczów jednocześnie.

- Np. https://api.open-meteo.com/v1/forecast?latitude=51.50,53.10,53.10&longitude=-0.13,-2.24,-3.10&current=temperature_2m,wind_speed_10m,precipitation,relative_humidity_2m,cloud_cover

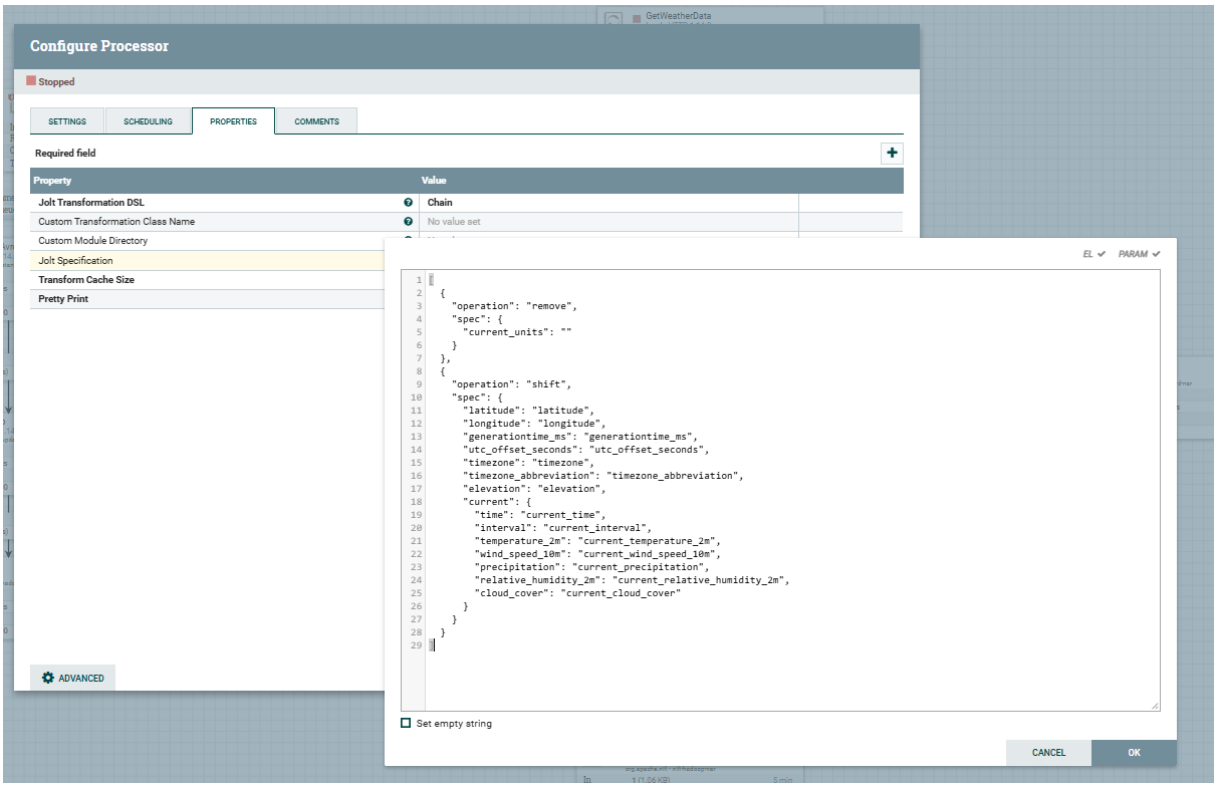


2. SplitJson

- Rozdzielamy odpowiedzi, tak aby mieć osobne odpowiedzi dla każdej lokalizacji.

3. JoltTransformJson

- Przekształcamy JSON, tak aby wydobyć z niego tylko interesujące nas informacje i przedstawić je w odpowiedniej strukturze.



4. MergeContent

- Łączymy wszystkie pliki (Flow File).

5. ConvertRecord

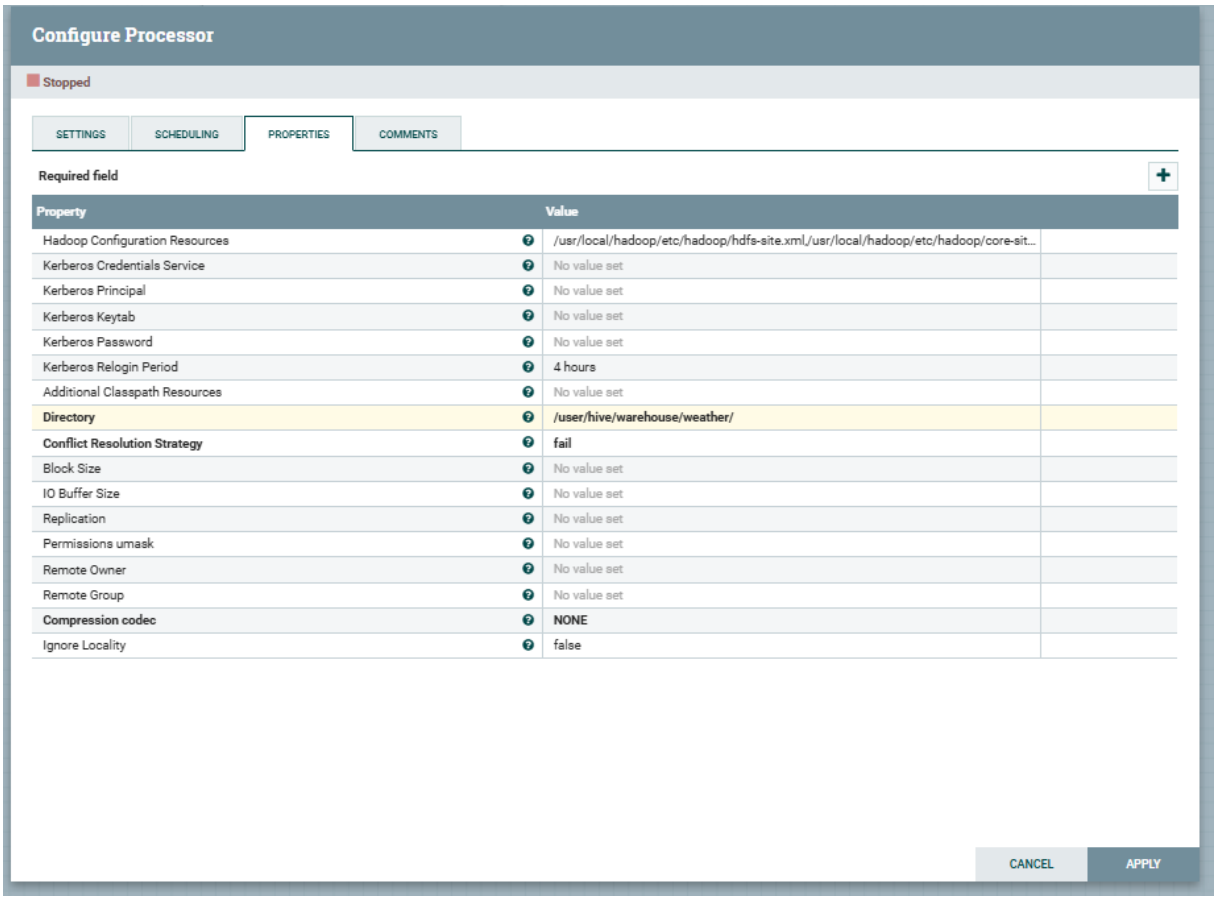
- Konwertujemy plik JSON na AVRO, który jest bardziej odpowiedni dla rozwiązań z zakresu Big Data.

6. UpdateAttribute

- Ustawiamy odpowiednią nazwę pliku wzbogaconą o timestamp.

7. PutHDFS

- Ładujemy plik do odpowiedniego folderu Apache Hadoop, z którego dane pobiera stworzona w Hive External Table weather.




```
hive> CREATE EXTERNAL TABLE weather (
. . >     latitude DOUBLE,
. . >     longitude DOUBLE,
. . >     generationtime_ms DOUBLE,
. . >     utc_offset_seconds INT,
. . >     timezone STRING,
. . >     timezone_abbreviation STRING,
. . >     elevation DOUBLE,
. . >     current_time STRING,
. . >     current_interval INT,
. . >     current_temperature_2m DOUBLE,
. . >     current_wind_speed_10m DOUBLE,
. . >     current_precipitation DOUBLE,
. . >     current_relative_humidity_2m INT,
. . >     current_cloud_cover INT
. . > )
. . > STORED AS AVRO
. . > LOCATION '/user/hive/warehouse/weather';
OK
No rows affected (0.735 seconds)
hive> select * from weather;
OK
51.5 -0.120000124 0.04398822784423828 0 GMT GMT 17.0 2024-12-15T11:45 900 10.5 11.8 0.0 90 91
53.302002 2.204 0.024080276489257812 0 GMT GMT 0.0 2024-12-15T11:45 900 9.4 42.1 0.0 97 100
2 rows selected (1.717 seconds)
```

- Pierwsze ładowanie danych do HDFS i tworzenie tabeli w Hive.

```
vagrant@node1:~$ hdfs dfs -ls /user/hive/warehouse/weather/
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.6/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/
SLF4J: Found binding in [jar:file:/usr/local/apache-tez-0.9.1-bin/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLogger
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Found 1 items
-rw-r--r--   1 root supergroup       1089 2024-12-15 11:59 /user/hive/warehouse/weather/20241215115945571_weather_data.avro
```

```
hive> select * from weather;
OK
51.5 -0.120000124 0.04398822784423828 0 GMT GMT 17.0 2024-12-15T11:45 900 10.5 11.8 0.0 90 91
53.302002 2.204 0.024080276489257812 0 GMT GMT 0.0 2024-12-15T11:45 900 9.4 42.1 0.0 97 100
2 rows selected (1.717 seconds)
```

- Widzimy, że tabela została poprawnie zasilona.

- Drugie ładowanie danych do HDFS.

```
vagrant@node1:~$ hdfs dfs -ls /user/hive/warehouse/weather/
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.6/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/
SLF4J: Found binding in [jar:file:/usr/local/apache-tez-0.9.1-bin/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLogger
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Found 2 items
-rw-r--r--   1 root supergroup       1089 2024-12-15 11:59 /user/hive/warehouse/weather/20241215115945571_weather_data.avro
-rw-r--r--   1 root supergroup       1089 2024-12-15 12:06 /user/hive/warehouse/weather/20241215120604641_weather_data.avro
vagrant@node1:~$
```

```
hive> select * from weather;
OK
51.5 -0.120000124 0.04398822784423828 0 GMT GMT 17.0 2024-12-15T11:45 900 10.5 11.8 0.0 90 91
53.302002 2.204 0.024080276489257812 0 GMT GMT 0.0 2024-12-15T11:45 900 9.4 42.1 0.0 97 100
51.5 -0.120000124 0.06091594696044922 0 GMT GMT 17.0 2024-12-15T12:00 900 10.6 12.1 0.0 89 98
53.302002 2.204 0.03600120544433594 0 GMT GMT 0.0 2024-12-15T12:00 900 9.4 43.2 0.0 97 100
4 rows selected (2.206 seconds)
```

- Tabela została zasilona nowymi danymi (dla godziny 12:00).

- Sprawdźmy czy możemy przeprowadzać analizy na tabeli.

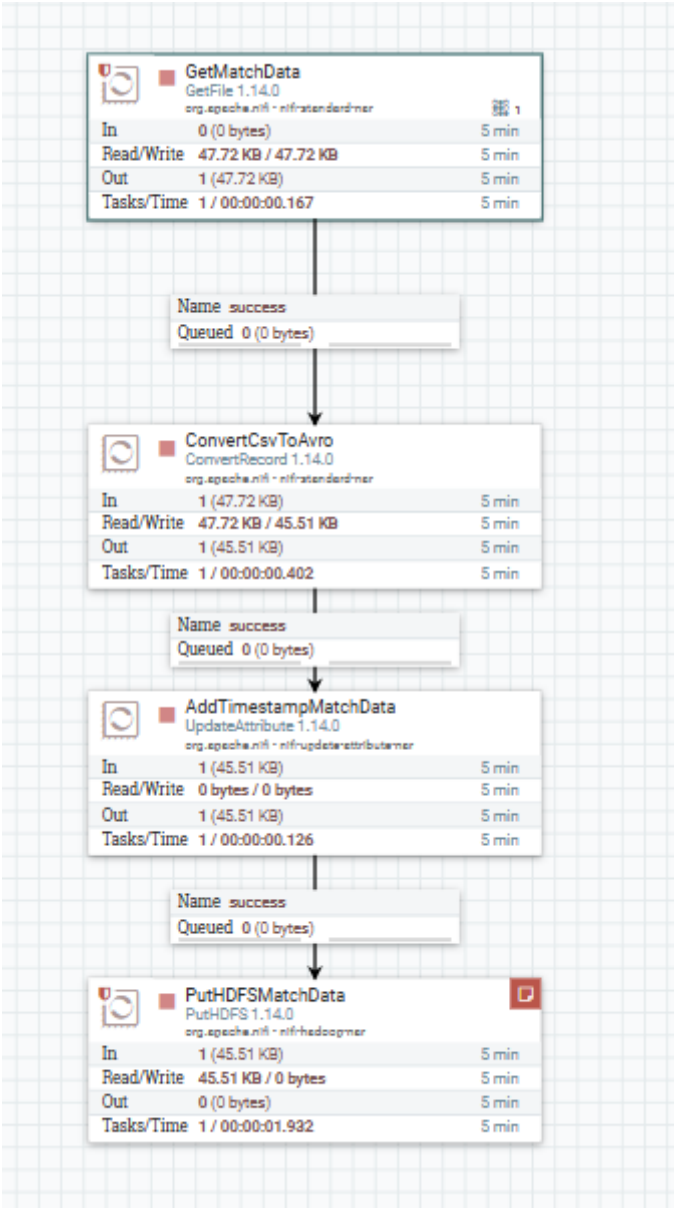
```
hive> select min(current_temperature_2m) from weather;
```

```
Stage-Stage-1:  HDFS Read: 0 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
9.4
1 row selected (42.032 seconds)
```

Składowanie danych z meczów piłkarskich:

Dane z meczów piłkarskich ładujemy w kolejnych porcjach (batchach). Na podstawie pobranych danych historycznych meczów Premier League zasymulowaliśmy napływ danych z meczów co minutę. (np. w 1 minucie meczu mamy 10 wykonanych podań, w 2 minucie 18 wykonanych podań itd.) Wyliczyliśmy je na podstawie rozkładów statystycznych. Pobieramy pliki CSV z lokalnego systemu plików i tak jak w przypadku danych pogodowych ładujemy je do folderu hdfs, z którego dane są zaciągane do EXTERNAL TABLE matches w Hive.

Procesory NiFi:



1. GetFile

- Pobieramy dane z lokalnego systemu plików

2. ConvertRecord

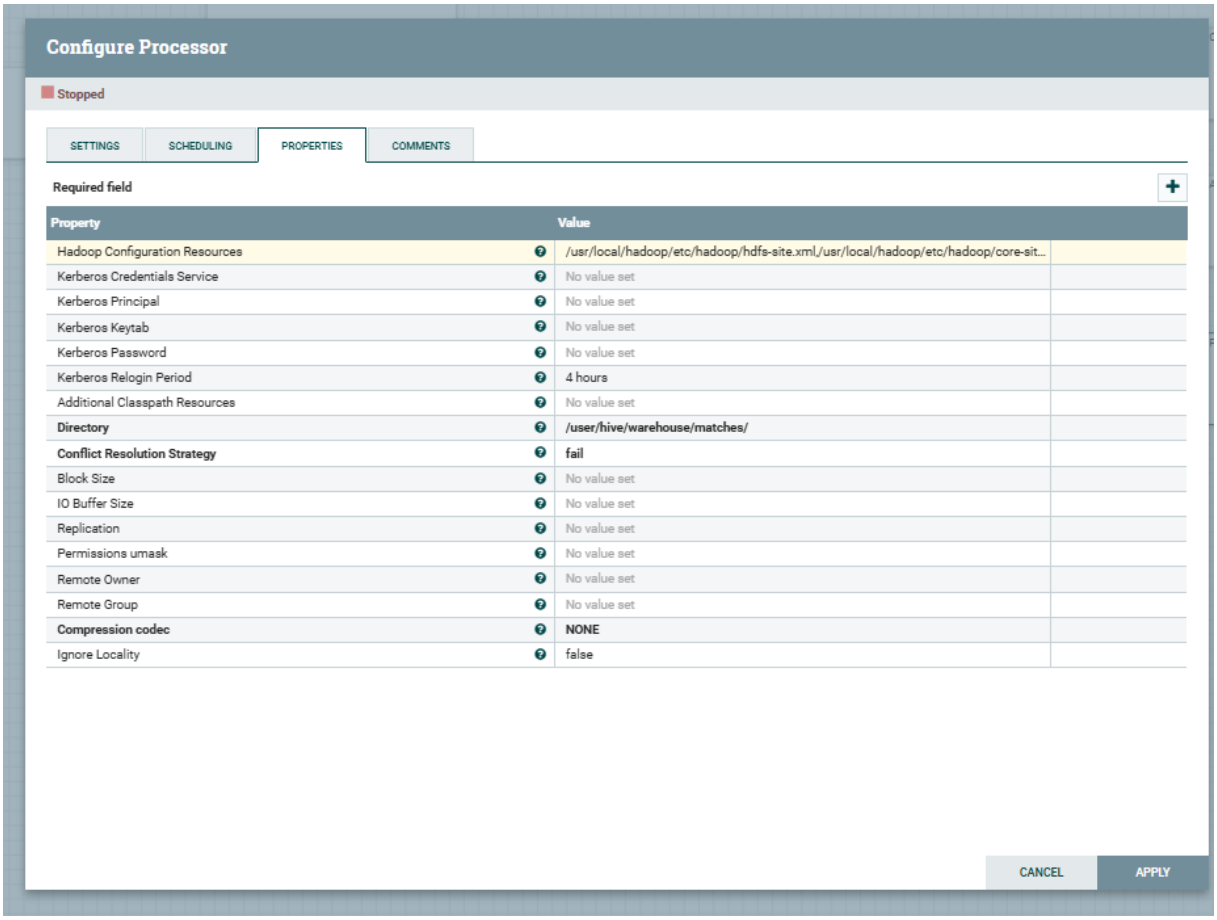
- Konwertujemy plik CSV na AVRO. W konfiguracji CsvReader ustawiamy Treat First Line as Header na true, Value Separator na ‘,’.

3. UpdateAttribute

- Ustawiamy odpowiednią nazwę pliku wzbogaconą o timestamp.

4.PutHFDS

- Ładujemy dane do odpowiedniego folderu w HDFS

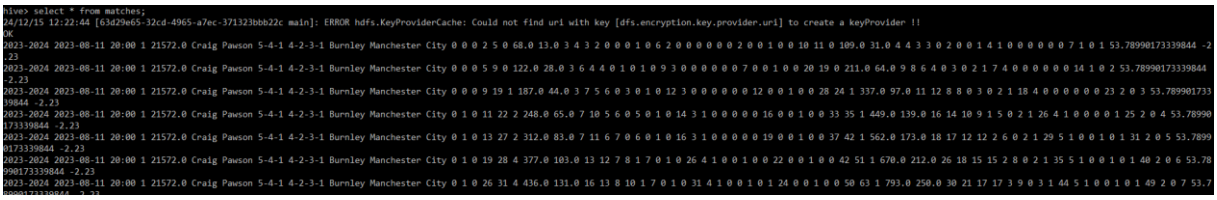



```
vagrant@node1:~$ hdfs dfs -ls /user/hive/warehouse/matches
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.6/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.c
lass]
SLF4J: Found binding in [jar:file:/usr/local/apache-tez-0.9.1-bin/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Found 1 items
-rw-r--r--   1 root supergroup      46603 2024-12-15 12:17 /user/hive/warehouse/matches/20241215121546798_simulated_matches_round.avro
```

```
vagrant@node1:~$ hdfs dfs -ls /user/hive/warehouse/matches
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.6/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerB
lass]
SLF4J: Found binding in [jar:file:/usr/local/apache-tez-0.9.1-bin/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Found 3 items
-rw-r--r--   1 root supergroup      46603 2024-12-15 12:17 /user/hive/warehouse/matches/20241215121546798_simulated_matches_round.avro
-rw-r--r--   1 root supergroup      46603 2024-12-15 12:32 /user/hive/warehouse/matches/20241215123138430_simulated_matches_round.avro
-rw-r--r--   1 root supergroup      46603 2024-12-15 12:35 /user/hive/warehouse/matches/20241215123543152_simulated_matches_round.avro
```

-Tworzymy EXTERNAL TABLE matches w Hive.

```
hive> CREATE EXTERNAL TABLE matches (
. . >     season STRING,
. . >     `date` STRING,
. . >     time STRING,
. . >     round INT,
. . >     attendance_value DOUBLE,
. . >     referee STRING,
. . >     formation_home STRING,
. . >     formation_away STRING,
. . >     home_team STRING,
. . >     away_team STRING,
. . >     home_goals INT,
. . >     home_shots INT,
. . >     home_shots_on_target INT,
. . >     home_passes_completed INT,
. . >     home_passes INT,
. . >     home_progressive_passes INT,
. . >     home_passes_total_distance DOUBLE,
. . >     home_passes_progressive_distance DOUBLE,
. . >     home_passes_completed_short INT,
. . >     home_passes_short INT,
. . >     home_passes_completed_medium INT,
. . >     home_passes_medium INT,
. . >     home_passes_completed_long INT,
. . >     home_passes_long INT,
. . >     home_assisted_shots INT,
. . >     home_passes_into_final_third INT,
. . >     home_passes_into_penalty_area INT,
. . >     home_passes_live INT,
. . >     home_passes_dead INT,
. . >     home_passes_free_kicks INT,
. . >     home_passes_switches INT,
. . >     home_passes_offsides INT,
. . >     home_passes_blocked INT,
. . >     home_blocked_shots INT,
. . >     home_blocked_passes INT,
. . >     home_passes_received INT,
. . >     home_progressive_passes_received INT,
. . >     home_own_goals INT,
. . >     away_goals INT,
. . >     away_shots INT,
. . >     away_shots_on_target INT,
. . >     away_passes_completed INT,
. . >     away_passes INT,
. . >     away_progressive_passes INT,
. . >     away_passes_total_distance DOUBLE,
. . >     away_passes_progressive_distance DOUBLE,
. . >     away_passes_completed_short INT,
. . >     away_passes_short INT,
. . >     away_passes_completed_medium INT,
. . >     away_passes_medium INT,
. . >     away_passes_completed_long INT,
. . >     away_passes_long INT,
. . >     away_assisted_shots INT,
. . >     away_passes_into_final_third INT,
. . >     away_passes_into_penalty_area INT,
. . >     away_passes_live INT,
. . >     away_passes_dead INT,
. . >     away_passes_free_kicks INT,
. . >     away_passes_switches INT,
. . >     away_passes_offsides INT,
. . >     away_passes_blocked INT,
. . >     away_blocked_shots INT,
. . >     away_blocked_passes INT,
. . >     away_passes_received INT,
. . >     away_progressive_passes_received INT,
. . >     away_own_goals INT,
. . >     minute INT,
. . >     latitude DOUBLE,
. . >     longitude DOUBLE
. . > )
. . > STORED AS AVRO
. . > LOCATION '/user/hive/warehouse/matches';
OK
No rows affected (1.128 seconds)
```



- Sprawdzamy, czy możemy wykonać analizy na tabeli

```
hive> select count(*) from matches where home_team="Arsenal";
```

```
Stage-Stage-1:  HDFS Read: 0 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
90
1 row selected (70.588 seconds)
hive>
```

Transformacja danych

Dane z Hive zostały wczytanie do sparka i tam przetransformowane.

Wczytanie danych z tabel Hive

[4]:
```
matches=spark.sql(f"""
    SELECT *
    FROM matches
""")

weather=spark.sql(f"""
    SELECT *
    FROM weather
""")
```

[5]:
```
matches.printSchema()
```

```
root
 |-- season: string (nullable = true)
 |-- date: string (nullable = true)
 |-- time: string (nullable = true)
 |-- round: integer (nullable = true)
 |-- attendance_value: double (nullable = true)
 |-- referee: string (nullable = true)
 |-- formation_home: string (nullable = true)
 |-- formation_away: string (nullable = true)
 |-- home_team: string (nullable = true)
 |-- away_team: string (nullable = true)
 |-- home_goals: integer (nullable = true)
 |-- home_shots: integer (nullable = true)
 |-- home_shots_on_target: integer (nullable = true)
 |-- home_passes_completed: integer (nullable = true)
 |-- home_passes: integer (nullable = true)
 |-- home_progressive_passes: integer (nullable = true)
 |-- home_passes_total_distance: double (nullable = true)
```

```
[8]: weather.show(5)
```

```
+---------+-----------+------------------+-----------------+--------+---------------------+---------+---------------+----------------+------------
---------+--------------------+--------------------+--------------------------+-----------------+
| latitude|  longitude|   generationtime_ms|utc_offset_seconds|timezone|timezone_abbreviation|elevation|    current_time|current_interval|current_tempe
rature_2m|current_wind_speed_10m|current_precipitation|current_relative_humidity_2m|current_cloud_cover|
+---------+-----------+------------------+-----------------+--------+---------------------+---------+---------------+----------------+------------
---------+--------------------+--------------------+--------------------------+-----------------+
|     51.5|-0.120000124| 0.04398822784423828|                0|     GMT|                  GMT|     17.0|2024-12-15T11:45|             900|
10.5|                  11.8|                 0.0|                          90|                 91|
|53.302002|       2.204|0.024080276489257812|                0|     GMT|                  GMT|      0.0|2024-12-15T11:45|             900|
9.4|                  42.1|                 0.0|                          97|                100|
|     51.5|-0.120000124| 0.06091594696044922|                0|     GMT|                  GMT|     17.0|2024-12-15T12:00|             900|
10.6|                  12.1|                 0.0|                          89|                 98|
|53.302002|       2.204| 0.03600120544433594|                0|     GMT|                  GMT|      0.0|2024-12-15T12:00|             900|
9.4|                  43.2|                 0.0|                          97|                100|
|     51.5|-0.120000124| 0.03898143768310547|                0|     GMT|                  GMT|     17.0|2024-12-15T12:30|             900|
10.9|                  12.6|                 0.0|                          88|                 98|
+---------+-----------+------------------+-----------------+--------+---------------------+---------+---------------+----------------+------------
---------+--------------------+--------------------+--------------------------+-----------------+
```

Na szczególną uwagę zasługuje transformacja danych pogodowych. Zapytanie API zwracało dane w specyficznej strukturze dlatego kluczowe było ich przetransformowanie do postaci odpowiedniej dla dalszej analizy. Po transformacji sprawdziliśmy, że przebiegła ona pomyślnie.

Przetwarzanie danych pogodowych

```
[9]: from pyspark.sql import functions as F

transformed_weather = weather.select(
    F.date_format(F.col('current_time'), 'yyyy-MM-dd').alias('date'),
    F.date_format(F.col('current_time'), 'HH:mm').alias('time'),
    F.col('longitude'),
    F.col('latitude'),
    F.col('current_temperature_2m').alias('weather_temperature'),
    F.col('current_precipitation').alias('weather_precipitation'),
    F.col('current_wind_speed_10m').alias('weather_wind'),
    F.col('current_relative_humidity_2m').alias('weather_humidity'),
    F.col('current_cloud_cover').alias('weather_cloud_cover')
)
```

```
10]: transformed_weather.show(5)
```

```
+----------+-----+-----------+---------+-------------------+---------------------+------------+----------------+-------------------+
|      date| time|  longitude| latitude|weather_temperature|weather_precipitation|weather_wind|weather_humidity|weather_cloud_cover|
+----------+-----+-----------+---------+-------------------+---------------------+------------+----------------+-------------------+
|2024-12-15|11:45|-0.120000124|     51.5|               10.5|                  0.0|        11.8|              90|                 91|
|2024-12-15|11:45|       2.204|53.302002|                9.4|                  0.0|        42.1|              97|                100|
|2024-12-15|12:00|-0.120000124|     51.5|               10.6|                  0.0|        12.1|              89|                 98|
|2024-12-15|12:00|       2.204|53.302002|                9.4|                  0.0|        43.2|              97|                100|
|2024-12-15|12:30|-0.120000124|     51.5|               10.9|                  0.0|        12.6|              88|                 98|
+----------+-----+-----------+---------+-------------------+---------------------+------------+----------------+-------------------+
```

Następnie dane meczowe zostały połączone z danymi pogodowymi.

```
[25]: updated_matches = updated_matches.withColumnRenamed('date', 'match_date') \
                                 .withColumnRenamed('time', 'match_time') \
                                 .withColumnRenamed('longitude', 'match_longitude') \
                                 .withColumnRenamed('latitude', 'match_latitude')

updated_weather = updated_weather.withColumnRenamed('date', 'weather_date') \
                                 .withColumnRenamed('time', 'weather_time') \
                                 .withColumnRenamed('longitude', 'weather_longitude') \
                                 .withColumnRenamed('latitude', 'weather_latitude')

matches_with_weather = updated_matches.join(
    updated_weather,
    (updated_matches['match_date'] == updated_weather['weather_date']) &
    (updated_matches['match_time'] == updated_weather['weather_time']) &
    (updated_matches['match_longitude'] == updated_weather['weather_longitude']) &
    (updated_matches['match_latitude'] == updated_weather['weather_latitude']),
    'inner'
)
```

```
[26]: matches_with_weather = matches_with_weather.drop('weather_date', 'weather_time', 'weather_longitude', 'weather_latitude')
```

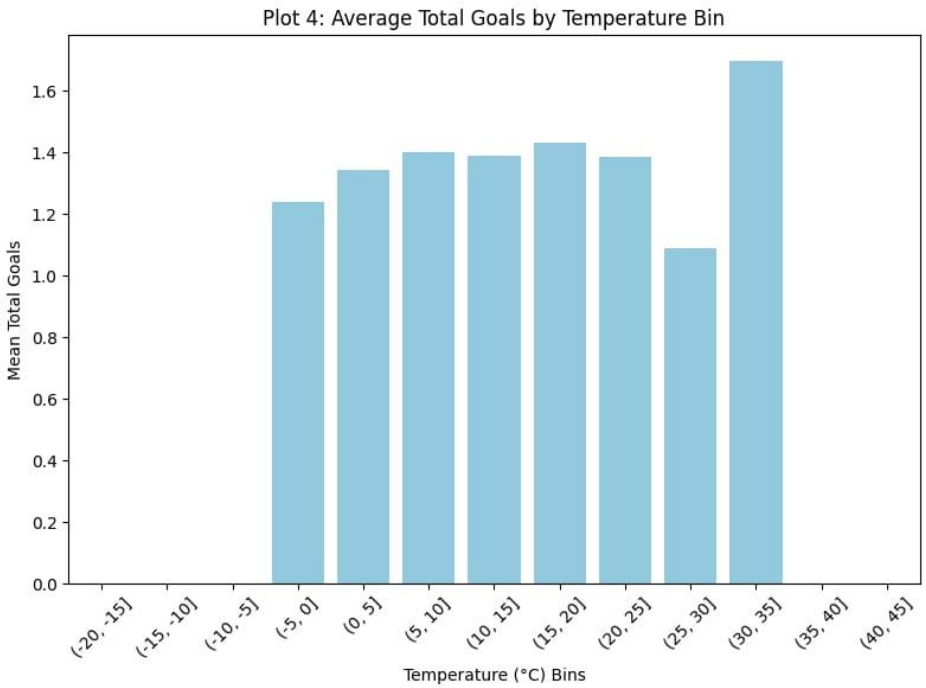
Kolejnym krokiem było załadowanie danych do Hbase:

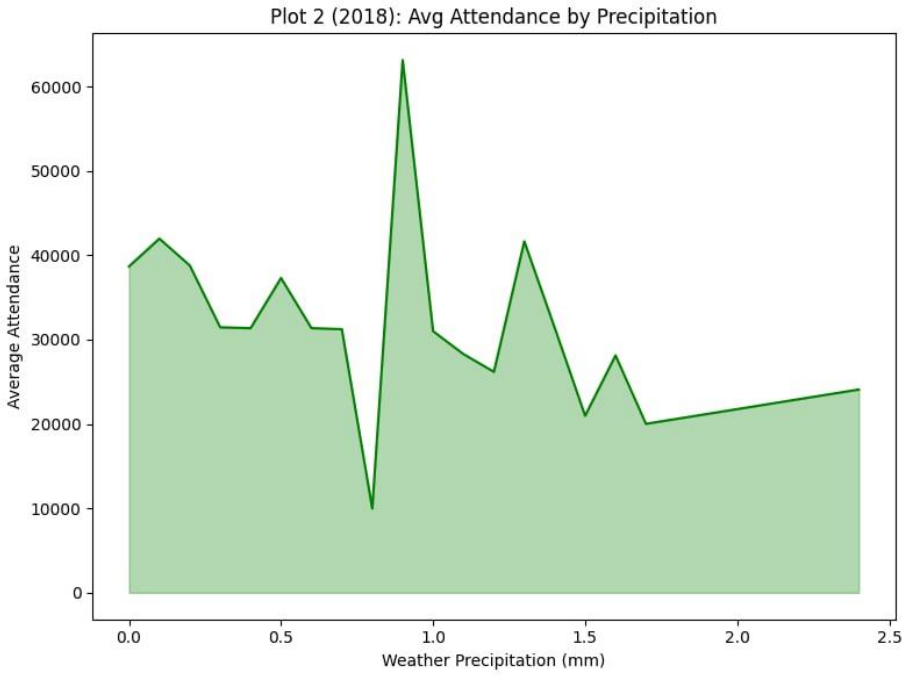
```
[5]: table = connection.table('matches_data')

[*]: table = connection.table('matches_data')
for key, data in table.scan():
    print(f'KEY: {key}')
    for column, value in data.items():
        print(f'\tCOLUMN: {column} VALUE: {value}')
```

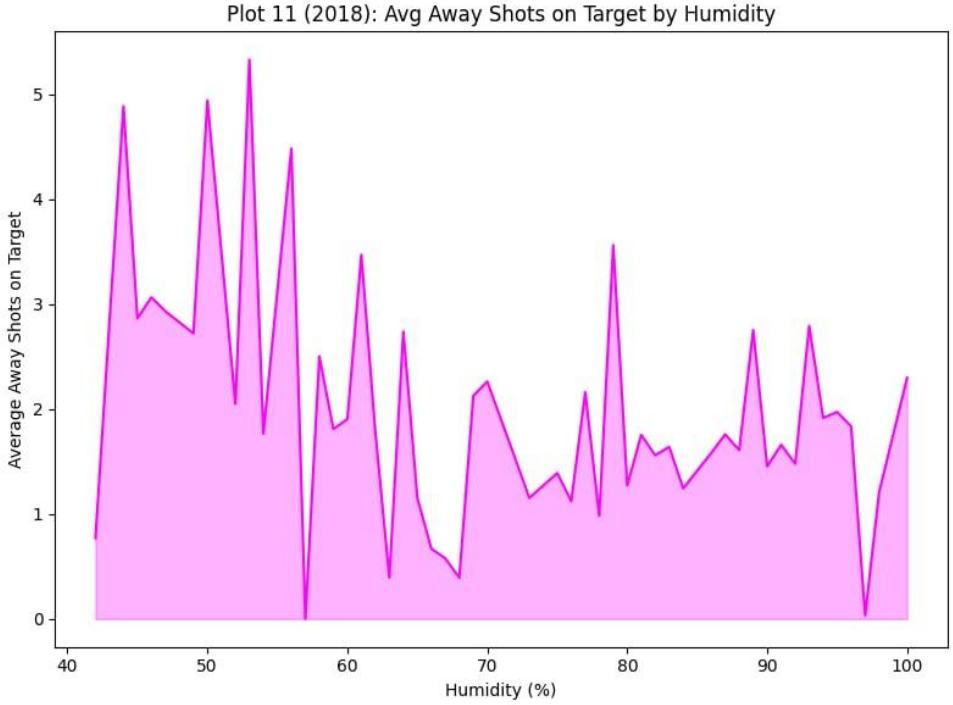
```
KEY: b'0000be23-ac12-4a4c-8486-47700a844c00'
        COLUMN: b'location:match_latitude' VALUE: b'51.3983'
        COLUMN: b'location:match_longitude' VALUE: b'-0.0856'
        COLUMN: b'match_info:attendance_value' VALUE: b'24741.0'
        COLUMN: b'match_info:match_date' VALUE: b'2023-09-03'
        COLUMN: b'match_info:match_time' VALUE: b'14:00'
        COLUMN: b'match_info:referee' VALUE: b'Robert Jones'
        COLUMN: b'match_info:round' VALUE: b'4'
        COLUMN: b'match_info:season' VALUE: b'2023-2024'
        COLUMN: b'match_timeline:minute' VALUE: b'16'
        COLUMN: b'stats:away_assisted_shots' VALUE: b'0'
        COLUMN: b'stats:away_goals' VALUE: b'0'
        COLUMN: b'stats:away_own_goals' VALUE: b'0'
        COLUMN: b'stats:away_passes' VALUE: b'98'
        COLUMN: b'stats:away_passes_completed' VALUE: b'76'
        COLUMN: b'stats:away_passes_into_final_third' VALUE: b'5'
        COLUMN: b'stats:away_passes_into_penalty_area' VALUE: b'0'
        COLUMN: b'stats:away_passes_progressive_distance' VALUE: b'452'
```

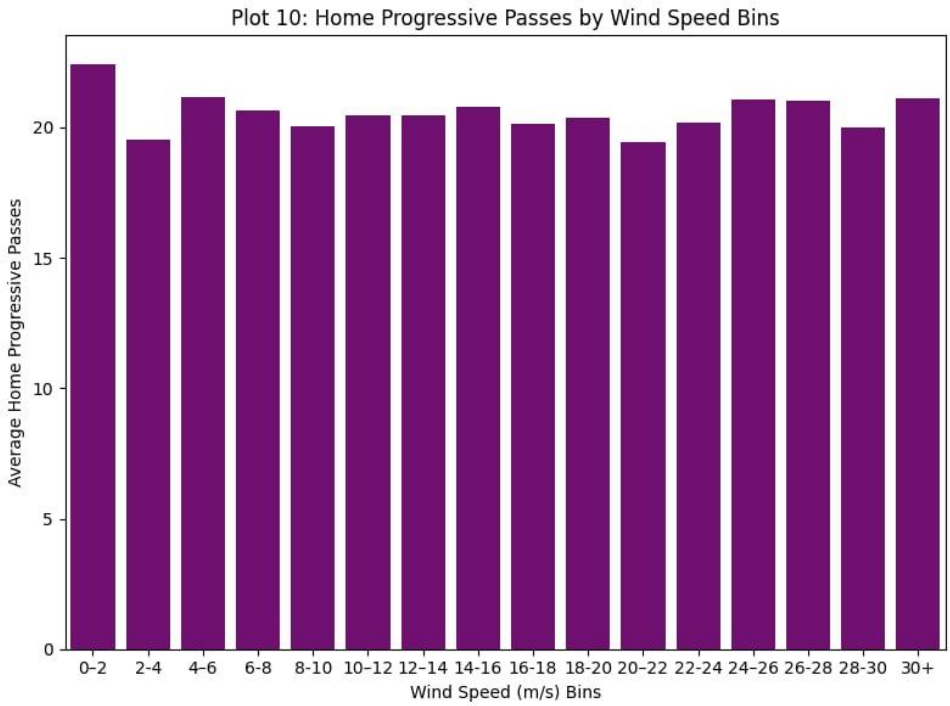
Wizualizacje

Podczas wykonywania wizualizacji skupiliśmy się przede wszystkim na sprawdzeniu jak przedstawiają się różne statystyki piłkarskie w zależności od pogody. Przykładowe wykresy omówione na prezentacji:




Plot 2 (2018): Avg Attendance by Precipitation
Average Attendance
60000
50000
40000
30000
20000
10000
0
0.0
0.5
1.0
1.5
2.0
2.5
Weather Precipitation (mm)


Plot 11 (2018): Avg Away Shots on Target by Humidity
Average Away Shots on Target
5
4
3
2
1
0
40
50
60
70
80
90
100
Humidity (%)


Plot 10: Home Progressive Passes by Wind Speed Bins
Average Home Progressive Passes
0
5
10
15
20
0–2
2-4
4–6
6-8
8-10
10–12
12–14
14-16
16-18
18-20
20–22
22-24
24–26
26-28
28-30
30+
Wind Speed (m/s) Bins