Big Data Application Final Project - FIFA World Cup

Introduction: Find the data

I like soccer a lot, so I decided to work on something related to soccer. I thought that since the world cup is coming soon, I can do something related to world cup soccer clubs. After searching around online, I saw most of them related to players. However, after searching for an hour and half, I found a csv file that records the data about each world cup team since 1993. One small issue with this dataset is that some columns only have zeros before 8/24/2011, which I believe it's because these weren't calculated back then.

Wikipedia:
"In the 2010s, teams realised the ranking system could be 'gamed', specifically by avoiding playing non-competitive matches, particularly against weaker opponents.[32] This was because the low weighting of friendlies meant that even victories could reduce a team's average score: in other words, a team could win a match and lose points."

Therefore, I will keep that in mind and be careful with how I run aggregation and groups. The file is 5MB (16 columns * 57793 rows), which is not large because the world cup is not as frequent as flights, but I will try to extract useful information from it using our cluster. One might think since world cup is once every 4 years, the data should be even smaller. But games around the countries happen all the time, the rank of countries are not only based on the world cup.

Link to data:
https://www.kaggle.com/datasets/tadhgfitzgerald/fifa-international-soccer-mens-ranking-1993now

Link that explains how the points work:
https://en.wikipedia.org/wiki/FIFA_Men%27s_World_Ranking

The data has important information like rank, total points of each counter, the rank change and so on. These data can help us see which teams are strong and strong consistently. We can potentially find out some patterns through this data and create an interesting app to let users see these results and make judgments on the upcoming world cup in the U.S. We also use it to do things like "What's the strongest team in 2008?" The strongest team is not always the one that wins the world cup. We can also do things like "Which team is consistently strong?" These

are questions I want to try. For the speed layer, I will see if I can let users submit new results (fake results, assumed by users) on teams so that they can get new information based on their alternative knowledge.

Step 1: Put the data onto the cluster

Download the csv file on my own laptop first.

Now let's first create a directory on cluster:
mkdir -p ~/wuh_fifa_project

Move the csv file to the cluster:
scp /where_it_is_on_my_laptop/fifa_ranking.csv
hadoop@ec2-34-230-47-10.compute-1.amazonaws.com:~/wuh_fifa_project/

Next, we want to move it to HDFS.

We first create an HDFS directory for the raw CSV:
hdfs dfs -mkdir -p /inputs/wuh_fifa

Move it from cluster to HDFS:
hdfs dfs -put ~/wuh_fifa_project/fifa_ranking.csv /inputs/wuh_fifa/

Step 2: Create a Hive table

Now I want to create a table in Hive so that I can write further queries. We learnt Hive from lecture 3, so I mainly referenced files from there.

My data has total of 16 columns:
rank
country_full
country_abrv
total_points
previous_points
rank_change
cur_year_avg
cur_year_avg_weighted
last_year_avg
last_year_avg_weighted
two_year_ago_avg
two_year_ago_weighted
three_year_ago_avg
three_year_ago_weighted
confederation

rank_date

The columns that are marked in bold have only zeros before 8/24/2011. But we will create our table with them in the first place, but we probably need to treat them differently during aggregation.

Again, they are:
total_points
cur_year_avg
cur_year_avg_weighted
last_year_avg
last_year_avg_weighted
two_year_ago_avg
two_year_ago_weighted
three_year_ago_avg
Three_year_ago_weighted

Create an external table for my data lake. I named it wuh_fifa_rankings_csv.

Open hive: hive

Drop the table since the csv file if the table is wrong:
hive> DROP TABLE IF EXISTS wuh_fifa_rankings_csv;

Create the table:
hive>
```
CREATE EXTERNAL TABLE IF NOT EXISTS wuh_fifa_rankings_csv (
   rank INT,
   country_full STRING,
   country_abrv STRING,
   total_points DOUBLE,
   previous_points DOUBLE,
   rank_change INT,
   cur_year_avg DOUBLE,
   cur_year_avg_weighted DOUBLE,
   last_year_avg DOUBLE,
   last_year_avg_weighted DOUBLE,
   two_year_ago_avg DOUBLE,
   two_year_ago_weighted DOUBLE,
   three_year_ago_avg DOUBLE,
   three_year_ago_weighted DOUBLE,
   confederation STRING,
   rank_date STRING
)
```

```
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION '/inputs/wuh_fifa'
TBLPROPERTIES ("skip.header.line.count"="1");
```

We must jump over the first row because the header of the csv file is the first row.

Check if the table is there:
```
hive> SELECT * FROM wuh_fifa_rankings_csv LIMIT 10;
```

Check the header:
```
hive> DESCRIBE wuh_fifa_rankings_csv;
```

All checked, now we have changed the csv to a hive table on our cluster.

Step 2: Convert to an ORC format

If we still have world cup in a thousand years or we add more columns in the future, we might have terabytes of data. Therefore, we make it ORC to gain efficiency for reading columns.

Let's create: wuh_fifa_rankings_orc
```
CREATE TABLE wuh_fifa_rankings_orc
STORED AS ORC
AS
SELECT * FROM wuh_fifa_rankings_csv;
```

With the ORC format, it would take me less time to create my aggregation tables.

Step 3: Aggregation Tables

We want to have some meaningful tables that can help us get the most out of the dataset. We need to keep notice of the following things:
1. Our work is across the entire 1993–2018 dataset.
2. Handle pre-2011 missing numeric values cleanly.
3. Meaningful results that can be queried by year/team throughout time.

So we see that most of our questions will either be answered by a certain year or answered throughout all the time. I therefore come up with 2 tables: wuh_fifa_yearly_team_stats and wuh_fifa_team_batch_stats.

With wuh_fifa_yearly_stats, we then will be able to know in each year, how did the teams perform:
```
CREATE TABLE wuh_fifa_yearly_team_stats AS
SELECT
```

```
    country_full AS team,
    confederation,
    year(to_date(rank_date)) AS year,
    MIN(rank) AS best_rank_this_year,
    MAX(rank) AS worst_rank_this_year,
    AVG(rank) AS avg_rank_this_year,
    AVG(ABS(rank_change)) AS avg_rank_volatility_this_year
FROM wuh_fifa_rankings_orc
GROUP BY
    country_full,
    confederation,
    year(to_date(rank_date));
```

wuh_fifa_yearly_team_stats can help me with questions like:

- “Which team was strongest in 2015?”
- “Which confederation dominated in 2000?”
- “Which teams improved from 2005 → 2010?”

wuh_fifa_yearly_team_stats will also allow me to filter by year range.

With wuh_fifa_team_batch_stats, we get to get a very aggregated result that shows performance throughout the time:

```
CREATE TABLE wuh_fifa_team_batch_stats AS
WITH confed AS (
    SELECT
        confederation,
        AVG(CASE WHEN total_points > 0 THEN total_points END) AS confed_avg_points
    FROM wuh_fifa_rankings_orc
    GROUP BY confederation
)
SELECT
    r.country_full AS team,
    r.confederation,
    MIN(r.rank) AS best_rank,
    MAX(r.rank) AS worst_rank,
    AVG(ABS(r.rank_change)) AS avg_abs_rank_change,
    MIN(year(to_date(r.rank_date))) AS first_year_active,
    MAX(year(to_date(r.rank_date))) AS last_year_active,
    COUNT(DISTINCT year(to_date(r.rank_date))) AS total_years_ranked,
    c.confed_avg_points
FROM wuh_fifa_rankings_orc r
LEFT JOIN confed c
    ON r.confederation = c.confederation
GROUP BY
    r.country_full, r.confederation, c.confed_avg_points;
```

It has the following features:

- Zero values before 2011 don't harm aggregates.
- Team-level stability questions become easy.
- Confederation comparisons work.
- Time-range analytics are possible

There are 6 confederations in FIFA world cup (AFC (Asia), CAF (Africa), CONCACAF (North, Central America, and the Caribbean), CONMEBOL (South America), OFC (Oceania), and UEFA (Europe)), so we might also want to know which confederation is strong and how did them evolve. We then have a table named wuh_fifa_confed_yearly_stats:

```
CREATE TABLE wuh_fifa_confed_yearly_stats AS
SELECT
   confederation,
   year(to_date(rank_date)) AS year,
   AVG(rank) AS avg_rank_confed,
   COUNT(*) AS num_teams_ranked
FROM wuh_fifa_rankings_orc
GROUP BY confederation, year(to_date(rank_date));
```

This allows me to quickly get results like:

- "Which confederation was strongest in 2002?"
- "How did UEFA's strength evolve over time?"

Up until now, we have 2 raw tables:

- wuh_fifa_rankings_csv
- Wuh_fifa_rankings_orc

We then generated 3 more tables:

- wuh_fifa_yearly_team_stats
- wuh_fifa_team_batch_stats
- wuh_fifa_confed_yearly_stats

Step 3: Move to HBase

Let's get the serving layer ready with what we had in lecture 4 so that my Node.js app will query HBase's REST API. The HBase batch view should have one row per team and row key using team name. It should also be the baseline for the upcoming speed layer. Let's only push 1 Hive table to HBase first and see if it works well. I picked wuh_fifa_team_batch_stats because it has the question I care about the most - "What's the best team?" If it works well, we can get wuh_fifa_yearly_team_stats and wuh_fifa_confed_yearly_stats to HBase next.

We do this: wuh_fifa_team_batch_stats -> wuh_fifa_team_stats (HBase)

By lecture 4 we create it as following:

hbase shell
create 'wuh_fifa_team_stats', 'stats'
Under wuh_fifa_project on cluster:
nano wuh_write_to_hbase.hql

The file wuh_write_to_hbase.hql is in repo, quite similar to what we had in lec_4_files's write_to_hbase.hql.

Run:
hive -f ~/wuh_fifa_project/wuh_write_to_hbase.hql

Check the HBase:
base:001:0> scan 'wuh_fifa_team_stats', {LIMIT => 10}

I see sorted by the alphabet of countries.

Step 4: Work on the app that connects to HBase

I have modified the original app to connect with what I have. The key files are:
app.js, team_result.mustache, public/index.html, submit-event.html.

If you search China, it will display an HBase Error because China is not in the database. To resolve this issue I use "hclient.table('wuh_fifa_team_stats').scan({ maxVersions: 1 }" so that I can get all the countries' names.
So there are 2 ways to see the result.

1. One is by going to "View all Available Countries" and browse for the country you are looking for there.
2. Two is by knowing the country name like how we knew the airport name and entering it.

Either way will show the same results.

If you search Kosovo and press enter, you will see its first ranking is in 2016:

Stats for Kosovo

Statistic	Value
Best Rank	141
Worst Rank	190
First Ranking Year	2016
Last Ranking Year	2018
Total Years Ranked	3
Avg Absolute Rank Change	4.208333333333333
Confederation Avg Points	651.9654763536685

Back

This is because Kosovo joined FIFA in 2016. This verifies the correctness of my HBase table.

The submit-event.html will let users enter rank and points, the idea is to simply have a speed layer that runs and see how it works. For now, it will tell you it's submitted but actually nothing happens. After we get the speed layer, we have ways to test:

1. Create a new country and see if we can search it.
2. Update the original data with a large number and see if anything changes.

Step 4: Build Speed Layer

HBase table for speed layer. Spark streaming will populate it.
In HBase shell:
create 'wuh_fifa_events_log', 'e'

We name the kafka topic: wuh_fifa_events

app.js update:
We need kafkajs to do this. The app is more like the app we have in lec_8_files.

For streaming, we take speed-layer-weather-archetype and replace the scala file with StreamFifaEvents.scala and deploy it. Remember to not leave it running. Submit the uber jar to wuh_fifa_project. I had it named untitled because it was on my local laptop.

Start the kafka topic:

```
kafka-topics.sh --create \
  --bootstrap-server
boot-publicbyg.mpcs53014kafka.2siu49.c2.kafka.us-east-1.amazonaws.com:9196 \
  --command-config ~/kafka.client.properties \
  --replication-factor 3 \
  --partitions 3 \
  --topic wuh_fifa_events
```

I have the scala file that streams named StreamFifaEvents.scala in my repo and put is jar under wuh_fifa_project, event though it's named untitled. I also updated the pom.xml file's spark dependency to match with what we have on cluster. I saw some errors reported about dependencies.

In /home/hadoop, run:

```
spark-submit \
  --driver-java-options "-Dlog4j.configuration=file:///home/hadoop/log4j.properties" \
  --master local[2] \
  --class StreamFifaEvents \
  --files /etc/hbase/conf/hbase-site.xml \
  wuh_fifa_project/target/uber-untitled-1.0-SNAPSHOT.jar \
  b-3-public.mpcs53014kafka.2siu49.c2.kafka.us-east-1.amazonaws.com:9196
```

It's been run successfully but since kafka topic is not created, it keeps reporting warnings and errors.

I then see someone downloaded python on a cluster so I tried to use python to create a kafka topic which worked. I put the python script in create-topic.py in my repo.

Step 5: Deploy the web app
Use what we had in lecture 8 to copy the app to the server.

My port is 3001:
pm2 start app.js --name wuh_app -- 3001
http://ec2-34-230-47-10.compute-1.amazonaws.com:8070/

I have successfully put it on port 3001 and can open at:
http://ec2-52-20-203-80.compute-1.amazonaws.com:3001/

Checked Kafka as I submitted Brazil:
[hadoop@ip-172-31-91-77 bin]$
kafka-console-consumer.sh \
  --bootstrap-server
boot-public-byg.mpcs53014kafka.2siu49.c2.kafka.us-east-1.amazonaws.com:9196 \
  --consumer.config ~/kafka.client.properties \
  --topic wuh_fifa_events \
  --from-beginning

{"country":"Brazil","new_rank":100,"points":200,"ts":1765063390411}

To remove it:
pm2 delete wuh_app

To check logs:
pm2 logs wuh_app

To show and to kill spark:
yarn application -list
yarn application -kill application_id

Check logs of spark:
yarn logs -applicationId application_1763001378295_1138 | grep Updated

To check the updates from speed layer:
hbase:008:0> get 'wuh_fifa_team_stats', 'Brazil'
COLUMN   CELL
 stats:avg_abs_rank_change   timestamp=2025-12-05T06:37:26.535, value=0.5

```
 stats:best_rank                    timestamp=2025-12-05T06:37:26.535, value=1
 stats:confed_avg_points            timestamp=2025-12-05T06:37:26.535,
value=929.0054096385551
 stats:first_year_active            timestamp=2025-12-05T06:37:26.535, value=1993
 stats:last_year_active             timestamp=2025-12-05T06:37:26.535, value=2018
 stats:latest_live_points           timestamp=2025-12-07T05:45:18.566,
value=@\x81X\x00\x00\x00\x00\x00
 stats:latest_live_rank             timestamp=2025-12-07T05:45:18.566, value=\x00\x00\x00o
 stats:latest_live_ts               timestamp=2025-12-07T05:45:18.566,
value=\x00\x00\x01\x9A\xF6C5\xB7
 stats:total_years_ranked           timestamp=2025-12-05T06:37:26.535, value=26
 stats:worst_rank                   timestamp=2025-12-05T06:37:26.535, value=22
1 row(s)
Took 0.0072 seconds
hbase:009:0>
hbase:010:0> scan 'wuh_fifa_events_log'
ROW                                COLUMN+CELL
 Brazil#1765063390411              column=e:points, timestamp=2025-12-07T05:45:18.561,
value=@i\x00\x00\x00\x00\x00\x00
 Brazil#1765063390411              column=e:rank, timestamp=2025-12-07T05:45:18.561,
value=\x00\x00\x00d
 Brazil#1765063390411              column=e:ts, timestamp=2025-12-07T05:45:18.561,
value=\x00\x00\x01\x9A\xF5\xF9\xF4\xCB
 Brazil#1765064093954              column=e:points, timestamp=2025-12-07T05:45:18.410,
value=@r\xC0\x00\x00\x00\x00\x00\x00
 Brazil#1765064093954              column=e:rank, timestamp=2025-12-07T05:45:18.410,
value=\x00\x00\x00\x96
 Brazil#1765064093954              column=e:ts, timestamp=2025-12-07T05:45:18.410,
value=\x00\x00\x01\x9A\xF6\x04\xB1\x02
 Brazil#1765066744671              column=e:points, timestamp=2025-12-07T05:45:18.561,
value=@t\xD0\x00\x00\x00\x00\x00\x00
 Brazil#1765066744671              column=e:rank, timestamp=2025-12-07T05:45:18.561,
value=\x00\x00\x00\xDE
 Brazil#1765066744671              column=e:ts, timestamp=2025-12-07T05:45:18.561,
value=\x00\x00\x01\x9A\xF6-#_
 Brazil#1765068191159              column=e:points, timestamp=2025-12-07T05:45:18.565,
value=@\x81X\x00\x00\x00\x00\x00\x00
 Brazil#1765068191159              column=e:rank, timestamp=2025-12-07T05:45:18.565,
value=\x00\x00\x00o
 Brazil#1765068191159              column=e:ts, timestamp=2025-12-07T05:45:18.565,
value=\x00\x00\x01\x9A\xF6C5\xB7
```