IPL DATA ANALYSIS
Using Hadoop Ecosystem & Power BI

Group members

Archana P Ajit (MT21ACS427)

Christo Joseph (MT21ACS426)

Jishnu Chandran (MT21ACS419)

Rahul Nambiar (MT21ACS421)

INDEX

INTRODUCTION

The Indian Premier League (IPL) is a professional Twenty20 cricket league in India contested during March or April and May of every year by eight teams representing eight different cities in India.The league was founded by the Board of Control for Cricket in India (BCCI) in 2008. The IPL has an exclusive window in ICC Future Tours Programme. Currently it's the 14th season of IPL. Sports data analytics started way back in the 1960s and is literally considered to be a game changer. These days with the rise in dynamic real time analysis teams are able to deploy various strategies and increase their probability of winning. Moreover, this plays a huge role in the aftermath of a season to improve the performance of the team. With the success of IPL since its fruition, people tend to invest more in the sport and hence rely on data analytics during the bidding/transfer process.

This project has made use of Hadoop Ecosystem components like Flume , Hive & PIG for analyzing the below statistics .

1. Power Play Analysis –

    - Runs in Powerplay of Each Match
    - Powerplay Average Runs & Dismissals

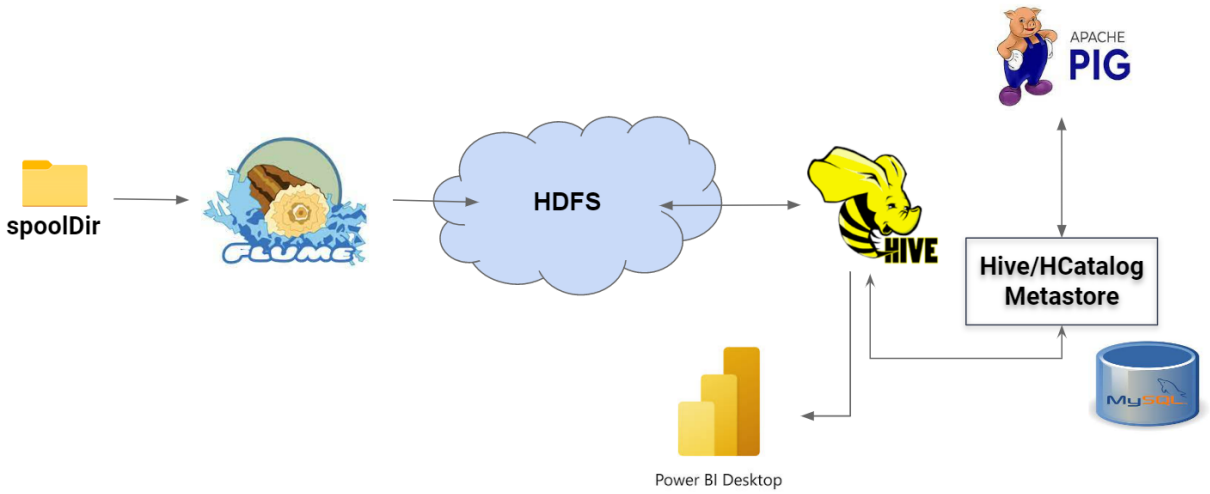
2. Target of 200 Runs or More -

    - How many times each Team scored > 200
    - Possibility to chase > 200 Target

3. Batsman Data –

    - Highest Average and Strike rate for >50 Matches
    - Top 10 Batsman in each run category

Data Pipeline



Data ingestion

Apache Flume is used to ingest data into the ecosystem. As input we have two files:

1. Match Details csv
2. Ball by ball Delivery Details csv

Hence two sources are configured as spool directory and respectieve channels sink the data to their respective destination folders in hdfs.

Additional configuration for controlling the channel flush are made appropriately like settings of rollSize, rollInterval, rollCount etc.

Data Preprocessing

Once the raw datas is available in hdfs, we utilize Apache Hive and Apache Pig to perform the preprocessing and data cleansing part. This involves two phases:

1. Hive Table creations - This sql script creates the necessary tables to store the result of cleansed data and also to externally manage the files ingested via flume in the previous step.
2. Pig Data cleansing part - This pig script will do the necessary cleansing and store the end result into hive tables.

As part of data cleansing the following are done:

- Team name : Certain IPL teams have rebranded during the course of the IPL seasons. So in order to update the same so as to help with the analysis, we update the old names with the new current names. This is done with the help of an excel sheet which holds the current names against an array of alternative names (old names)

- City Name: Certain rows contained ‘NA’, however the venue was available. So these were updated with the appropriate city with the help of an excel master data which contains the city and venue details.
- Venue Variations: Certain venue values were found to be variations of the same stadium. These were updated as a standard venue with a helper file. The same was used to rectify the city variations as well eg: (Bangalore < Bengaluru)
- Match Details: Few match records were having no outcome (suspect that the match was called off due to weather conditions etc.). These were excluded from the dataset from the both the Match details table as well as the Ball by ball delivery details tables.

Data Analysis

Analysis part is done with the help of PIG, which does all the processing. This is achieved via the -useHCatalog parameter, which enables us to refer to the schemas and data residing in HIVE.
Various analyses were performed (details mentioned later in this report) and the results were stored back into hive tables.
PIG was selected as the analysis tool mainly based on the performance analysis done for PIG vs HIVE which revealed that PIG provides an overall better performance than HIVE.
HIVE was selected to store the intermediary tables and the result tables which could then be leveraged by PowerBi tool for visualization.

Visualization

The PowerBi tool is used to visualize the results. The tool was linked to HIVE which was hosted by hiveserver2 with the help of Cloudera Driver. This allowed them to query directly into HIVE and use the tables as field models for visualization charts/graphs. As shown in the below figure PowerBi is linked directly to hive and with the help of thrift APIs pulls the data for visualization.

Hive Configuration

For the pipeline to work, we need to configure hive to a remote metastore architecture since other applications like pig, power bi are also using these tables for processing and visualization.

- We have used a MySQL database for the metastore, which is then configured with thrift MySQL requires the appropriate users to be added with the permissions/privileges.
- This enables the hive to act as a metastore as well as a hiversever. Pig was configured to use the HCatalog component via the thrift url setup. (-useHCatalog)
- Power bi was configured to Hive via the Cloudera ODBC hive connector driver to access the tables available in hive.

Shell Scripts

The pipeline was set up by writing shell scripts to execute the necessary hive and pig scripts for preprocessing and analysis. This had 2 phases with multiple scripts within:

1. ./preprocessing.sh: Which in turn calls the hive initial table creation scripts, and then invokes the pig script to perform the data cleansing.
   Appropriate logs are printed into the console and preprocess.log file.
2. ./analysis.sh: This is the main script which executes the various analysis processes.
   Appropriate logs are printed into the console and analysis.log file.

Steps to Execute

Below are the scripts/steps to follow to execute the pipeline.

1. Start HDFS, HIVE metastore (hive –service metastore), HIVE server2 (hive –service hiveserver2)

```
hive -service metastore
hive -service hiveserver2
```

2. Run flume agent with the below code:

```
flume-ng agent --conf $FLUME_HOME/conf   --conf-file lab.conf
--name a1 -Dflume.root.logger=INFO,console
```

3. Copy the match csv to ‘input/matches/’ and the ball to ball delivery to ‘input/deliveries/’
4. Once data is ingested, run the preprocessing script as follows

```
./preprocessing.sh
```

   To view logs use:

```
less preprocessing.log
```

5. Once the data ingestion is complete, run the analysis script.

```
./analysis.sh
```

   To view logs use:

```
 less analysis.log
```

6. Visualize the details from the respective tables in PowerBi by refreshing the data fields.

Note: Kindly make sure that src/source code is placed in /meda/sf_shareFolder/scripts.

MATCH DATA ANALYSIS

1. Powerplay Analysis.

First 6 overs in T20 international matches are limited for the bowling side. Fielders are not allowed to stand out of the 30-yard circle. Only two fielders can field in the boundary line. This fielding restriction in limited overs cricket is the powerplay. This period of 6 overs is mandatory.

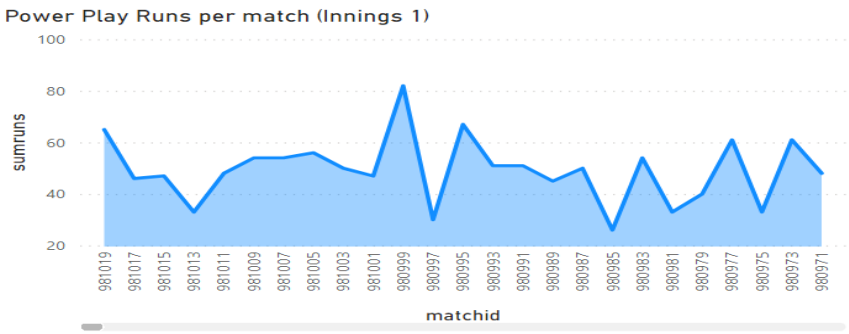
The analysis done here is to find runs scored in powerplay of each match and powerplay average Runs and Dismissals in IPL of different seasons.

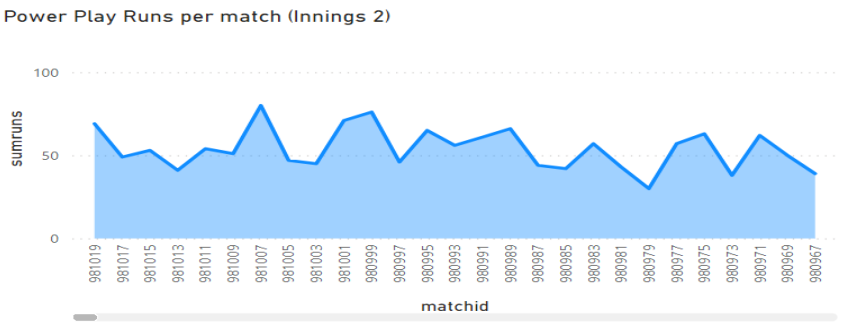
1.1 Runs in powerplay of each match

Total runs scored in each match is found for the first and the second innings of the match separately during the powerplay overs.

Steps taken:

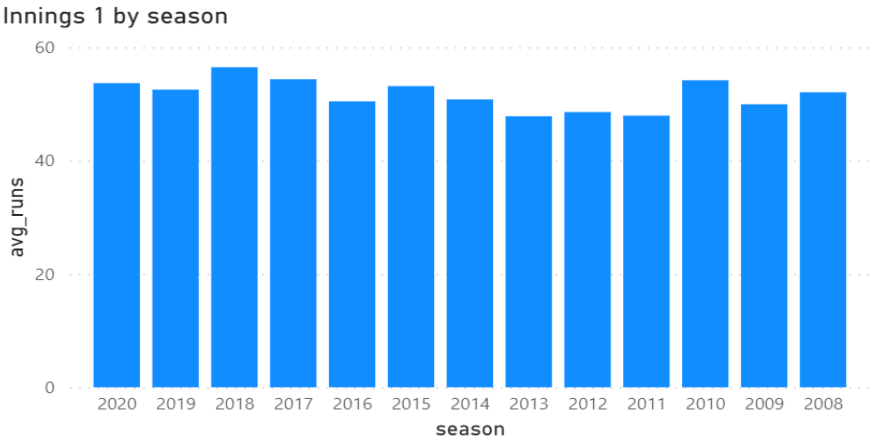
- Match and ball by ball data is loaded into two relations.
- They are joined on the basis of their id's.
- Filtered on the basis of "overs" less than 6 and whether it's the first or the second innings of the match.
- Then it's grouped on the basis of match id.
- SUM operation is performed to find the total runs scored in each match.
- Finally, the result is stored.

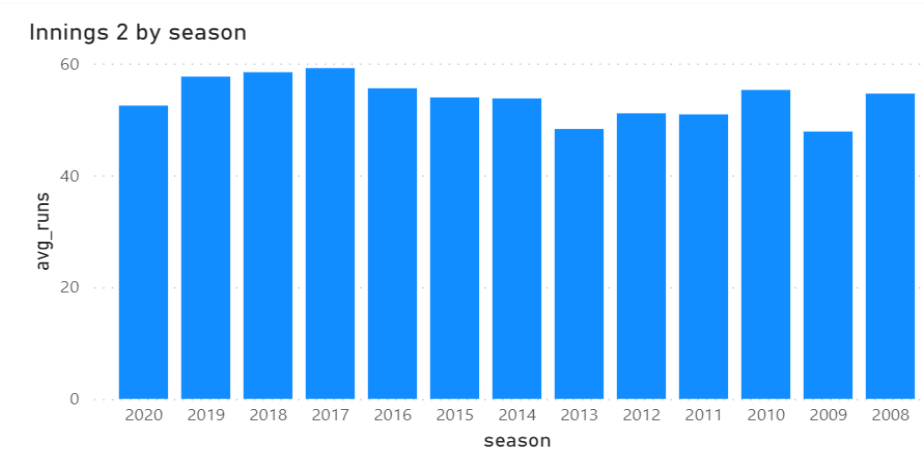




1.2 Powerplay Average Runs and Dismissals

- **Powerplay average runs :** Average runs scored in each season is found for the first and the second innings of the match separately during the powerplay overs.
  Steps taken:
  - Match and ball by ball data is loaded into two relations.
  - They are joined on the basis of their id's.
  - Filtered on the basis of "overs" less than 6 and whether it's the first or the second innings of the match.
  - Then its grouped on the basis of match id and season (year)
  - SUM operation is performed to find the total runs scored in each match in every season.
  - Then it's grouped on the basis of season (year).
- AVG operation is performed to find the average runs for each season.
- Finally, the result is stored.





- **Powerplay Average Dismissals**

  Maximum Dismissal – Max dismissals for each season during powerplay overs are found.

  Steps taken:

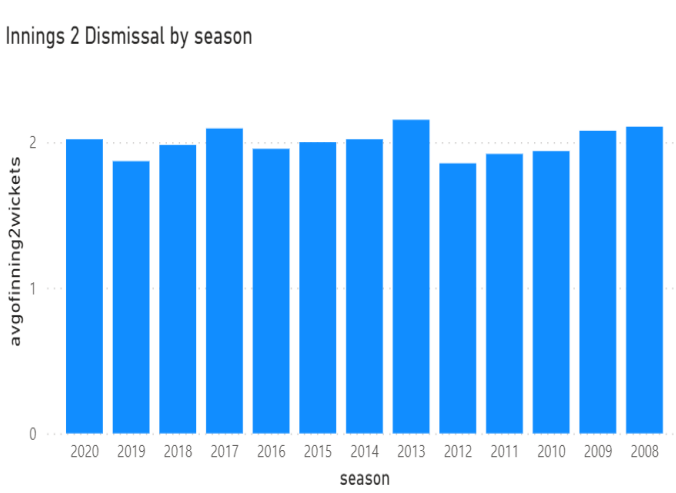
  - Match and ball by ball data is loaded into two relations.
  - They are joined on the basis of their id's.
  - Filtered on the basis of "overs" less than 6 and whether the wicket was taken.
  - Then it's grouped on the basis of match id, season (year) and inning.
  - COUNT operation is performed to find wickets taken for each match.
  - Then it's grouped on the basis of season (year).
  - MAX operation is performed to find the maximum wickets taken for each season.
  - Finally, the result is stored.

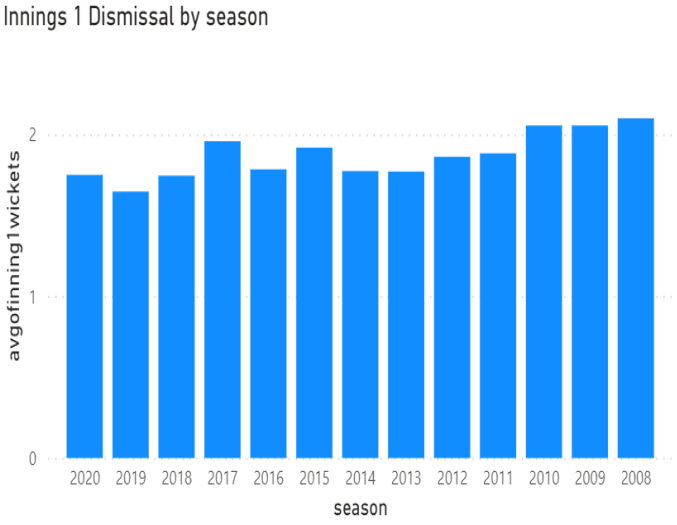
  Average Dismissal - Average dismissals in each season is found for the first and the second innings of the match separately during the powerplay overs.

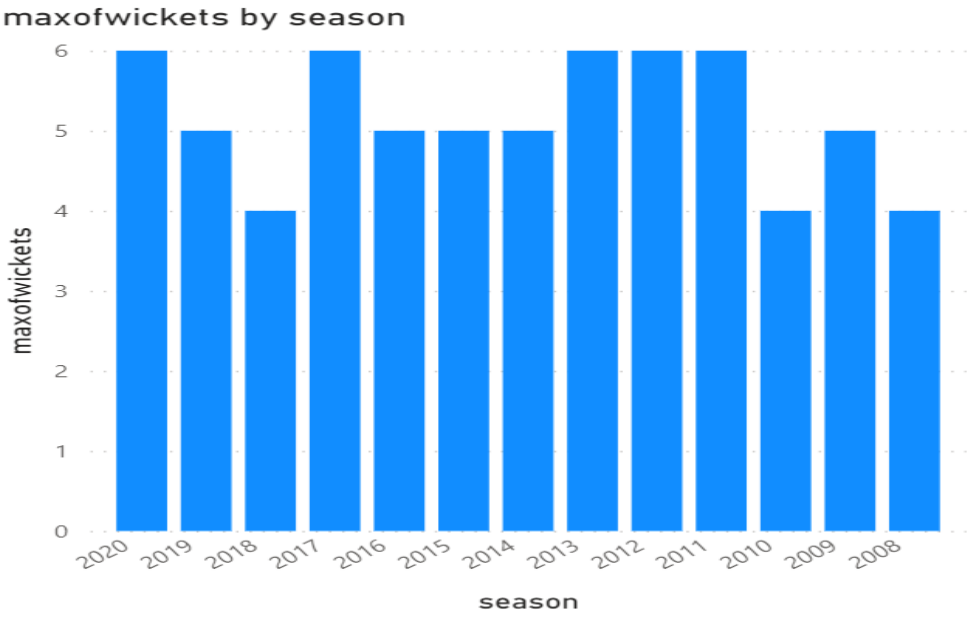
  Steps taken:

  - Match and ball by ball data is loaded into two relations.
  - They are joined on the basis of their id's.
  - Filtered on the basis of "overs" less than 6 and whether the wicket was taken.
  - Then it's grouped on the basis of match id, season (year) and inning.
  - COUNT operation is performed to find wickets taken for each match.
  - Then it's grouped on the basis of season (year).

- AVG operation is performed to find the average of wickets taken for each season.
- Finally, the result is stored.







2. Target of 200 Runs or More

Below Analysis is to find out the chance of victory for teams which are chasing a 200 Plus score. To analyze this we need to identify the teams which have scored runs more than 200 in each innings and then find out the team which is the winner.

2.1 How many times each Team scored > 200

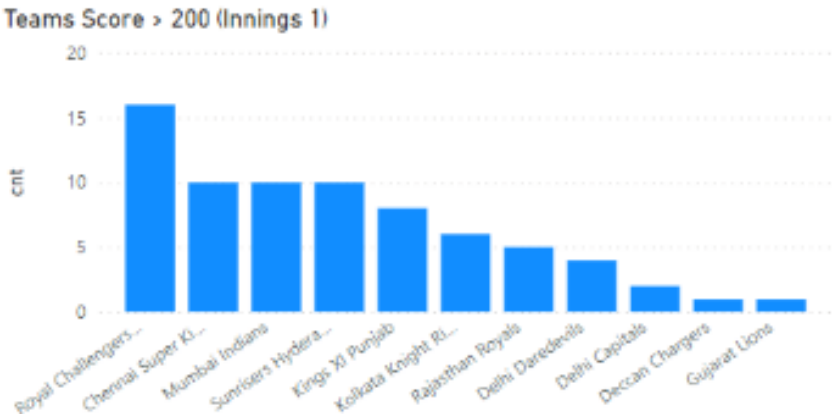
- Filter Innings wise match details
- Calculate total runs scored by Each team using the ball by ball data in each innings
- Filter out the details of batting teams and runs scored if match score >200
- Find the number of times each team scored greater than 200 score

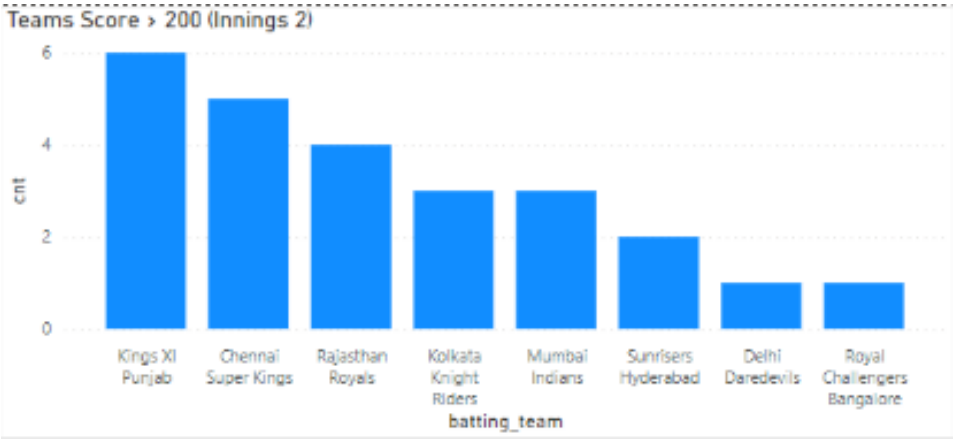
Innings 1 :

Team	Count
Gujarat Lions	1
Delhi Capitals	2
Mumbai Indians	10
Deccan Chargers	1
Kings XI Punjab	8
Delhi Daredevils	4
Rajasthan Royals	5
Chennai Super Kings	10
Sunrisers Hyderabad	10
Kolkata Knight Riders	6
Royal Challengers Bangalore	16

Innings 2:

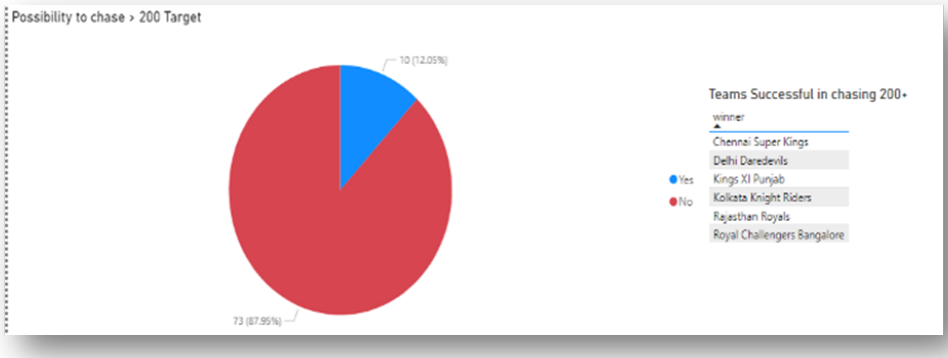
Team	Count
Kings XI Punjab	6
Delhi Daredevils	1
Rajasthan Royals	4
Chennai Super Kings	4
Sunrisers Hyderabad	2
Kolkata Knight Riders	3
Royal Challengers Bangalore	1
Mumbai Indians	3





2.2 Possibility to chase > 200 Target

- Find details of batting team won in 2nd innings
- Filter out details of teams which have won by chasing Score greater than 200.
- Plot the data for team wise score count & winning proportion of chasing team.



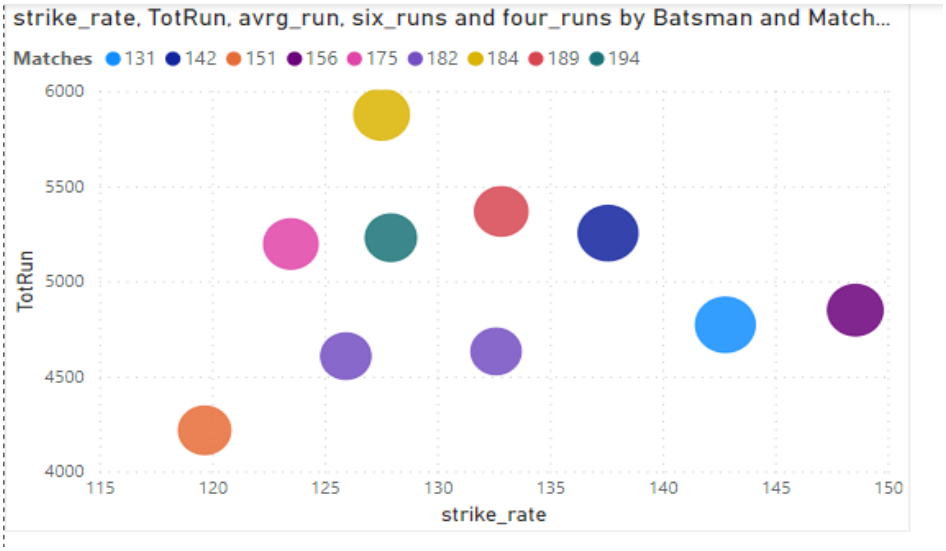
3. Batsman Data Analysis

We are finding best batsmen in different categories such as overall who is best, average run based analysis, top batsmen with strike rate, who score more runs from 6's and also who scores more from 4's.

3.1 Overall best top 10 batsman

We have compared different parameters such as strike rate, total run scored, average run score etc., to determine the overall top 10 batsmen.

Batsman	Matches	strike_rate	TotRun	avrg_run	six_runs	four_runs
V Kohli	184	127.53	5878	31.95	202	504
SK Raina	189	132.84	5368	28.40	194	493
DA Warner	142	137.58	5254	37.00	195	510
RG Sharma	194	127.94	5230	26.96	214	458
S Dhawan	175	123.50	5197	29.70	109	591
AB de Villiers	156	148.56	4849	31.08	235	390
CH Gayle	131	142.79	4772	36.43	349	384
MS Dhoni	182	132.61	4632	25.45	216	313
RV Uthappa	182	125.94	4607	25.31	163	454
G Gambhir	151	119.67	4217	27.93	59	492
Total		**1,318.96**	**50004**	**300.21**	**1936**	**4589**



3.2 Highest Average and Strike rate for >50 Matches.

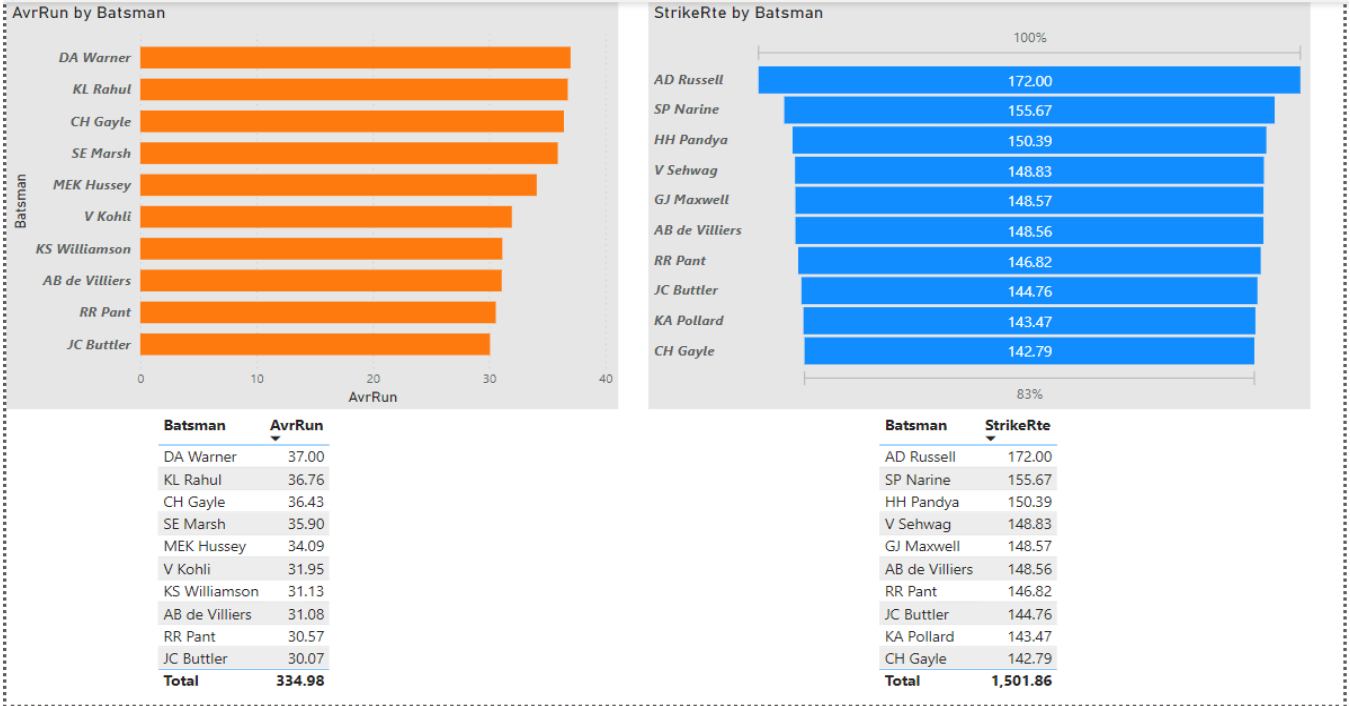
Added the total matches played by each batsman and filtered only those who have played more than 50 matches.

- **Average score** of a Batsman is calculated using Total runs scored by the batsman and the number of matches played. We took the top 10 batsmen.

Average score = Total runs scored / total number of unique matches he played

- **Batting strike rate** is a measure of how quickly a batsman gains runs. Batting strike rate (s/r) is defined for a batter as the average number of runs scored per 100 balls faced. The higher the strike rate, the more effective a batter is at scoring quickly.This was calculated from Total runs scored by the batsman and the total number of balls he faced. We took the top 10 batsmen.

Strike rate = (Total runs scored / total number of balls he faced) * 100



Batsman	AvrRun
DA Warner	37.00
KL Rahul	36.76
CH Gayle	36.43
SE Marsh	35.90
MEK Hussey	34.09
V Kohli	31.95
KS Williamson	31.13
AB de Villiers	31.08
RR Pant	30.57
JC Buttler	30.07
Total	**334.98**

Batsman	StrikeRte
AD Russell	172.00
SP Narine	155.67
HH Pandya	150.39
V Sehwag	148.83
GJ Maxwell	148.57
AB de Villiers	148.56
RR Pant	146.82
JC Buttler	144.76
KA Pollard	143.47
CH Gayle	142.79
Total	**1,501.86**

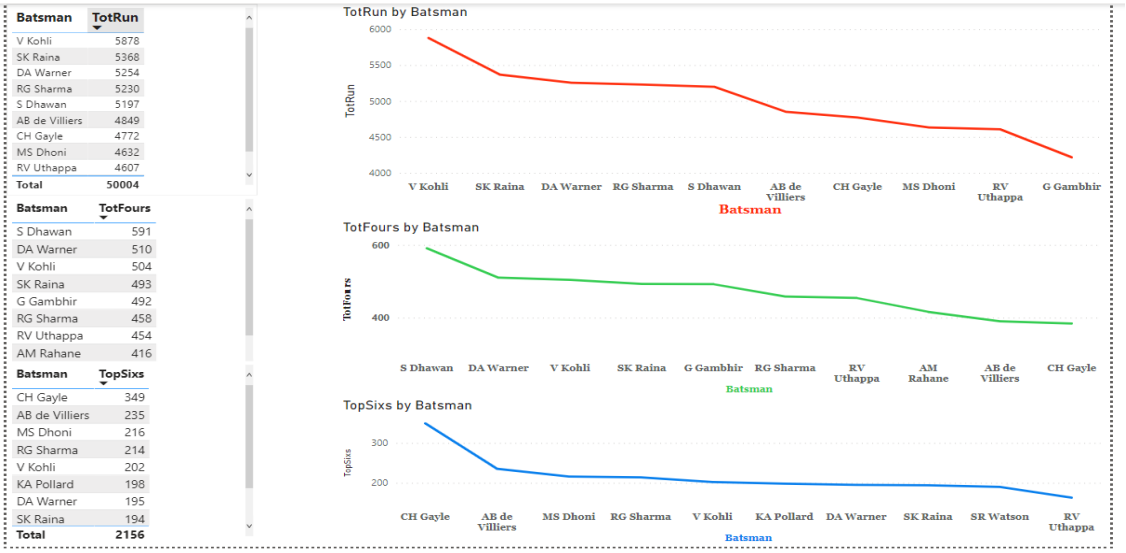
3.3 Top 10 Batsman in each run category

- We have taken top 10 batsman who took more run
  Added batsman_runs for each batsman and ordered based on this value in descending order.
- We have taken top 10 batsman who scored more from 6's
  Added 6's scored by each batsman and ordered based on this value in descending

order.

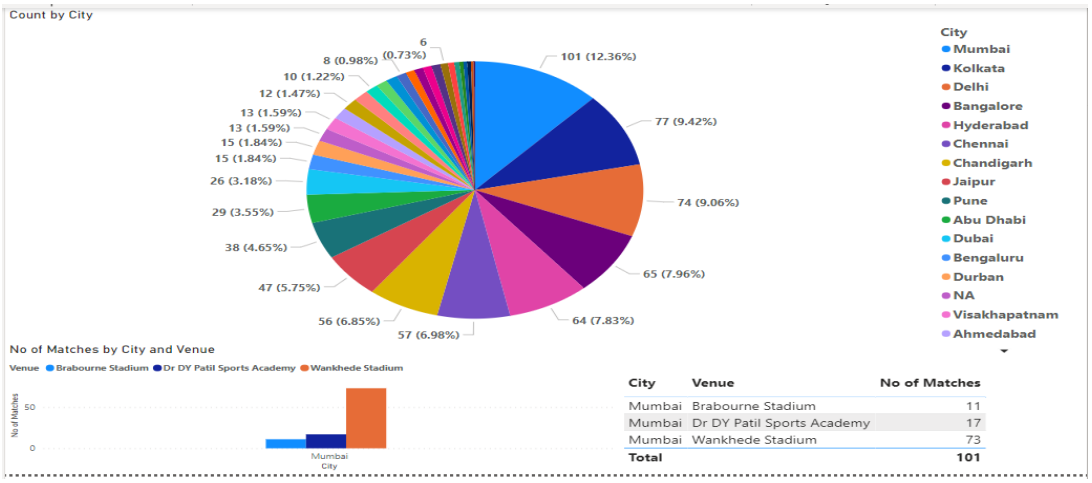
- We have taken top 10 batsman who scored more from 4's

  Added 4's scored by each batsman and ordered based on this value in descending order.



4. City Analysis

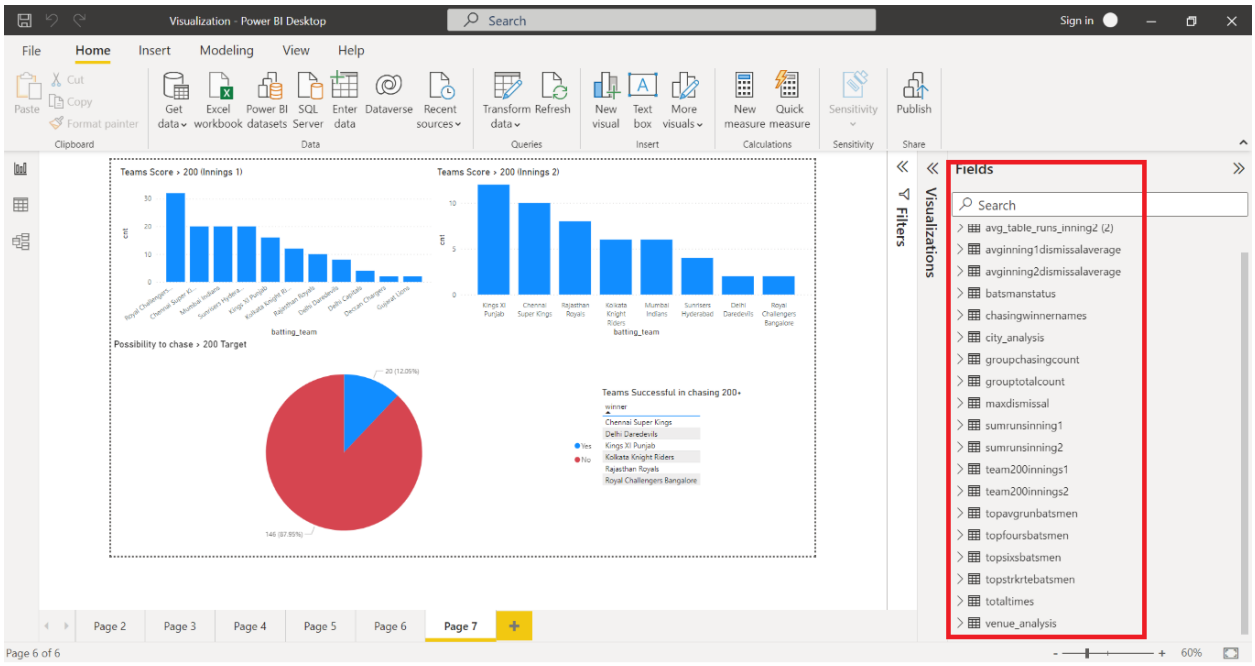
We are trying to see how many matches are held in each city here. We group the table based on the city and then count each game held in the city, giving the total number of matches held in that city. We are also grouping venues with cities, so that we will be getting a clear idea such as how many grounds are used in each city for the matches and the number of matches held under each ground.



Data Visualization using Power BI

Power BI is a collection of software services, apps, and connectors that work together **to turn your unrelated sources of data into coherent, visually immersive, and interactive insights**. Your data may be an Excel spreadsheet, or a collection of cloud-based and on-premises hybrid data warehouses.

Power BI is used as a Data Visualization tool in this project. Analysis Results stored in Hive tables are imported to Power BI to plot the Analysis. Choose Cloudera Hive DSN and The tables from your Hive Database will be visible in your Navigator column of Power BI. We can select the required columns projected for plotting the analysis and select the type of graphical representation.



Also we can directly fetch data from HDFS and plot it. We tried both methods. We stored the data from PIG to HDFS and then tried this option. We need to specify the HDFS server name in the below format in that case.

Eg: http://192.168.53.76:9870/webhdfs/v1/IPL_venue/part-r-00000

Additional Deliverable Achieved

- We have added City analysis additionally in the report to show match details in each city. We can see the number of matches in each ground(venue) under each city.
- Configured Hive remote metastore with mySQL to intercommunicate between the applications Pig and PowerBi with the help of ODBC driver. The data from the pig stored in hive tables and then fetched this table data from powerBI using the ODBC driver.

Major Challenges

1. While trying to setup Oozie for scheduling , distro creation failed

   Logs :

   [ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:2.22.2:test (default-test) on project oozie-core: There are test failures.

   [ERROR] Please refer to /home/hdoop/oozie/oozie-5.2.0/core/target/surefire-reports for the individual test results.
   [ERROR] Please refer to dump files (if any exist) [date].dump, [date]-jvmRun[N].dump and [date].dumpstream.
   [ERROR] The forked VM terminated without properly saying goodbye. VM crash or System.exit called?
   [ERROR] Command was /bin/sh -c cd /home/hdoop/oozie/oozie-5.2.0/core && /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java -Xmx2048m -da -XX:MetaspaceSize=512M -XX:MaxMetaspaceSize=1024M -XX:+CMSClassUnloadingEnabled -XX:+UseConcMarkSweepGC -jar

   Solution/Workaround : Using Shell Script for Scheduling the Pipeline.
2. Configuration issue while setting up Hue.
3. Initially started with fetching data from Twitter. But when we ingested the data using flume, the data format was Avro . It was time consuming to convert Avro data to proper data format and so we have used datasets from Kaggle instead.
4. CSVimport contains common input values containing ‘comma’ which was affecting the Column split based on Delimiter.
   - **Solution:**SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
5. While connecting Pig to Hive faced challenges in using HCatalog

- Setting up environment variables
- Moving from embedded metastore to Remote Metastore
- Connecting MySQL

Solution :

- http://www.thecloudavenue.com/2013/11/differentWaysOfConfiguringHiveMetastore.html
- https://docs.datafabric.hpe.com/70/Hive/Config-MySQLForHiveMetastore.html
- install mysql and create new user and provide grants
- mysql driver download and place in hive.

6. Repetitive INFO message alert in PIG during execution regarding the jobhistory node.
   Solution: Run the given command "mr-jobhistory-daemon.sh start historyserver"
7. Power BI data was coming as binary initially while fetching data using HDFS and we struggled to find how to convert it into normal format.
   Solution: was to use the edit query option within the powerBI.
   Later we decided to fetch the tables from HIVE instead of setting up the HDFS file urls.
8. Setting up hiveserver 2 for powerbi
   Reference:https://towardsdatascience.com/connecting-apache-hive-to-microsoft-power-bi-d460e2278720
   - Error starting HiveServer2 on attempt 1, will retry in 60000ms
   - java.lang.NoClassDefFoundError: org/apache/tez/dag/api/TezConfiguration

   Solution: update below property in hive-site.xml

```
<property>
        <name>hive.server2.active.passive.ha.enable</name>
        <value>false</value> # change false to true
</property>
```

9. Power BI ODBC access permission:
   Reference:https://stackoverflow.com/questions/60051970/org-apache-hadoop-security-accesscontrolexception-permission-denied
   Solution : add below property in hdfs-site.xml

```
<property>
        <name>dfs.permissions</name>
        <value>false</value>
</property>
```

Contributions

- Archana P Ajith strived to install and configure Oozie & Hue which couldn't be materialized for reasons stated above along with the implementation for the 200+ runs analysis sections and the sections of strike rate calculation.
- Christo Joseph contributed with the remote metastore configuration followed by data pipeline setup and preprocessing.
- Jishnu Chandran focused on the batsman analysis and city analysis in addition to the configuration and visualization of all the results in PowerBI via hdfs url and HIVE server.
- Rahul Nambiar implemented the power play analysis section and also contributed to the batmans analysis using PIG.

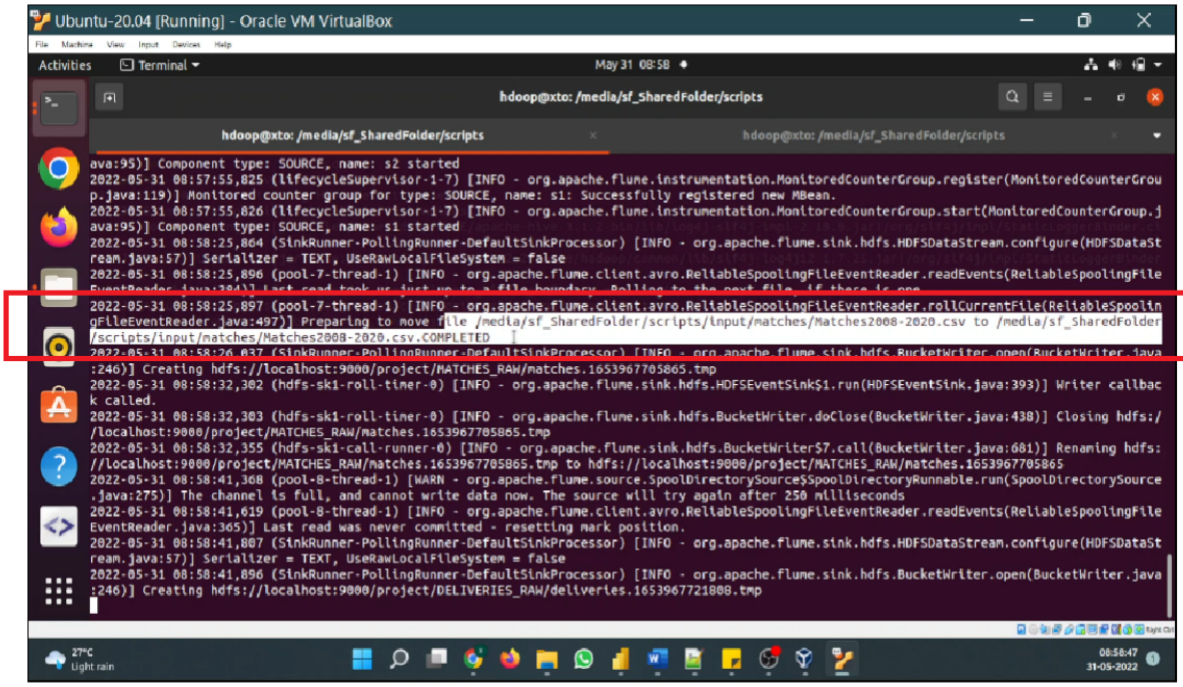
Result & Conclusion

Successfully implemented a pipeline from scratch starting with data ingestion all the way till the visualization of the analysis results. As part of this project the team was able to implement various technologies introduced during the course and get a better understanding of the Hadoop ecosystem and witness how various components fit into places.

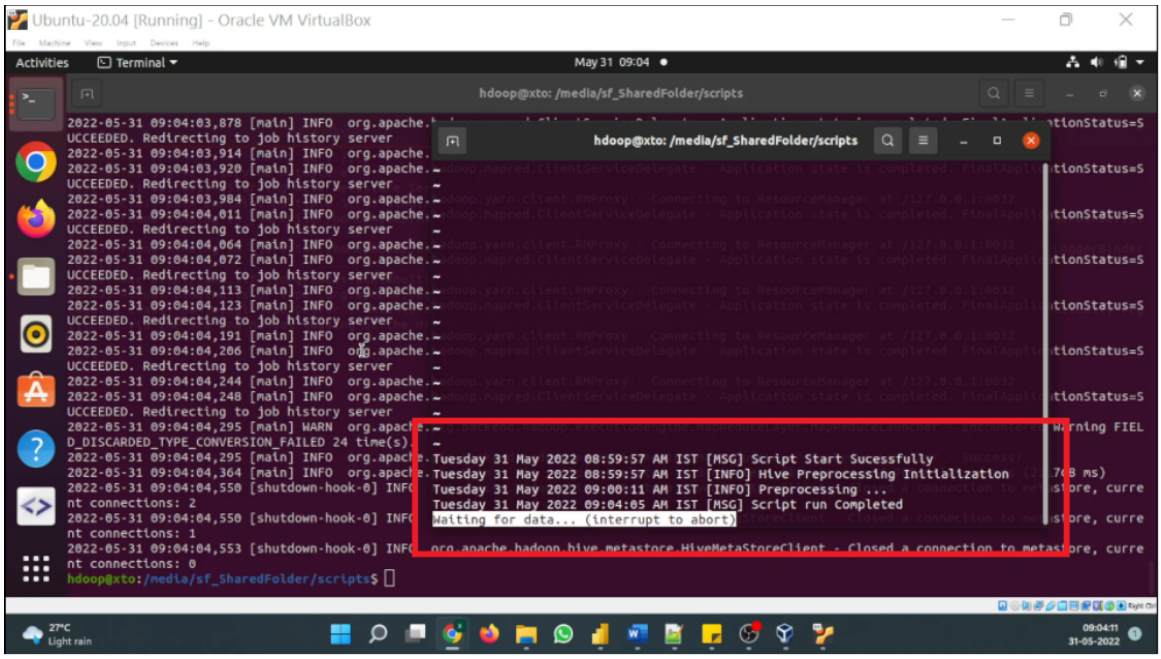
Below are the conclusions derived from the match data analysis:

- **Warner, Gayle and Pant** are best in terms of both strike rate and average run rate.
- Virat Kohli has the highest total runs across all seasons. Raina is just a few runs behind with the second spot.
- Average Score and 6"s by G Gambhir is lowest among top 10 batsmen.
- Dismissal Average for inning 1 is highest in 2008 IPL season.
- Dismissal Average for inning 2 is highest in 2013 IPL season.
- Highest average run score in inning 1 is in 2018.
- Highest average run score in inning 2 is in 2017.
- Target 200 Plus analysis results in a conclusion that 10 out of 83 matches, chasing team won after raising a 200 plus target score.

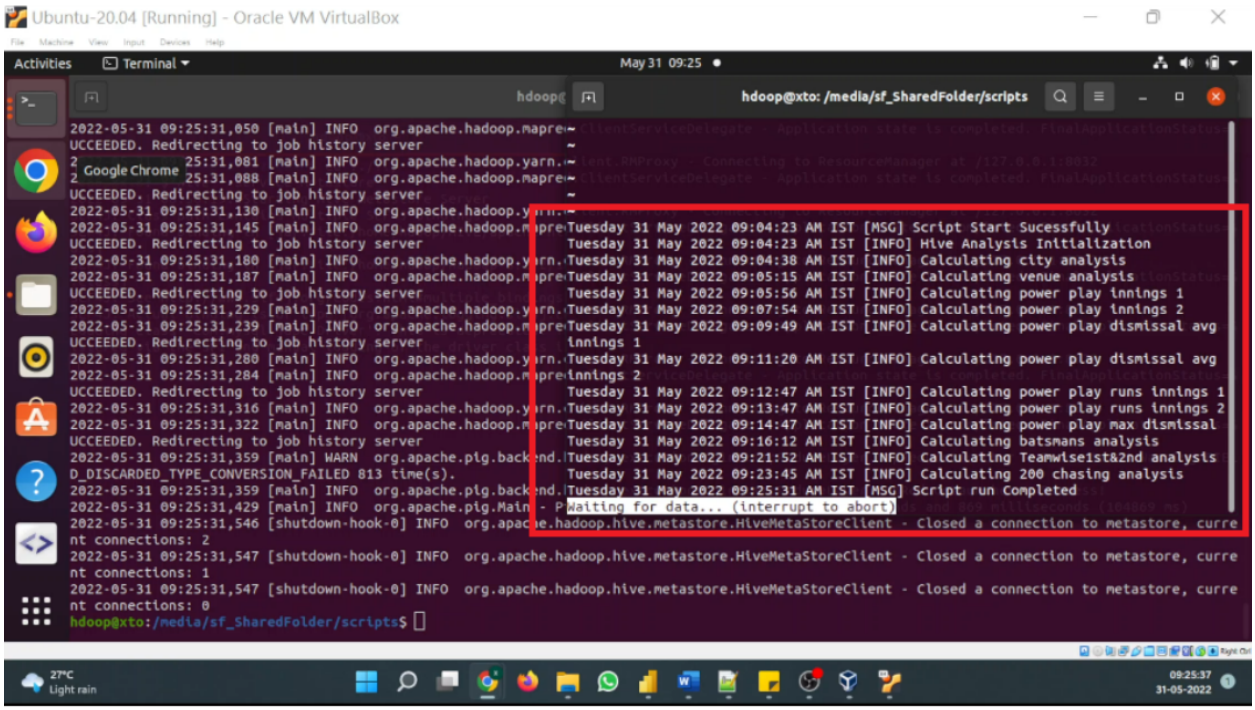
Screenshots of the project:



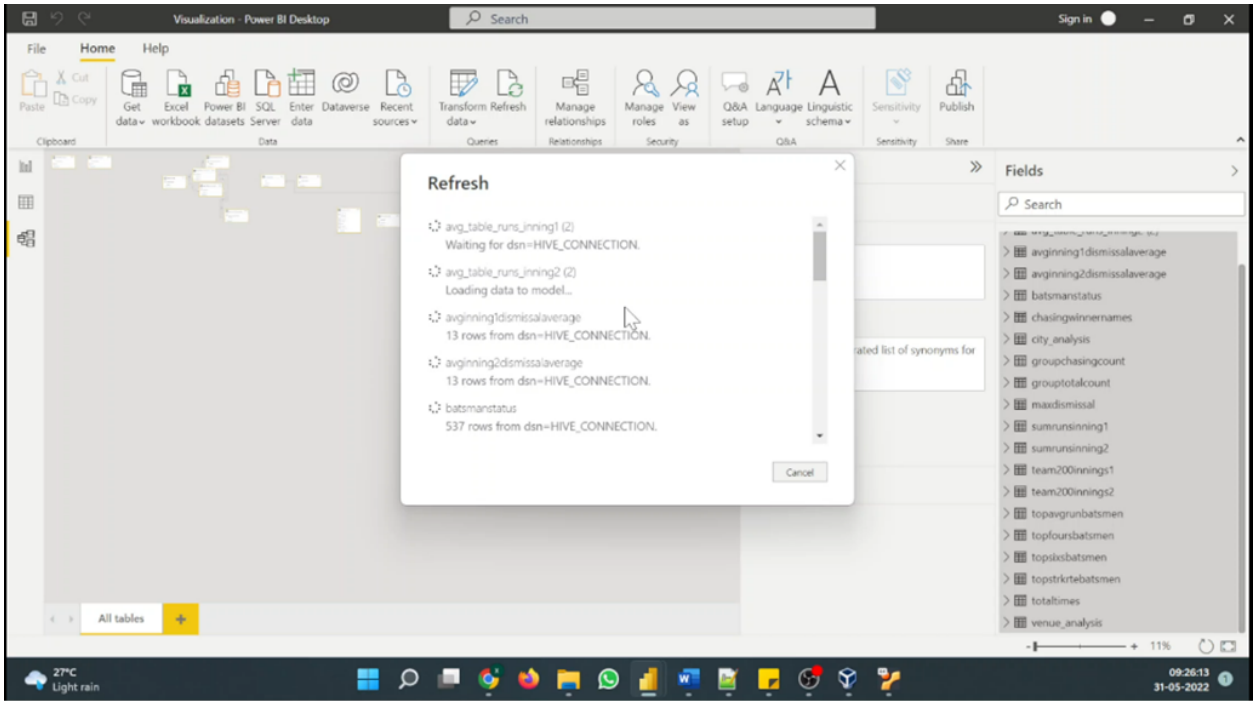
Implementation of flume



Data Cleansing run by preprocessing.sh
(preprocess.log output)



Analysis run by analysis.sh
(analysis.log output)



Refreshing all Fields in PowerBI after pipeline