**A picture containing logo

Description automatically generated**

**CA1**

Data Storage Solutions for Data Analytics

*2023*

**Submitted by**:

Student Name: Vaibhav Tiwari

Student ID: 10621051

Lecturer Name: Dr. Shazia A Afzal

**Group Members:**

Prathamesh Jadhav (10621081)

Aniket Bachewar (10621078)

Vaibhav Tiwari (10621051)

**Introduction:**

IPL is one of the most prominent cricket leagues with a huge fan base coming from all over the world. Originally formed by Board of Control of Cricket in 2007, it is held annually and had 14 different teams over the years. Also, the tournament has been organized in multiple countries like India, UAE, South Africa and multiple states across these countries.

As part of this project assignment, we will be using IPL dataset to obtain business intelligence from the data and develop a proof-of-concept data mart to benefit our key stakeholders. We will also be visualizing key findings using Tableau and SSRS reporting tools which can help in analyzing and exploring our data for making crucial business choices.

**Dataset:**

We have chosen an IPL dataset [Kaggle] which consists of all the details of the matches played from 2008 – 2019.

The dataset consists of match details for each season and shows important match information for each team, detailing key features such as, match winner, player of the match, toss decision, win by runs and much more.

**Reason for selecting subject area & benefit to businesses:**

With the help of this dataset, we can gain meaningful player and team insights which can help our key stakeholder with useful match facts and determining overall team performance. Fantasy cricket platforms can use this data to estimate each team capabilities by identifying statistical performance in actual game.

Popular teams and players are often compensated generously with a high pay, and it sometimes makes it tough for clubs, investors, and sponsors to determine the accurate value of a team or a player based on one’s potential. With important parameters from the dataset such as player of the match and total number of team wins, we can provide visual insights of overall team performance within that particular season. This will help sponsors and clubs gain a clear picture of one potential and worth for the upcoming seasons.

**Key stakeholders:**

* Fantasy cricket platforms
  + Fantasy cricket platform can utilize the data insights to display relevant information to their customers.
* Clubs, sponsors, and investors
  + Agencies and companies need large amount of information to decide which teams and players are the best fit for their business growth.

**Below steps were carried out for creating a data warehouse model for IPL:**

* **Data Modeling / ERD Diagram**  
  Data modelling is used to describe the relationships between the entities (also known as objects or tables in database).  
    
  **Schema:**  
  In data modeling there are 2 widely used schemas:
  + Star schema
  + Snowflake schema

Using MS SQL Server Management Studio [Moodle], we designed an ERD diagram which shows the column names, Primary Keys, Foreign Keys along with their entity relationships.

**ERD diagram demonstrated a star schema as seen below:**

Diagram

Description automatically generated

Relation was determined among entities by identifying dimension and fact tables and an ERD diagram was created. The design was created based on the category of records and entities.

* **SSIS: Implementing table and extracting, transforming, and loading data**

DDL SQL queries to define and create tables were executed and tables were generated to further load the data using SSIS.

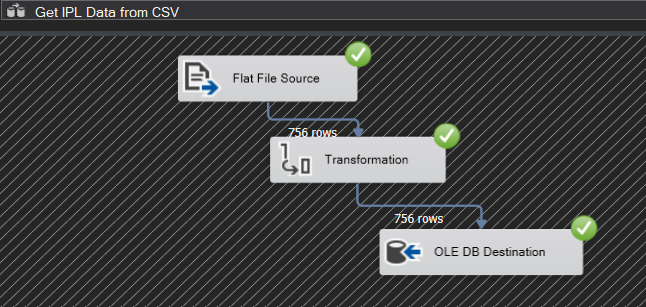
SSIS [Moodle] helps in importing and exporting bulk data to our entities using the Data Flow Task.

We have used staging table which stores the entire dataset into one single table and then the data is pushed to respective dimension and fact tables. This helps us in reducing the complexity for data transformation and loading data into respective tables as we have only OLE DB as the source for the entire lifecycle.

Below are the detailed steps of ETL for IPL dataset:

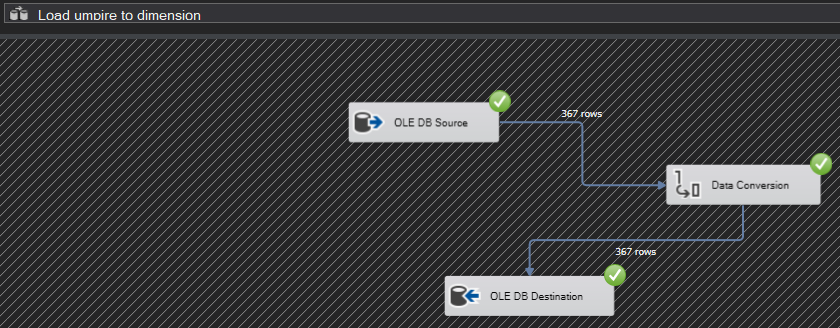
1. **Load IPL data into staging table**

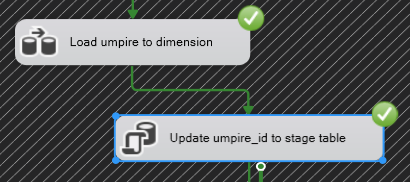
Flat file > Transformation > to OLE DB Destination

****

1. **Load umpire columns from staging table to umpire dimension table and update Primary Key of umpire table in staging table.**

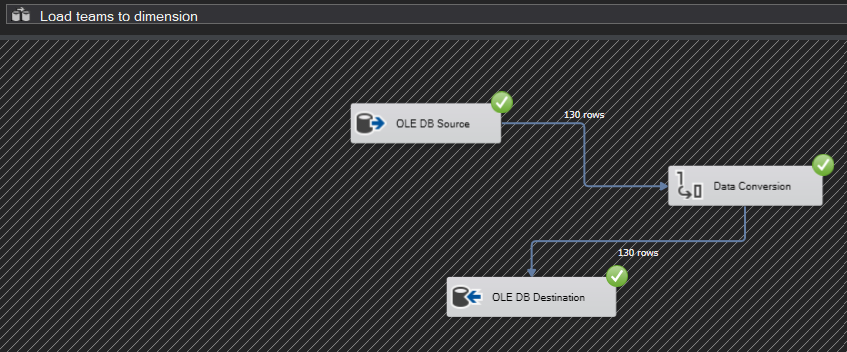
OLE DB source > Data Conversion > OLEB DB Destination

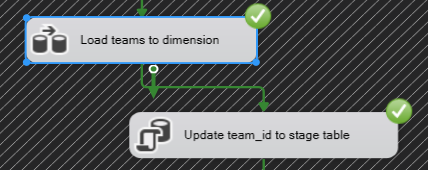
****



1. **Load team columns from staging table to team dimension table and update Primary Key of team table in staging table.**

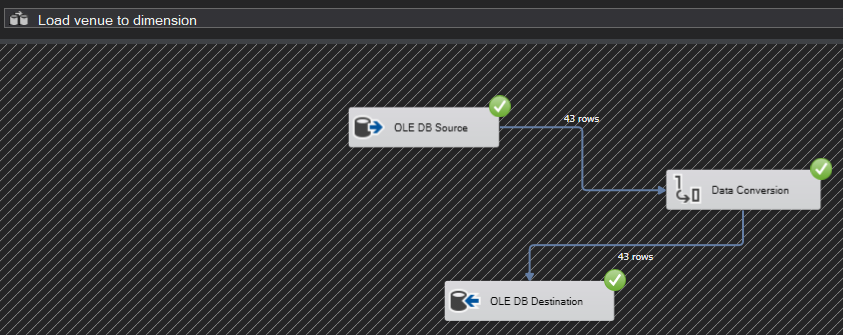
OLE DB source > Data Conversion > OLEB DB Destination

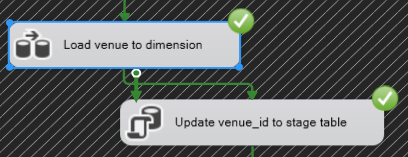




1. **Load venue columns from staging table to venue dimension table and update Primary Key of venue table in staging table.**

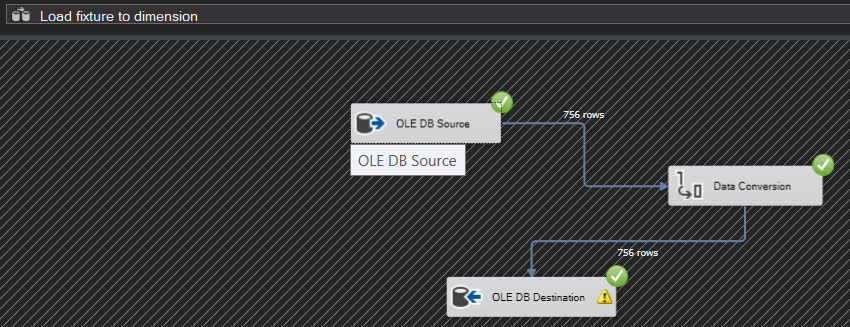
OLE DB source > Data Conversion > OLEB DB Destination

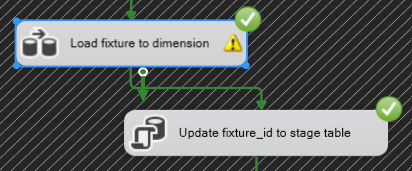




1. **Load match fixture columns from staging table to match fixture dimension table and update Primary Key of match fixture table in staging table.**

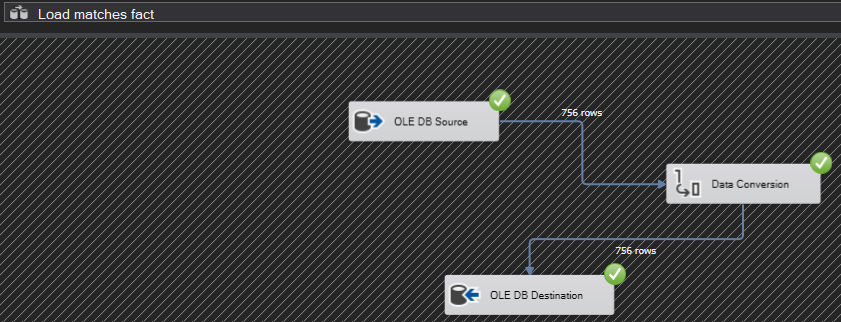
OLE DB source > Data Conversion > OLEB DB Destination





1. **Load data from staging table into matches fact table.**

OLE DB source > Data Conversion > OLEB DB Destination



1. **Overall ETL Process**

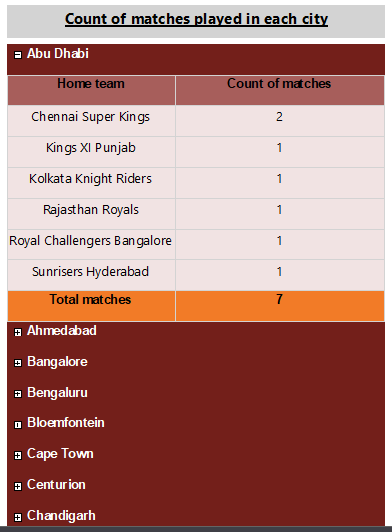


* **SSRS Reports and Tableau Visualization**

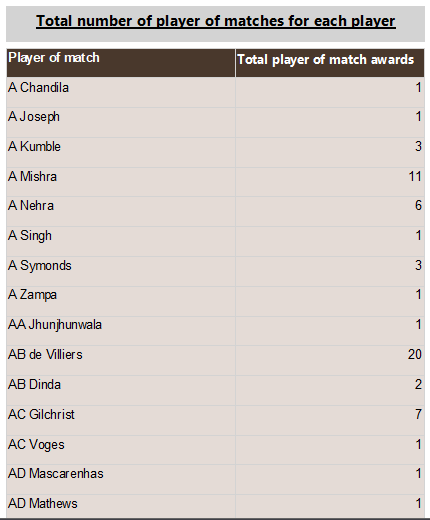
SSRS Reports [Moodle] are created using Microsoft Visual Studio with the help of SQL server as source and Tableau visualization uses tableau desktop application to generate data visualization with the help of input source.

Below are some reports and visualizations from IPL dataset generated using SSRS and Tableau.

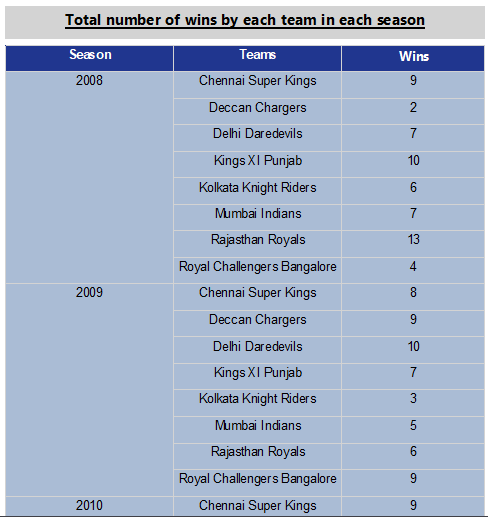
1. **Count of matches played in each city.**



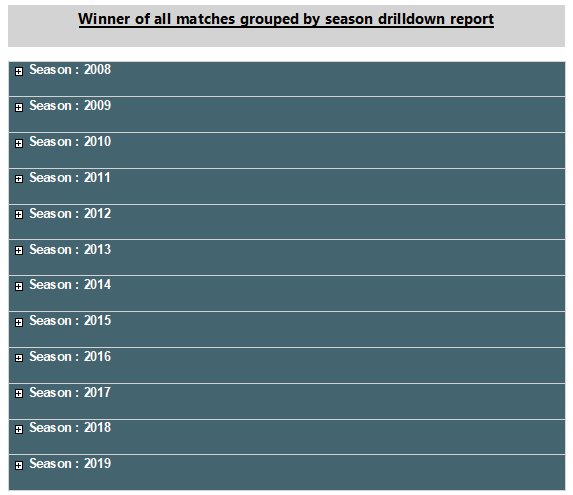
1. **Total number of player of matches for each player.**

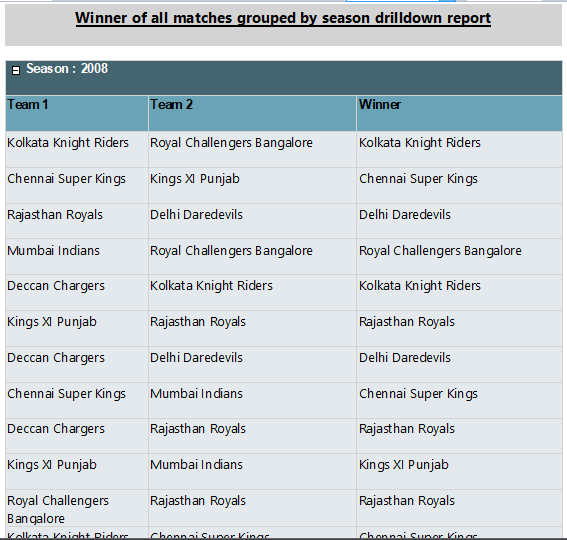


1. **Total number of wins by each team in each season.**

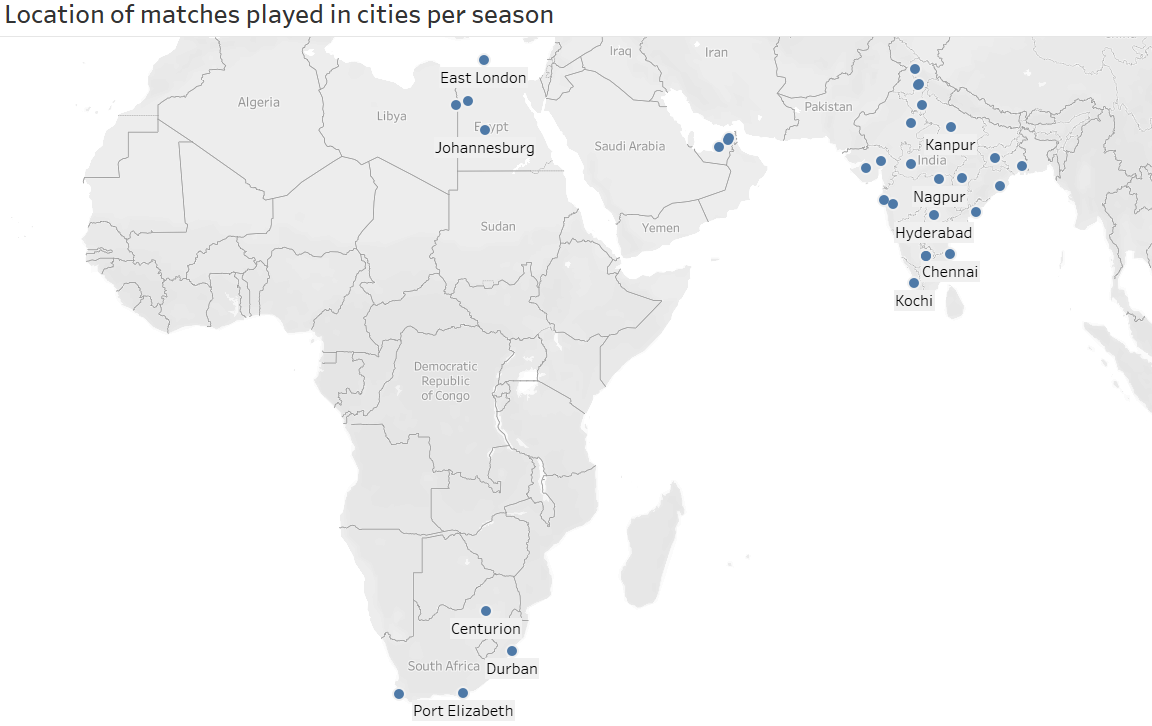


1. **Winner of all matches grouped by season drilldown report.**

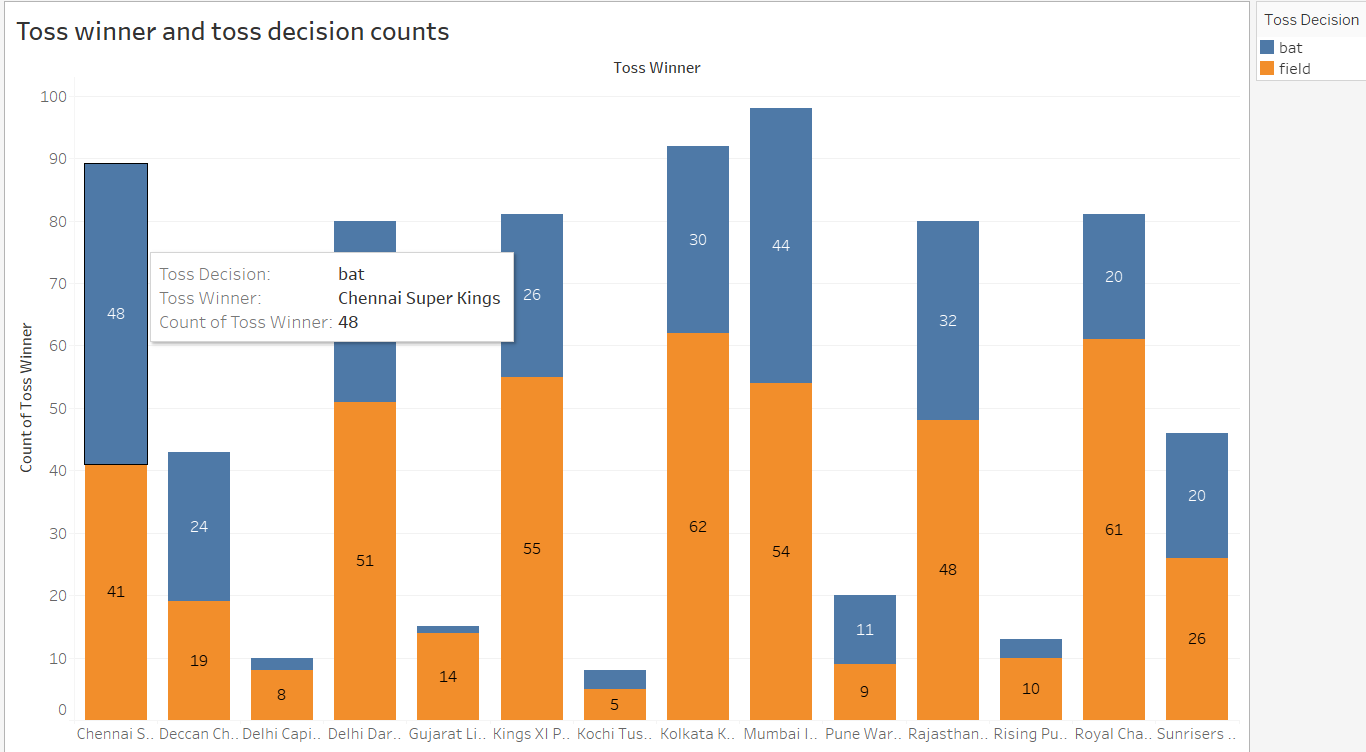




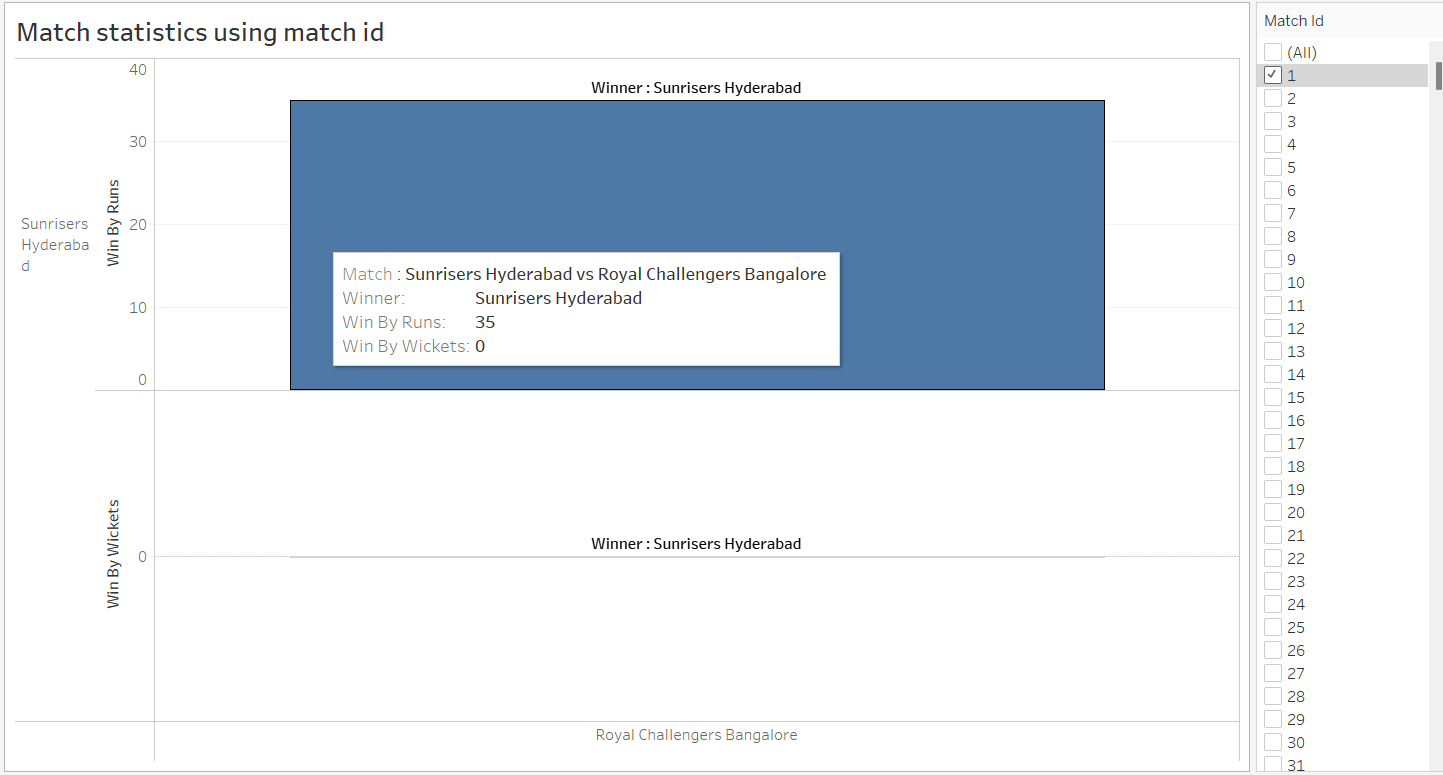
1. **Location of matches played in cities per season.**



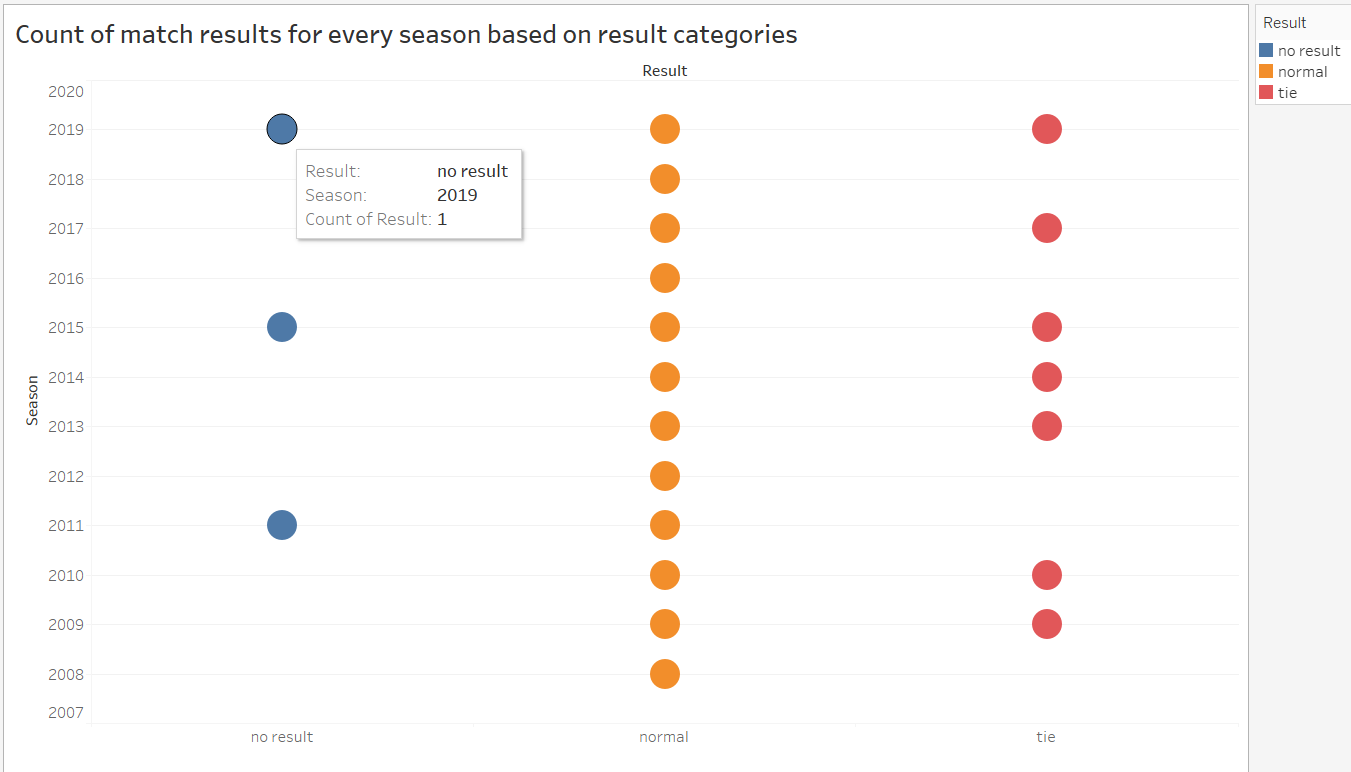
1. **Toss winner and toss decision counts.**



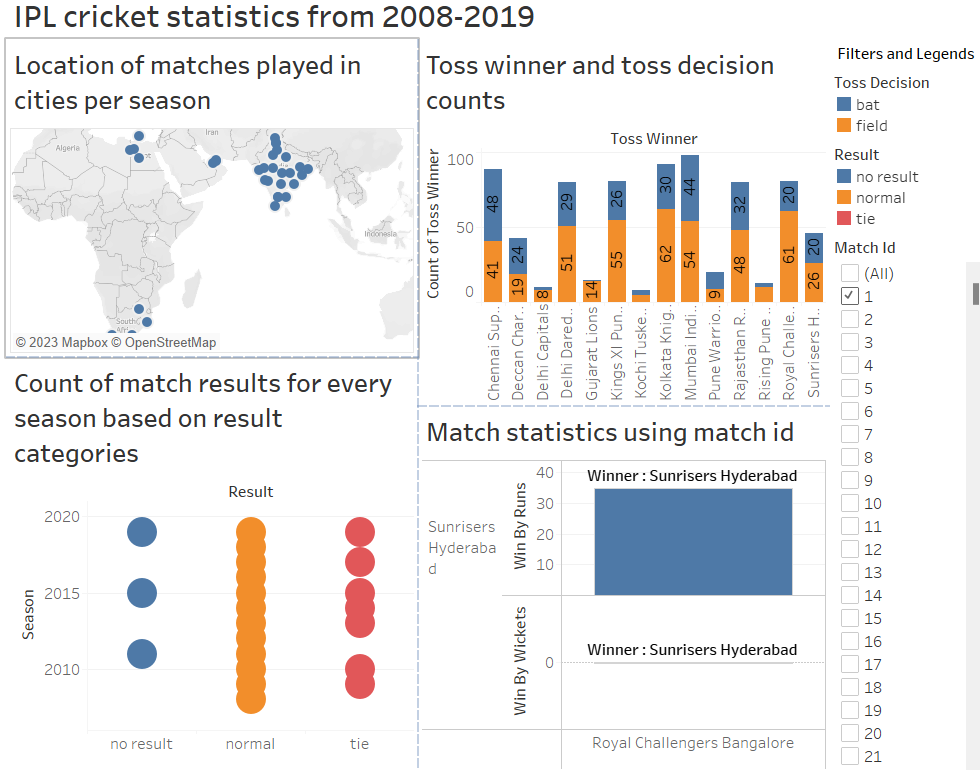
1. **Match statistics using match id.**



1. **Count of match results for every season based on result categories.**



1. **Dashboard of all tableau visualizations.**



**Neo4J**

Neo4j is graph database. The graph database is a database that represents and stores data using graph structures for semantic searches, such as nodes, edges, and properties.

**Comparing Graph Database and Relational Database**

Relational databases store data in tables with a fixed schema, and relationships between tables are defined through foreign keys. Whereas in graph databases, they store data as nodes, with the relationships between data unlike relational database which stores relation between data columns.

Each database has its own advantages and disadvantages based on the scenarios.

Initially all the data was loaded to Neo4J database using the LOAD CSV query.

The relational database, MS SQL already contained data inserted via SSIS process.

**Below show node creation and data insertion snippets from Neo4J Browser:**

Text

Description automatically generated

**CSV data insertion to matches\_fact node**

Text

Description automatically generated

**CSV data insertion to venue\_dim node**

Text

Description automatically generated

**CSV data insertion to umpire\_dim node**

Text

Description automatically generated

**CSV data insertion to teams\_dim node**

Text

Description automatically generated

**CSV data insertion to matches\_fixtures\_dim node**

**Create IDEXES for each node using queries below:**

* CREATE INDEX FOR (mf:matches\_fact) ON (mf.match\_id)
* CREATE INDEX FOR (vd:venue\_dim) ON (vd.venue\_id)
* CREATE INDEX FOR (um:umpire\_dim) ON (um.umpire\_id)
* CREATE INDEX FOR (td:teams\_dim) ON (td.team\_id)
* CREATE INDEX FOR (mfd:matches\_fixtures\_dim) ON (mfd.fixture\_id)

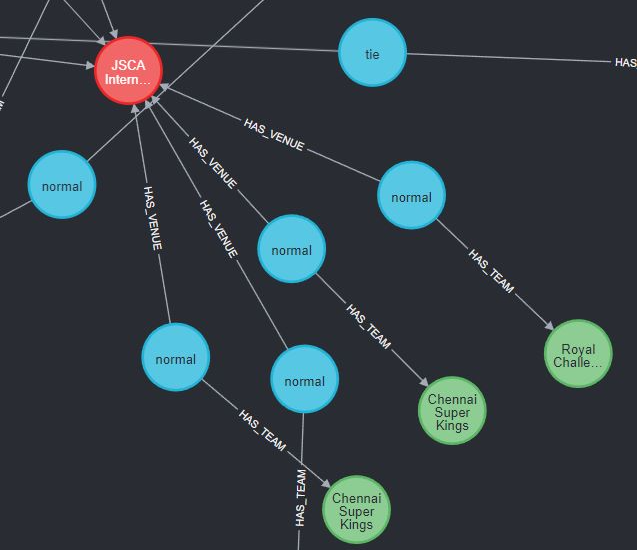
**Creating relationship between nodes:**

* MATCH (mf:matches\_fact),(td:teams\_dim) WHERE mf.team\_id=td.team\_id CREATE (mf) - [r:**HAS\_TEAM**] -> (td) RETURN mf,td
* MATCH (mf:matches\_fact),(vd:venue\_dim) WHERE mf.venue\_id=td.venue\_id CREATE (mf) - [r:**HAS\_VENUE**] -> (vd) RETURN mf,vd
* MATCH (mf:matches\_fact),(ud:umpire\_dim) WHERE mf.umpire\_id=ud.umpire\_id CREATE (mf) - [r:**HAS\_UMPIRE]** -> (ud) RETURN mf,ud

Graphical user interface

Description automatically generated

**Overview: Relationship between matches and team nodes**



**Relationship between matches with teams/matches node**

Below are 7 CQL and SQL queries which uses our IPL dataset to demonstrate differences between Graph and relational database along with providing useful stats using their respective queries. (All the CQL and SQL files for the CA assignment are provided)

Output count returned by both the CQL, and SQL exactly match and showcase a valid equivalent output for both graph and relational databases.

1. **SUM Aggregate function:** Retrieve total sum of win\_by\_runs by Mumbai Indians

**CQL query:**

Graphical user interface, text, application

Description automatically generated

**SQL query:**

Graphical user interface

Description automatically generated with medium confidence

The CQL query uses the MATCH operation to match match\_fact nodes with teams\_dim nodes, based on their relationships in the graph. The CQL query do not support subquery and instead rely on node relationship for faster retrieval of data. The SQL query uses a where clause with a join to match team\_id columns to retrieve the team1 for the Mumbai Indian teams. The sum aggregate function works identical for the scenarios and return total of win\_by\_runs column.

1. **Filtering data:** Retrieve list of all umpire2 who had done umpiring in Mumbai city.

**CQL query:**

MATCH ((mf:matches\_fact)-[:HAS\_UMPIRE]->(ud:umpire\_dim)), ((mf:matches\_fact)-[:HAS\_VENUE] ->(vd:venue\_dim)) WHERE vd.city = 'Mumbai' RETURN ud.umpire2;

Graphical user interface, application

Description automatically generated

**SQL query:**

select ud.umpire2 from ipl\_sch.matches\_fact mf, ipl\_sch.venue\_dim vd, ipl\_sch.umpire\_dim ud

where mf.venue\_id = vd.venue\_id and mf.umpire\_id = ud.umpire\_id and vd.city = 'Mumbai';

Graphical user interface, application, Word

Description automatically generated

SQL is a declarative language, so it used structure language to fetch data from relational tables, while CQL is a graph traversal language and uses pattern matching to traverse the graph and retrieve data from nodes.

1. **COUNT:** Retrieve count of total number of matches played by team1 against team2.

**CQL query:**

MATCH ((mf:matches\_fact) - [:HAS\_TEAM] ->(td:teams\_dim)) RETURN td.team1,td.team2, COUNT(mf.match\_id) AS Team\_matches\_played\_count;

A screenshot of a computer

Description automatically generated with medium confidence

**SQL query:**

select td.team1,td.team2,COUNT(mf.match\_id) AS Team\_matches\_played\_count from ipl\_sch.matches\_fact mf, ipl\_sch.teams\_dim td

where mf.team\_id = td.team\_id GROUP BY td.team1,td.team2;

Graphical user interface, text, application

Description automatically generated

The SQL query uses the traditional relational database model and count aggregate function for match\_id and grouping the data based on the team1 and team2 column. In comparison, the CQL query uses the graph database model, where the match\_fact and teams\_dim nodes are connected by a "HAS\_TEAM" relationship.

In general, graph databases are best suited for complex data with many relationships between components. Graph databases excel in traversing and querying these relationships, and they can handle large-scale, highly connected data with better performance and scalability than traditional relational databases.

On the other hand, relational databases are ideal for structured data, where the schema is well defined, and data is normalized. Relational databases excel in performing aggregations, filtering, and sorting on large data sets with high efficiency.

1. **WHERE condition:** Retrieve all match\_ids who have won with more than 5 wickets in a match.

**CQL query:** MATCH (mf:matches\_fact) WHERE tofloat(mf.win\_by\_wickets) > 5 RETURN mf.match\_id;

Graphical user interface, application

Description automatically generated with medium confidence

**SQL query:**

select mf.match\_id from ipl\_sch.matches\_fact mf where mf.win\_by\_wickets > 5;

Graphical user interface, application

Description automatically generated

In scenario of retrieving records, SQL queries are faster than the CQL queries for medium size datasets. However, for larger datasets with complex relationships between entities/nodes, the CQL queries are faster and more efficient.

1. **TOP 5**: Retrieve top 5 match ids where player of the match was Rashid Khan.

**CQL query:**

A screenshot of a computer

Description automatically generated with medium confidence

**SQL query:**

Graphical user interface, text, application, email

Description automatically generated

The SQL queries use the TOP clause to restrict the number of rows returned from the database, while the CQL queries use the LIMIT clause to restrict the number of nodes or relationships returned from the graph. SQL databases are faster for medium sized databases whereas CQL should be preferred for larger datasets which have complex relationship structure amount themselves.

1. **AVG aggregate function:** Evaluate average of win\_by\_wickets for Mumbai Indians.

**CQL query:**

Text

Description automatically generated

**SQL query:**

Graphical user interface, text, application

Description automatically generated

The SQL and CQL both have and aggregate function to return average of the parameter.

Aggregate functions perform better in case of traditional relational databases like SQL and should be preferred where aggregation is need for large number of records.

1. **Retrieve records:** Retrieve all records for matches by joining venue dimension and team dimension tables.

**CQL query:**

Graphical user interface, text, application

Description automatically generated

**SQL query:**

Graphical user interface

Description automatically generated

SQL is a declarative language, so it used structure language to fetch data from relational tables, while CQL is a graph traversal language and uses pattern matching to traverse the graph and retrieve data from nodes.

**Overall differences between relational and graph databases:**

* Relational databases store data in tables with a fixed schema, and relationships between tables are defined through foreign keys. In contrast, graph databases store data as nodes and edges, with the relationships between data being an integral part of the data model. This difference in data model affects how data is stored and retrieved.
* In relational databases, retrieving data from multiple tables involves joining them together, which can be computationally expensive for large datasets. In graph databases, traversing relationships between nodes is a fast operation and is typically done through graph algorithms.
* Graph databases are well-suited for use cases where the relationships between data are important, and the structure of the data is flexible. Relational databases are better suited for use cases where data is well-structured and relationships between data are simple.

**Conclusion:**

We have modelled DataMart for IPL matches from 2008 to 2019. The Kaggle dataset contains all the information regarding all the matches played and the match outcomes for all the teams. Considering IPL cricket popularity, we have gained useful match insights and team impact report to assist key stakeholder such a Fantasy cricket platforms, sponsors, investors, or clubs in making their business decisions. Fantasy cricket platforms require player and teams’ historical data for creating appropriate biddings whereas clubs, investors or sponsors can determine team’s performance based on the match stats and accordingly make an offer or sign a team.

After finalizing an ER schema – star schema, we started by creating with SSIS, which defines a ETL process to load data in various dimension and fact tables. We further created SSRS reports and Tableau visualization to give better understanding of the matches and teams which had significant impact in that IPL season. We also created a graph database using Neo4J which uses the same IPL dataset and compared the identical CQL queries with the traditional SQL database queries.

**Bibliography:**

Dataset: <https://www.kaggle.com/datasets/ashraykothari/ipldataset?select=matches.csv>

Moodle: <https://elearning.dbs.ie/course/view.php?id=17152>