

IPL Cricket Match Data Integration

A PROJECT REPORT

Submitted by

Bhushan Sagar (INT683)



CONTENTS

- CHAPTER 1: - INTRODUCTION
- CHAPTER 2: - PROJECT OVERVIEW
- CHAPTER 3: - DATA PIPELINE DESIGN (APPROCHES)
- CHAPTER 4: - DATA MODELLING
- CHAPTER 5: - IPL CRICKET MATCH DATA INTEGRATION (SQL SERVER)
- CHAPTER 6: - IPL CRICKET MATCH DATA INTEGRATION (SNOWFLAKE)
- CHAPTER 7: - IPL CRICKET MATCH DATA INTEGRATION (AWS ACCOUSTIC)
- CHAPTER 8: - IPL CRICKET MATCH DATA INTEGRATION (REAL TIME STREAMING)
- CHAPTER 9: - APROCHES WORKING ON
- CHAPTER 10: - POWER BI (DATA VISUALIZATION)
- CHAPTER 11: - COST ANALYSIS FOR DIFFERENT ETL AND BI APPROCHES
- CHAPTER 12: - LINK FOR PROJECT
- CHAPTER 13: - CONCLUSION AND LEARNINGS

CHAPTER 1: INTRODUCTION

Introduction:

In the realm of IPL cricket match data integration, organizations face the challenge of efficiently extracting, transforming, and loading (ETL) vast amounts of data from diverse sources to derive actionable insights. This project addresses this challenge by exploring multiple solutions to the single problem statement of integrating IPL cricket match data for visualization and analysis. Each solution offers a unique approach to ETL processes and data visualization, catering to different infrastructural requirements and preferences of organizations. Let's delve into the problem statement and the proposed solutions:

Problem Statement: The objective is to design a data integration solution for IPL cricket match data that enables visualization and analysis for stakeholders. The challenge lies in efficiently handling the ETL process from disparate sources, ensuring data quality, and providing seamless integration with visualization tools.

Proposed Solutions:

1. **SSIS to SQL Server and Power BI:** Utilizing SQL Server Integration Services (SSIS), IPL cricket match data is extracted, transformed, and loaded into SQL Server. Subsequently, Power BI connects to SQL Server for visualization, providing stakeholders with interactive dashboards and reports.
2. **SSIS to Snowflake and Power BI:** Leveraging SSIS, IPL cricket match data is loaded into Snowflake Data Warehouse. Power BI connects to Snowflake via ODBC for visualization, enabling scalable analytics and insights.
3. **AWS Cloud Architecture:** IPL cricket match data is retrieved from S3 using AWS Glue for ETL processes. Data quality checks are performed, and erroneous data is queried using Athena. Clean data is loaded into Redshift Data Warehouse, connecting to Power BI for visualization.
4. **Azure Cloud Architecture:** IPL cricket match data is sourced from Azure Blob Storage and processed in Azure Synapse Analytics using ETL processes. The transformed data is stored in Azure Synapse Analytics Warehouse, integrated with Power BI for visualization.
5. **On-Premises or Traditional Approach:** IPL cricket match data is extracted from HDFS and processed using PySpark, then stored in Apache Hive. Airflow orchestrates the ETL workflow, connecting to Power BI for visualization.
6. **Streaming and NoSQL:** IPL cricket match data is transformed into JSON format using Python and streamed with timestamp data. Cosmos DB consumes the data for real-time analytics, enabling dynamic visualization in Power BI.

Data Pipeline:

Our objective is to develop a robust data pipeline tailored to handle the wealth of information generated during IPL matches. This pipeline will facilitate the extraction, transformation, and loading (ETL) of ball-by-ball data, enabling stakeholders to derive valuable insights and make data-driven decisions.

Dataset Description:

The dataset encompasses comprehensive ball-by-ball data captured during IPL matches. It includes a multitude of dimensions, such as:

- **Match:** Contains details about each cricket match, including match ID, teams involved, match date, venue, toss information, outcome, and more.
- **Player:** Stores information about cricket players, such as player ID, name, date of birth, batting hand, bowling skill, and country.
- **Team:** Represents cricket teams participating in the IPL, including team ID, team name, and logos.
- **Venue:** Provides information about cricket venues, including venue ID, venue name, and city.
- **Innings:** Contains data about innings played in each match, including inning number, batting team, and bowling team.
- **Ball_by_ball:** Contains detailed ball-by-ball data for each match, including match ID, over ID, ball ID, innings number, batting team, bowling team, players involved, and more.
- **Batsman_scored:** Stores information about runs scored in each ball, including match ID, over ID, ball ID, runs scored, and innings number.
- **Batting_style:** Defines batting styles of players.
- **Bowling:** Defines bowling skills of players.
- **City:** Contains information about cities where matches are played.
- **Country:** Provides information about countries.
- **Extra_run:** Contains details about extra runs scored in each ball.
- **Extra_type:** Defines types of extra runs (leg byes, wides, etc.).
- **Out_type:** Defines types of player dismissals (caught, bowled, run out, etc.).
- **Outcome:** Defines match outcomes (result, no result, super over).
- **Player_match:** Maps players to matches, including match ID, player ID, role ID, and team ID.
- **Role:** Defines roles of players in matches.
- **Season:** Contains information about IPL seasons, including season ID, man of the series, orange cap winner, purple cap winner, and season year.
- **Win_by:** Defines win types (runs, wickets, etc.).
- **Wicket_taken:** Contains information about wickets taken in each match.
- **Toss_decision:** Defines toss decisions (field, bat).

Our endeavour is to construct a data pipeline that not only efficiently manages this dataset but also ensures data integrity, accuracy, and timeliness throughout the ETL process. By adhering to best practices and leveraging tools like SSIS (SQL Server Integration Services), we aim to create a scalable and sustainable solution that meets the evolving needs of the cricket ecosystem.

CHAPTER 2: PROJECT OVERVIEW

The Indian Premier League (IPL) stands as a pinnacle of modern cricket, blending sporting prowess with entertainment, attracting a global audience, and revolutionizing the economics of cricket. This project delves into the realm of IPL cricket match data integration, aiming to leverage the wealth of data generated during each match to drive insights, inform decision-making, and enhance the overall cricket ecosystem.

Significance of IPL: The IPL has transcended traditional cricket boundaries, emerging as a premier Twenty20 cricket league renowned for its international appeal, high-octane matches, and star-studded line-ups. With its unique franchise-based model, the IPL has not only redefined cricketing dynamics but also catalysed the growth of cricket as a global sport, attracting talent from around the world and captivating audiences across demographics.

Dataset Description: At the heart of this project lies a comprehensive dataset encompassing ball-by-ball data from IPL matches. This dataset captures intricate details of player performances, team statistics, match outcomes, and various other dimensions essential for dissecting and analysing cricket matches. From match details to player profiles, batting styles to bowling skills, the dataset offers a holistic view of the IPL cricketing landscape.

Project Objectives: The primary objective of this project is to design and implement a robust data pipeline using SQL Server Integration Services (SSIS) for IPL cricket match data integration. The pipeline aims to extract, transform, and load (ETL) data from diverse sources, ensuring its accuracy, consistency, and timeliness. By adhering to best practices and leveraging SSIS capabilities, the project seeks to empower stakeholders with actionable insights derived from IPL cricket match data.

Expected Outcomes:

- Development of a scalable and efficient data pipeline for IPL cricket match data integration.
- Creation of a relational database schema optimized for storing and querying IPL cricket match data.
- Implementation of ETL processes that adhere to data validation, transformation, and loading requirements.
- Delivery of fully functional SSIS packages, accompanied by comprehensive documentation covering pipeline architecture, data model, ETL processes, and package configurations.

Impact and Implications: The successful implementation of the IPL Cricket Match Data Integration project is poised to have far-reaching implications across the cricket ecosystem. By providing stakeholders with timely, accurate, and insightful data, the project aims to:

- Enable cricket franchises to optimize player selection, team strategies, and performance analysis.
- Empower broadcasters and analysts to enhance viewer engagement through data-driven storytelling and visualizations.
- Facilitate research and innovation in sports analytics, contributing to the advancement of cricketing strategies and techniques.
- Foster a culture of data-driven decision-making within the cricketing community, driving continuous improvement and innovation in the sport.

CHAPTER 3: DATA PIPELINE DESIGN

Solution 1: SSIS to SQL Server and Power BI

1. Data Extraction:

- IPL cricket match data is extracted from various sources such as databases or files using SSIS Data Flow tasks.
- Source systems like SQL Server or flat files are queried to retrieve the required data.

2. Data Transformation:

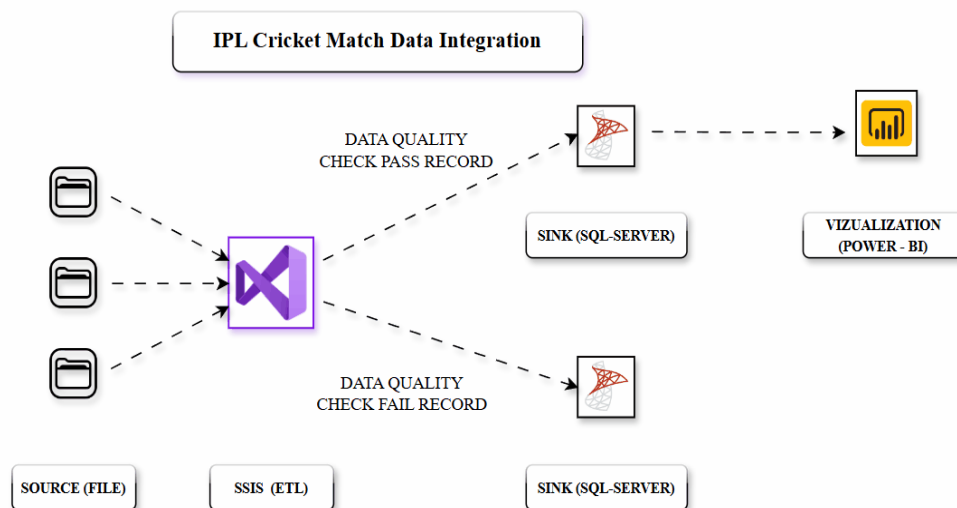
- SSIS transformations are applied to cleanse and transform the extracted data.
- Data cleansing techniques are used to remove duplicates, correct errors, and standardize formats.
- Business logic is applied to derive insights or aggregate data for analysis.

3. Data Loading:

- The transformed data is loaded into SQL Server, which serves as the target destination for storage.
- SSIS Data Flow destinations are used to load data into SQL Server tables efficiently.

4. Visualization and Analysis:

- Power BI connects to SQL Server to visualize the IPL cricket match data.
- Interactive dashboards, reports, and visualizations are created in Power BI to enable stakeholders to analyse the data effectively.



Solution 2: SSIS to Snowflake and Power BI

1. Data Extraction:

- IPL cricket match data is extracted using SSIS from various sources such as databases or files.
- Source systems like SQL Server or flat files are queried to retrieve the required data.

2. Data Transformation:

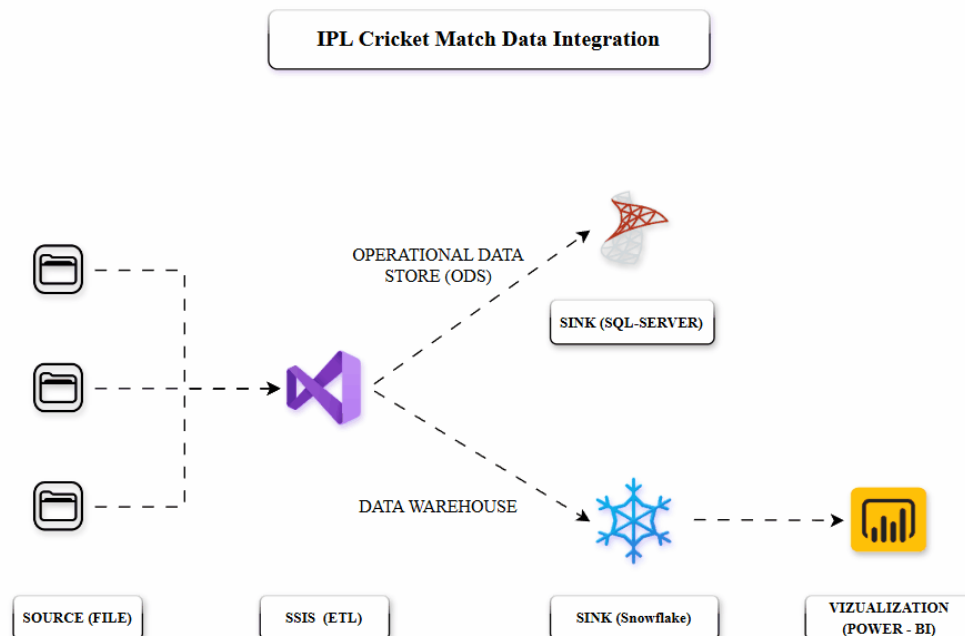
- SSIS transformations are applied to cleanse and transform the extracted data.
- Data cleansing techniques are used to remove duplicates, correct errors, and standardize formats.
- Business logic is applied to derive insights or aggregate data for analysis.

3. Data Loading:

- The transformed data is loaded into Snowflake Data Warehouse, which serves as the target destination for storage.
- SSIS Data Flow destinations are used to load data into Snowflake tables efficiently.

4. Visualization and Analysis:

- Power BI connects to Snowflake Data Warehouse to visualize the IPL cricket match data.
- Interactive dashboards, reports, and visualizations are created in Power BI to enable stakeholders to analyse the data effectively.



Solution 3: AWS Cloud Architecture

1. Data Extraction:

- IPL cricket match data is read from S3 storage using AWS Glue for ETL processes.
- Glue crawlers are utilized to infer schema and metadata from the data stored in S3.

2. Data Transformation:

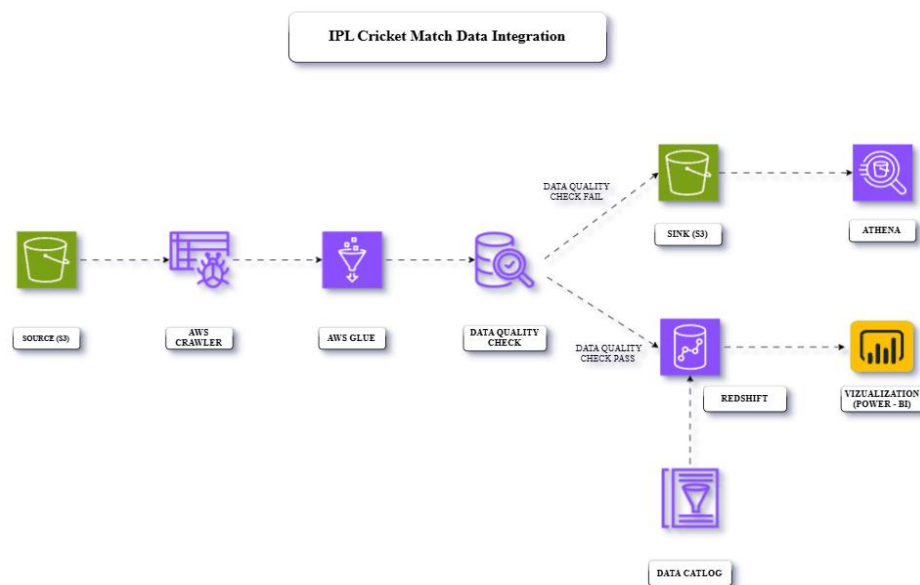
- AWS Glue ETL jobs are used to cleanse, transform, and enrich the IPL cricket match data.
- Data quality checks are performed to ensure the integrity and reliability of the data.

3. Data Loading:

- Good quality data is loaded into Redshift Data Warehouse for storage and analysis.
- Bad quality data is stored in S3 for further investigation or processing.

4. Visualization and Analysis:

- Power BI connects to Redshift Data Warehouse to visualize and analyze the IPL cricket match data.
- Interactive dashboards, reports, and visualizations are created in Power BI to facilitate data-driven decision-making.



Solution 4: Azure Cloud Architecture

1. Data Extraction:

- IPL cricket match data is read from Azure Blob Storage using Azure Data Factory for ETL processes.
- Azure Data Factory pipelines are used to orchestrate data movement and transformations.

2. Data Transformation:

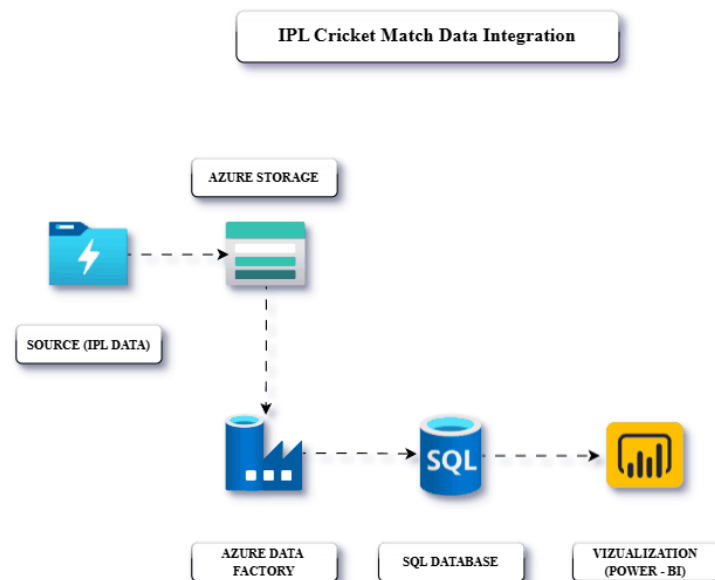
- Data flows within Azure Synapse Analytics are employed to cleanse, transform, and aggregate the IPL cricket match data.
- Azure Synapse Analytics provides capabilities for data preparation and data integration tasks.

3. Data Loading:

- The transformed data is stored in the data warehouse of Azure Synapse Analytics for storage and analysis.
- Azure Synapse Analytics provides scalable and high-performance storage for analytical workloads.

4. Visualization and Analysis:

- Power BI connects to Azure Synapse Analytics to visualize and analyze the IPL cricket match data.
- Interactive dashboards, reports, and visualizations are created in Power BI to facilitate data exploration and insights.



Solution 5: Streaming and NoSQL

1. Data Streaming:

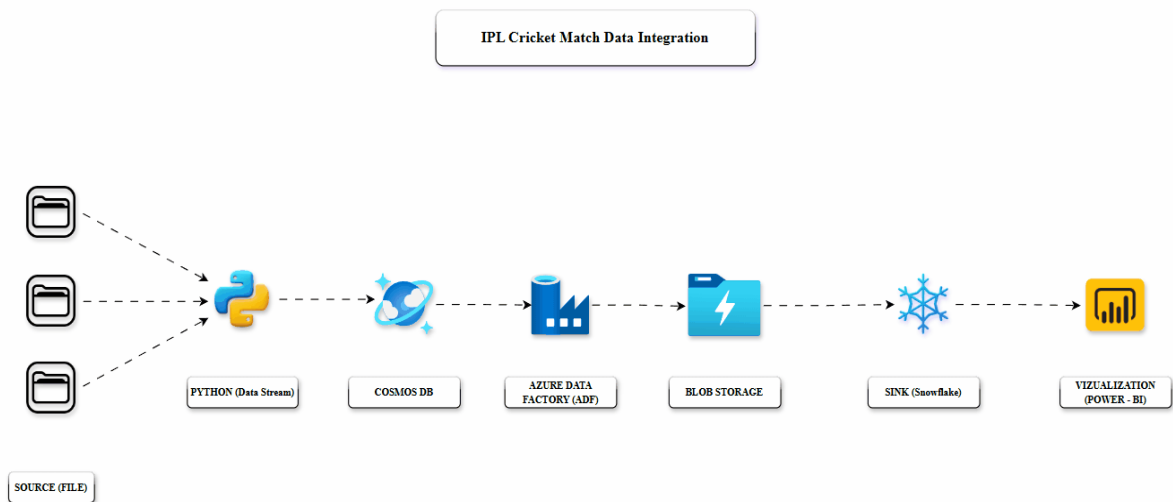
- IPL cricket match data is converted into JSON format using Python scripts.
- The data is streamed in real-time with timestamp information for continuous ingestion.

2. Data Storage:

- Cosmos DB is used as a NoSQL database to store and manage the streamed IPL cricket match data.
- Cosmos DB provides scalable and globally distributed storage for real-time analytics.

3. Visualization and Analysis:

- Power BI connects to Cosmos DB to visualize and analyze the streamed IPL cricket match data.
- Interactive dashboards, reports, and visualizations are created in Power BI to facilitate real-time data exploration and insights.



Solution 6: Traditional approach:

1. Data Extraction:

- IPL cricket match data is read from HDFS using Py-spark for ETL processes.
- Apache Airflow pipelines are used to orchestrate data movement and transformations.

2. Data Transformation:

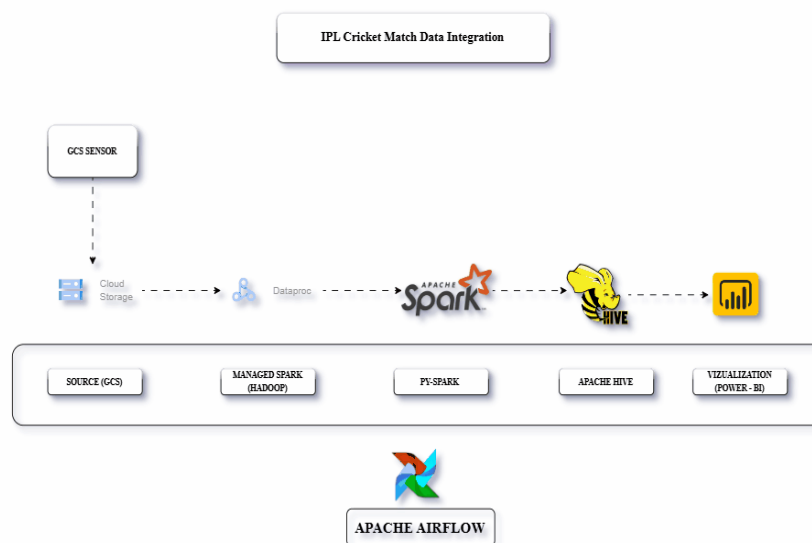
- GCP Data Proc Cluster (py-spark) are employed to cleanse, transform, and aggregate the IPL cricket match data.
- Py-spark provides capabilities for data preparation and data integration tasks.

3. Data Loading:

- The transformed data is stored in the data warehouse of Apache Hive for storage and analysis.
- Apache Hive provides scalable and high-performance storage for analytical workloads, mostly when data volume is huge.

4. Visualization and Analysis:

- Power BI connects to for visualize and analyse the IPL cricket match data.
- Interactive dashboards, reports, and visualizations are created in Power BI to facilitate data exploration and insights.



Key points Take care during Performing all approaches:

- **Data Validation:**

- Data quality checks are performed to ensure accuracy, completeness, and consistency of the loaded data.
- Validation rules are applied to detect and handle errors, duplicates, missing values, or outliers.

- **Incremental Loading:**

- Incremental loading techniques are implemented to update only the changed or new data since the last extraction, reducing processing time and resources.

- **Error Handling and Logging:**

- Mechanisms for error handling and logging are put in place to track and manage errors or exceptions during the pipeline execution.
- Logs provide visibility into the pipeline's performance, monitoring its progress and identifying any issues that need attention.

- **Scheduling and Automation:**

- The data pipeline is scheduled to run at regular intervals or triggered by events, ensuring timely and automated data processing.
- Scheduling tasks may be managed by cron jobs, job schedulers, or cloud-based orchestration services.

- **Integration with Visualization Tools:**

- Once data is loaded into the destination, it can be integrated with visualization tools like Power BI for analysis and reporting.

CHAPTER 4: DATA MODELLING

The data model for IPL cricket match data integration consists of a set of tables that organize and store different aspects of match-related information. overview of the key components of the data model:

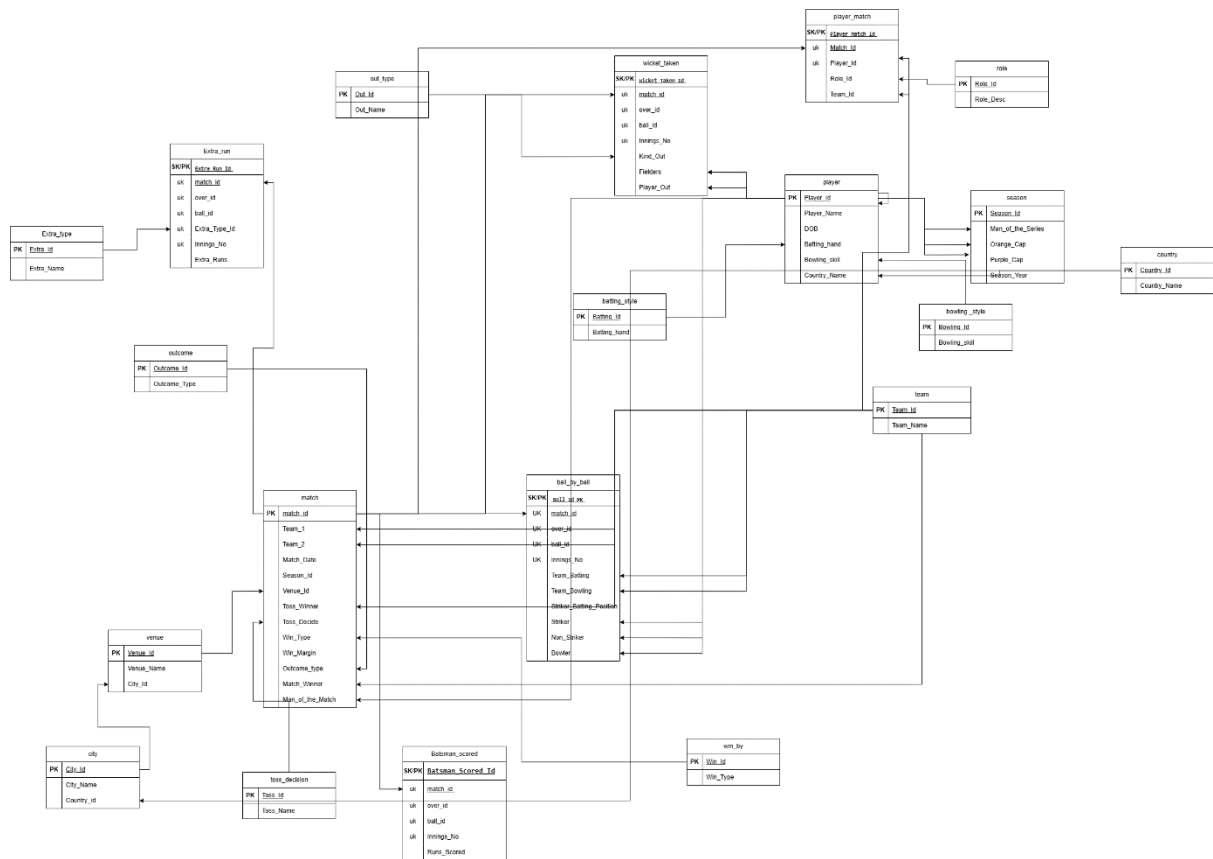
1. Dimension Tables:

- **Dim_Team:** Contains details of IPL teams participating in matches.
- **Dim_Country:** Stores information about countries.
- **Dim_City:** Holds data related to cities, with references to countries.
- **Dim_Batting_Style:** Defines various batting styles.
- **Dim_Bowling_Style:** Represents different bowling techniques.
- **Dim_Extra_Type:** Defines types of extra runs (like wides, leg-byes, etc.).
- **Dim_Role:** Stores player roles in matches.
- **Dim_Player:** Contains player details along with batting and bowling attributes.
- **Dim_Season:** Holds data about IPL seasons, including awards.
- **Dim_Toss_Decision:** Represents decisions made during the toss (e.g., bat or field).
- **Dim_Venue:** Stores information about match venues.
- **Dim_Win_Type:** Defines different win types (by runs or wickets).
- **Dim_Out_Type:** Represents various types of player outs.
- **Dim_Outcome:** Defines the outcome of matches (result or no result).
- **Dim_batsman_scored:** Records runs scored by batsmen.
- **Dim_extra_run:** Records extra runs conceded.
- **Dim_player_match:** Contains player match details.
- **Dim_wicket_taken:** Records wickets taken during matches.
- **Dim_match:** Stores comprehensive details of IPL matches.
- **Dim_ball_by_ball:** Records ball-by-ball details of matches.

2. Fact Table:

- **Fact_Match_Details:** Contains detailed information about IPL cricket match events, including player actions, scores, and match outcomes.

This data model provides a structured framework for organizing IPL cricket match data, enabling efficient storage, retrieval, and analysis. Dimension tables capture descriptive attributes, while the fact table contains detailed event information. The model ensures data integrity and facilitates comprehensive analysis of IPL match data across different dimensions.



Link - https://drive.google.com/file/d/1f4knkxhQgeUbpgUn8z1S6uj4irbPy6M/view?usp=drive_link

CHAPTER 5: IPL CRICKET MATCH DATA INTEGRATION (SQL SERVER)

- **SSIS (SQL Server Integration Services):** SSIS is a Microsoft tool used for data integration, transformation, and migration. It allows users to create workflows to extract data from various sources, transform it according to business requirements, and load it into target destinations, such as databases or data warehouses.
- **ETL (Extract, Transform, Load):** ETL is a process used to extract data from various sources, transform it into a desired format or structure, and load it into a target destination. It involves three main stages:
 - **Extract:** Retrieve data from one or multiple sources, such as databases, files, or APIs.
 - **Transform:** Modify, clean, or enrich the extracted data to meet business needs or comply with data standards.
 - **Load:** Load the transformed data into a target database, data warehouse, or other storage systems.

SSIS to SQL Server Solution:

The SSIS (SQL Server Integration Services) solution is employed for extracting, transforming, and loading IPL cricket match data into SQL Server for subsequent analysis and visualization in Power BI. Below is an overview of the components used in the SSIS package for this solution:

1. **Data Flow Task:**
 - The Data Flow Task is the core component of the SSIS package, responsible for defining data flow operations.
 - It consists of various data flow components connected via data paths, facilitating the movement and transformation of data.
2. **For Each Loop Container (For Dynamic Source):**
 - The For Each Loop Container iterates over a collection of objects, such as files in a directory or tables in a database.
 - It dynamically generates the list of files or tables to be processed based on specified criteria, enabling dynamic data source handling.
3. **Flat File Source:**
 - The Flat File Source component reads data from flat files, such as CSV or text files.
 - It extracts raw data from the source files and passes it to subsequent data flow components for processing.
4. **OLE DB Destination (SQL Server):**
 - The OLE DB Destination component loads data into a SQL Server database.
 - It maps the incoming data columns to the corresponding columns in the destination SQL Server table for insertion.

5. Derived Column Transformation:

- The Derived Column Transformation component creates new columns or modifies existing ones based on expressions or conditions.
- It is used to handle null values or perform data manipulations before loading into the destination.
- Ex- Dim wicket taken table- ISNULL(Fielders)? 0: Fielders

6. Fuzzy Lookup Transformation:

- The Fuzzy Lookup Transformation is used to perform approximate string matching, especially useful for handling player name mismatches or variations.
- It compares input data against a reference dataset (e.g., player names) and identifies potential matches based on similarity.

7. Slowly Changing Dimension (SCD) Transformation:

- The SCD Transformation detects and manages changes to dimension data over time.
- Used to handle changes over time in dimension data, especially numeric data with frequent updates
- It handles updates, inserts, and historical tracking of dimension records, ensuring data consistency and accuracy.

8. Lookup Transformation (For Incremental Load):

- The Lookup Transformation is used to perform lookups against reference datasets or tables.
- It is employed for incremental loading by comparing incoming data with existing data in the destination and determining whether to update or insert records.

9. Multicast:

- The Multicast component duplicates data rows and sends them to multiple outputs.
- Duplicates data rows and sends them to multiple outputs, facilitating parallel processing or branching logic. (Note- utilize but we can use)
- It is used to direct data flow to different downstream components for parallel processing or branching logic.

10. Conditional Split:

- The Conditional Split component routes rows based on specified conditions.
- It is utilized for error handling, segregating data based on predefined criteria for further processing or handling.

11. Script Component:

- The Script Component allows custom script code to be executed within the data flow pipeline.
- Executes custom script code within the data flow pipeline, essential for handling data mismatches in the venue column or implementing complex logic not achievable with built-in components.
- It is used for complex data transformations or custom logic not achievable through built-in SSIS components.

12. File System Task:

- The File System Task is used to perform file system operations such as moving, copying, or deleting files.
- It moves processed files to an archive folder after successful processing to maintain data integrity and organization.

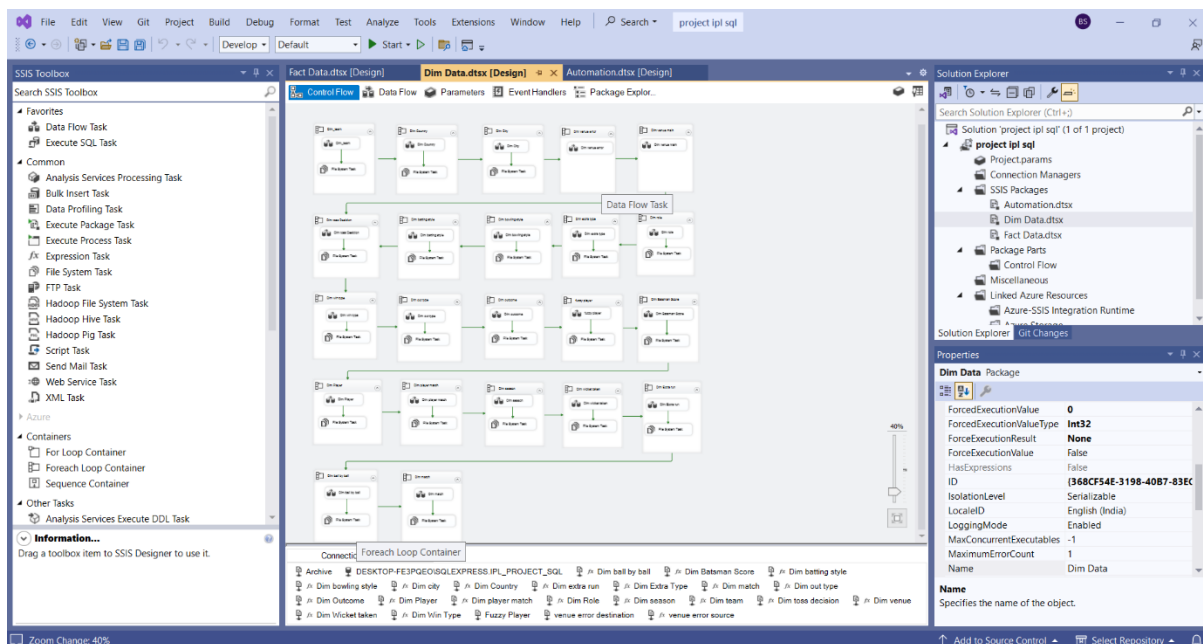
13. Deploy the SSIS Package:

- **Build** the SSIS project to create the .dtsx package file.
- **Connect to SQL Server** using SQL Server Management Studio (SSMS).
- **Integration Services CatLog:**
 - create SSISDB and Create Folder.
 - In IPL project folder and select Deploy Project.
 - Follow the Deployment Wizard to deploy the SSIS package.

14. Schedule the SSIS Package:

- In **SQL Server Agent**, creating a new job.
- Add a new job step with type **SQL Server Integration Services Package**.
- Configure the step to run the deployed SSIS package.
- Schedule the job to run at desired intervals.

SSIS Configuration:



Package Control Flow

- **Data Loading:**
 - Multiple Data Flow Tasks are responsible for loading data into various dimension tables.
- **Sequential Execution:**
 - Data Flow Tasks run sequentially due to preceding constraints.
 - But we can do it parallel also.
- **File Handling:**
 - Each Data Flow Task is followed by a File System Task.
 - The File System Task moves raw files from a source folder to an archive folder using two variables: `source_filepath` and `archive_filepath`.
- **Iteration Over Files:**
 - All Data Flow Tasks, File System Tasks, and Execute SQL tasks for the imported task are placed into a Foreach Loop Container.
 - The Foreach Loop Container iterates over each file in a defined folder path specified by a variable.

CHAPTER 6: IPL CRICKET MATCH DATA INTEGRATION (SNOWFLAKE)

- **ETL Process in SSIS:**

The initial part of the ETL process is executed using SSIS (SQL Server Integration Services), which includes extracting data from various sources, transforming it according to business logic, and loading it into a SQL Server database.

- **ETL Tasks in Snowflake:**

After the initial data load into the SQL Server database, further transformations and data management are performed in Snowflake.

Detailed ETL Process in SSIS:

1. **Data Extraction:**

- **For Each Loop Container:** Used to dynamically set the source, allowing the ETL process to handle multiple data files or sources.
- **Flat File Source:** Reads data from flat files.
- **OLE DB Source:** Extracts data from SQL Server or other OLE DB-compliant databases.

2. **Data Transformation:**

- **Data Conversion:** Converts data types as required.
- **Derived Column:** Creates new columns or modifies existing ones, handling null values, for example:

```
sql
Copy code
ISNULL(Fielders) ? 0 : Fielders
```

- **Fuzzy Lookup:** Handles minor data mismatches or variations, especially useful for ensuring data consistency (e.g., player names).
- **Slowly Changing Dimension (SCD):** Manages changes over time for dimension data, crucial for maintaining historical accuracy.
- **Lookup:** Performs lookups for foreign key integration, often used for incremental loads to improve performance.
- **Multicast:** Splits the data flow into multiple paths.
- **Conditional Split:** Directs rows to different outputs based on conditions, often used for error handling.
- **Script Component:** Custom processing of data, such as handling columns with mixed data types (e.g., int and string in a venue column).
- **File System Task:** Moves processed files to an archive folder to ensure no reprocessing.

3 Data Loading:

- **OLE DB Destination:** Loads data into SQL Server. Consider it as ODS Operational Data Store.
- **ODBC Destination:** Exports data to Snowflake for further processing and storage.

4 Data Loading:

- Load the transformed data into the destination database, which can be SQL Server, using SSIS ODBC Destination.

5 Error Handling:

- Implement error handling mechanisms using SSIS components and redirect to SQL table.

6 Incremental Loading:

- Utilize SSIS technique of Lookup to perform incremental loading and update only the changed or new records in the destination.

7 Logging and Monitoring:

- Configure logging in SSIS packages to capture execution details and monitor the ETL process for any failures or issues.

8 Deploy the SSIS Package:

- **Build** the SSIS project to create the .dtsx package file.
- **Connect to SQL Server** using SQL Server Management Studio (SSMS).
- **Integration Services CatLog:**
 - create SSISDB and Create Folder.
 - In Snowflake project folder and select Deploy Project.
 - Follow the Deployment Wizard to deploy the SSIS package.

h. Schedule the SSIS Package:

- In **SQL Server Agent**, creating a new job.
- Add a new job step with type SQL Server Integration Services Package.
- Configure the step to run the deployed SSIS package.
- Schedule the job to run at desired intervals.

Post-ETL Processing in Snowflake and Snowflake Implementation:

a. Stream Creation Change Data Capture (CDC):

- Create a stream on the relevant tables in Snowflake to capture changes in the data. For Change Data Capture.
- Leverage Snowflake's stream feature to implement CDC by capturing data changes in the stream tables and applying them to the target tables.
- For example:

```
sql
Copy code
CREATE OR REPLACE STREAM my_stream ON TABLE DIM_MATCH;
```

b. Time Travel:

- Utilize Snowflake's time travel feature to query historical data versions and perform analyses on different points in time.

c. Data Masking:

- Implement data masking policies in Snowflake to obfuscate sensitive data based on role-based access control.
- On Dim player – Player Name Has mask and Date of Birth also For User Data Analyst Because to hide the important info of client. Ex- Mobile Number

2024-05-25 10:24am

Databases Worksheets

DATA_ANALYST COMPUTE_WH (X-Small) Share

```
651 WHEN CURRENT_ROLE() IN ('Data_Engineer', 'ACCOUNTADMIN') THEN TO_CHAR(DOB, 'YYYY-MM-DD') /
652 ELSE CONCAT(TO_CHAR(DOB, 'YYYY'), '***-**-**')
653 END;
654
655
656
657
```

	PLAYER_ID	PLAYER_NAME	DOB	BATTING_HAND_ID	BOWLING_SKILL_ID	COUNTRY_ID
1	1	SC G*****	9999-01-01	1	1	1
2	2	BB M*****	9999-01-01	2	1	4
3	3	RT P*****	9999-01-01	2	1	5
4	4	DJ H*****	9999-01-01	2	2	5
5	5	Moha*****	9999-01-01	2	2	6
6	6	R D*****	9999-01-01	2	2	1
7	7	W Ja*****	9999-01-01	2	2	1
8	8	V Ko*****	9999-01-01	2	1	1
9	9	JH K*****	9999-01-01	2	3	2
10	10	CL W*****	9999-01-01	2	4	5
11	11	MV B*****	9999-01-01	2	1	2

Results Chart

Query Details

Query duration 132ms

Rows 470

Query ID 01b4cffc-3201-1a8d-00...

PLAYER_ID #

PLAYER_NAME 100% filled

DOB

d. Data Sharing:

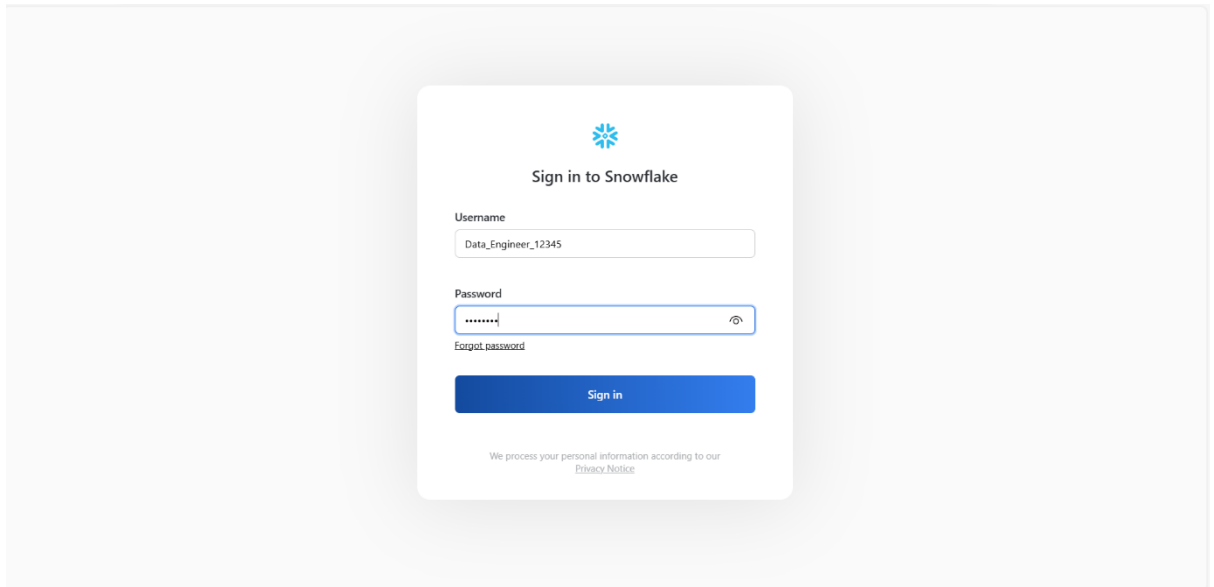
- Share relevant data and insights with other users or accounts by creating shares and granting appropriate permissions.
- **You Can Try:**

Link: https://lpihhit-read_only_user.snowflakecomputing.com

Admin name = Data_Engineer_12345

admin password = De@12345

Enter ID and Password:



Open Worksheet:

Click the file shown in image

A screenshot of the Snowflake "Worksheets" page. On the left is a sidebar with navigation options: "Create", "Search", "Projects", "Worksheets" (highlighted), "Notebooks", "Streamlit", "Dashboards", "App Packages", "Data", "Data Products", "AI & ML", "Monitoring", and "Admin". At the bottom of the sidebar, the user "DATA_ENGINEER_12345" is logged in as "SYSADMIN". The main area is titled "Worksheets" and features a "Recommended" section with a card titled "Simplify batch and streaming data pipelines with Dynamic Tables". Below this is a table with tabs for "Recent", "Shared with me", "My Worksheets", and "Folders". The "Recent" tab is active, showing a table with columns: "TITLE", "TYPE", "VIEWED", "UPDATED", and "ROLE".

TITLE	TYPE	VIEWED	UPDATED	ROLE
2024-06-06 1:18am	SQL	2 minutes ago	just now	ACCOUNTADMIN

A red curved arrow points from the bottom of the page towards the "2024-06-06 1:18am" entry in the table.

Select Query and Run:

The screenshot shows the Snowflake SQL Editor interface. The left sidebar displays the database structure with 'DATA_SHARE_DB' and 'SNOWFLAKE' visible. The main editor contains a SQL query with comments and a SELECT statement. The query is executed, and the results are displayed in a table below the editor. The 'Query Details' panel on the right shows the execution duration as 904ms and the number of rows as 136.6K.

```
1  -- use role
2
3  use role accountadmin;
4
5  -- To Check Datawarehouse
6
7  use schema ipl_data;
8
9  -- To Check Data in Tables
10
11
12 select * from Fact_Match_Details
13 select * from Dim_ball_by_ball
14 select * from Dim_match
15 select * from Dim_wicket_taken
16 select * from Dim_player_match
17 select * from Dim_extra_run
18 select * from Dim_batsman_scored
19 select * from Dim_city
```

	MATCHDETAIL_ID	MATCH_ID	OVER_ID	BALL_ID	INNINGS_NO
1	1	335987	1	1	1
2	2	335987	1	1	2
3	3	335987	1	2	1

Query Details

- Query duration: 904ms
- Rows: 136.6K
- Query ID: 01b4d018-3201-1aa3-0...

e. Performance Optimization:

- Optimize query performance by clustering tables based on frequently queried columns or using materialized views for pre-aggregated data.
- Also utilizing partitioning
- Ex- Before optimization for 1.36 lakh rows Execution time 882 ms

The screenshot shows the Snowflake SQL Editor interface. The left sidebar displays the database structure with 'IPL_PROJECT', 'SNOWFLAKE', and 'SNOWFLAKE_SAMPLE_DATA' visible. The main editor contains a SQL query with comments and a SELECT statement. The query is executed, and the results are displayed in a table below the editor. The 'Query Details' panel on the right shows the execution duration as 882ms and the number of rows as 136.6K.

```
516
517 insert into time_travel values(1),(2),(5)
518
519
520 delete from time_travel
521 where id = 1
522
523
524 select * from time_travel
525
526
527
528
529 -- Query Optimization
530 -- before clustering
531
532 select * from Fact_Match_Details where match_id > 50000
533
534
```

	MATCHDETAIL_ID	MATCH_ID	OVER_ID	BALL_ID	INNINGS_NO	TEAM_BATTING	TEAM_BOWLING
1	1	335987	1	1	1	1	
2	2	335987	1	1	2	2	
3	3	335987	1	2	1	1	

Query Details

- Query duration: 882ms
- Rows: 136.6K
- Query ID: 01b4cd9d-3201-1aa3-0...

- After optimization for 1.36 lakh rows Execution time 73 ms it reduces by 91.74%

The screenshot displays the Snowflake SQL Editor interface. The top bar shows the current database as 'IPL_PROJECT.IPL_DATA'. The left sidebar contains a 'Databases' tab and a 'Worksheets' tab. The main editor area shows a SQL query with line numbers 520 to 538. The query is a SELECT statement from 'Fact_Match_Details' with a WHERE clause 'match_id > 50000'. Below the query, the 'Results' tab is active, showing a table with 3 rows and 7 columns: MATCHDETAIL_ID, MATCH_ID, OVER_ID, BALL_ID, INNINGS_NO, TEAM_BATTING, and TEAM_BOWLING. The 'Query Details' panel on the right shows the query duration as 73ms and the number of rows as 136.6K. An orange arrow points from the 'Query Details' panel to the 'Results' tab.

- The significant volume of Big Data creates a substantial impact.

Zero Copy Clone:

Snowflake's zero-copy cloning feature allows you to create a clone of a database, schema, or table without actually copying the data. This is efficient for creating environments for development, testing, or creating data marts.

Clones are created instantaneously and do not consume additional storage beyond the metadata until changes are made to the clone or the original.

Data Marts:

Data marts are subsets of data warehouses tailored for specific business lines or departments. They provide focused data access for analytics and reporting.

DATAMART FOR DATA ANALYST TEAM

The screenshot shows the Snowflake web interface. The sidebar on the left displays a tree view of the database structure. The 'IPL_PROJECT' database is expanded, showing a 'DATAMART' schema. Under 'DATAMART', there are several tables: 'DATAMART_PLAYER', 'DATAMART_PLAYER_MATCH', 'DATAMART_TEAM', 'INFORMATION_SCHEMA', and 'IPL_DATA'. The 'IPL_DATA' schema is also expanded, showing tables like 'DIM_BALL_BY_BALL', 'DIM_BATSMAN_SCORED', 'DIM_BATTING_STYLE', 'DIM_BOWLING_STYLE', 'DIM_CITY', 'DIM_COUNTRY', and 'DIM_EXTRA_RUN'. The central query editor shows a SQL query: `select * from datamart_team`. The results pane at the bottom displays a table with two columns: 'TEAM_ID' and 'TEAM_NAME'. The table contains six rows of data. The right sidebar shows query details, including 'Query duration: 66ms', 'Rows: 13', and 'Query ID: 01b4d169-3201-1ac3-0...'. The 'TEAM_ID' column is highlighted in the results table.

TEAM_ID	TEAM_NAME
1	Kolkata Knight Riders
2	Royal Challengers Bangalore
3	Chennai Super Kings
4	Kings XI Punjab
5	Rajasthan Royals
6	Delhi Daredevils

PRODUCTION DATABASES

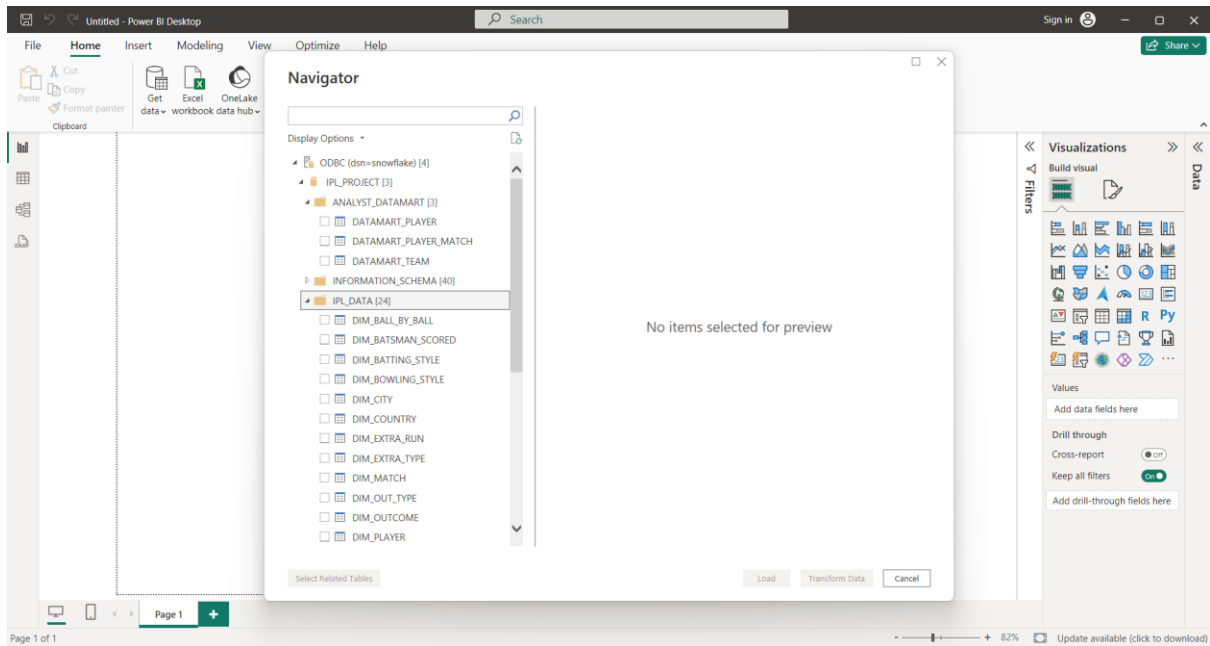
Cloning the tables ensures that the data mart has an up-to-date snapshot of the data without affecting the performance of the production database.

By following these steps, I effectively perform ETL processes both in SSIS and Snowflake, ensuring data integrity, accuracy, and performance in your analytics environment.

Snowflake to Power bi:

Creating the connection from Snowflake to Power BI by using ODBC connection

For that install the ODBC driver of snowflake.



CHAPTER 7: IPL CRICKET MATCH DATA INTEGRATION (AWS ACCOUSTIC)

Introduction:

This report outlines the development and implementation of an AWS Cloud Acoustic Data Pipeline. The pipeline is designed to ingest acoustic data from S3, perform data quality checks and transformations, and manage data storage and analysis using various AWS services such as Glue, Redshift, and Athena. Visualizations are performed using Power BI.

Objectives:

Ingest Acoustic Data: Read data from S3 storage.

Data Crawling and Schema Discovery: Use AWS Glue Crawler to identify the schema.

Data Quality Checks: Implement quality checks on the data.

Data Transformation: Transform the data to meet business requirements.

Data Storage and Management:

- Store data failing quality checks in a separate S3 bucket for further analysis.
- Move data passing quality checks to Redshift for efficient querying and reporting.

Data Analysis and Visualization: Utilize Athena for analysis of failed data and Power BI for visualizing data in Redshift.

Architecture:

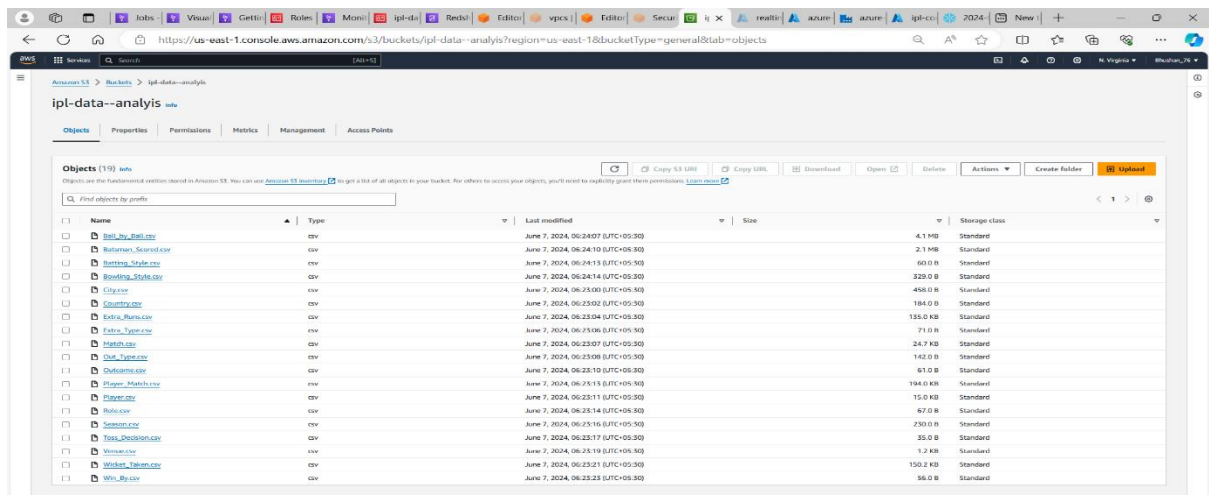
The architecture of the acoustic data pipeline involves several AWS services working in conjunction:

1. **Amazon S3:** Storage for raw and processed acoustic data.
2. **AWS Glue:** For ETL (Extract, Transform, Load) operations and schema discovery.
3. **Amazon Redshift:** Data warehouse for storing clean and transformed data.
4. **Amazon Athena:** Querying and analysing data directly from S3.
5. **Power BI:** Business intelligence tool for visualization.

Detailed Workflow:

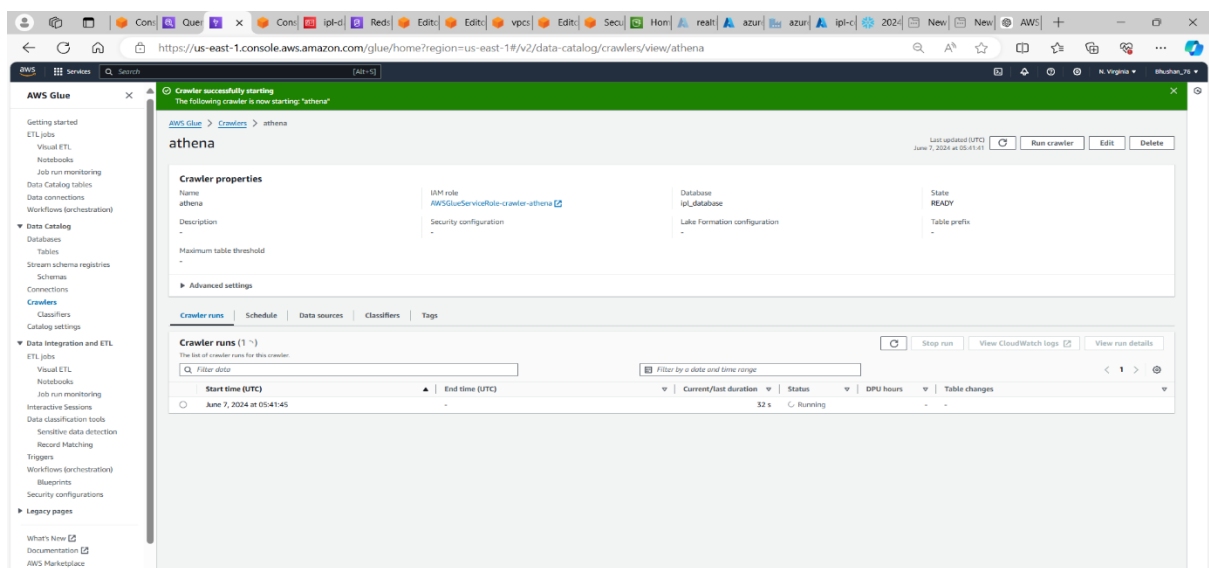
1. Data Ingestion:

- Acoustic data is uploaded to an S3 bucket (raw-acoustic-data).



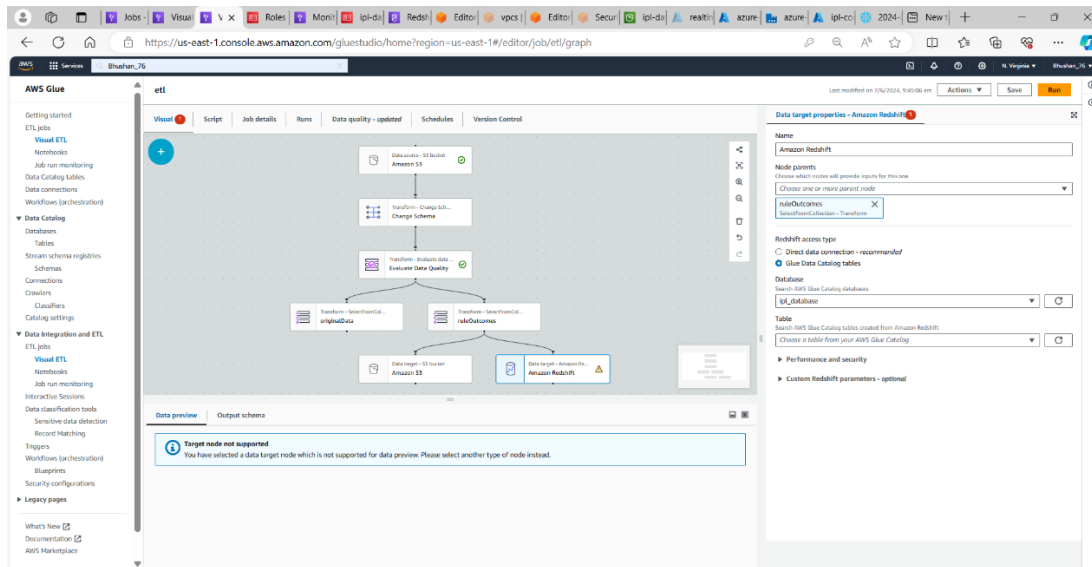
2. Schema Discovery:

- AWS Glue Crawler runs on the raw-acoustic-data bucket to detect the schema and create a table in the Glue Data catalogue.



3. Data Quality Check:

- AWS Glue ETL job reads the data from the Glue Data catalogue.
- Data quality checks are performed to ensure the integrity and accuracy of the data.
- Data failing the quality checks is moved to a separate S3 bucket (failed-quality-check-data).

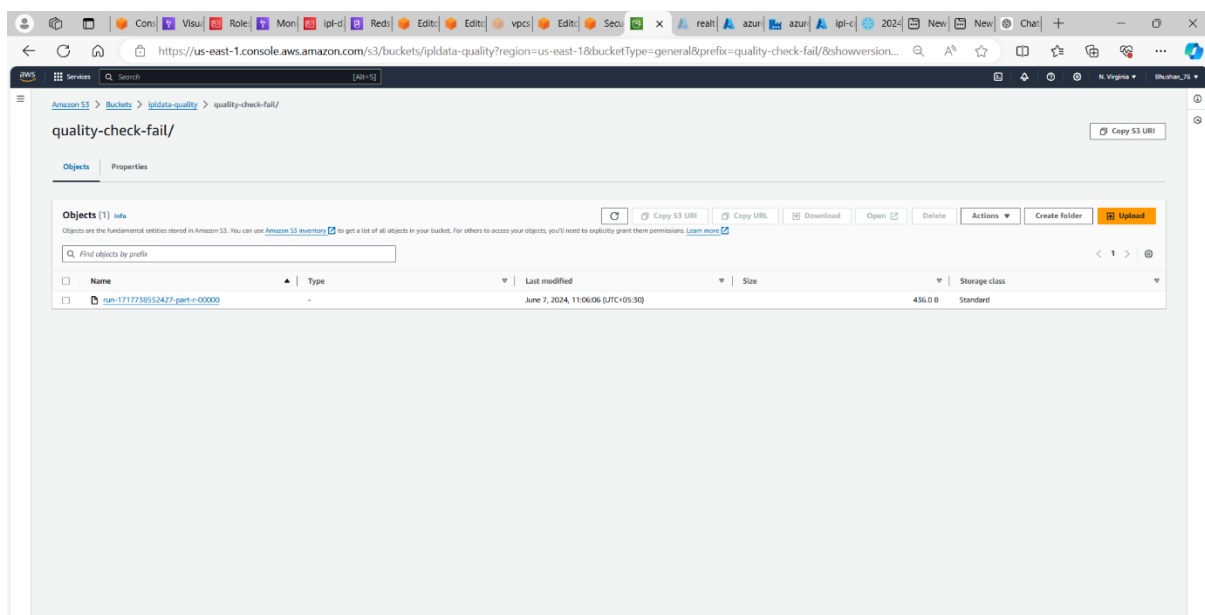


4. Data Transformation:

- Data passing the quality checks is transformed as needed (e.g., normalization, aggregation).

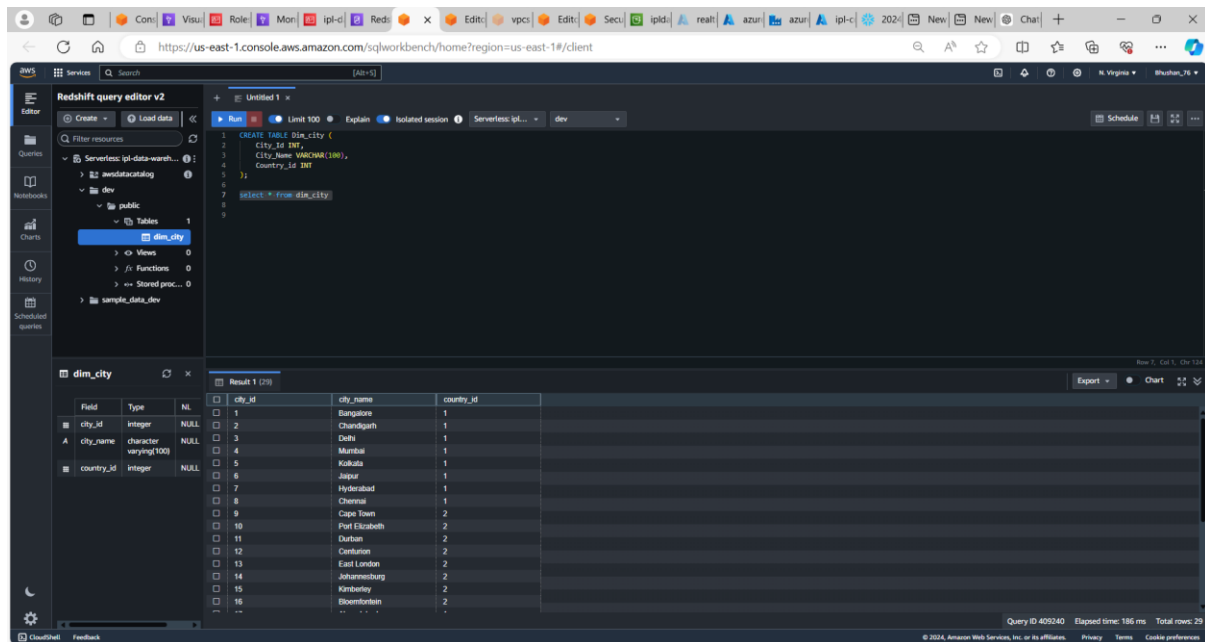
5. Data Storage:

- Transformed data is loaded into Amazon Redshift (acoustic-data-warehouse).
- Failed data remains in S3 (failed-quality-check-data) for further analysis using Athena.



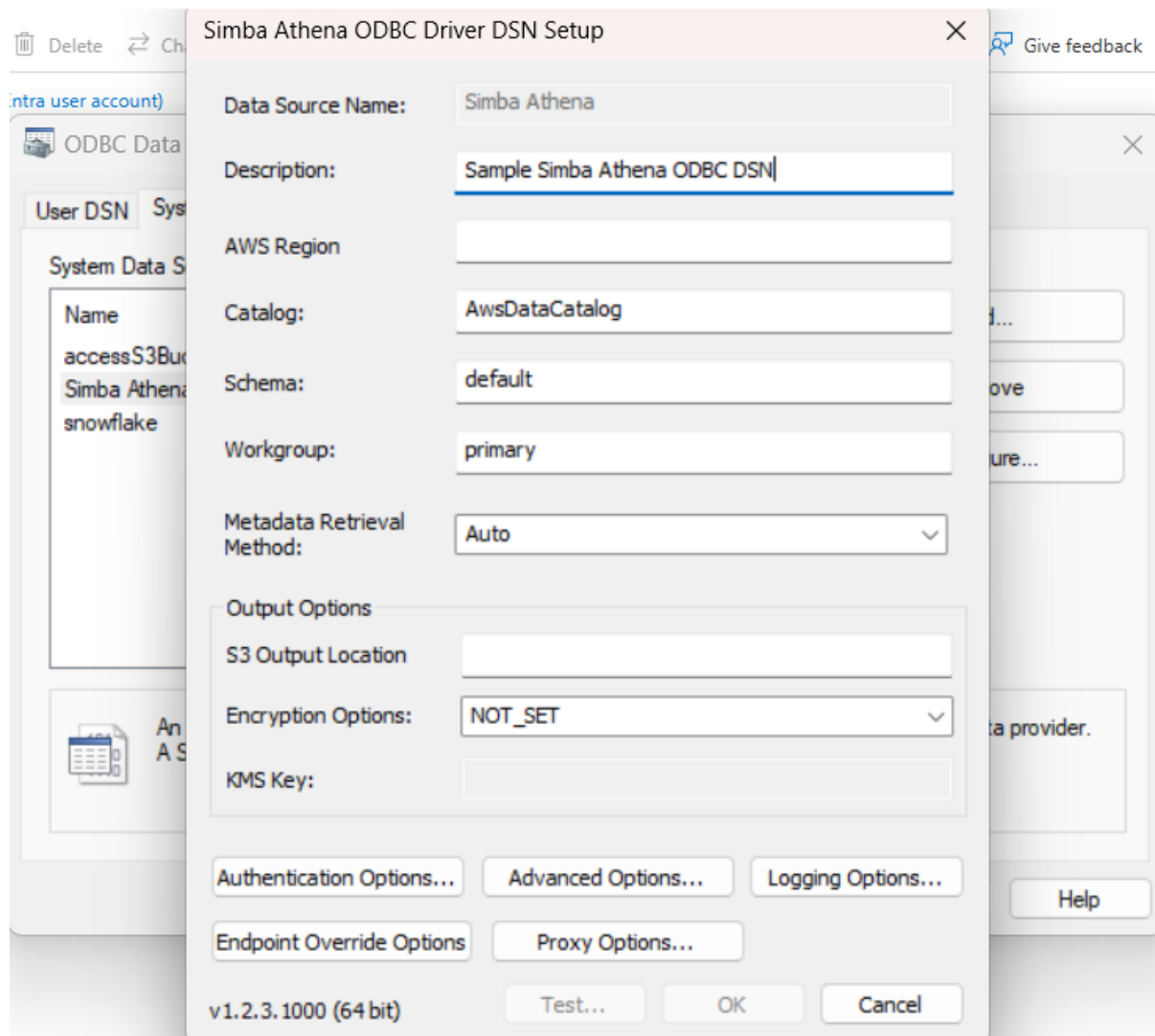
6. Data Analysis:

- Analysts use Amazon Athena to run queries on the failed data directly from S3.
- Clean and transformed data in Redshift is made available for visualization.



7. Visualization:

- Power BI connects to Amazon Redshift to generate interactive reports and dashboards for business stakeholders.
- Athena ODBC connection for power bi



Implementation Details:

AWS Glue Crawler and ETL:

1. **Crawler Configuration:**

- A Glue Crawler is configured to scan the raw-acoustic-data S3 bucket.
- The Crawler runs at scheduled intervals to keep the Glue Data Catalog updated with the latest schema information.

2. **ETL Job:**

- A Glue ETL job is developed using Py-Spark to read data from the Glue Data Catalog.
- The ETL job performs the following:
 - Reads the raw data.
 - Applies data quality checks (e.g., missing values, outlier detection).
 - Separates data into passing and failing categories.
 - Transforms passing data (e.g., normalization, feature engineering).
 - Writes the passing data to Amazon Redshift and failing data back to S3.

Data Quality Checks:

- Checks include:
 - Completeness: Ensuring no missing values in critical columns.
 - Validity: Ensuring data types and formats are correct.
 - Uniqueness: Ensuring no duplicate records.
 - Consistency: Ensuring data values are consistent across records.

Data Transformation:

- Transformation steps:
 - Normalization of acoustic measurements.
 - Aggregation of data for summary statistics.
 - Feature engineering for machine learning models.

Data Storage and Analysis:

1. **Amazon Redshift:**

- A Redshift cluster (acoustic-data-warehouse) is configured.
- Transformed data is loaded into Redshift tables for efficient querying.

2. **Amazon Athena:**

- Athena is configured to query data from the failed-quality-check-data S3 bucket.
- Analysts can use SQL to explore and analyse the failed data.

Visualization with Power BI:

- Power BI connects to Amazon Redshift using the provided ODBC/JDBC connection.
- Interactive reports and dashboards are created to visualize acoustic data trends, quality metrics, and other key insights.

Conclusion:

The AWS Cloud Acoustic Data Pipeline provides a robust solution for ingesting, processing, storing, and analysing acoustic data. By leveraging AWS services such as S3, Glue, Redshift, and Athena, the pipeline ensures data quality, enables efficient storage and querying, and supports comprehensive data analysis and visualization through Power BI. This architecture facilitates better decision-making and insights from acoustic data.

CHAPTER 8: - IPL CRICKET MATCH DATA INTEGRATION (REAL TIME STREAMING)

Solution for real-time data streaming and processing, leveraging Python scripts, Cosmos DB, Azure Data Factory (ADF), Azure Blob Storage, and Snowflake. The approach ensures efficient data ingestion, storage, and transformation, enabling robust data analysis and reporting.

objective is to design a real-time data pipeline that addresses the following challenges:

1. Efficiently ingesting and streaming data in real-time.
2. Handling semi-structured data formats such as JSON.
3. Ensuring data quality and consistency across various storage systems.
4. Enabling scalable and efficient data transformation and storage for analysis.

Proposed Solution

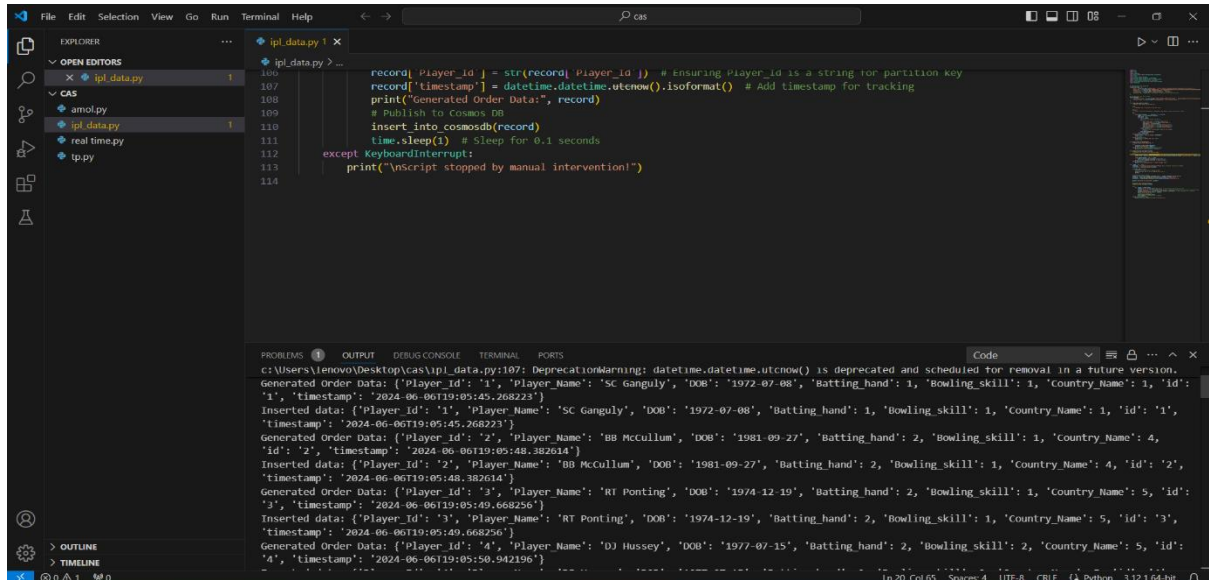
Our solution involves a multi-step process that integrates various tools and technologies to achieve real-time data streaming and processing:

1. **Ingest Data using Python Script:** Real-time data is ingested using a Python script that publishes data to Cosmos DB.
2. **Move Data to Blob Storage using ADF:** Azure Data Factory (ADF) pipelines are utilized to move data from Cosmos DB to Azure Blob Storage.
3. **Load Data into Snowflake:** Data from Blob Storage is loaded into Snowflake staging tables.
4. **Transform JSON Data in Snowflake:** JSON data is converted into a structured SQL format within Snowflake for further processing.

ETL Process:

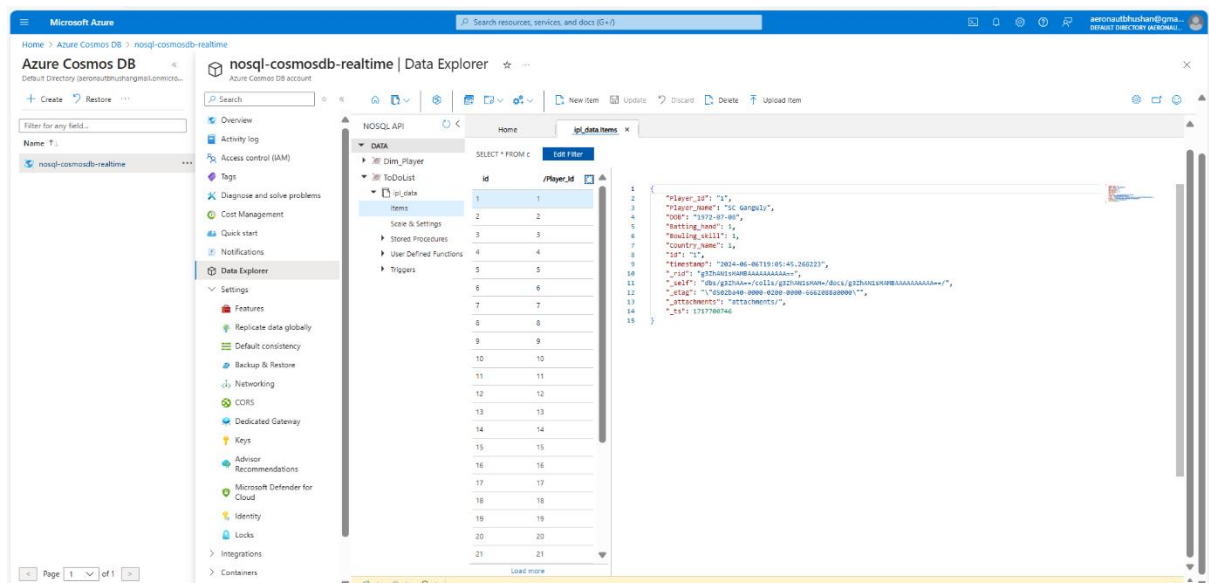
1. Ingest Data using Python Script:

A Python script is employed to publish real-time data to Cosmos DB.



```
106 record['Player_Id'] = str(record['Player_Id']) # Ensuring Player_Id is a string for partition key
107 record['timestamp'] = datetime.datetime.utcnow().isoformat() # Add timestamp for tracking
108 # Publish to Cosmos DB
109 insert_into_cosmosdb(record)
110 time.sleep(1) # Sleep for 0.1 seconds
111 except KeyboardInterrupt:
112     print("Script stopped by manual intervention!")
113
114
```

```
c:\Users\lenovo\Desktop\cas\ipl_data.py:107: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version.
Generated Order Data: {'Player_Id': '1', 'Player_Name': 'SC Ganguly', 'DOB': '1972-07-08', 'Batting_hand': 1, 'Bowling_skill': 1, 'Country_Name': 1, 'id': '1', 'timestamp': '2024-06-06T19:05:45.268223'}
Inserted data: {'Player_Id': '1', 'Player_Name': 'SC Ganguly', 'DOB': '1972-07-08', 'Batting_hand': 1, 'Bowling_skill': 1, 'Country_Name': 1, 'id': '1', 'timestamp': '2024-06-06T19:05:45.268223'}
Generated Order Data: {'Player_Id': '2', 'Player_Name': 'BB McCullum', 'DOB': '1981-09-27', 'Batting_hand': 2, 'Bowling_skill': 1, 'Country_Name': 4, 'id': '2', 'timestamp': '2024-06-06T19:05:48.382614'}
Inserted data: {'Player_Id': '2', 'Player_Name': 'BB McCullum', 'DOB': '1981-09-27', 'Batting_hand': 2, 'Bowling_skill': 1, 'Country_Name': 4, 'id': '2', 'timestamp': '2024-06-06T19:05:48.382614'}
Generated Order Data: {'Player_Id': '3', 'Player_Name': 'RT Ponting', 'DOB': '1974-12-19', 'Batting_hand': 2, 'Bowling_skill': 1, 'Country_Name': 5, 'id': '3', 'timestamp': '2024-06-06T19:05:49.668256'}
Inserted data: {'Player_Id': '3', 'Player_Name': 'RT Ponting', 'DOB': '1974-12-19', 'Batting_hand': 2, 'Bowling_skill': 1, 'Country_Name': 5, 'id': '3', 'timestamp': '2024-06-06T19:05:49.668256'}
Generated Order Data: {'Player_Id': '4', 'Player_Name': 'DJ Hussey', 'DOB': '1977-07-15', 'Batting_hand': 2, 'Bowling_skill': 2, 'Country_Name': 5, 'id': '4', 'timestamp': '2024-06-06T19:05:50.942196'}
```



2. Move Data to Blob Storage using Azure Data Factory (ADF):

Azure Data Factory (ADF) is used to transfer data from Cosmos DB to Azure Blob Storage.

1. **Create Linked Services:** Establish connections to Cosmos DB and Blob Storage.
2. **Create Datasets:** Define datasets for both data sources.
3. **Create Pipeline:** Set up a pipeline with a copy activity.

Microsoft Azure | Data Factory | adf-ipl-data | Search factory and documentation

Factory Resources

- Pipelines: 1
 - pipeline1
- Change Data Capture (preview): 0
- Datasets: 2
 - CosmosDBNoSqlContainer1
 - Json1
- Data flows: 0
- Power Query: 0

Activities

- Move and transform
- Synapse
- Azure Data Explorer
- Azure Function
- Batch Service
- Databricks
- Data Lake Analytics
- General
- HDInsight
- Iteration & conditionals
- Machine Learning
- Power Query

Copy data

move data from cosmos to blob

Run Succeeded

Successfully ran pipeline1 (Pipeline).
View pipeline run

Parameters Variables Settings Output

Pipeline run ID: a4ff5d5b-ae2b-4d36-bffd-a4b468a62263

Pipeline status: Succeeded

All status

Showing 1 - 1 of 1 items

Activity name	Activity status	Activity type	Run start	Duration	Integration runtime	User properties	Activity run ID
move data from cosmos to bl...	Succeeded	Copy data	6/7/2024, 12:34:08 AM	14s	AutoResolveIntegration		dbc6426b-27e5-48a8-86a2-f22121c57dbb

Microsoft Azure | Storage accounts | ipldatastorage | Containers

ipldatacontainer

Overview

Authentication method: Access key (Switch to Microsoft Entra user account)

Location: ipldatacontainer

Search blobs by prefix (case-sensitive)

Show deleted blobs

Add filter

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
data_6afe3744-1207-458b-93fe-15a6afe4042d_0283fc1d-3019-445e-8336-68ee5374cc8b.json	6/7/2024, 12:36:56 AM	Hot (Inferred)		Block blob	10.91 KB	Available

88°F Smoke

Search

ENG IN

00:54

07-06-2024

3. Load Data into Snowflake:

Create a Snowflake external stage to access the data in Azure Blob Storage and load the data into a staging table.

The screenshot shows the Snowflake web interface with a query executed against the `ipldata_real_time.ipl_dats.JSON_ROW` table. The query is `select * from ipldata_real_time.ipl_dats.JSON_ROW`. The results are displayed in a JSON format, showing 12 rows of player data. The right sidebar shows query details: Query duration 18ms, Rows 107, and Query ID 01b4d58c-3201-1a3-0...

Row	Batting_hand	Bowling_skill	Country_Name	DOB	Player_Id	Player_Name	attachments	etag	
1	{	"Batting_hand": 1,	"Bowling_skill": 1,	"Country_Name": 1,	"DOB": "1972-07-08",	"Player_Id": "1",	"Player_Name": "SC Ganguly",	"attachments": "attachments/",	"etag": "\f9f5"
2	{	"Batting_hand": 2,	"Bowling_skill": 1,	"Country_Name": 4,	"DOB": "1981-09-27",	"Player_Id": "2",	"Player_Name": "BB McCullum",	"attachments": "attachments/",	"etag": "\f9f5"
3	{	"Batting_hand": 2,	"Bowling_skill": 1,	"Country_Name": 5,	"DOB": "1974-12-19",	"Player_Id": "3",	"Player_Name": "RT Ponting",	"attachments": "attachments/",	"etag": "\f9f5"
4	{	"Batting_hand": 2,	"Bowling_skill": 2,	"Country_Name": 5,	"DOB": "1977-07-15",	"Player_Id": "4",	"Player_Name": "DJ Hussey",	"attachments": "attachments/",	"etag": "\f9f5"
5	{	"Batting_hand": 2,	"Bowling_skill": 2,	"Country_Name": 6,	"DOB": "1980-10-17",	"Player_Id": "5",	"Player_Name": "Mohammad Hafeez",	"attachments": "attachments/",	"etag": "\f9f5"
6	{	"Batting_hand": 2,	"Bowling_skill": 2,	"Country_Name": 1,	"DOB": "1973-01-11",	"Player_Id": "6",	"Player_Name": "R Dravid",	"attachments": "attachments/",	"etag": "\f9f5"
7	{	"Batting_hand": 2,	"Bowling_skill": 2,	"Country_Name": 1,	"DOB": "1978-02-16",	"Player_Id": "7",	"Player_Name": "W Jaffer",	"attachments": "attachments/",	"etag": "\f9f5"
8	{	"Batting_hand": 2,	"Bowling_skill": 1,	"Country_Name": 1,	"DOB": "1988-11-05",	"Player_Id": "8",	"Player_Name": "V Kohli",	"attachments": "attachments/",	"etag": "\f9f5"
9	{	"Batting_hand": 2,	"Bowling_skill": 3,	"Country_Name": 2,	"DOB": "1975-10-16",	"Player_Id": "9",	"Player_Name": "JH Kallis",	"attachments": "attachments/",	"etag": "\f9f5"
10	{	"Batting_hand": 2,	"Bowling_skill": 4,	"Country_Name": 5,	"DOB": "1983-08-18",	"Player_Id": "10",	"Player_Name": "CL White",	"attachments": "attachments/",	"etag": "\f9f5"
11	{	"Batting_hand": 2,	"Bowling_skill": 1,	"Country_Name": 2,	"DOB": "1976-12-03",	"Player_Id": "11",	"Player_Name": "MV Boucher",	"attachments": "attachments/",	"etag": "\f9f5"
12	{	"Batting_hand": 2,	"Bowling_skill": 5,	"Country_Name": 1,	"DOB": "1977-10-07",	"Player_Id": "12",	"Player_Name": "B Akhil",	"attachments": "attachments/",	"etag": "\f9f5"

4. Convert JSON to SQL Format:

In Snowflake, transform the JSON data into a structured SQL format.

The screenshot shows the Snowflake web interface with a query executed against the `ipldata_real_time.ipl_dats.JSON_ROW` table. The query is `select raw_file:Batting_hand::int as Batting,raw_file:Bowling_skill::int as Bowling_skill,raw_file:Country_Name::int as Country_Name,raw_file:DOB::Date as`. The results are displayed in a structured SQL format, showing 12 rows of player data. The right sidebar shows query details: Query duration 30ms, Rows 107, and Query ID 01b4d58c-3201-1a3-0... Below the query details, there are two bar charts: one for Batting (showing a distribution from 1 to 2) and one for Bowling_Skill (showing a distribution from 0 to 11).

	BATTING	BOWLING_SKILL	COUNTRY_NAME	DOB	PLAYER_ID	PLAYER_NAME
1	1	1	1	1972-07-08	1	SC Ganguly
2	2	1	4	1981-09-27	2	BB McCullum
3	2	1	5	1974-12-19	3	RT Ponting
4	2	2	5	1977-07-15	4	DJ Hussey
5	2	2	6	1980-10-17	5	Mohammad Hafeez
6	2	2	1	1973-01-11	6	R Dravid
7	2	2	1	1978-02-16	7	W Jaffer
8	2	1	1	1988-11-05	8	V Kohli
9	2	3	2	1975-10-16	9	JH Kallis
10	2	4	5	1983-08-18	10	CL White
11	2	1	2	1976-12-03	11	MV Boucher
12	2	5	1	1977-10-07	12	B Akhil

This outlines a robust and scalable real-time data streaming and processing solution. By integrating Python scripts, Cosmos DB, Azure Data Factory, Azure Blob Storage, and Snowflake, the solution effectively addresses the challenges of real-time data ingestion, storage, and transformation. Advanced Snowflake features like CDC, Zero Copy Clone, and Time Travel enhance data management and analysis capabilities, ensuring reliable and efficient data processing for business intelligence and analytics.

Video version available –

https://drive.google.com/file/d/1iRiVEPcFIZ5Wy1Ks8Ev2SMBKSoOyRkHm/view?usp=drive_link

CHAPTER 9: APPROCHES IN-PROGRESS

Traditional approach:

1. **Data Extraction:**
 - IPL cricket match data is read from HDFS using Py-spark for ETL processes.
 - Apache Airflow pipelines are used to orchestrate data movement and transformations.
2. **Data Transformation:**
 - GCP Data Proc Cluster (py-spark) are employed to cleanse, transform, and aggregate the IPL cricket match data.
 - Py-spark provides capabilities for data preparation and data integration tasks.
3. **Data Loading:**
 - The transformed data is stored in the data warehouse of Apache Hive for storage and analysis.
 - Apache Hive provides scalable and high-performance storage for analytical workloads, mostly when data volume is huge.
4. **Visualization and Analysis:**
 - Power BI connects to for visualize and analyse the IPL cricket match data.
 - Interactive dashboards, reports, and visualizations are created in Power BI to facilitate data exploration and insights.

Azure Cloud Architecture:

1. **Data Extraction:**
 - IPL cricket match data is read from Azure Blob Storage using Azure Data Factory for ETL processes.
 - Azure Data Factory pipelines are used to orchestrate data movement and transformations.
2. **Data Transformation:**
 - Data flows within Azure Synapse Analytics are employed to cleanse, transform, and aggregate the IPL cricket match data.
 - Azure Synapse Analytics provides capabilities for data preparation and data integration tasks.
3. **Data Loading:**
 - The transformed data is stored in the data warehouse of Azure Synapse Analytics for storage and analysis.
 - Azure Synapse Analytics provides scalable and high-performance storage for analytical workloads.
4. **Visualization and Analysis:**
 - Power BI connects to Azure Synapse Analytics to visualize and analyze the IPL cricket match data.
 - Interactive dashboards, reports, and visualizations are created in Power BI to facilitate data exploration and insights.

CHAPTER 10: - POWER BI (DATA VISUALIZATION)

As a cricket enthusiast and data Engineer, I have created an interactive Power BI dashboard to delve deep into IPL match statistics, providing stakeholders with actionable insights into team performance, player contributions, and match outcomes.

Objective: To develop a dynamic and visually appealing Power BI dashboard that facilitates in-depth analysis of IPL matches, allowing stakeholders to make informed decisions and devise strategies based on data-driven insights.

User Story: As a Data Engineer, I want to access an intuitive Power BI dashboard to explore IPL match data comprehensively. This dashboard should provide detailed insights into various aspects of match performance, including team scores, batting and bowling statistics, player contributions, and match outcomes. By having access to this rich data visualization tool, I aim to extract valuable insights that can inform strategic decisions for teams, sponsors, and broadcasters.

Acceptance Criteria:

1. Match Outcome Analysis:

- The dashboard should display a visual representation of match outcomes, highlighting the number of matches won by wickets versus runs.
- Percentage breakdowns of match outcomes should be provided, indicating the dominance of wicket-taking strategies.

2. Team Performance Metrics:

- Detailed metrics should be available for each team, including total runs scored, total wickets taken, run rates, and remaining balls.
- Historical trends of team performances should be accessible, enabling stakeholders to identify patterns and trends.

3. Player Statistics:

- The dashboard should feature individual player statistics such as batting strike rates, number of boundaries (fours and sixes) scored, and bowling figures.
- Special recognition should be given to players receiving awards such as Orange Cap (leading run-scorer), Purple Cap (leading wicket-taker), and Man of the Series.

4. Dynamic Filters and Slicers:

- Users should be able to filter data based on specific teams, seasons, or players of interest.
- Interactive slicers should allow users to focus on particular match formats (e.g., T20, playoffs) or stages of the tournament.

5. Data Integrity and Accuracy:

- The dashboard should ensure that data is accurately represented and updated regularly to reflect the latest match results and player performances.
- Measures and calculations should be thoroughly tested to maintain data integrity and consistency.

Dashboard Components:

- Visualizations: Cards and Donut charts representing various metrics.
- Tables and Matrices: Detailed tables showcasing match statistics and player performances.
- Filters and Slicers: Interactive elements enabling users to customize data views.

Data Model:

- Utilize a robust data model comprising fact and dimension tables to organize IPL match data efficiently.
- Ensure seamless integration of data sources to provide a unified view of match statistics.

Measures and Calculations:

- Implement DAX measures to calculate key performance indicators such as run rates, wicket counts, and player statistics.
- Validate measures to ensure accuracy and reliability in data analysis.

Deliverables:

- A visually appealing and user-friendly Power BI dashboard offering comprehensive insights into IPL match data.
- Detailed documentation outlining dashboard functionalities, data sources, and calculation methodologies.



Match Number

980912

Home

Live Score

Points Table

Royal Challengers Bangalore opted for batting first

Remaining Balls: 0

Run Needed: 46

Required Run Rate: 0

Royal Challengers Bangalore

227/4 (20.0)

Run Rate:11.35

Sunrisers Hyderabad

182/6 (20.0)

Run Rate:9.1

10 Twos

15 Four

9 Sixes

Team_Bowling

Bowler	Overs	Runs	Four	Sixes	Wicket
Royal Challengers Bangalore Adam Fraser Milne	4	44	4	2	
Royal Challengers Bangalore Harshal Vikram Patel	4	33	3	1	
Royal Challengers Bangalore Parvez Ghulam Rasool	4	31	2	2	
Total	20	182	15	9	

Innings_No

2

Season_Year

2016

Team_Batting

Striker	Runs	Four	Sixes	Strike_Rate
Sunrisers Hyderabad Akshath Ashish Reddy	32	2	3	177.78
Sunrisers Hyderabad David Andrew Warner	63	4	5	232.00
Sunrisers Hyderabad Deepak Jagbir Hooda	7	1		100.00
Sunrisers Hyderabad Eoin Joseph Gerard Morgan	23	2		129.41
Sunrisers Hyderabad Karn Sharma	28	3	1	162.50
Sunrisers Hyderabad Moises Constantino Henriques	20	2		82.61
Sunrisers Hyderabad Naman Vinaykumar Ojha	1			0.00
Sunrisers Hyderabad Shikhar Dhawan	8	1		88.89
Total	182	15	9	144.92

Sixes Contribute

Four Contribute

Wickets Contribute

Season_Year

2012

Season

Home

Live Score

Points Table

Orange

Christopher Henry Gayle

Purple Cap

Morne Morkel

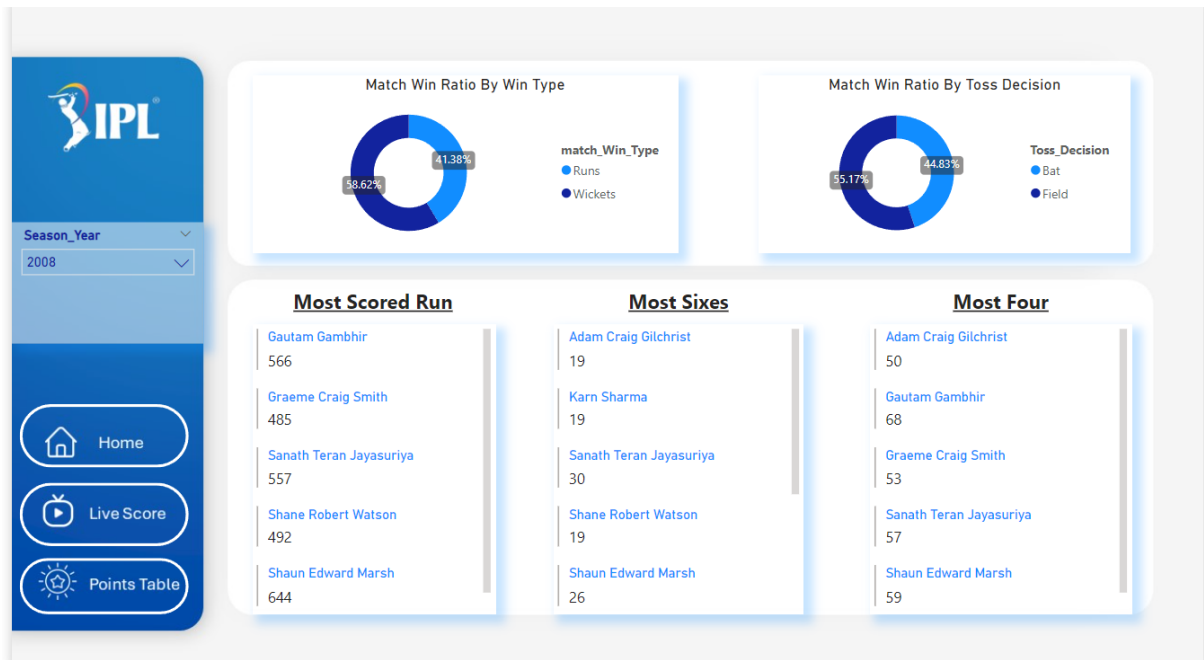
Total Matches: 74

Team	Team Name	Captain	Total Matches	Match Won
	Chennai Super Kings		9	6
	Delhi Daredevils		10	5

Man Of The Series

Sunil Philip Narine

44



Video Available:

https://drive.google.com/file/d/1Nb-87QHMSCGjydjIBTg1do-bXi7Z1Sb/view?usp=drive_link

CHAPTER 11: - COST ANALYSIS FOR DIFFERENT ETL AND BI APPROCHES

The costs associated with each solution will vary based on several factors including data volume, frequency of processing, compute requirements, and licensing. Below is a general cost comparison for each approach.

1. SSIS to SQL Server and Power BI:

Component	Cost Type	Estimated Monthly Cost
SQL Server License	Subscription/License	\$3,700 (Standard Edition)
SQL Server Hardware	Hardware/VM	\$1,000
SSIS License	Included with SQL Server	\$0
Power BI Pro	Subscription	\$9.99/user

Total: ~\$4,800/month

2. SSIS to Snowflake and Power BI:

Component	Cost Type	Estimated Monthly Cost
Snowflake Storage	Pay-as-you-go	\$40/TB
Snowflake Compute	Pay-as-you-go	\$2.00/credit
SSIS License	Subscription/License	\$1,300
Power BI Pro	Subscription	\$9.99/user

Total: ~\$1,940/month

3. AWS Cloud Architecture:

Component	Cost Type	Estimated Monthly Cost
S3 Storage	Pay-as-you-go	\$0.023/GB
AWS Glue	Pay-as-you-go	\$0.44/DPU-hour
Redshift Storage	Pay-as-you-go	\$0.25/GB
Redshift Compute	Pay-as-you-go	\$0.85/hour (dc2.large)
Power BI Pro	Subscription	\$9.99/user

Total: ~\$1,280/month

4. Azure Cloud Architecture:

Component	Cost Type	Estimated Monthly Cost
Azure Blob Storage	Pay-as-you-go	\$0.0184/GB
Azure Data Factory	Pay-as-you-go	\$1.50/DIU-hour
Synapse Analytics	Pay-as-you-go	\$5/100 DWU-hours
Power BI Pro	Subscription	\$9.99/user

Total: ~\$719/month

5. Streaming and NoSQL:

Component	Cost Type	Estimated Monthly Cost
Cosmos DB	Pay-as-you-go	\$0.008/hour per RU/s
Python Scripts	Development	\$0 (assuming existing resources)
Power BI Pro	Subscription	\$9.99/user

Total: ~\$600/month

6. Traditional Approach:

Component	Cost Type	Estimated Monthly Cost
HDFS Storage	Open-source	\$0 (hardware cost varies)
GCP Data Proc	Pay-as-you-go	\$0.01/CPU/hour
Apache Hive	Open-source	\$0 (hardware cost varies)
Airflow	Open-source	\$0 (hardware cost varies)
Power BI Pro	Subscription	\$9.99/user

Total: ~\$200/month (excluding hardware)

Conclusion:

Selecting the right approach depends on your specific requirements, existing infrastructure, and budget. The traditional approach may be cost-effective for organizations with existing big data infrastructure, while cloud-based solutions like AWS and Azure offer scalability and flexibility with varying costs based on usage.

CHAPTER 12: - LINK FOR PROJECT

Github - [aeronaut2001/IPL-Data-Analysis \(github.com\)](https://github.com/aeronaut2001/IPL-Data-Analysis)

Snowflake Role -

Link: https://lpihhit-read_only_user.snowflakecomputing.com

Admin name = Data_Engineer_12345

admin password = De@12345

Video Version of Real Time streaming Project -

https://drive.google.com/file/d/1iRiVEPcFIZ5Wy1Ks8Ev2SMBKSoOyRkHm/view?usp=drive_link

Video Version of Dashboard –

https://drive.google.com/file/d/1Nb-87QHMSCGjvdjIBTg1do-bXi7Z1Sb/view?usp=drive_link

CHAPTER 13: - CONCLUSION AND LEARNINGS

Conclusion and Learning:

Each of the six solutions outlined provides a distinct approach to extracting, transforming, loading, and visualizing IPL cricket match data, leveraging different tools and technologies suited for various environments and requirements. The key takeaway is that the choice of solution depends on the specific needs of the organization, including existing infrastructure, data volume, real-time processing requirements, and budget considerations.

Solution Summaries:

1. SSIS to SQL Server and Power BI:

- **Strengths:** Utilizes familiar SQL Server environment, easy integration with Power BI, cost-effective for organizations already using SQL Server.
- **Use Case:** Suitable for organizations with existing SQL Server infrastructure looking for a straightforward ETL and BI solution.

2. SSIS to Snowflake and Power BI:

- **Strengths:** Leverages Snowflake's scalable and performant cloud data warehouse, easy integration with Power BI.
- **Use Case:** Ideal for organizations seeking to utilize cloud data warehousing with robust ETL capabilities.

3. AWS Cloud Architecture:

- **Strengths:** Fully managed services with AWS Glue and Redshift, scalable and efficient data processing, real-time capabilities with S3 and Glue.
- **Use Case:** Suitable for organizations already invested in AWS, needing scalable and managed cloud services for ETL and analytics.

4. Azure Cloud Architecture:

- **Strengths:** Seamless integration with Azure Data Factory, Synapse Analytics, and Power BI, scalable and high-performance storage and analytics.
- **Use Case:** Best for organizations using Azure services, requiring a comprehensive and integrated cloud solution for data processing and analytics.

5. Streaming and NoSQL:

- **Strengths:** Real-time data ingestion and analytics with Cosmos DB, scalable and globally distributed storage.
- **Use Case:** Ideal for use cases requiring real-time data processing and analysis, particularly for applications needing globally distributed data access.

6. Traditional Approach:

- **Strengths:** Utilizes open-source technologies like PySpark and Hive, suitable for large-scale data processing with HDFS.
- **Use Case:** Suitable for organizations with big data infrastructure using Hadoop ecosystems, focusing on large data volumes and batch processing.

Key Considerations:

- **Data Validation:**
 - Ensure data accuracy, completeness, and consistency.
 - Implement validation rules to detect and handle errors, duplicates, and outliers.
- **Incremental Loading:**
 - Update only changed or new data to reduce processing time and resources.
- **Error Handling and Logging:**
 - Track and manage errors during pipeline execution with logging mechanisms.
- **Scheduling and Automation:**
 - Automate data processing tasks with scheduled intervals or event triggers for timely data updates.
- **Integration with Visualization Tools:**
 - Ensure seamless integration with Power BI for effective data visualization and analysis.

Conclusion:

Selecting the appropriate solution involves evaluating the existing infrastructure, understanding the specific needs of the organization, and considering factors like scalability, performance, and cost. Each approach has its strengths and is best suited for different scenarios. Proper implementation of data validation, incremental loading, error handling, scheduling, and integration with visualization tools is crucial for ensuring a robust and efficient data pipeline.

By leveraging the strengths of each solution and adhering to best practices in data management, organizations can derive valuable insights from IPL cricket match data, driving strategic decision-making and gaining a competitive edge in their analysis efforts.