IPL CRICKET LEAGUE MATCH DATA INTEGRATION USING SSIS

PROJECT TITLE : SSIS FINAL PROJECT – IPL CRICKET MATCH DATA INTEGRATION USING SSIS

NAME : Krimisha Vaghela

Tool Used : Microsoft SQL Server Integration (SSIS)

Table of Contents

1. Introduction

1.1 Overview of the Project

This project titled "IPL Cricket League – Data Engineering Case Study" focuses on building a complete data pipeline using SQL Server Integration Services (SSIS) to process and manage structured cricket data from multiple CSV files. The data consists of match-level, player-level, and ball-by-ball statistics from the Indian Premier League (IPL), which is cleaned, validated, transformed, and stored in Microsoft SQL Server for analysis and reporting.

The primary objective is to build an efficient, reliable, and scalable ETL (Extract, Transform, Load) solution that mimics real-world data engineering scenarios like incremental loading, error handling, data validation, and staging.

1.2 Tools Used

- **SQL Server Integration Services (SSIS)** – for ETL operations
- **Microsoft SQL Server** – for final data storage and querying
- **Microsoft Excel/CSV** – as source files
- **Visual Studio 2022** – for developing and managing SSIS packages
- **SSMS (SQL Server Management Studio)** – for querying and managing SQL Server

1.3 Project Objectives

- Extract raw data from multiple CSV files
- Cleanse and transform the data to maintain quality and consistency
- Handle errors such as nulls, duplicate IDs, and invalid foreign key relationships
- Use incremental loading to improve performance and avoid reprocessing
- Load validated data into corresponding fact and dimension tables in SQL Server
- Provide a well-organized and scalable SSIS package structure

1.4 Key Highlights

- 20+ CSV source files processed
- 2 main SSIS packages for Fact and Dimension data loading
- Custom error handling logic using Conditional Splits and Redirects
- Use of Execute SQL Tasks for DDL and pre/post-processing operations
- Data Flow Tasks with Lookup transformations for foreign key checks
- All packages successfully deployed and tested with clean final data in SQL Server

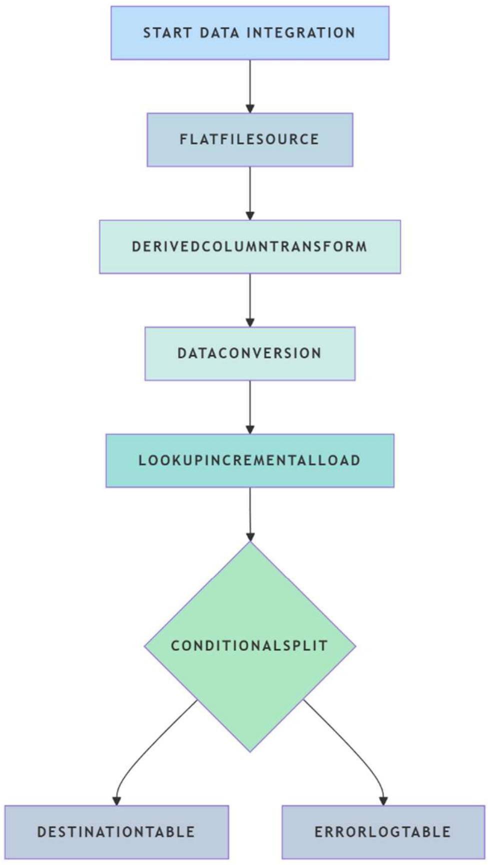
2. Project Architecture

2.1 ETL Design Flow

The ETL pipeline consists of the following stages:

1. **Extraction**:
   - Source CSV files are read using Flat File Connections.
   - Files are loaded into SSIS data flow pipelines.
2. **Transformation**:
   - Data cleaning (e.g., removing nulls, trimming strings)
   - Key validation (e.g., foreign key match checks)
   - Deduplication and business logic (e.g., resolving name-ID mismatches)
   - Conversion of data types (e.g., date formats)
3. **Loading**:
   - Transformed data is written into staging tables (optional)
   - Final validated data is loaded into fact and dimension tables in SQL Server.
4. **Error Handling**:
   - Rows failing constraints or validation rules are redirected to error logs or error tables.
   - Logs are maintained for each type of error scenario.
5. **Incremental Load**:
   - Lookup transformations identify new vs existing records.
   - Only new or updated data is inserted/updated into destination tables.

2.2 High-Level Architecture Diagram



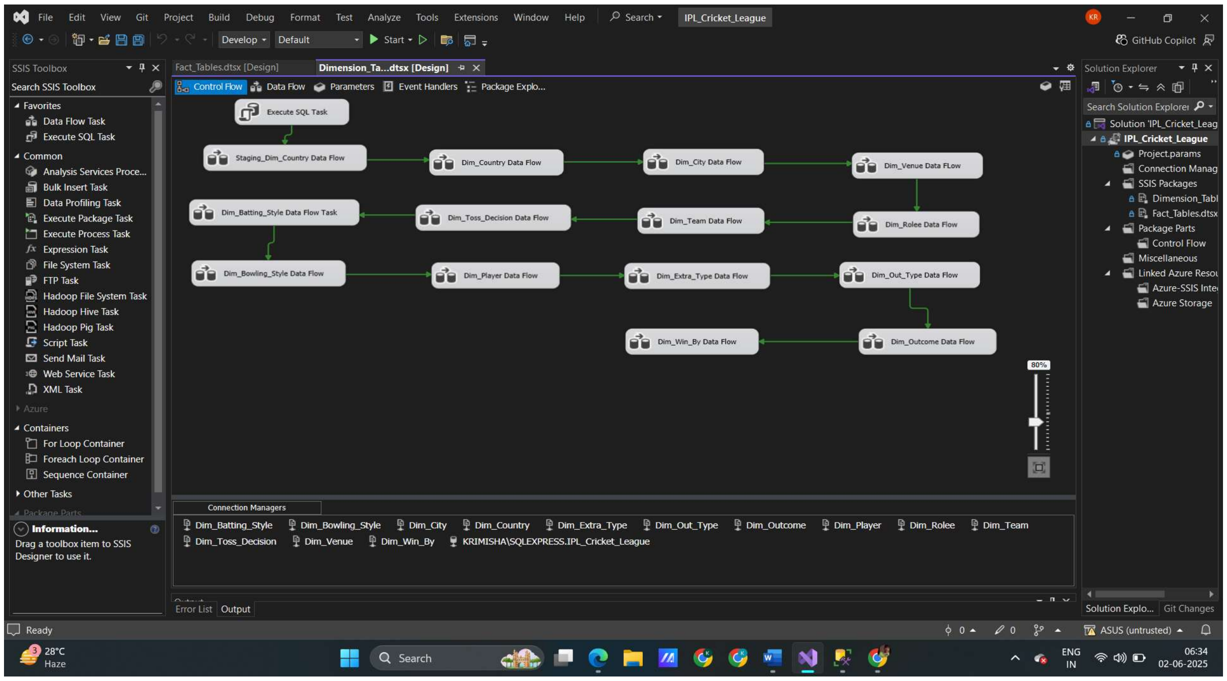
2.2.1 Flowchart

2.3 Source Layer Description

Table Name	**Columns Name**
Country	Country Id, Country Name
City	City Id, City Name , Country Id
Venue	Venue Id, Venue Name, City Id
Rolee	Role Id, Role Desc
Team	Team Id, Team Name
Toss Decision	Toss Id, Toss Name
Batting Style	Batting Id, Batting Hand
Bowling Style	Bowling Id, Bowling Skill
Player	Player Id, Player Name, DOB, Batting Id, Bowling Id, Country Id
Extra Type	Extra Id, Extra Name
Outcome	Out Id, Out Name
Win by	Win Id, Win Type
Season	Season Id, Man of the Series, Orange Cap, Purple Cap, Season Year
Match	Match Id, Team 1, Team 2, Match Date, Season Id, Venue Id, Team Toss Winner, Toss Id, Win Id, Win Margin, Outcome Id, Match Winner Team Name, Man of the Match
Ball by Ball	Match Id, Over Id, Ball Id, Innings No, Team Battings, Team Bowlings, Striker Batting Position, Striker, Non-Striker, Bowler
Batsman Scored	Match Id, Over Id, Ball Id, Innings No, Runs Scored
Extra Runs	Match Id, Over Id, Ball Id, Innings No, Extra Runs Id, Extra Type Id
Wicket Taken	Match Id, Over Id, Ball Id, Innings No, Player Out, Kind Out, Fielders
Player Match	Match Id, Player Id, Role Id, Team Id

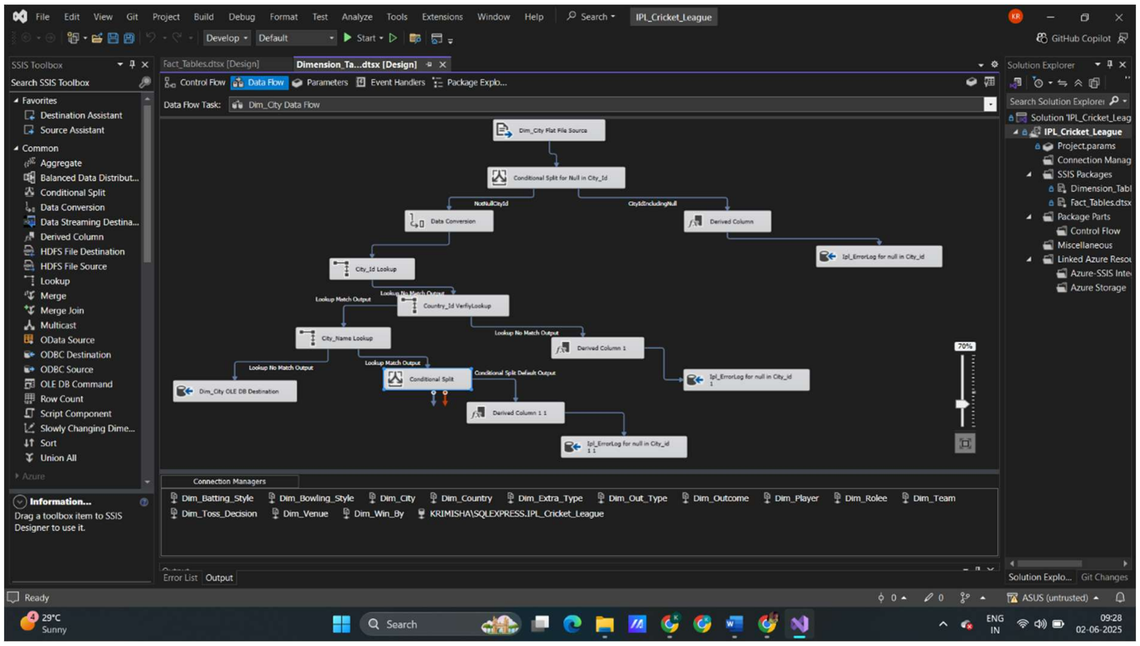
2.4 SSIS Layer Structure

- **Control Flow:** Sequence of tasks

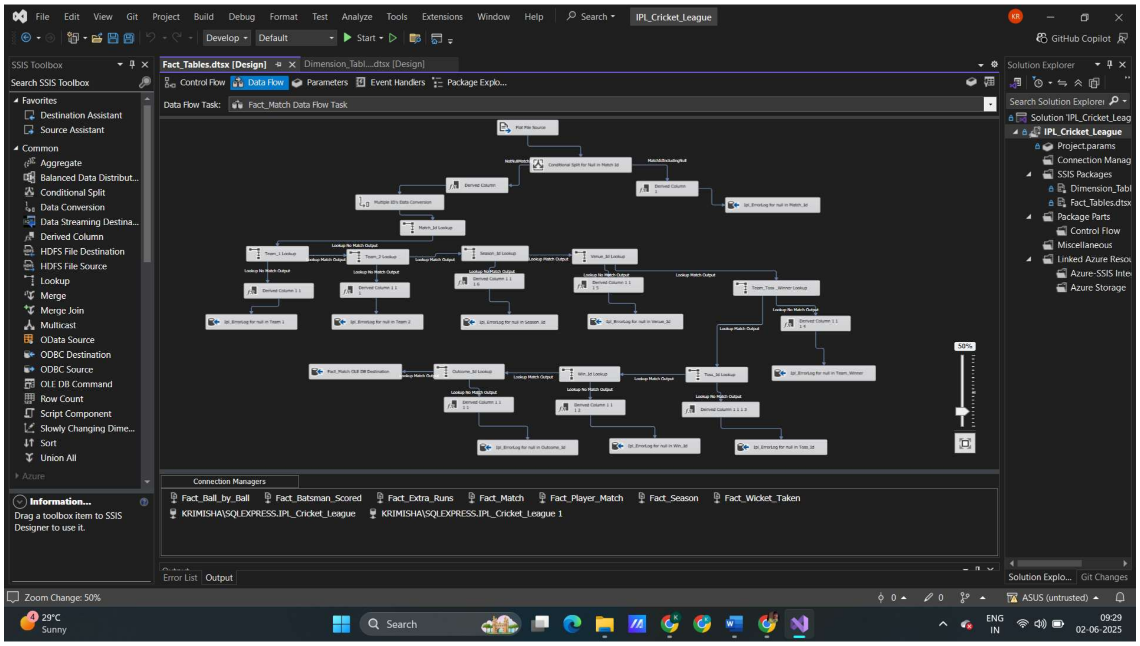


2.4.1 Control Flow Components for Dimension Tables

- **Data Flow:**
  - Flat File Source
  - Dervied Column
  - Data Conversion
  - Lookup
  - Conditional Split
  - Merge
  - Sort
  - Union All
  - OLE DB Destination



2.4.2 Data Flow Components for Dim_City Table



2.4.3 Data Flow Components for Fact_Match Table

2.5 Destination Layer Schema

Final dimension and fact tables in SQL Server

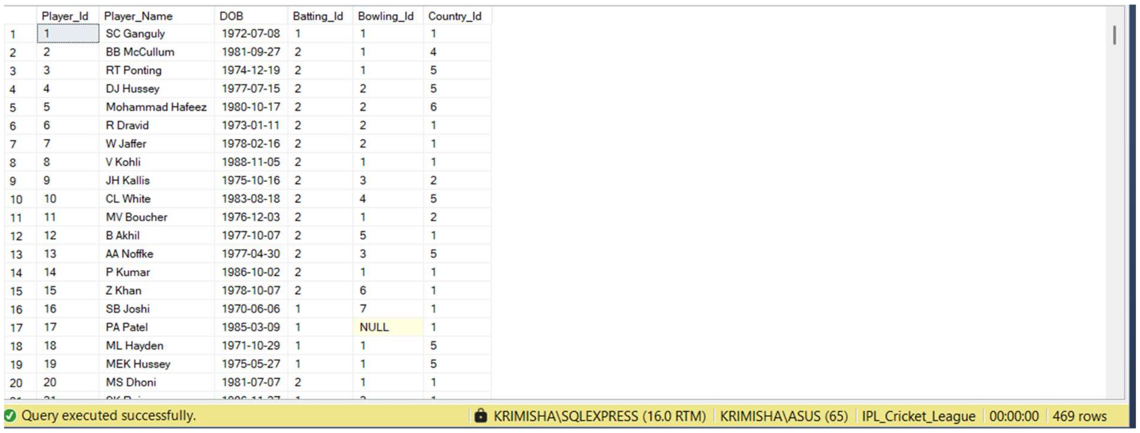
Table Name	Column Name	Data Type	Constraints
Staging_Dim_Country	Country_Id	int	—
	Country_Name	varchar(255)	—
Dim_Country	Country_Id	int	Primary Key
	Country_Name	varchar(255)	NOT NULL
Dim_Cty	City_Id	int	Primary Key
	City_Name	varchar(255)	NOT NULL
	Country_Id	int	Foreign Key → Dim_Country(Country_Id), ON DELETE/UPDATE CASCADE
Dim_Venue	Venue_Id	int	Primary Key
	Venue_Name	varchar(255)	NOT NULL
	City_Id	int	Foreign Key → Dim_City(City_Id), ON DELETE/UPDATE CASCADE
Dim_Rolee	Role_Id	int	Primary Key
	Role_Desc	varchar(255)	NOT NULL
Dim_Team	Team_Id	int	Primary Key

	Team_Name	varchar(255)	NOT NULL
Dim_Toss_Decision	Toss_Id	int	Primary Key, CHECK (Toss_Id IN (1, 2))
	Toss_Name	varchar(255)	NOT NULL
Dim_Batting_Style	Batting_Id	int	Primary Key
	Batting_Hand	varchar(255)	NOT NULL
Dim_Bowling_Style	Bowling_Id	int	Primary Key
	Bowling_Skill	varchar(255)	NOT NULL
Dim_Player	Player_Id	int	Primary Key
	Player_Name	varchar(255)	NOT NULL
	DOB	date	—
	Batting_Id	int	Foreign Key → Dim_Batting_Style(Batting_Id), Nullable
	Bowling_Id	int	Foreign Key → Dim_Bowling_Style(Bowling_Id), Nullable
	Country_Id	int	Foreign Key → Dim_Country(Country_Id), Nullable
Dim_Extra_Type	Extra_Id	int	Primary Key
	Extra_Name	varchar(255)	NOT NULL
Dim_Out_Type	Out_Id	int	Primary Key
	Out_Name	varchar(255)	NOT NULL
Dim_Outcome	Outcome_Id	int	Primary Key
	Outcome_Type	varchar(255)	NOT NULL
Dim_Win_By	Win_Id	int	Primary Key
	Win_Type	varchar(255)	NOT NULL
Fact_Season	Season_Id	int	Primary Key
	Man_of_the_Series	int	Foreign Key → Dim_Player(Player_Id)
	Orange_Cap	int	Foreign Key → Dim_Player(Player_Id)
	Purple_Cap	int	Foreign Key → Dim_Player(Player_Id)
	Season_Year	int	CHECK (Season_Year BETWEEN 1900 AND 2100)
Fact_Match	Match_Id	int	Primary Key
	Team_1	int	Foreign Key → Dim_Team(Team_Id), NOT NULL

	Team_2	int	Foreign Key → Dim_Team(Team_Id), NOT NULL
	Match_Date	date	NOT NULL
	Season_Id	int	Foreign Key → Fact_Season(Season_Id)
	Venue_Id	int	Foreign Key → Dim_Venue(Venue_Id)
	Team_Toss_Winner	int	Foreign Key → Dim_Team(Team_Id)
	Toss_Id	int	Foreign Key → Dim_Toss_Decision(Toss_Id)
	Win_Id	int	Foreign Key → Dim_Win_By(Win_Id)
	Win_Margin	int	Nullable
	Outcome_Id	int	Foreign Key → Dim_Outcome(Outcome_Id)
	Match_Winner_Team_Name	int	Foreign Key → Dim_Team(Team_Id), Nullable
	Man_of_the_match	int	Foreign Key → Dim_Player(Player_Id)
Fact_Ball_by_Ball	Match_Id	int	Foreign Key → Fact_Match(Match_Id)
	Over_Id	int	Composite PK
	Ball_Id	int	Composite PK
	Innings_No	int	Composite PK, CHECK (1–4)
	Team_Battings	int	Foreign Key → Dim_Team
	Team_Bowlings	int	Foreign Key → Dim_Team
	Striker_Batting_Position	int	—
	Striker	int	Foreign Key → Dim_Player
	Non_Striker	int	Foreign Key → Dim_Player
	Bowler	int	Foreign Key → Dim_Player
Fact_Batsman_Scored	Match_Id, Over_Id, Ball_Id, Innings_No	Int	Composite PK, FK → Fact_Ball_by_Ball
	Runs_Scored	Int	—
Fact_Extra_Runs	Match_Id, Over_Id, Ball_Id, Innings_No	Int	Composite PK, FK → Fact_Ball_by_Ball
	Extra_Runs_Id	Int	—
	Extra_Type_Id	int	—

Fact_Wicket_Taken	Match_Id, Over_Id, Ball_Id, Innings_No	int	Composite PK, FK → Fact_Ball_by_Ball
	Player_Out	int	Foreign Key → Dim_Player
	Kind_out	int	—
	Fielders	int	Nullable, FK → Dim_Player
Fact_Player_Match	Match_Id	int	Foreign Key → Fact_Match
	Player_Id	int	Foreign Key → Dim_Player
	Role_Id	int	Foreign Key → Dim_Rolee
	Team_Id	int	Foreign Key → Dim_Team

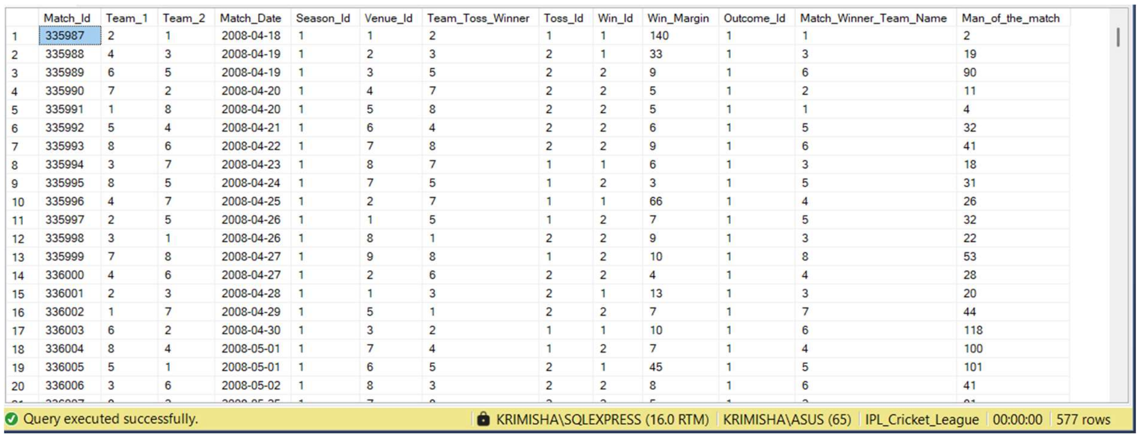
2.6 Destination Layer Output



	Player_Id	Player_Name	DOB	Batting_Id	Bowling_Id	Country_Id
1	1	SC Ganguly	1972-07-08	1	1	1
2	2	BB McCullum	1981-09-27	2	1	4
3	3	RT Ponting	1974-12-19	2	1	5
4	4	DJ Hussey	1977-07-15	2	2	5
5	5	Mohammad Hafeez	1980-10-17	2	2	6
6	6	R Dravid	1973-01-11	2	2	1
7	7	W Jaffer	1978-02-16	2	2	1
8	8	V Kohli	1988-11-05	2	1	1
9	9	JH Kallis	1975-10-16	2	3	2
10	10	CL White	1983-08-18	2	4	5
11	11	MV Boucher	1976-12-03	2	1	2
12	12	B Akhil	1977-10-07	2	5	1
13	13	AA Noffke	1977-04-30	2	3	5
14	14	P Kumar	1986-10-02	2	1	1
15	15	Z Khan	1978-10-07	2	6	1
16	16	SB Joshi	1970-06-06	1	7	1
17	17	PA Patel	1985-03-09	1	NULL	1
18	18	ML Hayden	1971-10-29	1	1	5
19	19	MEK Hussey	1975-05-27	1	1	5
20	20	MS Dhoni	1981-07-07	2	1	1

```
Query executed successfully.    KRIMISHA\SQLEXPRESS (16.0 RTM) | KRIMISHA\ASUS (65) | IPL_Cricket_League | 00:00:00 | 469 rows
```

2.6.1 Output of Dim_Player



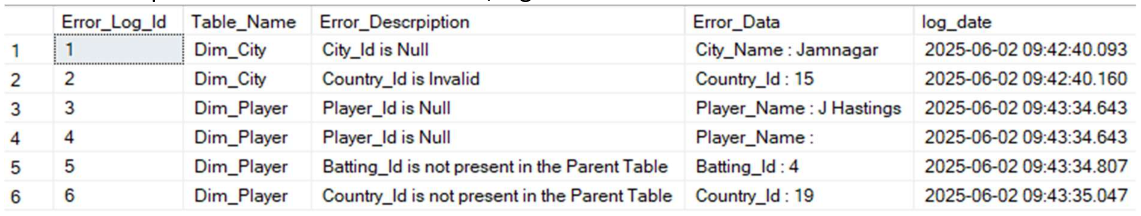
	Match_Id	Team_1	Team_2	Match_Date	Season_Id	Venue_Id	Team_Toss_Winner	Toss_Id	Win_Id	Win_Margin	Outcome_Id	Match_Winner_Team_Name	Man_of_the_match
1	335987	2	1	2008-04-18	1	1	2	1	1	140	1	1	2
2	335988	4	3	2008-04-19	1	2	3	2	1	33	1	3	19
3	335989	6	5	2008-04-19	1	3	5	2	2	9	1	6	90
4	335990	7	2	2008-04-20	1	4	7	2	2	5	1	2	11
5	335991	1	8	2008-04-20	1	5	8	2	2	5	1	1	4
6	335992	5	4	2008-04-21	1	6	4	2	2	6	1	5	32
7	335993	8	6	2008-04-22	1	7	8	2	2	9	1	6	41
8	335994	3	7	2008-04-23	1	8	7	1	1	6	1	3	18
9	335995	8	5	2008-04-24	1	7	5	1	2	3	1	5	31
10	335996	4	7	2008-04-25	1	2	7	1	1	66	1	4	26
11	335997	2	5	2008-04-26	1	1	5	1	2	7	1	5	32
12	335998	3	1	2008-04-26	1	8	1	2	2	9	1	3	22
13	335999	7	8	2008-04-27	1	9	8	1	2	10	1	8	53
14	336000	4	6	2008-04-27	1	2	6	2	2	4	1	4	28
15	336001	2	3	2008-04-28	1	1	3	2	1	13	1	3	20
16	336002	1	7	2008-04-29	1	5	1	2	2	7	1	7	44
17	336003	6	2	2008-04-30	1	3	2	1	1	10	1	6	118
18	336004	8	4	2008-05-01	1	7	4	1	2	7	1	4	100
19	336005	5	1	2008-05-01	1	6	5	2	1	45	1	5	101
20	336006	3	6	2008-05-02	1	8	3	2	2	8	1	6	41

```
Query executed successfully.    KRIMISHA\SQLEXPRESS (16.0 RTM) | KRIMISHA\ASUS (65) | IPL_Cricket_League | 00:00:00 | 577 rows
```

2.6.2 Output of Fact_Player

2.7 Error Handling Layer

Error outputs redirected to error tables/logs



	Error_Log_Id	Table_Name	Error_Descrpiption	Error_Data	log_date
1	1	Dim_City	City_Id is Null	City_Name : Jamnagar	2025-06-02 09:42:40.093
2	2	Dim_City	Country_Id is Invalid	Country_Id : 15	2025-06-02 09:42:40.160
3	3	Dim_Player	Player_Id is Null	Player_Name : J Hastings	2025-06-02 09:43:34.643
4	4	Dim_Player	Player_Id is Null	Player_Name :	2025-06-02 09:43:34.643
5	5	Dim_Player	Batting_Id is not present in the Parent Table	Batting_Id : 4	2025-06-02 09:43:34.807
6	6	Dim_Player	Country_Id is not present in the Parent Table	Country_Id : 19	2025-06-02 09:43:35.047

2.7.1 Output of IPL_ErrorLog Table

3. Data Model Design

3.1 Purpose of the Data Model

The primary goal of the data model is to organize and store historical IPL cricket match data in a structured and optimized format that supports analytics and reporting. The model follows a star schema design, ensuring ease of querying and high performance for analytical workloads.

3.1 1 Schema Type

- **Star Schema**
  The central Fact tables (e.g., Fact_Match, Fact_Ball_by_Ball, Fact_Player_Match) are connected to surrounding Dimension tables (e.g., Dim_Player, Dim_Team, Dim_Venue) via foreign key relationships.

3.2 Key Components

3.2.1 Dimension Tables

These provide descriptive context to the facts and enable slicing and dicing of data.

- Dim_Country: Contains country names
- Dim_City: Cities linked to countries
- Dim_Venue: Match venues linked to cities
- Dim_Team: List of participating teams
- Dim_Player: All registered players with additional details like batting and bowling style
- Dim_Rolee: Role of players (e.g., batsman, bowler, all-rounder)
- Dim_Toss_Decision: Toss choices (bat/field)
- Dim_Batting_Style: Batting preferences (left/right-hand)
- Dim_Bowling_Style: Bowling styles (off-spin, fast, etc.)
- Dim_Extra_Type: Types of extras (no-ball, wide, etc.)
- Dim_Out_Type: Types of dismissals
- Dim_Outcome: Match outcomes (win/loss/draw)
- Dim_Win_By: Win by runs or wickets

3.2.2 Fact Tables

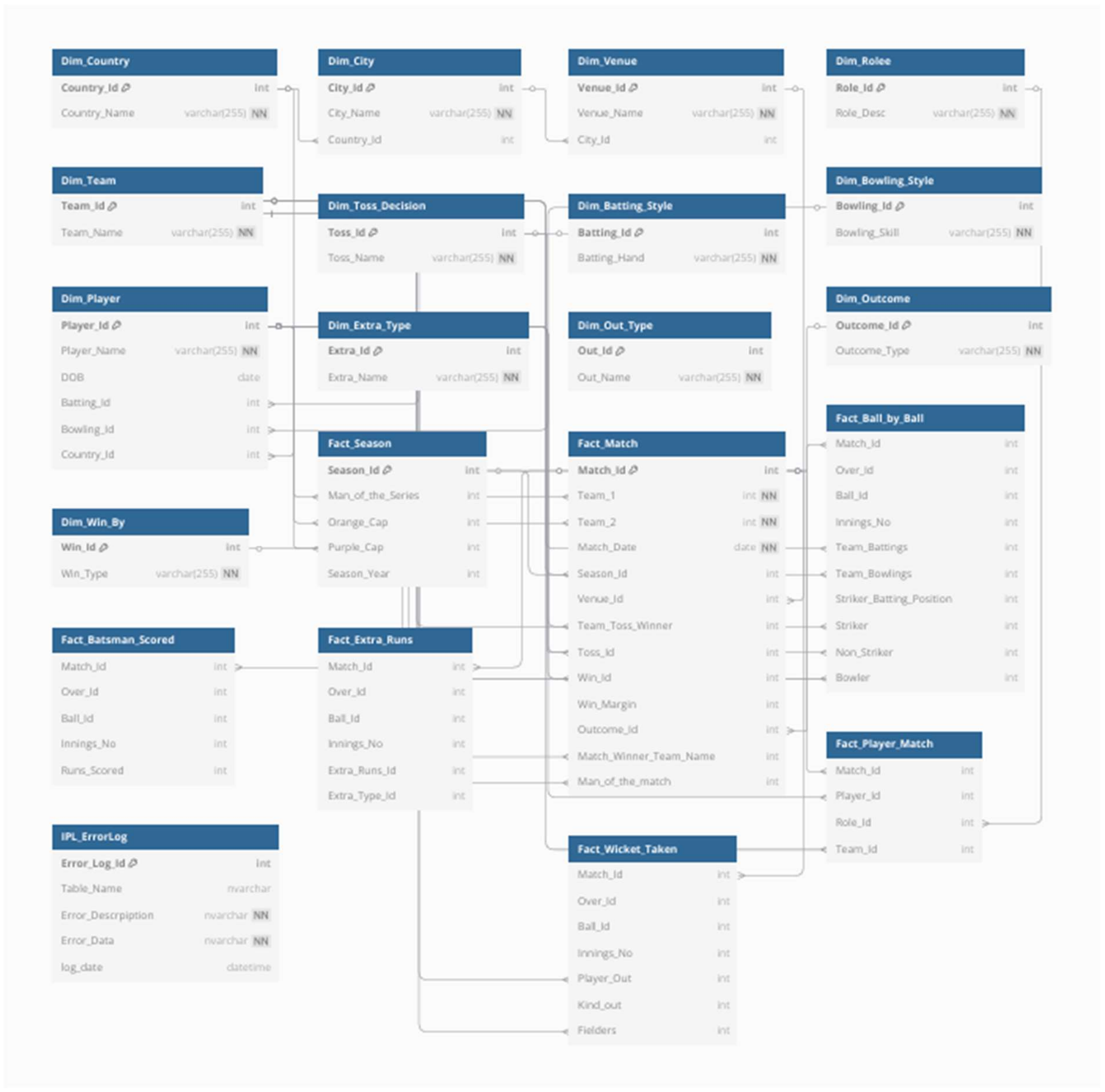
These store the measurable data points, typically transactional or event-based.

- Fact_Season: Season-wise stats including award winners
- Fact_Match: Match-level data like teams, venue, outcome, toss

- Fact_Ball_by_Ball: Ball-level transactional data for every delivery
- Fact_Batsman_Scored: Batsman performance per delivery
- Fact_Extra_Runs: Tracks extra runs per delivery
- Fact_Wicket_Taken: Tracks dismissals per delivery
- Fact_Player_Match: Player participation and roles per match

3.3 Relationships

- **One-to-Many (1:N)**:
  For example, one country has many cities (Dim_Country → Dim_City), one player can appear in many matches (Dim_Player → Fact_Player_Match).
- **Many-to-One (N:1)**:
  Many records in Fact_Ball_by_Ball reference one match in Fact_Match.
- **Composite Primary Keys**:
  Used in transactional fact tables like Fact_Ball_by_Ball to uniquely identify a delivery using (Match_Id, Over_Id, Ball_Id, Innings_No).



3.3.1 Data Modeling

3.4 Normalization

- The Dimension tables are normalized to avoid redundancy (e.g., styles, teams, roles).
- The Fact tables are denormalized for fast aggregations and joins.

3.5 Integrity Constraints

- Primary keys are applied to uniquely identify each row.
- Foreign key constraints ensure data consistency across fact and dimension tables.
- NOT NULL constraints ensure mandatory values.
- CHECK constraints are used for domain validation (e.g., innings number between 1 and 4, valid toss decisions, year range).

3.6 Advantages of This Design

- Fast performance for read-heavy analytics workloads
- Supports slicing by players, teams, seasons, match outcomes, etc.
- Easily extendable to include more facts (like ticket sales, audience data)
- Ensures referential integrity with clean error handling and data quality

4. SSIS Data Flow & Transformations

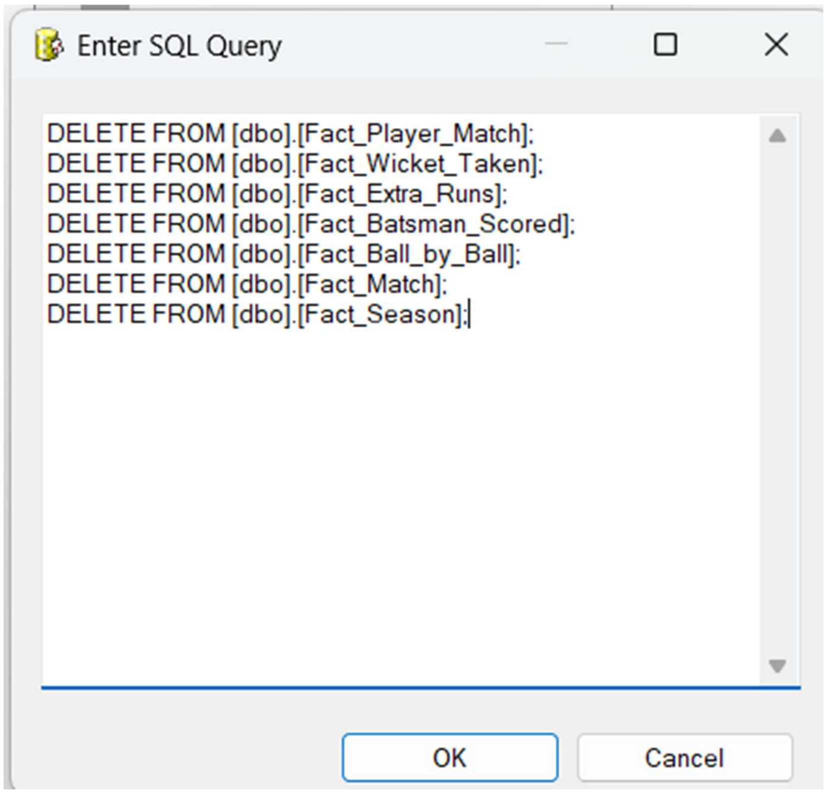
In this project, **SQL Server Integration Services (SSIS)** was used to design a scalable and modular ETL solution for processing IPL match data. The SSIS packages consist of structured **Control Flow** and **Data Flow** elements, with robust transformation logic and error handling mechanisms.

4.1 Control Flow Tasks

The Control Flow defines the sequence and orchestration of tasks within the SSIS package. It handles the high-level logic such as preparing the database, executing individual Data Flow tasks, and managing dependencies.

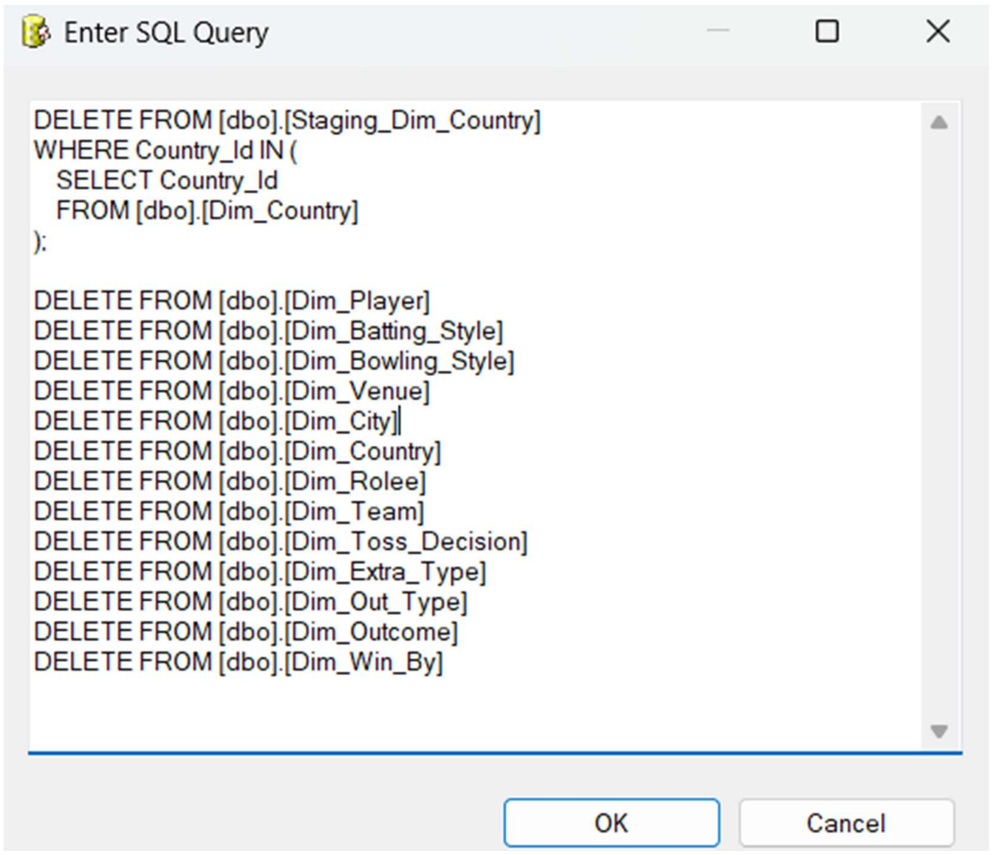
Tasks used:

- **Execute SQL Task**: Used for pre-loading scripts such as truncating staging tables or checking conditions.



```
DELETE FROM [dbo].[Fact_Player_Match];
DELETE FROM [dbo].[Fact_Wicket_Taken];
DELETE FROM [dbo].[Fact_Extra_Runs];
DELETE FROM [dbo].[Fact_Batsman_Scored];
DELETE FROM [dbo].[Fact_Ball_by_Ball];
DELETE FROM [dbo].[Fact_Match];
DELETE FROM [dbo].[Fact_Season];
```

4.1.1 Execute SQL Task for Fact Tables



```
DELETE FROM [dbo].[Staging_Dim_Country]
WHERE Country_Id IN (
  SELECT Country_Id
  FROM [dbo].[Dim_Country]
);

DELETE FROM [dbo].[Dim_Player]
DELETE FROM [dbo].[Dim_Batting_Style]
DELETE FROM [dbo].[Dim_Bowling_Style]
DELETE FROM [dbo].[Dim_Venue]
DELETE FROM [dbo].[Dim_City]
DELETE FROM [dbo].[Dim_Country]
DELETE FROM [dbo].[Dim_Rolee]
DELETE FROM [dbo].[Dim_Team]
DELETE FROM [dbo].[Dim_Toss_Decision]
DELETE FROM [dbo].[Dim_Extra_Type]
DELETE FROM [dbo].[Dim_Out_Type]
DELETE FROM [dbo].[Dim_Outcome]
DELETE FROM [dbo].[Dim_Win_By]
```

4.1.1 Execute SQL Task for Dimension Tables

- **Data Flow Task**: Triggers the actual ETL logic where data is extracted, transformed, and loaded.
- **Sequence Container**: Organizes tasks for better readability and modularity.
- **Precedence Constraints**: Manage task dependencies and execution order (e.g., only proceed if the previous task succeeds).

4.2 Data Flow Tasks

The Data Flow is where the actual ETL transformation occurs. Each Data Flow task includes components for:

- Reading from source files (Flat File Source)
- Applying transformations
- Writing data to SQL Server tables

Flat File Source → Derived Column / Data Conversion → Lookup → Conditional Split → OLE DB Destination

Each fact or dimension table had its own Data Flow pipeline for isolation and easier debugging.

4.3 Lookup, Derived Column, Data Conversion

Lookup Transformation

- Used to validate foreign key relationships.
- Matches values (e.g., City_Name, Bowling_Style) with the corresponding dimension table.
- Helps in identifying whether a record already exists (useful for incremental loads).

Derived Column Transformation

- Used to create or replace column values.
- Example: Mapping city names to their respective City_Id using conditional expressions.

```
TRIM(City_Id) == "Mohali" ? 2 :
TRIM(City_Id) == "Uppal" ? 7 :
TRIM(City_Id) == "Chepauk" ? 8 :
TRIM(City_Id) == "Motera" ? 17 :
TRIM(City_Id) == "Jamtha" ? 19 : 0
```

Data Conversion

- Ensures data type compatibility between source and destination.
- Common conversions: String to Int, DateTime formatting, etc.

4.4 Conditional Split

Conditional Split plays a crucial role in:

- Separating **valid vs. invalid** records
- Managing different business logic paths (e.g., null handling, ID mismatches)

Examples Used in Project:

- **Null Check for Primary Key (City_Id)**

  ```
  ISNULL(City_Id) || TRIM(City_Id) == ""
  ```

- **Mismatch Check for Existing Names with New IDs**

  ```
  DataType_City_Id_Changed != Existing_City_Id
  ```

Outputs:

- Valid rows → Sent to OLE DB Destination
- Invalid rows → Sent to Ipl_ErrorLog via another data path

4.5 Error Redirection and Logging

SSIS allows for redirection of bad rows through Error Outputs in components like Lookup, OLE DB Destination, and Conditional Split.

Implementation:

- **Redirect row** option selected in components
- Error data is sent to a custom table: Ipl_ErrorLog
- This includes metadata like:
  - Table_Name
  - Error_Description
  - Error_Data
  - Log_Date

Benefits:

- Keeps the main ETL pipeline clean
- Enables debugging without discarding bad data
- Supports future correction and reprocessing

5. Error Logging Strategy

5.1 Error Log Table: Ipl_ErrorLog

To ensure data integrity and enable traceability of rejected records, an Error Log Table named Ipl_ErrorLog was implemented. This table captures records that fail validation or referential integrity checks during the ETL process.

5.2 Purpose:

- Store rows with null or invalid foreign keys (e.g., City_Id)
- Capture records where business rules are violated (e.g., mismatched City_Name with existing City_Id)
- Help in debugging and data correction for future loads

5.3 Implementation:

Error rows are identified using Conditional Split, and then directed to the Ipl_ErrorLog table using an OLE DB Destination.

Scenarios Captured:

1. **City_Id is null or empty**
   - Split condition: ISNULL(City_Id) || TRIM(City_Id) == ""
   - Action: Redirect to Ipl_ErrorLog
2. **City_Name exists but City_Id mismatches**
   - Split condition: DataType_City_Id_Changed != Existing_City_Id
   - Action: Redirect to Ipl_ErrorLog
3. **Bowling_Id is null when it should not be** (for Player records)
   - Conditional split on: ISNULL(Bowling_Id)
   - Action: Log to Ipl_ErrorLog and/or direct to alternate processing flow

5.4 Incremental Load Strategy

To avoid reprocessing all records during each ETL run, an incremental load strategy is implemented. This ensures only new or updated data is processed and inserted into the target tables.

5.5 Lookup & Conditional Split for Change Detection

- Existing records are checked using Lookup Transformation.
- If a new record or updated value is found, it is routed using Conditional Split.

5.6 Benefits of This Approach

- Optimized ETL performance
- Accurate tracking of data changes
- Easier debugging and data correction through error logs
- Clean separation of valid and invalid data flows

6. Challenges Faced and Solutions

1. Handling Inconsistent City_Id Values in the Venue Table

Challenge:
In the Venue table, the City_Id column is expected to contain integer values that reference the City dimension table. However, certain rows contained text values like "Mohali", "Uppal", etc., instead of the correct integer IDs. This caused referential integrity issues during data load.

Solution:
To resolve this issue, a Derived Column transformation was implemented to convert known city names into their corresponding integer City_Id values. The following conditional logic was used:

```
TRIM(City_Id) == "Mohali" ? 2 :

TRIM(City_Id) == "Uppal" ? 7 :

TRIM(City_Id) == "Chepauk" ? 8 :

TRIM(City_Id) == "Motera" ? 17 :

TRIM(City_Id) == "Jamtha" ? 19 : 0
```

Any unmatched value defaults to 0, which can later be filtered or logged for quality control.

Additionally, a string parsing expression was tested:

```
RIGHT(City_Id, LEN(City_Id) - FINDSTRING(City_Id, ",", 1))
```

This was used to isolate part of the city name if combined data was present, although not implemented in the final flow due to inconsistency in source formatting.

Improvement Suggestion:
Using a Lookup Transformation against the City table based on the city name would make this logic more dynamic and easier to maintain, avoiding hardcoded mappings.

2. Handling Null Bowling_Skill in the Player Table

Challenge:
The Bowling_Skill column in the Player table contains null values. Since this column references the Bowling dimension table, nulls would break the foreign key relationship during lookup.

Solution:
A Conditional Split transformation was applied to handle null and non-null values separately:

- **Condition 1:** ISNULL(DataType_Bowling_Skill_Including_Null_Changed)

- **Condition 2:** !ISNULL(DataType_Bowling_Skill_Including_Null_Changed)

The two streams were then recombined using a Union All transformation:

- Non-null values were sent through a Lookup Transformation to fetch valid Bowling_Ids.
- Null values were bypassed or assigned a default/unknown Bowling_Id.

This ensured that the ETL pipeline maintained data integrity while handling missing skill information gracefully.

7 Conclusion

The IPL Cricket League Data Warehouse project successfully demonstrates the design and implementation of a robust ETL process using SQL Server Integration Services (SSIS). Through carefully planned data flow pipelines, conditional logic, and error handling mechanisms, the project achieves:

7.2 Key Outcomes:

- Clean and validated data from flat file sources loaded into dimension and fact tables.
- Effective use of Conditional Split and Lookup transformations to handle data discrepancies and integrity checks.
- Error logging mechanism through Ipl_ErrorLog table to track invalid records and support data correction.
- An incremental load strategy ensures only new and changed records are processed, boosting performance and efficiency.
- Modular SSIS package design for scalability, maintainability, and debugging.

7.2 Challenges Overcome:

- Transformation of non-numeric City_Id values to correct FK references using Derived Column logic.
- Handling of null and mismatched values across foreign key relationships (City_Id, Bowling_Id) using Conditional Split, Union All, and Lookups.
- Implementation of error logging and data validation flow using multiple data paths in the control flow.