IPL Data Warehouse - End-to-End Data Flow Documentation

1. Overview

This document explains the complete end-to-end flow for processing and storing IPL cricket match data using a modern data warehouse architecture. The pipeline extracts raw match and delivery data, stages it, curates the data, and transforms it into a star schema with fact and dimension tables for analytics.

---

2. Data Sources

- **match_info**: Contains match-level details such as teams, date, toss information, venue, winner, and players.
- **deliverables**: Contains ball-by-ball delivery data, including batsman, bowler, runs, extras, and dismissals.

---

3. Data Pipeline Architecture

Layers

1. **Staging Layer**
2. Purpose: Load raw data without transformation.

3. Tables:

    - `staging_matches`: Raw match data.
    - `staging_deliveries`: Raw ball-by-ball data.

4. **Curated Layer**

5. Purpose: Clean and standardize data, remove duplicates.

6. Tables:

    - `curated_matches`: Cleaned match-level data.
    - `curated_deliveries`: Cleaned ball-by-ball data.

7. **Dimensional Layer (Star Schema)**

8. Dimensions:
    - `dim_date`: Stores unique dates with day, month, quarter, year.
    - `dim_team`: Stores unique team names.

- `dim_player`: Stores player details.
- `dim_venue`: Stores venue and city information.
- `dim_match`: Stores match metadata linked to date and venue.

9. Facts:
    - `fact_deliveries`: Ball-by-ball fact table.
    - `fact_matches`: Match-level fact table.

---

4. Visual Architecture Diagram

```
[Raw Data Sources]
      |
      v
[Staging Layer]
(staging_matches, staging_deliveries)
      |
      v
[Curated Layer]
(curated_matches, curated_deliveries)
      |
      v
[Dimensional Layer]
(dim_date, dim_team, dim_player, dim_venue, dim_match)
      |
      v
[Fact Tables]
(fact_deliveries, fact_matches)
      |
      v
[BI & Analytics Tools]
(Power BI, Tableau, Looker)
```

This diagram represents the flow of data from raw sources to analytical dashboards.

---

5. Data Flow Steps

4.1 Data Ingestion

- Raw CSV files for `match_info` and `deliverables` are loaded into **staging tables** using `LOAD DATA INFILE` or ETL tools.

4.2 Data Curation

- Remove duplicates.

• Ensure consistent naming of teams and players.
• Filter irrelevant matches if needed.

4.3 Dimension Creation

• **Date Dimension**: Extracts date, day, month, quarter, year, weekday.
• **Team Dimension**: Unifies teams from matches and deliveries.
• **Player Dimension**: Collects players from deliveries and player of the match.
• **Venue Dimension**: Stores venue and city information.
• **Match Dimension**: Combines match metadata with foreign keys for date and venue.

4.4 Fact Table Population

• **fact_deliveries**: Joins curated deliveries with teams, players, and matches.
• **fact_matches**: Stores one row per match with result details.

4.5 Data Consumption

• Business users can query the data using BI tools like Power BI, Looker, or Tableau.
• Common use cases:
• Player performance analytics.
• Team win/loss trends.
• Venue-specific performance.
• Toss decision impact.

---

6. Benefits of This Architecture

• **Separation of Concerns**: Raw, curated, and dimensional layers prevent data corruption.
• **Historical Analysis**: Date dimension enables time-based analysis.
• **Query Optimization**: Star schema supports fast analytical queries.
• **Scalability**: New seasons or teams can be added with minimal changes.

---

7. Next Steps

• Automate ETL using Apache Airflow, dbt, or Dataflow.
• Add incremental load logic to avoid full reloads.
• Create materialized views for most-used KPIs (e.g., runs per match, strike rates).
• Integrate with a dashboarding solution for real-time insights.

---

8. Example Queries

1. **Top 5 Run Scorers**:

```sql
SELECT p.player_name, SUM(fd.batsman_runs) AS total_runs
FROM fact_deliveries fd
JOIN dim_player p ON fd.batter_id = p.player_id
GROUP BY p.player_name
ORDER BY total_runs DESC
LIMIT 5;
```

2. **Team Win Count**:

```sql
SELECT winner, COUNT(*) AS wins
FROM dim_match
GROUP BY winner
ORDER BY wins DESC;
```

3. **Matches per Venue**:

```sql
SELECT v.venue_name, COUNT(*) AS total_matches
FROM dim_match m
JOIN dim_venue v ON m.venue_id = v.venue_id
GROUP BY v.venue_name
ORDER BY total_matches DESC;
```

---

9. Conclusion

This end-to-end pipeline enables reliable and scalable IPL data analytics. The layered architecture ensures clean data processing, and the star schema design supports quick, insightful reporting.