Final Project Documentation (NHL)

The Uncalled Four

Chan Pin Han Chang Wai Yuen, Kith Tay Hui Ping

Content Page

Overview

What is NHL and Seasons?

The NHL, also known as the National Hockey League, is a professional ice hockey league in North America that was established in 1917 and is one of the major professional sports in the United States and Canada (James, 2012). NHL season is split into 3 parts: the pre-season, the regular season, and the Stanley Cup.

The pre-season is generally held during the last 2 weeks of September, 32 teams will each play about six to eight weeks of exhibition games for their coach to evaluate their team, new and unfamiliar players to try out for roster and position spots, and for experienced players to practice for the competition.

In the regular season, teams are split between 2 conferences depending on where their home team is located (Western and Eastern) and further divided into 4 divisions (Pacific, Central, Atlantic and Metropolitan), and will need to play a total of 82 games from early October through early April using the point system to determine their standing, with the top 16 teams qualify for the Stanley Cup (8 spots for each conference, the top 3 teams from each division and 2 wild-card teams in each conference (teams with the most points in each conference after the top 6 teams)). (Cheap Seats Sport, 2021)

The Stanley Cup takes place after the regular season, with the top 16 teams from the regular season qualifies for the playoffs, an elimination tournament consisting of 3 rounds of matches to determine (also known as wild cards) with the highest points.

In the first round, teams are split into 2 separate brackets by conference and each bracket consists of the top 3 divisional qualifiers and 1 of the wild card teams will play a best-of-seven series of matches to determine the 2 winners of each conference. The winners move up to the second round, also known as the Conference Finals, where they will play each other until a winner from each conference move to the final round.(Cheap Seats Sport, 2021)

The final round, widely known as the Stanley Cup Finals, will determine which conference champions will be the Stanley and NHL winner.

Team Composition

A team must comprise of 20 players: 2 goaltenders, 6 defenders and 12 forwards.
In a game, 6 players will be in the rink; 1 goaltender, 2 defenders, 3 forwards.
Each player has their own role:

- Goaltender: last line of defense, to prevent the puck from entering into their net.
- Defender: safeguarding their goal by halting opposing forwards and breaking up their formation.

- Forwards: primarily responsible for the offense, divided into 3 positions
    - Center: win faceoffs, score goals, defend and disturb opposing players.
    - Wingers (left and right): mainly assisting by creating scoring opportunities and attacking the opponent's goal.

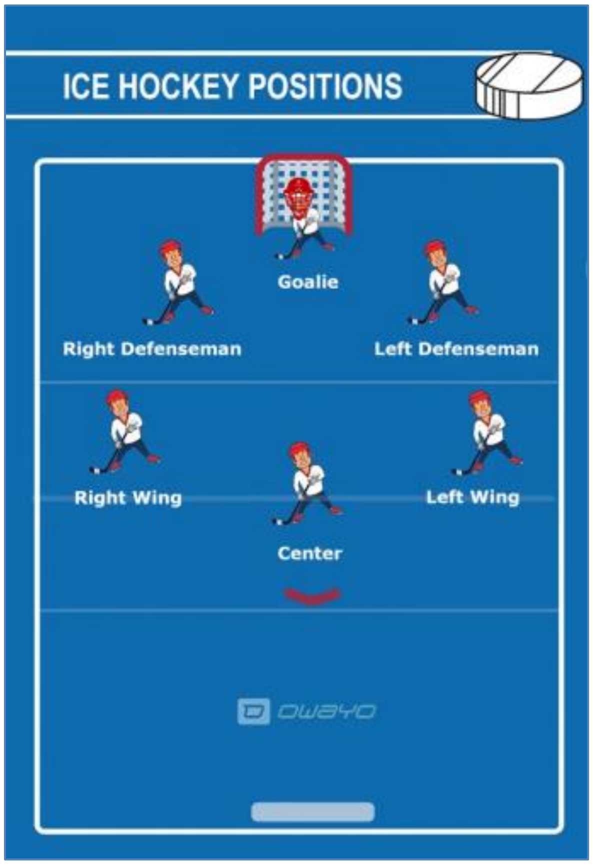


Figure 1: Ice hockey team positions

Ice hockey Rink

The ice hockey rink is an area where ice hockey games are played. There are many markings that are important for both the game and the rules that govern it.

<u>Rink</u>
The rink is measured at about 60-61 m in length and about 26-30 m in width, depending on where the game is held (North America or International).

At the length side, there are benches for the teams and the officials, as well as special benches for penalties. On one side, there are the two player benches, one for each team for the players to wait for their turn and for their coaches to observe the game and to call for substitution of players when necessary.

At the opposite side are the two penalty benches, one for each team when their player commits a rule violation/foul, and is sent to serve a timeout for a certain number of minutes depending on the severity of the penalty. In between the penalty benches is the scorekeeper bench where the scorekeepers record the game stats, make sure the game clock is accurate, communicate with officials, enforce rules and other responsibilities to ensure the game runs smoothly.

Lines

There are 5 lines in the rink for ruling purposes:

- Center line (Thick red dotted):
  This indicates the absolute middle of the rink and the middle of that center line is the faceoff spot where the game begins, be it from the start of a period. after a team scores a goal, or a call for faceoff due to a penalty by the officials.

- Defending line (Blue):
  The defending lines divide the rink into three parts, called zones. The zone in the middle is called the neutral zone and the team side is called the defending zone where it is used to judge if a player is offside. If an attacking player crosses the line into the other team's defending zone before the puck does, it is said to be offside.

- Goal line (Thin red solid):
  The goal line is used to judge goals and icing calls.

Faceoff spots and circle

There are 9 faceoff spots where all faceoffs take place. One in the middle of the center line, 4 in the neutral zone and 4 in the defending zone (2 on each team's defending zone).
Out of the 9 faceoff spots, 5 of which have the faceoff circle, the center and the 4 defending zone.
The center faceoff spot and circle is colored blue so that players don't get confused with the red dotted line. The defending zone however is colored red with hash marks to show the players where they can place themselves legally during faceoff or in-game play.

Goal posts and goal crease

The goal posts are placed in the middle of each of the goal lines, and a goal crease in front of each goalpost is an area for the goalie to perform without interference.

Referee crease

The referee crease is an area for referees to consult or report to the game officials on the scorekeeper bench, no players are allowed to enter it when such discussion takes place or they will be issued with a misconduct penalty.

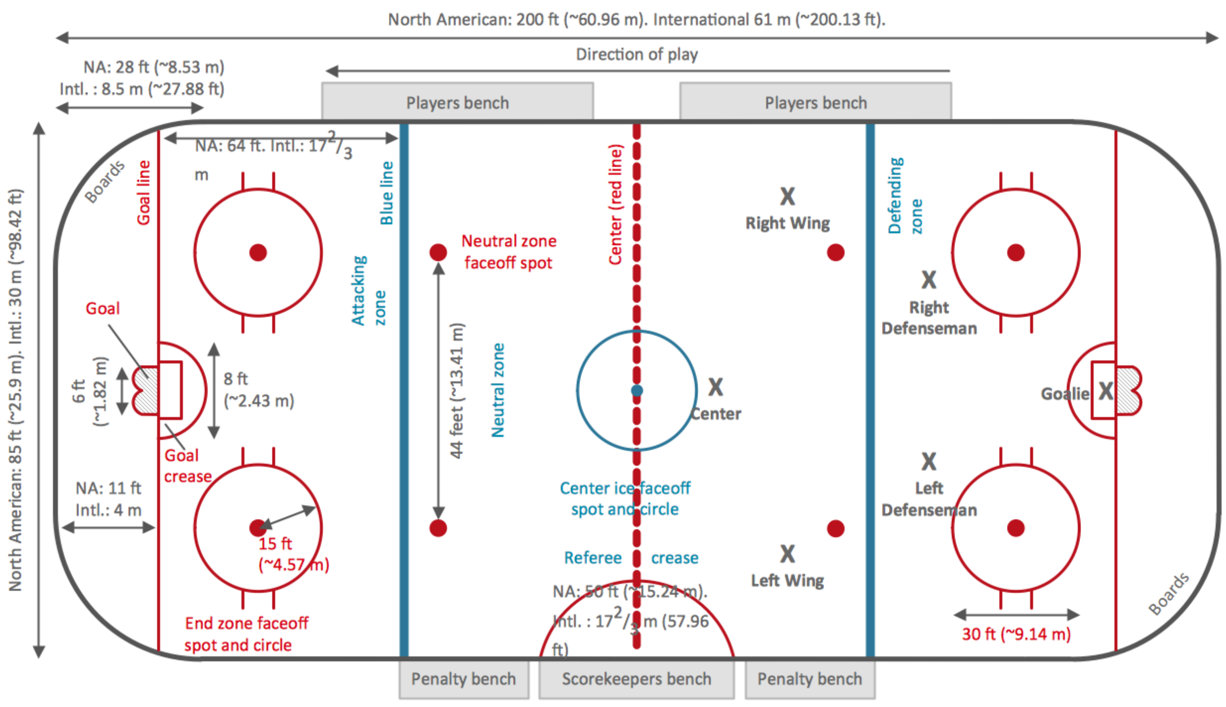


Figure 2: Detailed diagram of an ice hockey rink

Gameplay

Ice hockey is a high physical contact 6-man team sport the main goal is to score more goals than the opposing team. The game is divided into 3 periods, each 20 minutes long, with 15 minute intervals between each period for breaks and ice resurfacing. If the game is tied at the end of full time, it will go into a 5 minute overtime game. And if the game is still tied, then it will shift to best of three penalty shootout, and if it is still tied the shootout will continue until a winner is declared.

During the game

At the start of each period, each team will place their players in positions in figure 1 at the start with their center in the Center faceoff circle in the middle of the center line where they will fight for initial possession of the puck. The team that has the puck will attack with their forwards in order to score goals in the opposing team goal post.

Penalty

When there is a foul committed by a player, a penalty will be imposed on the said player. Depending on the severity of the penalty, it may range from the player just sitting in the penalty box for a certain amount of time during the game, to ejecting and banning them from the current and the next game.

There are many types of penalty, from minor to match penalty:

- Minor penalty

The player will sit in the penalty box for 2 minutes and his team will play short-handed, while the opposing team will go on a power play.

- Major penalty
  A more severe infraction of rules than a minor penalty, where the player sits in the penalty box for 5 minutes in which his team will be short-handed.

- Misconduct penalty
  Player who receive it will sit in the penalty box for 10 minutes, but may be able to be substituted for other team member on ice. This usually occurs when the player is having an altercation with the opposing player or the on-ice officials (referee) with the exception of fighting.

- Game misconduct
  This is usually given for various violations such as getting out of the penalty box before the penalty time is served, attempting to join or break up a fight (as the third man) or having a second misconduct penalty in the same game. The offending player will be ejected from the entire game and will immediately be substituted by another team member.

- Match penalty
  This is the most severe penalty whenever a player deliberately injures or attempts to injure another player. This punching, kicking, tackling or any forms of physical assault that are not in line with the rules. It usually needs referees and game officials to review footage in order to confirm the match penalty call or reduce it to a minor penalty.
  The offending player will be suspended for the next game or possible future games if necessary.

<u>Power Play and Short-handed</u>
In the event such as when a player is sent to the penalty box to serve time for a minor offense, there is now a Power Play and Short-handed event where there is a numerical advantage/ disadvantage on ice.

- Power Play
  An event where the team has numerical advantage and often employs more offensive tactics to score as many goals as possible.
- Short-handed
  An event where the team has numerical disadvantage and often employs defensive tactics in order to wait for the penalized player to finish his penalty time.

Problem Statement

<u>Problem</u>
NHL is a very competitive, high contact sport and the top few teams have had a monopoly on the biggest competition, the Stanley Cup, for many years. This causes the stagnation of the competitive spirit amongs the other teams, or the more talented player of the lower team being poached by the better team for a more lucrative future.

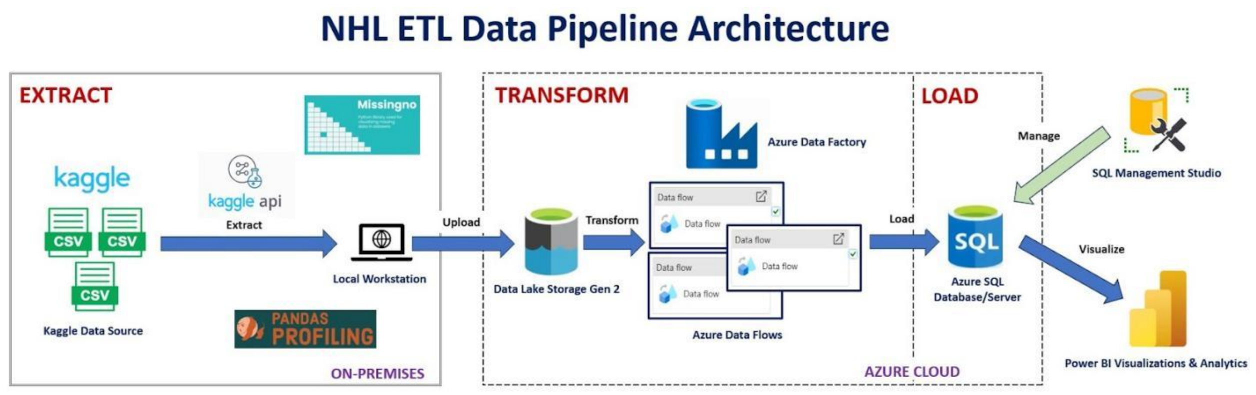
<u>Background</u>
NHL big teams such as Tampa Bay Lightning and Pittsburgh Penguins, and occasionally less well-known teams, have been dominating the Stanley Cup championship for a couple of years, and this could be the result of variables such as the team cohesion, coach teaching method.

To break the cycle we use about 10 years' worth of datasets to find the talented team players among the teams and assemble a fantasy team.

<u>Objective</u>
The primary objective is to find the best players with specific set of variables which include:

- The high shot/goal ratio for forwards, mainly center as they are the main attackers.
- Number of assists for wingers who help the center the most, and also shows how good their teamwork is.
- Save percentage for defenders and goalie to show their decision making and their reaction in defending their goal.
- Primary positions of the players as they are familiar with their roles.

Architecture



The extraction phase utilizes the **Kaggle API** and CSV files, allowing for programmatic access to datasets, reducing manual intervention. Initial data profiling and quality checks are performed on a local workstation using **Pandas Profiling** and **Missingno**, ensuring that missing values and anomalies are detected before cloud processing.

For transformation, the architecture leverages **Azure Data Lake Storage Gen 2**, providing a scalable and cost-effective storage solution. **Azure Data Factory** and **Data Flows** are used to orchestrate data transformations efficiently through a low-code/no-code approach, enabling parallel processing for better performance. This cloud-based transformation layer eliminates the need for extensive on-premises infrastructure.

In the loading phase, data is stored in **Azure SQL Database**, a managed cloud solution that ensures high availability, security, and scalability. This database can be managed **using SQL Management Studio**, allowing for seamless query execution and optimization. Additionally, the architecture integrates **Power BI** for visualization and analytics, enabling real-time reporting and interactive dashboards to provide valuable insights.

A key strength of this architecture is its **hybrid** approach, combining on-premises pre-processing for data quality checks with cloud-based transformation and storage, optimizing both performance and cost. The use of Azure Data Factory ensures orchestration, making the ETL pipeline more efficient. Moreover, **the pay-as-you-go cloud** pricing model minimizes costs compared to traditional ETL tools.

ETL Architecture Tech Stack

Dataset

The NHL Game Data dataset on Kaggle contains comprehensive information about NHL games, teams, and players. The datasets includes:

1. Game data from the 2000-2001 season through the 2022-2023 season

2. 13 CSV files with detailed information such as:
    - Game-by-game results
    - Player statistics
    - Team performance metrics
    - Goalie statistics
    - Game events (goals, penalties, etc.)

The datasets are regularly updated, providing researchers and analysts with current NHL statistics. It serves as a valuable resource for sports analytics, allowing users to perform in-depth analyses of NHL trends, player performance, and team strategies over multiple seasons.

It is used for various purposes, including predictive modeling, historical analysis of NHL trends, and exploring factors that contribute to team and player success in professional ice hockey.

Data Schema & Entity Relationship Diagram

Data Model and Entity Relationship Design

In our analysis of the NHL datasets, we focused on optimizing the data structure for efficient querying and comprehensive insights. Here's an overview of our approach:

Data Selection and Refinement

From the original 13 CSV files, we identified 6 critical files essential for our analysis. This selection process ensured a focused and manageable datasets while maintaining the integrity of key information.

Entity Relationship Diagram (ERD) Design

We developed an ERD to visualize and structure the relationships between entities. The key relationships identified are as follows:

1. Game to Player/Goalie/Skater: Many-to-Many
2. Game to Game Play: One-to-Many
3. Game to Game Team Stats: One-to-One

Normalization Process

To optimize data integrity and reduce redundancy, we normalized the many-to-many (M:M) relationships:

1. Game to Player/Goalie/Skater:
    - Intermediary tables: game_shifts, game_skater_stats, game_goalie_stats

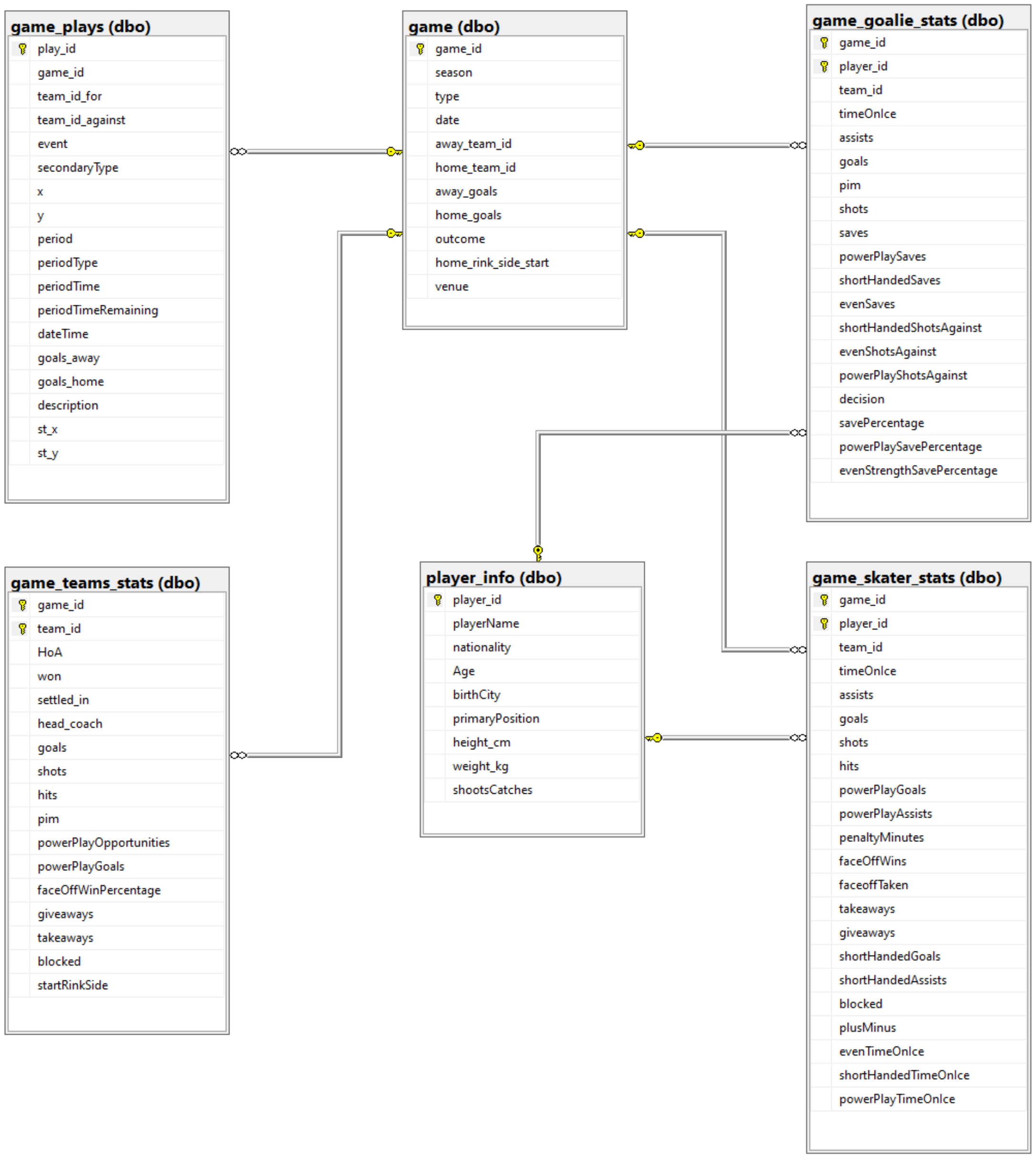
Resulting Normalized Structure

This process resulted in 6 normalized tables, providing a robust foundation for our analysis:

1. Games
2. Players_Info
3. Game_Team_Stats
4. Game_Plays
5. Game_Skater_Stats
6. Game_Goalie_Stats

This optimized structure enables efficient querying, maintains data integrity, and supports complex analytical operations across various aspects of NHL game data.

Data Schema

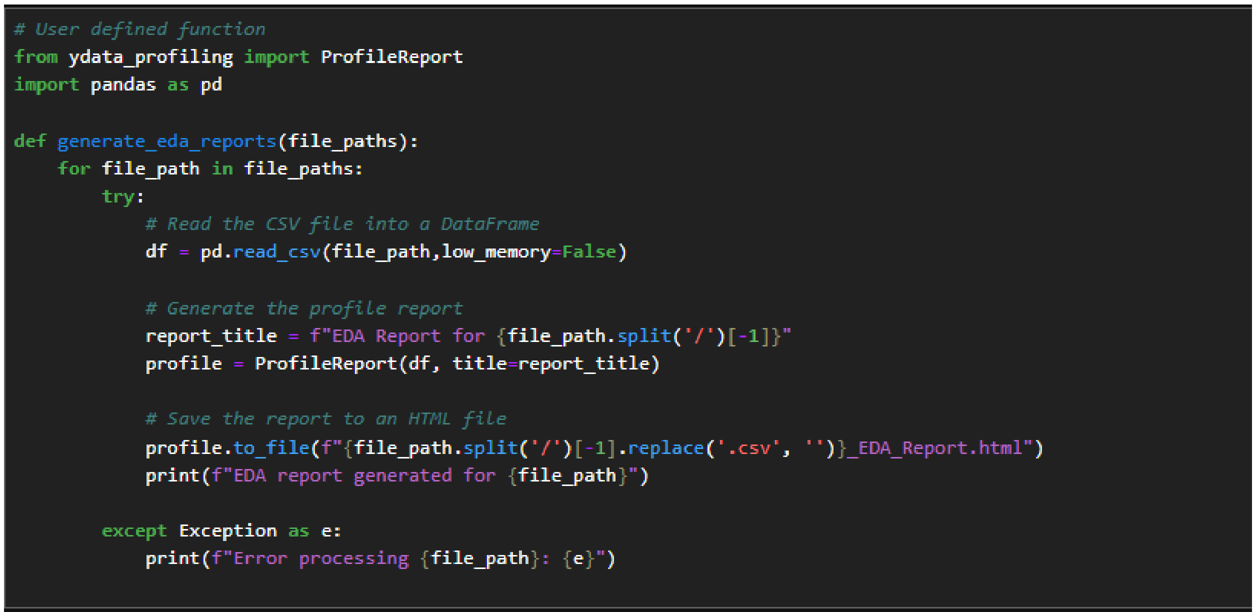


Approach

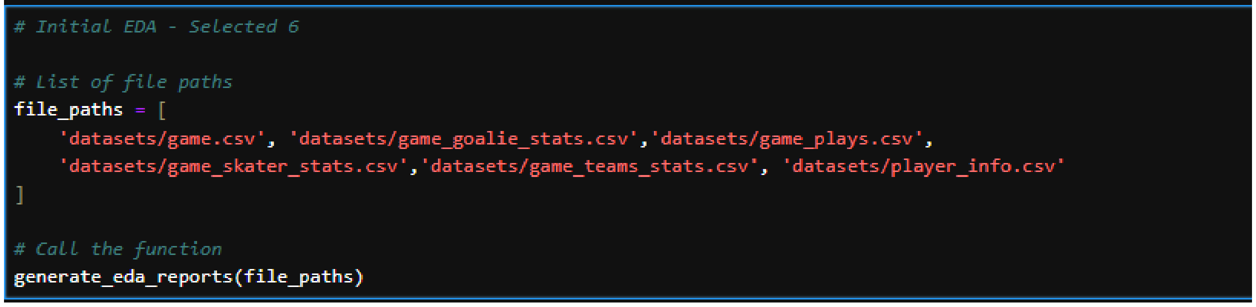
Data Exploration & Assessment

Exploratory Data Analysis (EDA) plays a vital role in understanding the main characteristics of the datasets, identifying patterns, and uncovering insights. To achieve this, our team utilized **ydata-profiling** (formerly known as pandas-profiling) to generate comprehensive EDA reports. These reports provide a detailed overview of the datasets, including statistics, missing values, duplicate records, and correlation heatmaps.

Below is the Python script used to generate the EDA reports for six selected datasets. The function processes multiple datasets, creates HTML reports, and displays valuable insights.



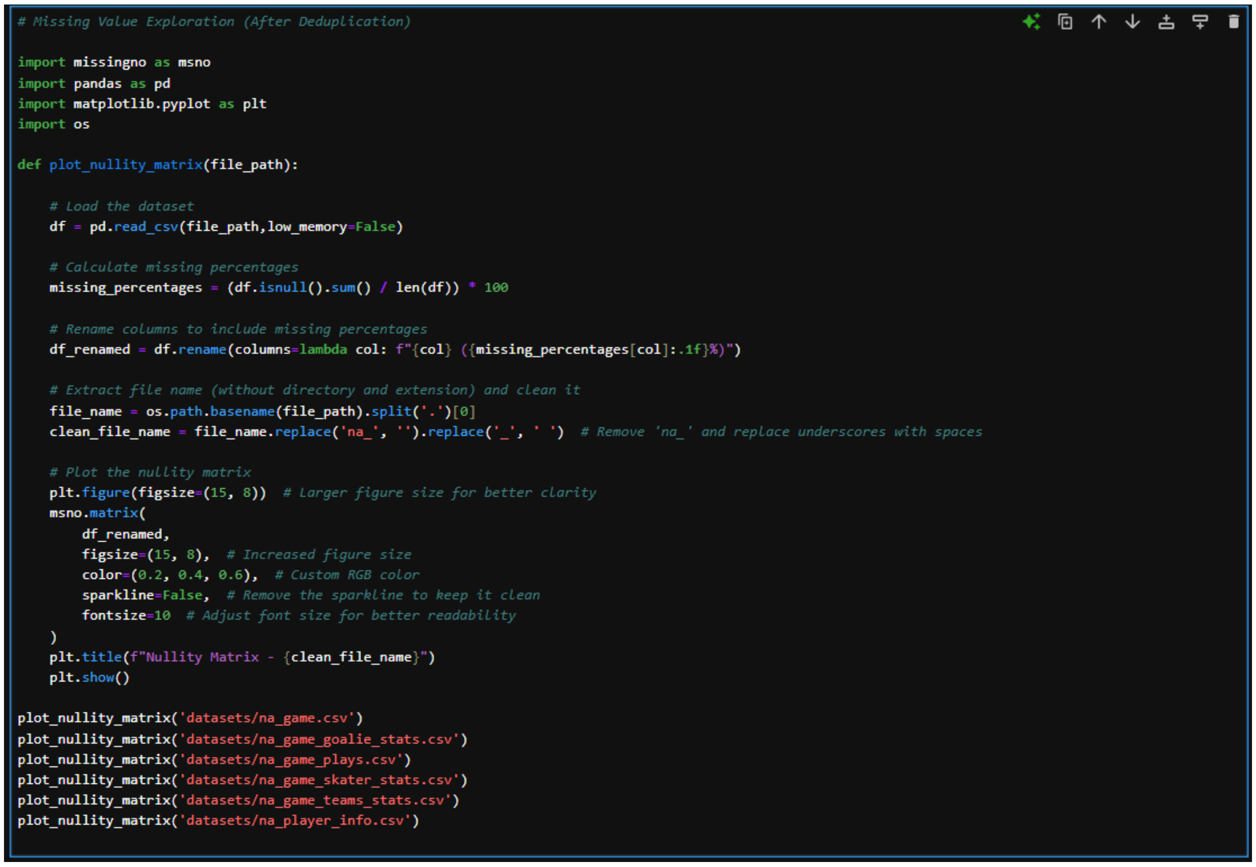
```python
# User defined function
from ydata_profiling import ProfileReport
import pandas as pd

def generate_eda_reports(file_paths):
    for file_path in file_paths:
        try:
            # Read the CSV file into a DataFrame
            df = pd.read_csv(file_path,low_memory=False)

            # Generate the profile report
            report_title = f"EDA Report for {file_path.split('/')[-1]}"
            profile = ProfileReport(df, title=report_title)

            # Save the report to an HTML file
            profile.to_file(f"{file_path.split('/')[-1].replace('.csv', '')}_EDA_Report.html")
            print(f"EDA report generated for {file_path}")

        except Exception as e:
            print(f"Error processing {file_path}: {e}")
```



```python
# Initial EDA - Selected 6

# List of file paths
file_paths = [
    'datasets/game.csv', 'datasets/game_goalie_stats.csv','datasets/game_plays.csv',
    'datasets/game_skater_stats.csv','datasets/game_teams_stats.csv', 'datasets/player_info.csv'
]

# Call the function
generate_eda_reports(file_paths)
```

To analyze missing data effectively, we used the **missingno** Python library. This tool visualizes missing values in the form of a Nullity Matrix, offering a graphical representation of the missing data patterns. This provides better insights into data completeness.

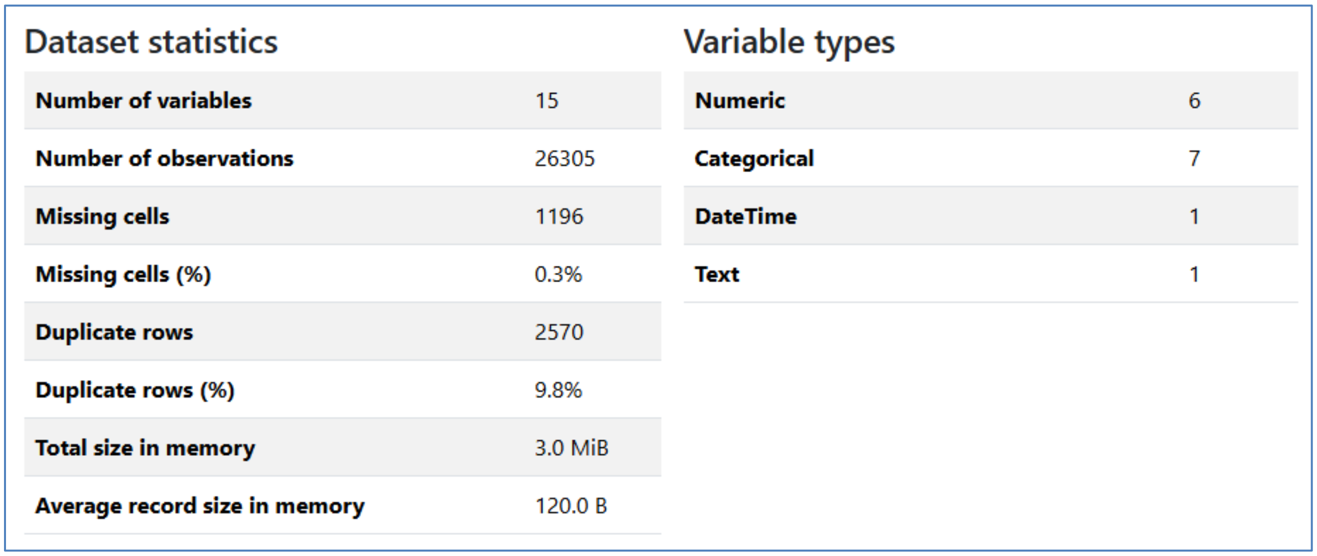
Below is the Python script to generate the matrix for each of the six datasets:



```python
# Missing Value Exploration (After Deduplication)

import missingno as msno
import pandas as pd
import matplotlib.pyplot as plt
import os

def plot_nullity_matrix(file_path):

    # Load the dataset
    df = pd.read_csv(file_path,low_memory=False)

    # Calculate missing percentages
    missing_percentages = (df.isnull().sum() / len(df)) * 100

    # Rename columns to include missing percentages
    df_renamed = df.rename(columns=lambda col: f"{col} ({missing_percentages[col]:.1f}%)")

    # Extract file name (without directory and extension) and clean it
    file_name = os.path.basename(file_path).split('.')[0]
    clean_file_name = file_name.replace('na_', '').replace('_', ' ')  # Remove 'na_' and replace underscores with spaces

    # Plot the nullity matrix
    plt.figure(figsize=(15, 8))  # Larger figure size for better clarity
    msno.matrix(
        df_renamed,
        figsize=(15, 8),  # Increased figure size
        color=(0.2, 0.4, 0.6),  # Custom RGB color
        sparkline=False,  # Remove the sparkline to keep it clean
        fontsize=10  # Adjust font size for better readability
    )
    plt.title(f"Nullity Matrix - {clean_file_name}")
    plt.show()

plot_nullity_matrix('datasets/na_game.csv')
plot_nullity_matrix('datasets/na_game_goalie_stats.csv')
plot_nullity_matrix('datasets/na_game_plays.csv')
plot_nullity_matrix('datasets/na_game_skater_stats.csv')
plot_nullity_matrix('datasets/na_game_teams_stats.csv')
plot_nullity_matrix('datasets/na_player_info.csv')
```

By leveraging these tools and methods, we aim to gain critical insights into data quality and readiness for further processing.

Dataset - game.csv

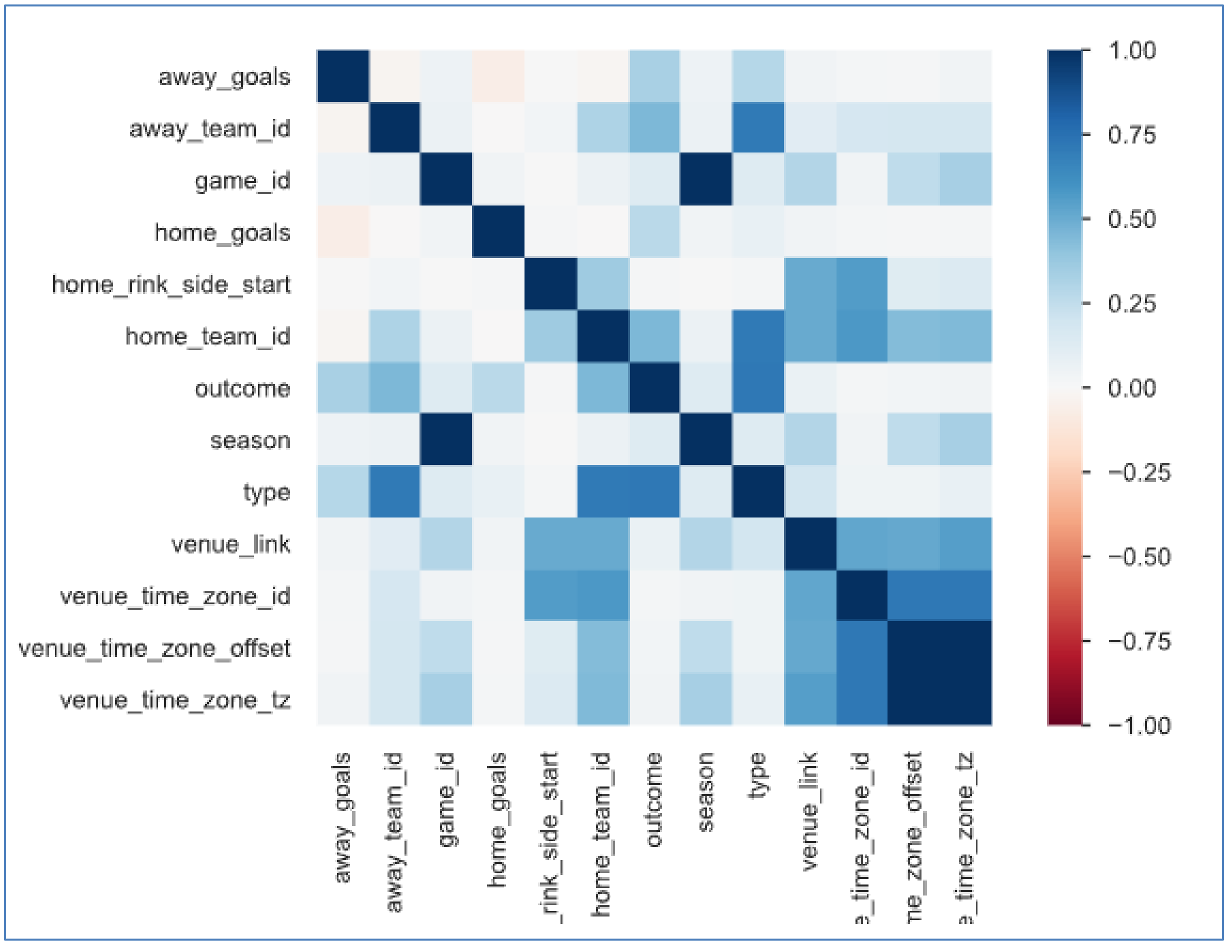


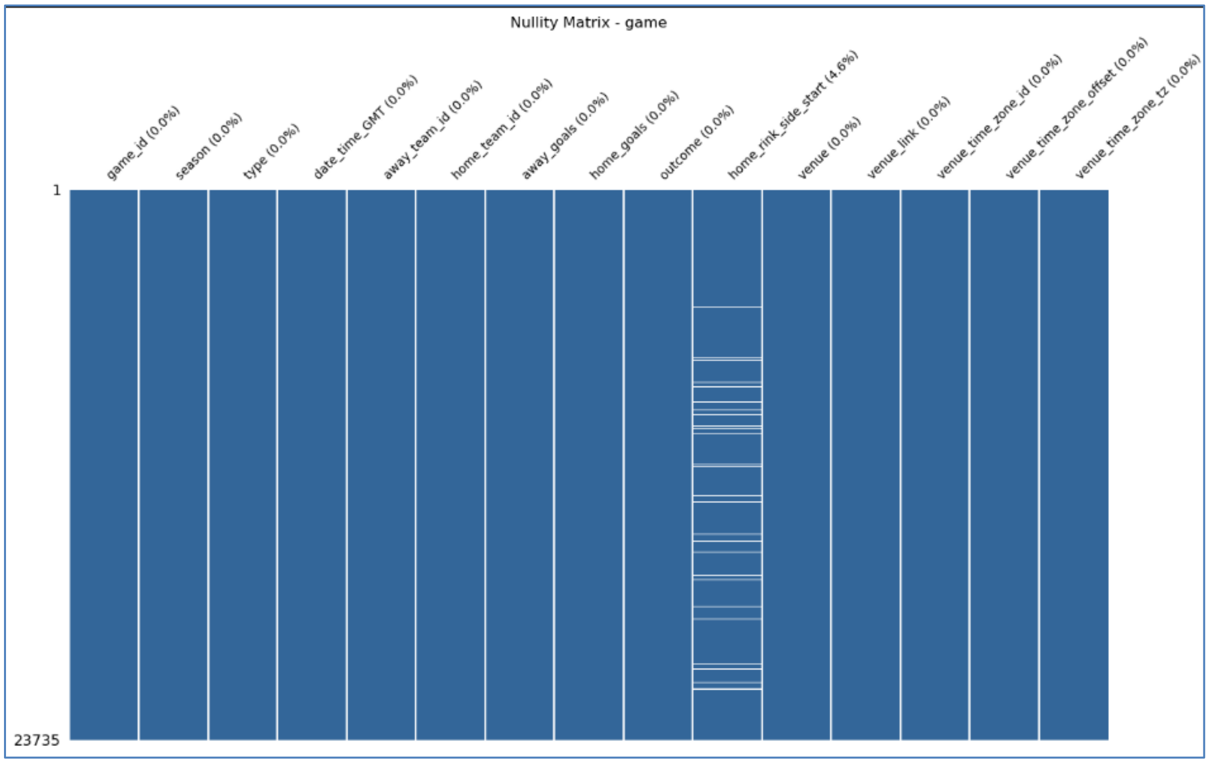
Dataset statistics

Number of variables	15
Number of observations	26305
Missing cells	1196
Missing cells (%)	0.3%
Duplicate rows	2570
Duplicate rows (%)	9.8%
Total size in memory	3.0 MiB
Average record size in memory	120.0 B

Variable types

Numeric	6
Categorical	7
DateTime	1
Text	1





The **game.csv** dataset consists of **15 variables** and **26,305 observations**, making it one of the key datasets for our analysis. This dataset includes variables of numeric, categorical, and date-time types, offering a broad spectrum of data for exploratory analysis.

The dataset contains **1,196 missing cells**, accounting for approximately **0.3%** of the data. This minimal percentage indicates that it is relatively complete, requiring minor pre-processing efforts to handle the missing values. However, the dataset contains **2,570 duplicate rows**, which represent about **9.8%** of the total records. Removing these duplicates is crucial for ensuring data integrity and avoiding skewed insights during analysis.

The dataset consists of **6 numeric variables**, including **home_goals** and **away_goals**, which capture game-related performance metrics. There are **7 categorical variables** that serve as identifiers, such as season, outcome, and type, classifying different game attributes. Additionally, a **single Date-Time variable** records time-based information, aiding in chronological analysis. Lastly, **one text variable**, likely **venue_link**, provides descriptive contextual information about the game venue

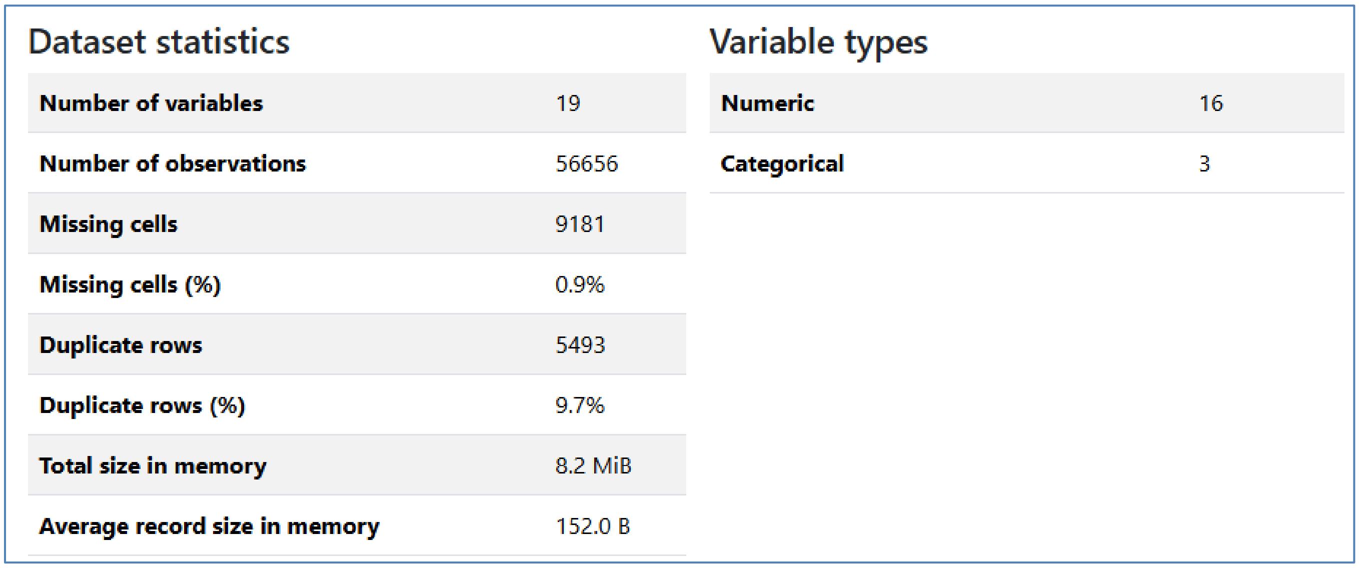
From the **Nullity Matrix** visualization, the variable **home_rink_side_start** stands out as the only column with missing values. Approximately **4.6%** of the values are missing, equating to around **1,196** cells. While its absence may not be significantly impactful, it might play a critical role in venue-related analysis.

Column Name	Description
game_id	Unique identifier for each game (YYYYTTGGGG)
season	The season in which the game was played
type	Type of game
date_time	Date and time of the game
away_team_id	IDs of the AWAY Teams
home_team_id	IDs of the HOME Teams
away_goals	Final scores for AWAY Team
home_goals	Final scores for HOME Team
outcome	Result of the game
home_rink_side_start	Indicates which side of the rink the home team started on
venue	Name of the venue where the game was played
venue_link	API link associated with the venue for additional details
venue_time_zone_id	Time zone of the venue, given as a location-based identifier
venue_time_zone_offset	The time difference from GMT
venue_time_zone_tz	Abbreviation for the time zone at the venue

The above data dictionary for **game.csv** was created based on research and interpretation of the columns and variables in the dataset. Each column's meaning was inferred from its name, context, and relevance to the dataset's subject matter. This dictionary serves as a reference to clarify the purpose of each variable, such as game identifiers, team and score details, outcomes, and venue information.

The dictionary will provide a structured understanding of the dataset, enabling accurate analysis and insights.

Dataset - game_goalie_stats.csv

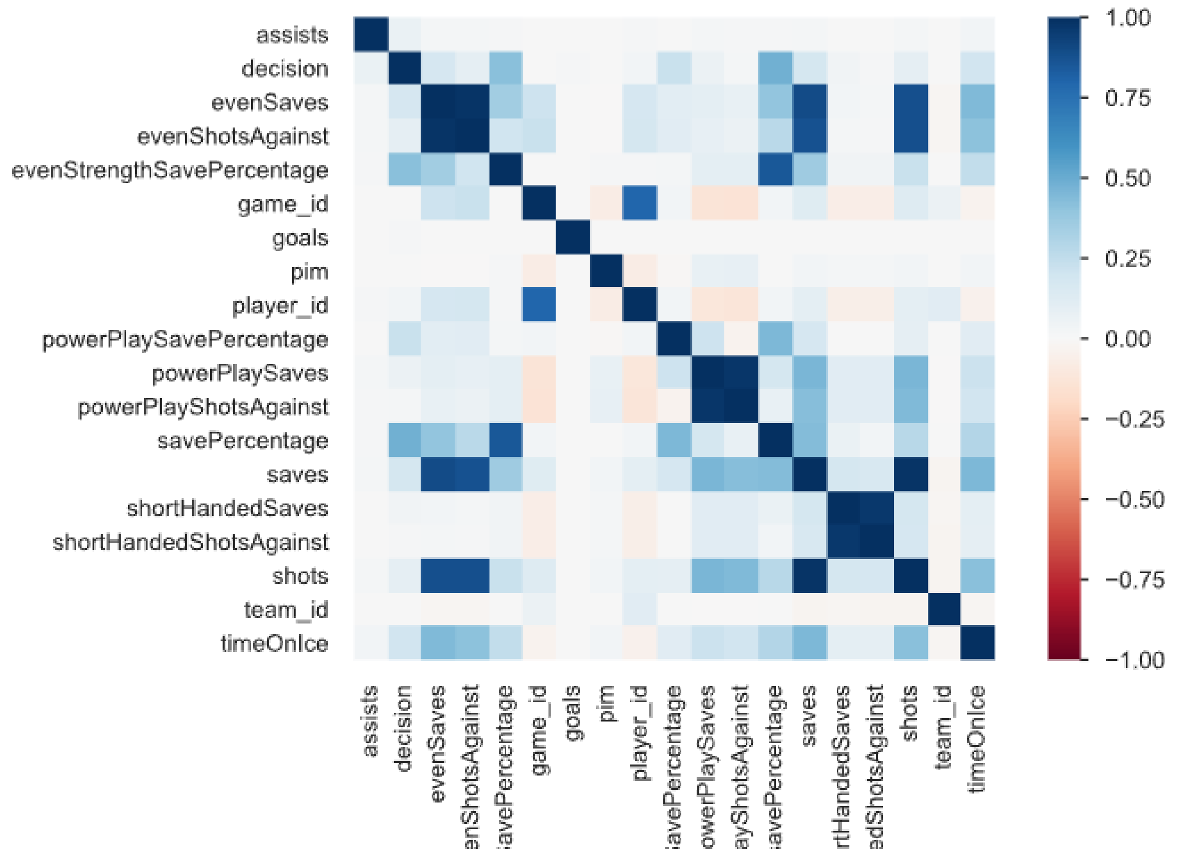


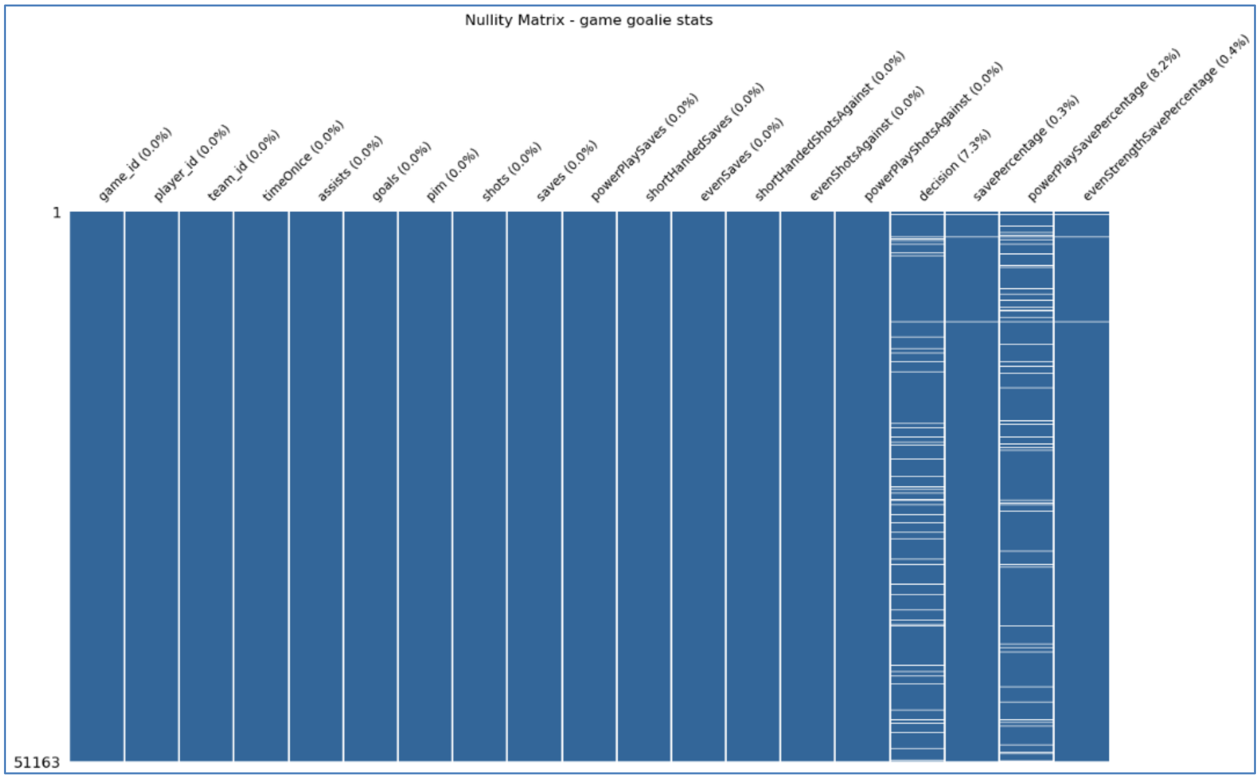
Dataset statistics

Number of variables	19
Number of observations	56656
Missing cells	9181
Missing cells (%)	0.9%
Duplicate rows	5493
Duplicate rows (%)	9.7%
Total size in memory	8.2 MiB
Average record size in memory	152.0 B

Variable types

Numeric	16
Categorical	3





The **game_goalie_stats.csv** dataset contains **19 variables** and **56,656 observations**, providing comprehensive data on goalie performance in hockey games. The dataset includes **16 numeric variables** and **3 categorical variables**, offering a mix of quantitative and descriptive insights.

There are **9,181 missing cells**, accounting for **0.9%** of the data. This low percentage indicates a high level of completeness, requiring minimal handling of missing data.

The dataset contains **5,493 duplicate rows**, which represent **9.7%** of the total data. Addressing these duplicates is essential for ensuring data accuracy and consistency.

The dataset consists of **16 numeric variables** that capture key goalie performance metrics, including saves, shots, goals, savePercentage, and timeOnIce, providing detailed insights into goalkeeping effectiveness. Additionally, there are **3 categorical variables**, which include identifiers such as **player_id** and **team_id**, as well as **decision**, which classifies game outcomes for goalies, such as a win or loss

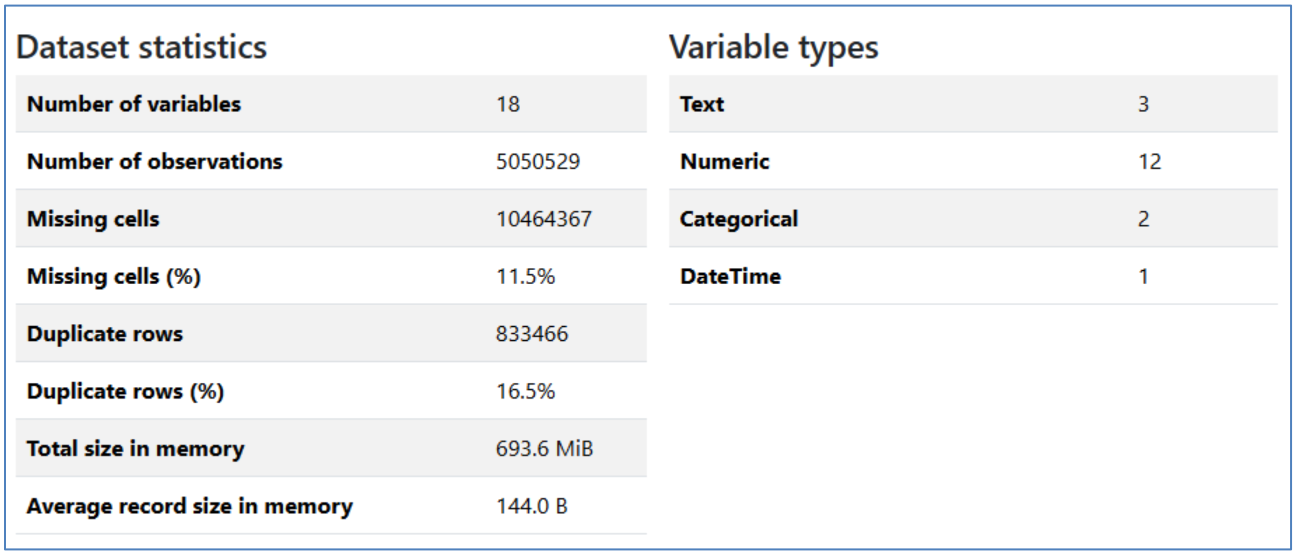
The heatmap of numeric variables reveals several notable patterns. Variables like **shots**, **saves**, and **timeOnIce** are closely related, reflecting that goalies who face more shots tend to spend more time on the ice and record higher save counts. The variable **goals** is negatively correlated with **savePercentage**, indicating that higher goals allowed result in lower save efficiency.

Column Name	Description
game_id	unique identifier for each game
player_id	unique identifier for the goalie player
team_id	The team ID corresponding to the team for which the goalie played in the specific game
timeOnIce	The total time (in seconds) the goalie spent on the ice during the game
assists	The number of assists made by the goalie during the game
goals	The number of goals scored by the goalie during the game (usually 0 for goalies)
pim	Penalty minutes incurred by the goalie during the game
shots	The total number of shots faced by the goalie during the game
saves	The total number of saves made by the goalie (shots faced minus goals allowed)
powerPlaySaves	Saves made by the goalie while the opposing team was on a power play
shortHandedSaves	Saves made by the goalie while their team was short-handed (on a penalty kill)
evenSaves	Saves made by the goalie during even-strength play
shortHandedShotsAgainst	The number of shots against the goalie while opposing team was short-handed
evenShotsAgainst	The number of shots against the goalie during even-strength play
powerPlayShotsAgainst	The number of shots against the goalie while the opposing team was on a power play
decision	The game outcome for the goalie, such as Win (W), Loss (L), or other results (e.g., OT or SO for overtime/shootout decisions)
savePercentage	The goalie's overall save percentage for the game (saves divided by total shots faced)
powerPlaySavePercentage	The save percentage for shots faced during power plays
evenStrengthSavePercentage	The save percentage for shots faced during even-strength play

The above data dictionary for **game_goalie_stats.csv** was created based on research and interpretation of the dataset's variables. Each column's meaning was deduced from its name and its likely context within hockey statistics. This dictionary provides definitions for key variables such as game identifiers, goalie performance metrics, and situational statistics like saves, shots against, and save percentages in various scenarios (e.g., power plays, even-strength).

This dictionary will serve as a reference to guide data exploration and analysis, ensuring accurate understanding of the dataset and enabling deeper insights into goalie performance during games.

Dataset - game_plays.csv

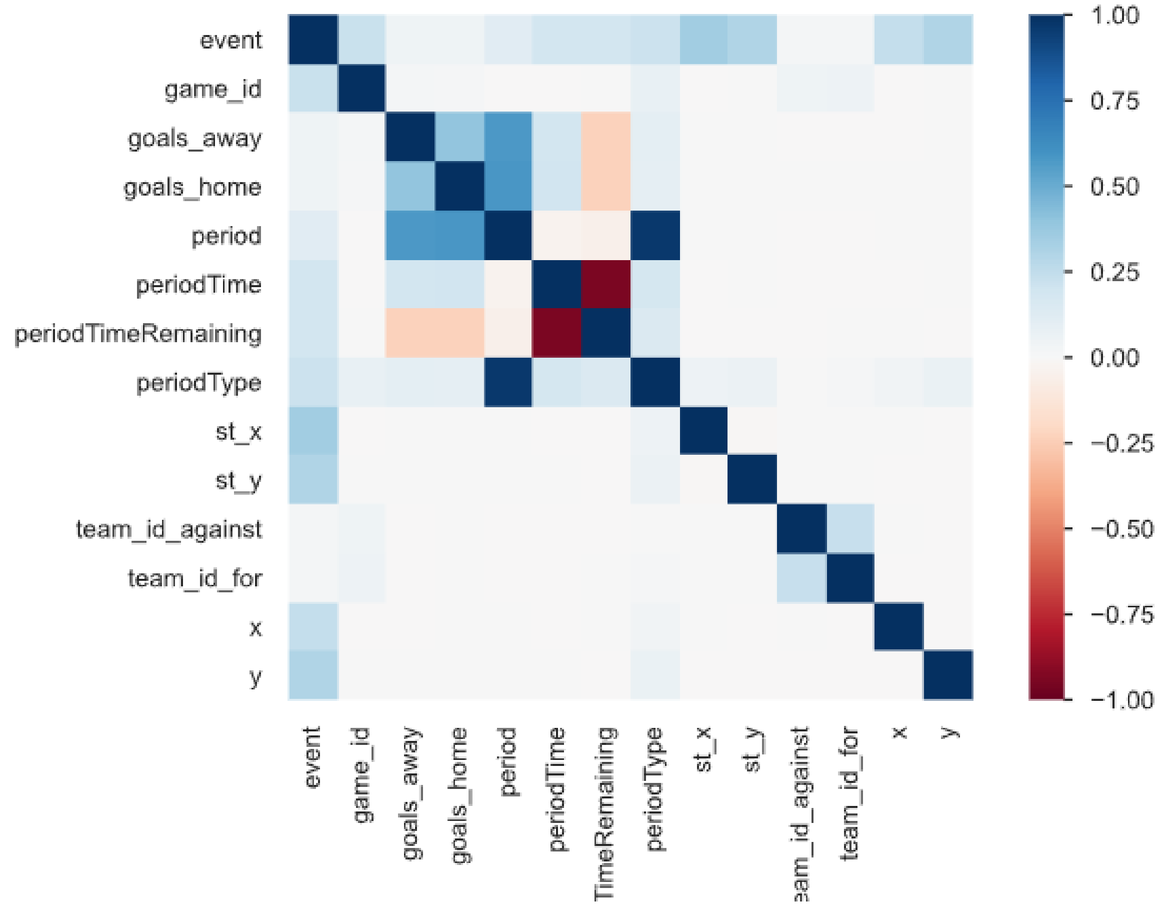


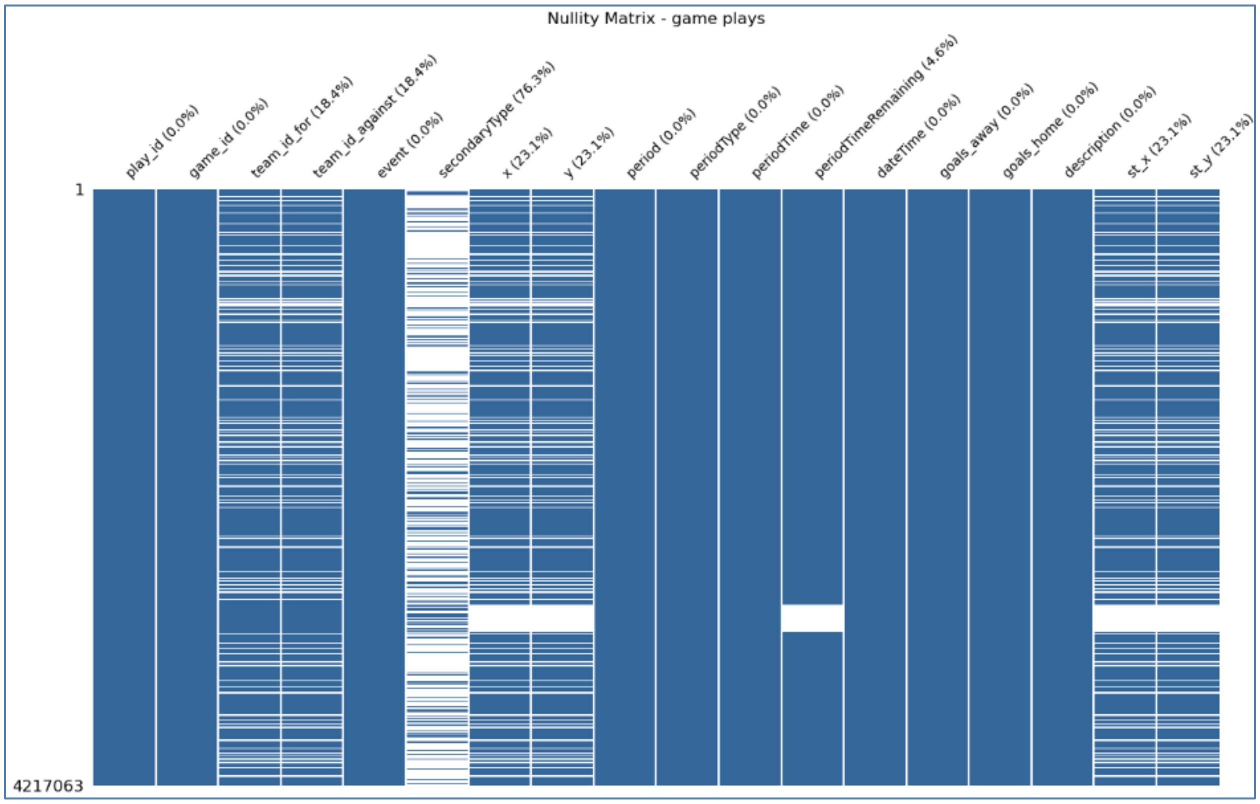
Dataset statistics

Number of variables	18
Number of observations	5050529
Missing cells	10464367
Missing cells (%)	11.5%
Duplicate rows	833466
Duplicate rows (%)	16.5%
Total size in memory	693.6 MiB
Average record size in memory	144.0 B

Variable types

Text	3
Numeric	12
Categorical	2
DateTime	1





The **game_plays.csv** dataset provides detailed information on individual game events, containing **18 variables** and over **5 million observations**. This dataset is vital for analyzing game dynamics and event-level insights.

The dataset has **10,464,367 missing cells**, accounting for **11.5%** of the total data. This indicates a significant number of missing values,

The dataset contains **833,466 duplicate rows**, comprising **16.5%** of the total records. Removing these duplicates is essential for ensuring the accuracy of event-level analysis.

The dataset includes **3 text variables**, which provide categorical information such as event types and descriptive labels. Additionally, there are **12 numeric variables** representing various game metrics, including goals, positions, and time-related data. Lastly, **3 categorical and DateTime variables** capture details on game periods, period types, and event timestamps,

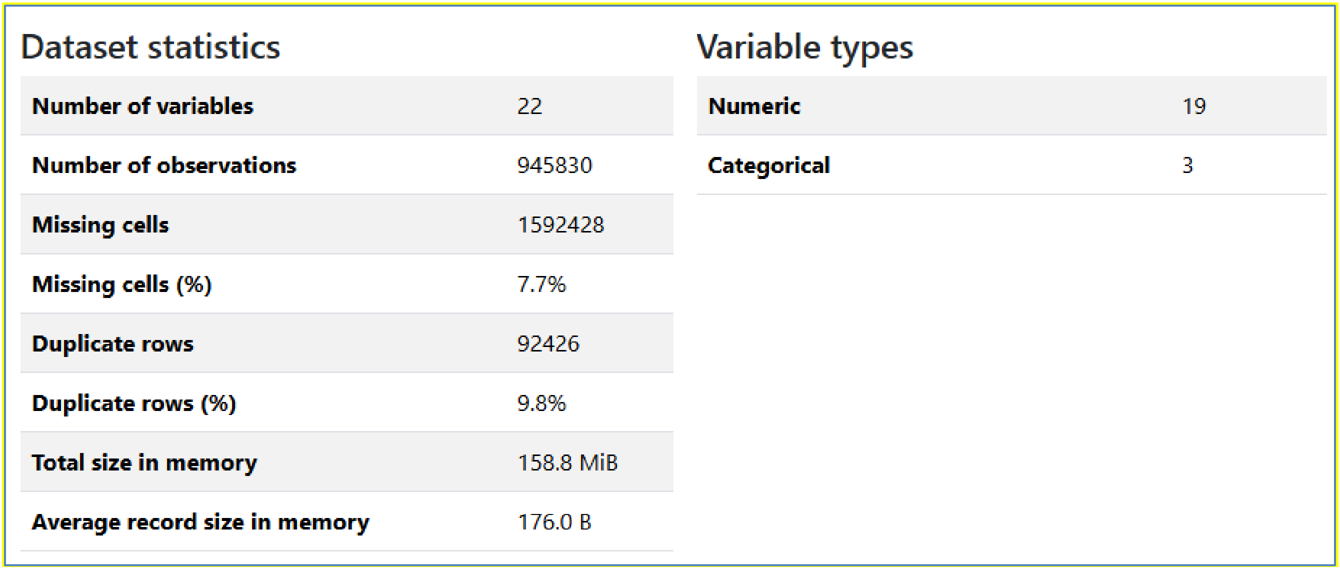
The heatmap reveals strong correlations between **periodTimeRemaining** and **period**, reflecting temporal dependencies within each period.Spatial variables **st_x** and **st_y** exhibit moderate correlations, suggesting structured event patterns.

Column Name	Description
play_id	Unique identifier for each play in a game
game_id	Unique identifier for each game
team_id_for	The ID of the team performing the event or action
team_id_against	The ID of the opposing team during the event or action
event	The type of event that occurred
secondaryType	Provides additional context or classification for the event
x	The RAW x-coordinate of the event on the rink
y	The RAW y-coordinate of the event on the rink
period	The period of the game in which the event occurred
periodType	The type of period
periodTime	The elapsed time in seconds from the start of the period when the event occurred
periodTimeRemaining	The remaining time in seconds in the period when the event occurred
dateTime	The exact timestamp when the event occurred (in ISO format)
goals_away	The number of goals scored by the away team at the time of the event
goals_home	The number of goals scored by the home team at the time of the event
description	A textual description of the event, often including player names and specific actions
st_x	The STANDARDIZED x-coordinate of the event on the rink
st_y	The STANDARDIZED y-coordinate of the event on the rink

The above data dictionary for **game_plays.csv** was constructed based on research and interpretation of the dataset's variables. It provides descriptions for each column, explaining key elements such as play identifiers, event details, spatial coordinates, team involvement, and game timing. For example, **x** and **y** capture the raw coordinates of events on the rink, and **periodTimeRemaining** provides temporal context within a game period.

This dictionary will serve as a foundational reference for understanding the dataset, enabling event-level analysis and insights into game dynamics.

Dataset - game_skater_stats.csv

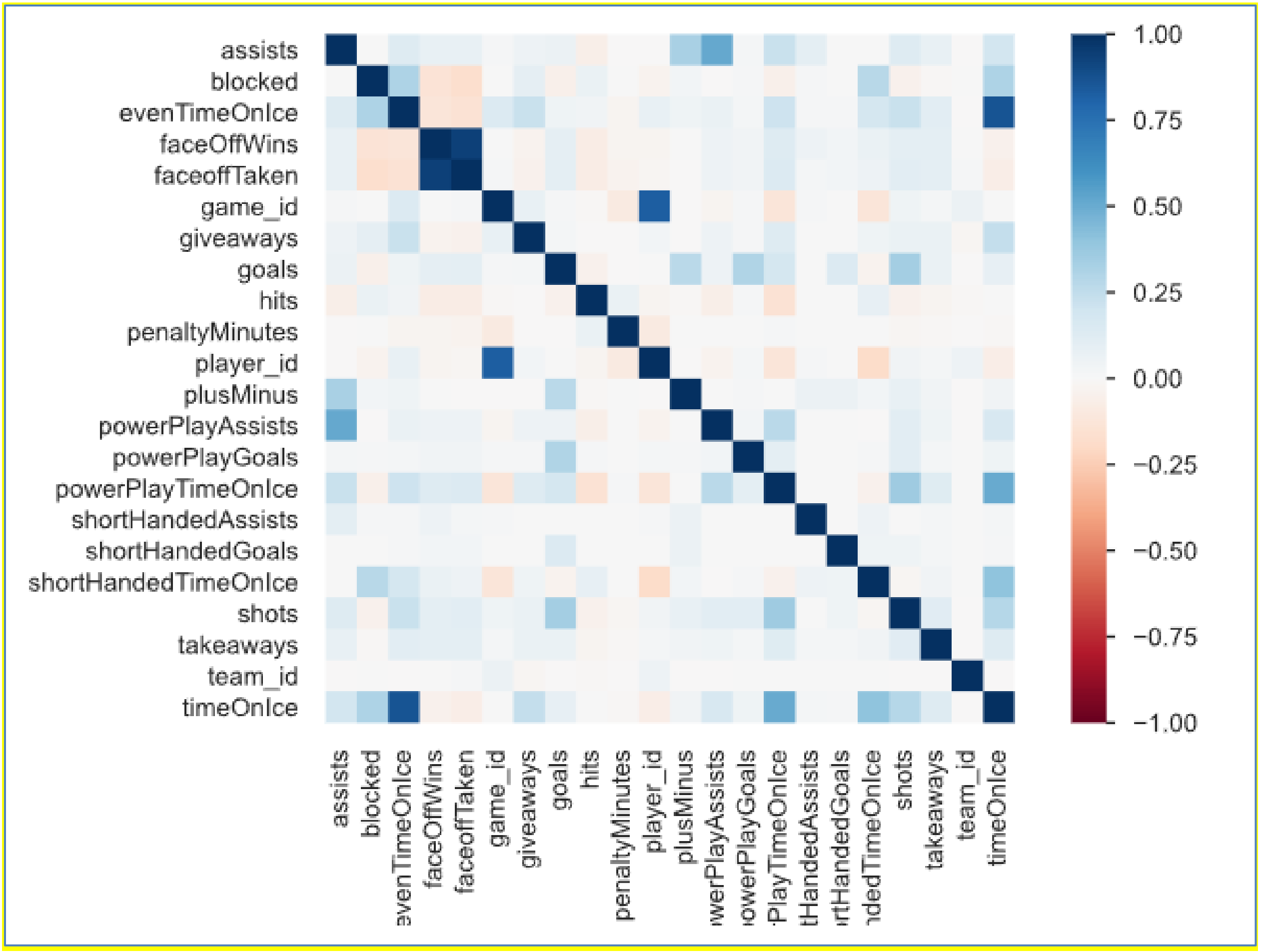


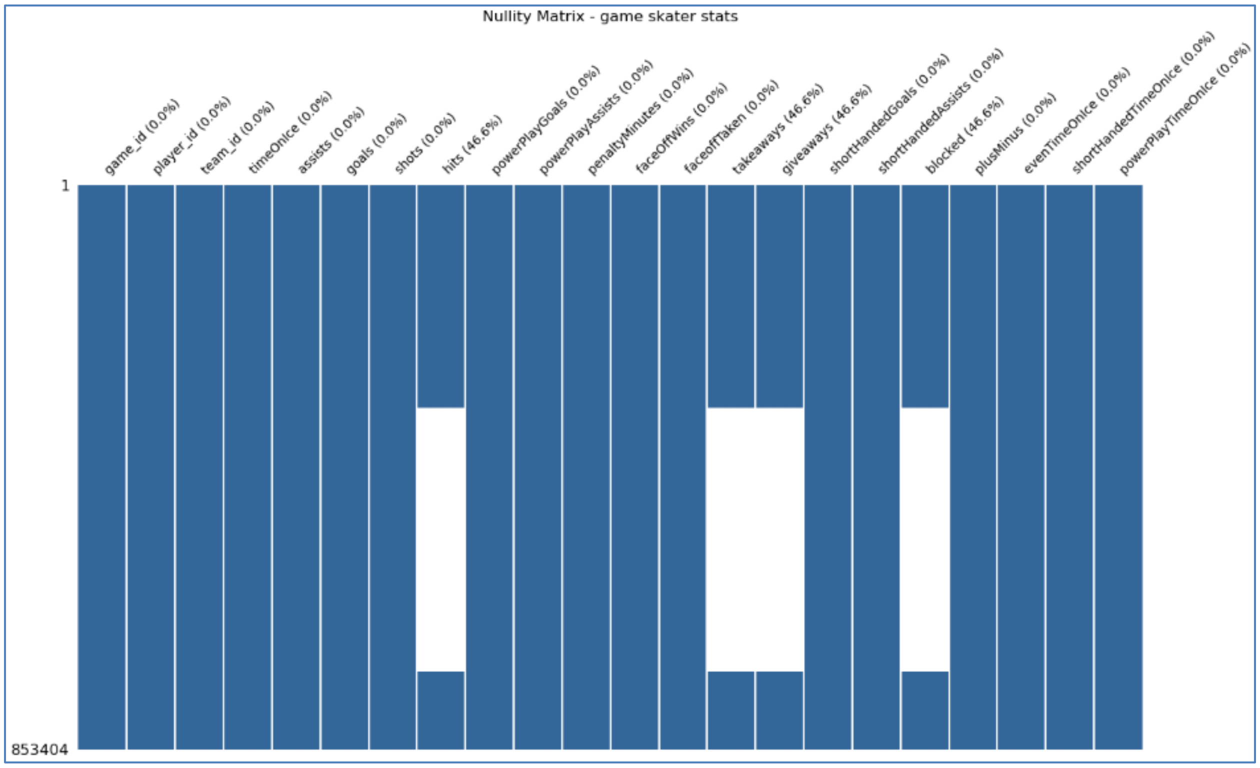
Dataset statistics

Number of variables	22
Number of observations	945830
Missing cells	1592428
Missing cells (%)	7.7%
Duplicate rows	92426
Duplicate rows (%)	9.8%
Total size in memory	158.8 MiB
Average record size in memory	176.0 B

Variable types

Numeric	19
Categorical	3





The game_skater_stats.csv dataset provides detailed information about player (skater) statistics during hockey games. It contains **22 variables** and **945,830 observations**, offering comprehensive data on player performance across various metrics.

The dataset contains **1,592,428 missing** cells, accounting for **7.7%** of the total data. This highlights the need for careful handling of missing values, particularly for key performance metrics.

There are **92,426 duplicate** rows, comprising **9.8%** of the total observations. Removing these duplicates is crucial to ensure data accuracy and reliability.

The dataset includes **19 numeric variables**, which represent performance metrics such as **goals**, **assists**, **hits**, and **penalty minutes**, providing quantitative insights into player contributions during games. Additionally, there are **3 categorical variables**, such as **team** and **player IDs**, which facilitate grouping and comparisons across games for deeper analysis.

The heatmap reveals strong correlations between **timeOnIce** and situational metrics like **evenTimeOnIce**, **powerPlayTimeOnIce**, and **shortHandedTimeOnIce**.

There are positive relationships between offensive metrics like **goals**, **assists**, and **plusMinus**, reflecting overall player impact.

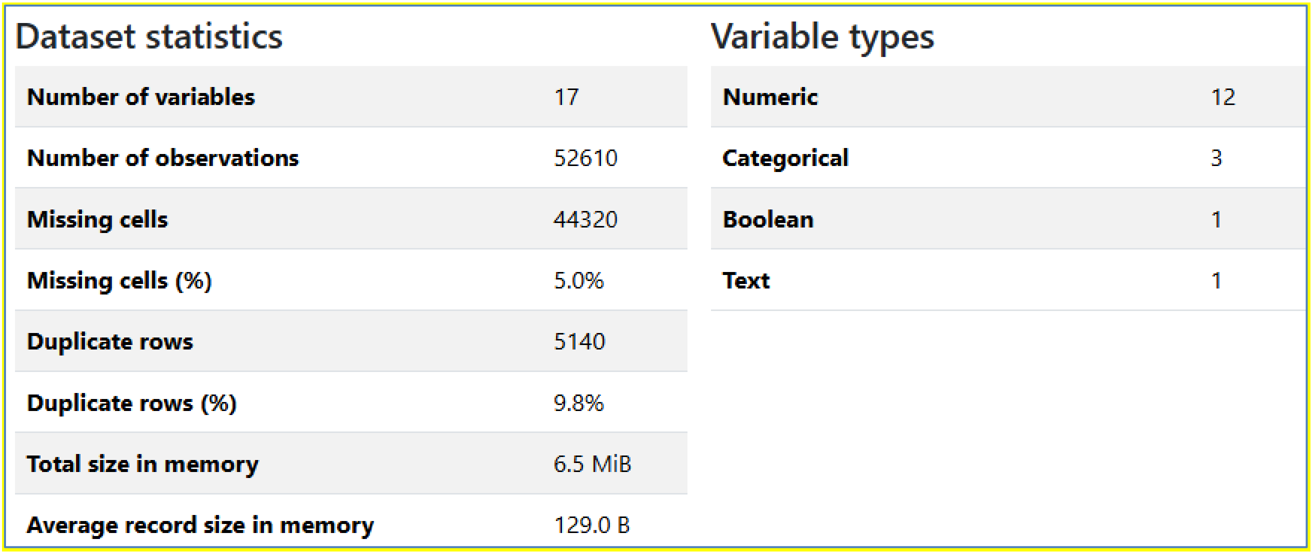
There are weak or negative correlations for defensive metrics like **giveaways** and **penaltyMinutes**, which often indicate detrimental plays.

Column Name	Description
game_id	Unique identifier for the game
player_id	Unique identifier for the player
team_id	Unique identifier for the team the player belongs to
timeOnIce	Total time (in seconds) the player spent on the ice during the game
assists	Number of assists made by the player in the game
goals	Number of goals scored by the player
shots	Number of shots the player attempted
hits	Number of hits delivered by the player (physically taking possession from opponent)
powerPlayGoals	Goals scored by the player during a power play situation
powerPlayAssists	Assists made by the player during a power play
penaltyMinutes	Total penalty minutes the player accumulated during the game
faceOffWins	Number of face-offs won by the player
faceoffTaken	Total number of face-offs taken by the player
takeaways	Number of takeaways (stealing the puck from the opponent) by the player without contact
giveaways	Number of giveaways (losing the puck to the opponent) by the player without contact
shortHandedGoals	Goals scored by the player during a short-handed situation
shortHandedAssists	Assists made during a short-handed situation
blocked	Number of shots blocked by the player
plusMinus	Player's "plus-minus" statistic, which tracks their goal differential while on the ice
evenTimeOnIce	Time on ice (in seconds) during even-strength play
shortHandedTimeOnIce	Time on ice (in seconds) during a short-handed situation
powerPlayTimeOnIce	Time on ice (in seconds) during a power play situation

The data dictionary for **game_skater_stats.csv** was created through research and interpretation of the dataset's variables. Each column's meaning was inferred based on its name and context within hockey statistics. This dictionary provides detailed descriptions of key metrics, such as player performance (**goals**, **assists**, **shots**), situational contributions (**powerPlayGoals**, **shortHandedAssists**), and game participation (**timeOnIce**, **faceOffWins**). It also includes team and player identifiers, facilitating analysis at both individual and team levels.

This dictionary will aid in understanding the dataset's structure and supports accurate analysis of player performance and game dynamics.

Dataset - game_team_stats.csv

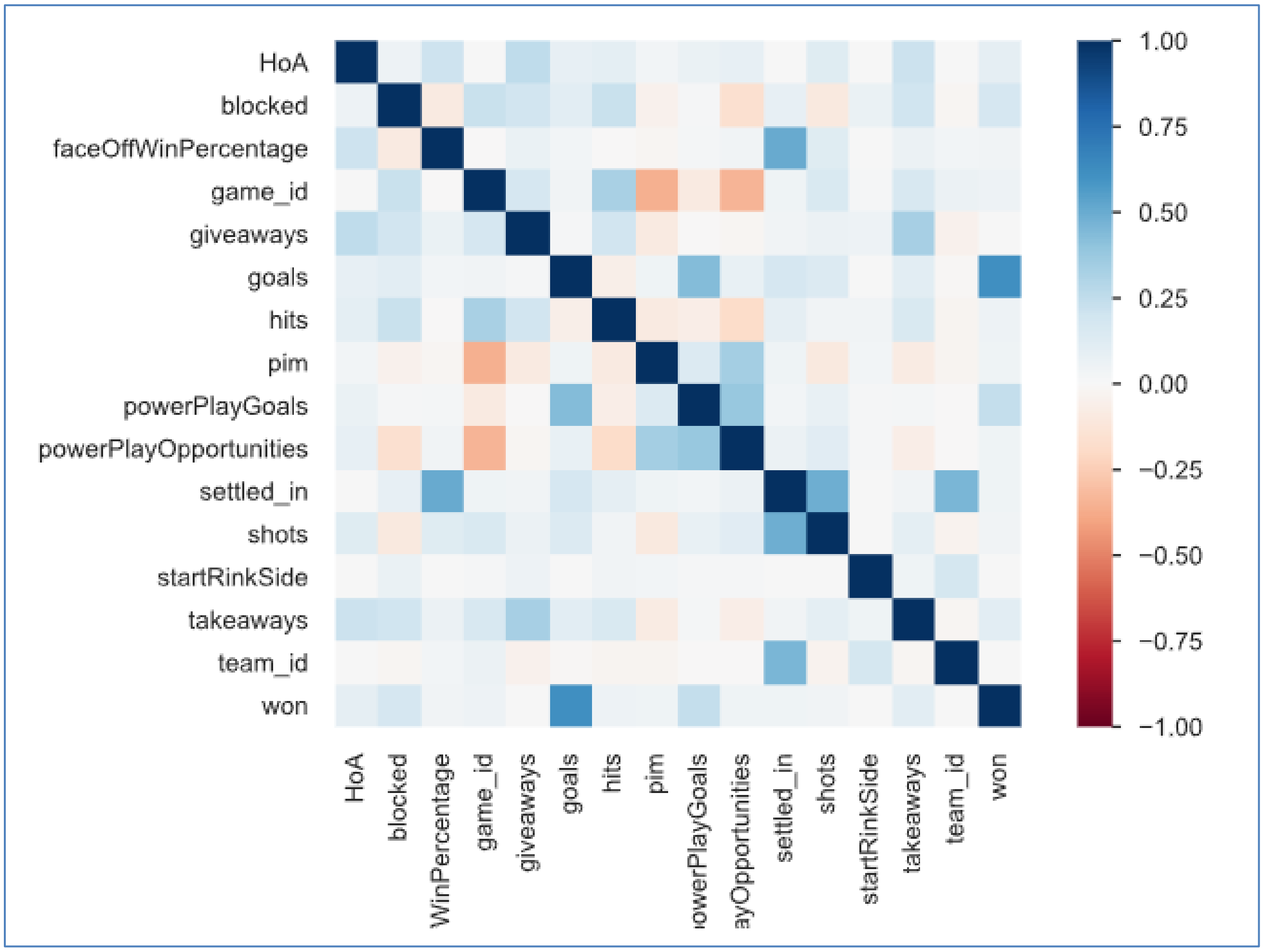


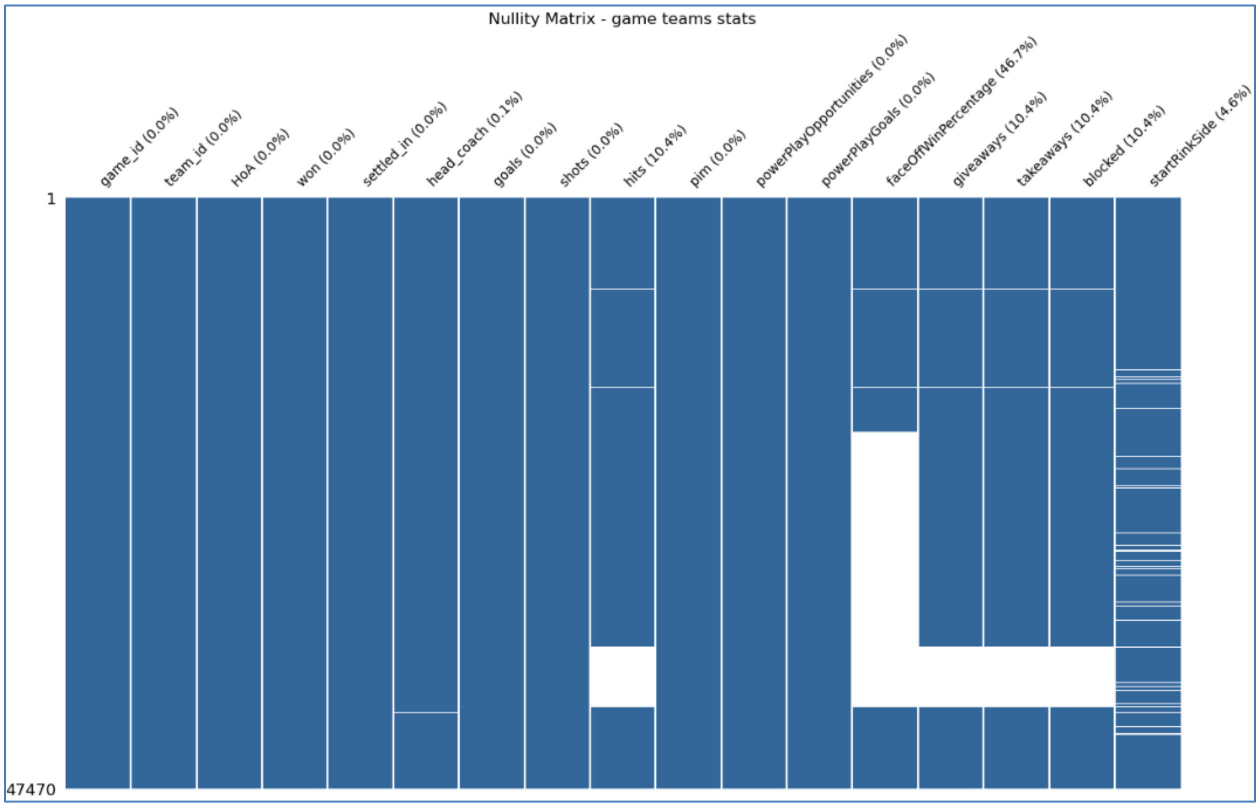
Dataset statistics

Number of variables	17
Number of observations	52610
Missing cells	44320
Missing cells (%)	5.0%
Duplicate rows	5140
Duplicate rows (%)	9.8%
Total size in memory	6.5 MiB
Average record size in memory	129.0 B

Variable types

Numeric	12
Categorical	3
Boolean	1
Text	1





The team_stats.csv dataset provides summarized performance data for hockey teams across games. It contains **17 variables** and **52,610 observations**, offering insights into team-level statistics and game outcomes.

There are **44,320 missing cells**, accounting for **5.0%** of the data. While this percentage is moderate, handling these missing values is essential for accurate analysis.

The dataset includes **5,140 duplicate** rows, making up 9.8% of the total data. Removing these duplicates is critical to ensure data reliability.

The dataset includes **12 numeric variables**, representing key performance metrics such as **goals**, **shots**, **hits**, and **takeaways**, which provide quantitative insights into team performance. Additionally, there are **3 categorical variables**, capturing team-related attributes like **team_id** and contextual factors such as **startRinkSide**, which help analyze team dynamics and game settings. The dataset also contains **1 boolean variable**, won, which indicates whether a team won the game, offering a direct measure of game outcomes. Lastly, there is **1 text variable** that provides descriptive data or labels, adding contextual information to the dataset.

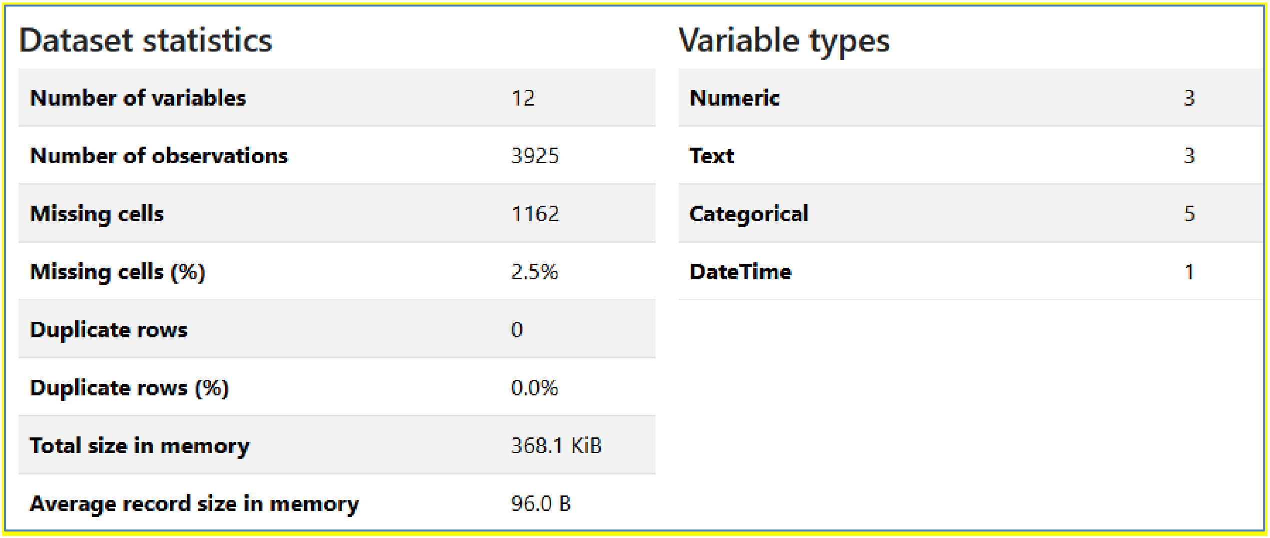
The heatmap reveals strong positive correlations between offensive metrics such as **goals**, **shots**, and **powerPlayGoals**, indicating their interconnected role in driving team performance. In contrast, defensive metrics like **blocked** and **hits** show weaker correlations with offensive performance, emphasizing the distinct nature of defensive play styles. Additionally, the **won** variable correlates positively with metrics like **goals** and **faceOffWinPercentage**, underscoring their significance in determining game outcomes and team success.

Column Name	**Description**
game_id	Unique identifier for each NHL game
team_id	Unique identifier for the team that participated in the game
HoA	Home or Away status of the team
won	Boolean indicating if the team won the game
settled_in	It describes the method by which the game ended
head_coach	Name of the head coach for the team in that game
goals	Number of goals scored by the team in the game
shots	Number of shots on goal taken by the team
hits	Number of hits delivered by the team during the game
pim	Penalty Infraction Minutes, the total number of **minutes** the team spent penalized
powerPlayOpportunities	Number of power play opportunities the team had in the game
powerPlayGoals	Number of goals scored while the team was on a power play
faceOffWinPercentage	Percentage of faceoffs won by the team during the game
giveaways	Number of times the team gave the puck away to the opposing team
takeaways	Number of times the team took the puck away from the opposing team
blocked	Number of shots blocked by the team
startRinkSide	The side of the rink the team started on

The data dictionary for **team_stats.csv** was created based on research and interpretation of the dataset's variables. It provides detailed explanations of each column, such as **game_id** and **team_id**, which uniquely identify games and teams, and **HoA**, which indicates whether the team played at home or away. Key performance metrics like **goals**, **shots**, **hits**, and **blocked** offer insights into offensive and defensive contributions, while **powerPlayGoals** and **powerPlayOpportunities** focus on special team performance. Contextual columns like **startRinkSide** and **settled_in** provide additional game-specific information.

This dictionary will serve as a guide for analyzing team performance and game outcomes.

Dataset - player_info.csv

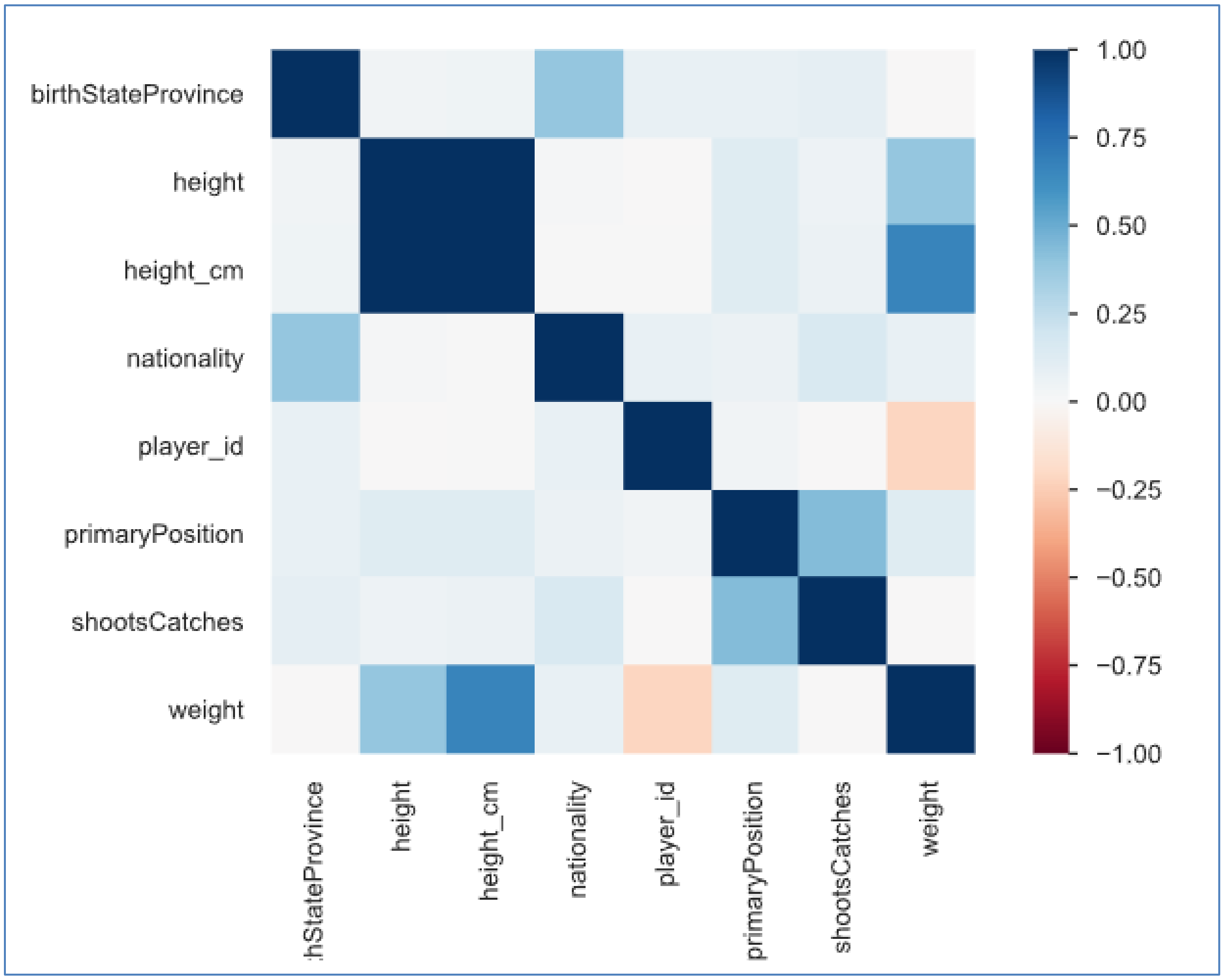


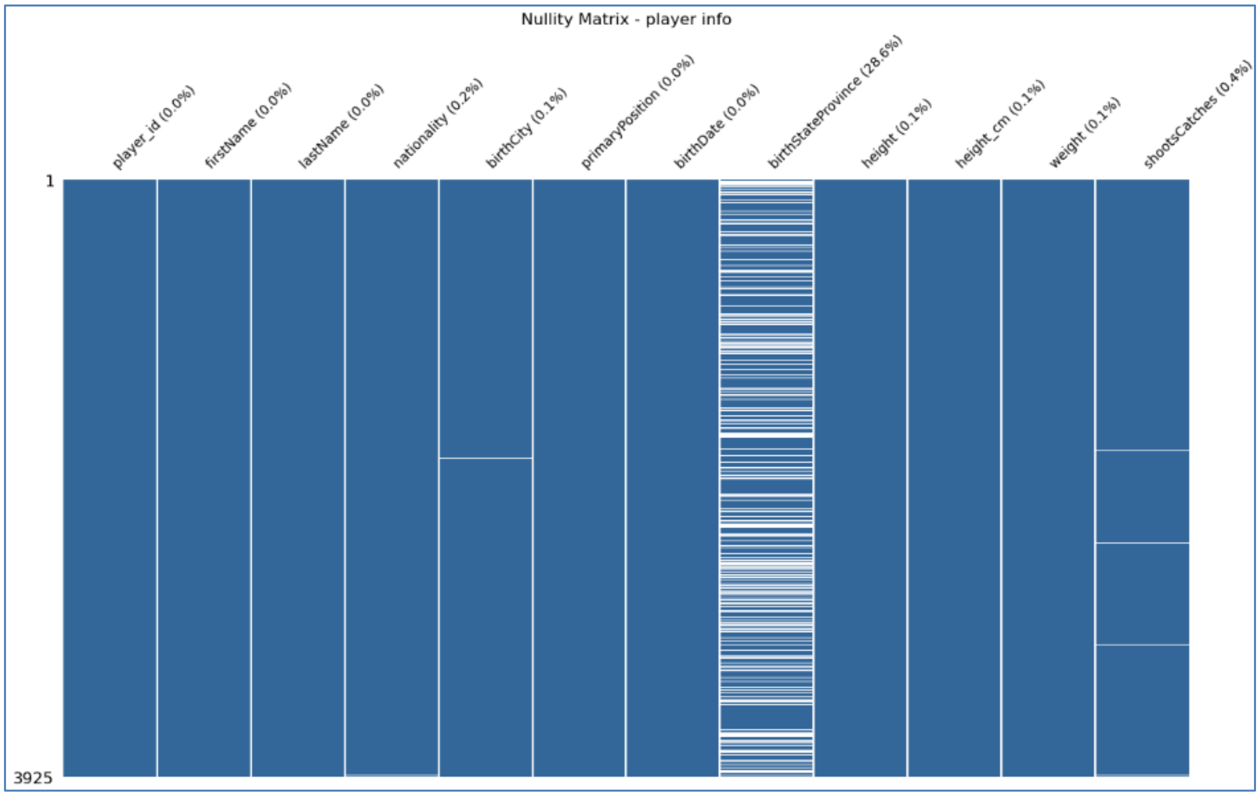
Dataset statistics

Number of variables	12
Number of observations	3925
Missing cells	1162
Missing cells (%)	2.5%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	368.1 KiB
Average record size in memory	96.0 B

Variable types

Numeric	3
Text	3
Categorical	5
DateTime	1





The player_info.csv dataset contains **12 variables** and **3,925 observations**, providing key demographic and physical information about hockey players.

There are **1,162 missing cells**, which account for **2.5%** of the data. This low percentage indicates relatively high completeness, with minimal effort required for handling missing values.

The dataset contains **0 duplicate rows**, ensuring data integrity without the need for deduplication.

The dataset contains **3 numeric variables**, such as **height_cm** and **weight**, which describe the physical attributes of players. These variables provide quantitative measures of player characteristics, essential for analysing trends related to size and performance.

Additionally, there are **3 text variables**, including **birthStateProvince** and **nationality**, which offer descriptive information about players' geographic origins and cultural backgrounds. These variables are valuable for demographic analysis.

The dataset also includes **5 categorical variables**, such as **primaryPosition** and **shootsCatches**, which classify players based on their role on the ice and their shooting or catching preferences. These categories are key for understanding player specialization and style.

Lastly, there is **1 DateTime variable**, likely capturing time-related data such as birthdates or career milestones. This variable provides temporal context for age-related or career progression analyses.

The heatmap reveals strong positive correlation between **height_cm** and **weight**, as expected in physical attributes.

It also shows positive relationships between **primaryPosition** and physical attributes like **weight** and **height_cm**. This may provide insights into position-specific requirements.

Key Columns	Description
player_id	A unique identifier for each player
firstName	The first name of the player
lastName	The last name of the player
nationality	The player's nationality
birthCity	The city where the player was born
primaryPosition	The player's primary position on the team
birthDate	The date of birth of the player
birthStateProvince	The state or province of the birth location
height	The height of the player in feet and inches format
height_cm	The player's height converted to centimetres
weight	The player's weight in pounds
shootsCatches	Indicates the player's shooting or catching hand

The data dictionary for **player_info.csv** was created based on research and inferred meanings of the dataset's variables. It provides detailed descriptions of each column, including identifiers like **player_id**, demographic details such as nationality, **birthCity**, and **birthStateProvince**, and physical attributes like **height**, **height_cm**, and **weight**. The dataset also includes player-specific details such as **primaryPosition** and shoots Catches, which describe their role and preferences on the ice.

This dictionary will offer a comprehensive understanding of the dataset, supporting player profiling and analysis in hockey-related contexts.

ETL Process

Data Extraction Using KaggleApi

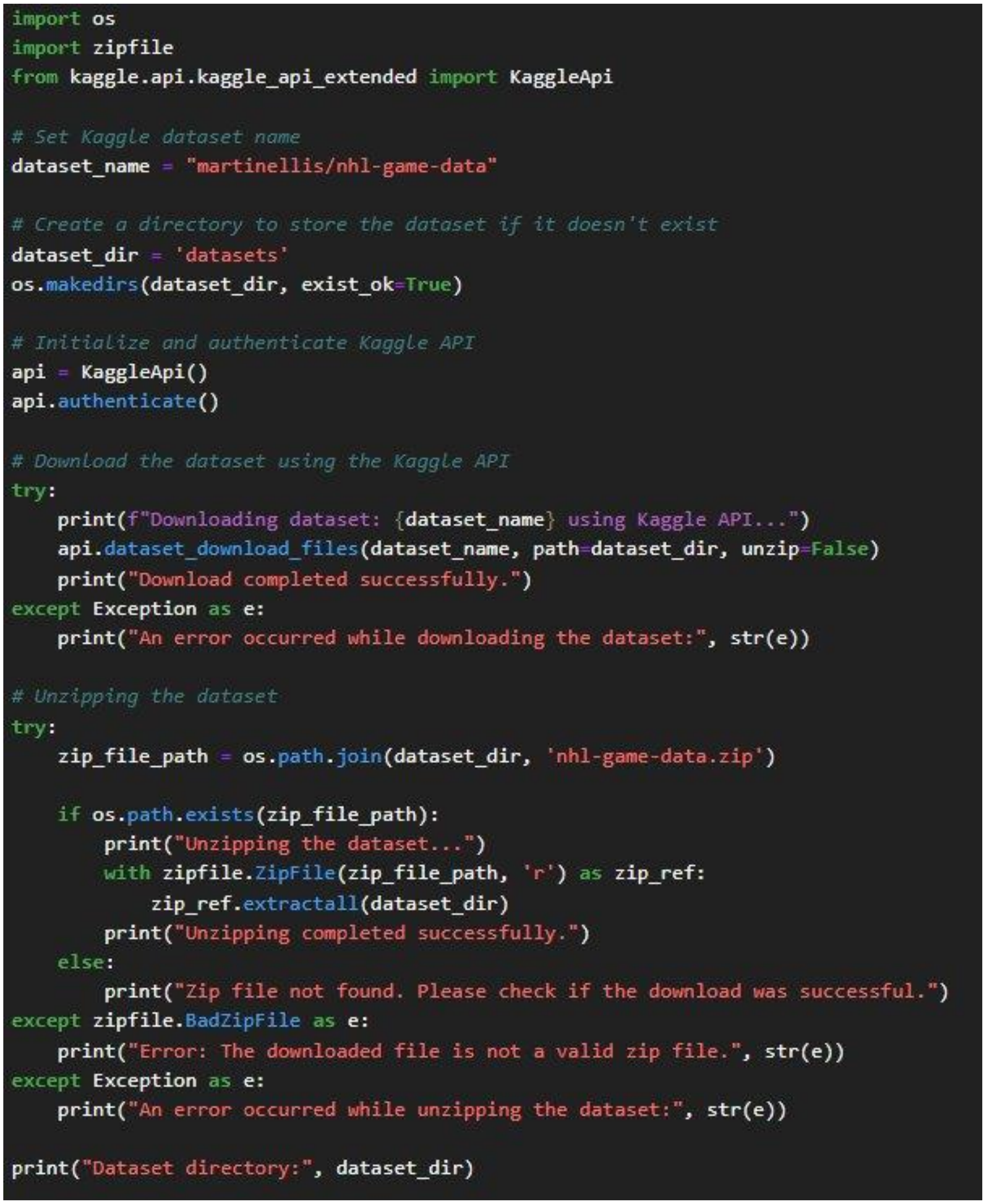
The team employs an **ETL (Extract, Transform, Load)** process to efficiently manage and process data for the project. This approach ensures seamless data integration, transformation, and loading into the target system for analysis and reporting.

The extraction phase retrieves NHL game-related data from Kaggle using the Kaggle API using a Python script (See below). The dataset is stored as CSV files for further processing.

Implementation Details

- Kaggle API Authentication:
  - The script imports the necessary libraries, including **KaggleApi** from **kaggle.api.kaggle_api_extended**, to authenticate and download the dataset.
  - The authentication process requires a **kaggle.json** file containing the user's Kaggle API key.
- Dataset Download and Storage:
  - The script sets up a directory (datasets/) to store the downloaded dataset.
  - The **api.dataset_download_files()** function fetches the dataset (**martinellis/nhl-game-data**) and stores it in a compressed format.
- Unzipping the Dataset:
  - The script checks for the downloaded ZIP file and extracts its contents using the **zipfile.ZipFile** module.
- Error Handling:
  - The script implements exception handling (**try-except** statements) to manage potential download and extraction issues.
  - It verifies the integrity of the downloaded file before proceeding with extraction.

This ensures efficient and structured extraction of NHL game datasets for further data transformation and analysis.



```python
import os
import zipfile
from kaggle.api.kaggle_api_extended import KaggleApi

# Set Kaggle dataset name
dataset_name = "martinellis/nhl-game-data"

# Create a directory to store the dataset if it doesn't exist
dataset_dir = 'datasets'
os.makedirs(dataset_dir, exist_ok=True)

# Initialize and authenticate Kaggle API
api = KaggleApi()
api.authenticate()

# Download the dataset using the Kaggle API
try:
    print(f"Downloading dataset: {dataset_name} using Kaggle API...")
    api.dataset_download_files(dataset_name, path=dataset_dir, unzip=False)
    print("Download completed successfully.")
except Exception as e:
    print("An error occurred while downloading the dataset:", str(e))

# Unzipping the dataset
try:
    zip_file_path = os.path.join(dataset_dir, 'nhl-game-data.zip')

    if os.path.exists(zip_file_path):
        print("Unzipping the dataset...")
        with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
            zip_ref.extractall(dataset_dir)
        print("Unzipping completed successfully.")
    else:
        print("Zip file not found. Please check if the download was successful.")
except zipfile.BadZipFile as e:
    print("Error: The downloaded file is not a valid zip file.", str(e))
except Exception as e:
    print("An error occurred while unzipping the dataset:", str(e))

print("Dataset directory:", dataset_dir)
```

As part of the **ETL process**, the team verifies the integrity of the downloaded dataset to ensure accuracy and consistency. This is done using **MD5Checker**, which compares the **MD5 hash values** of the dataset obtained through two different download methods:

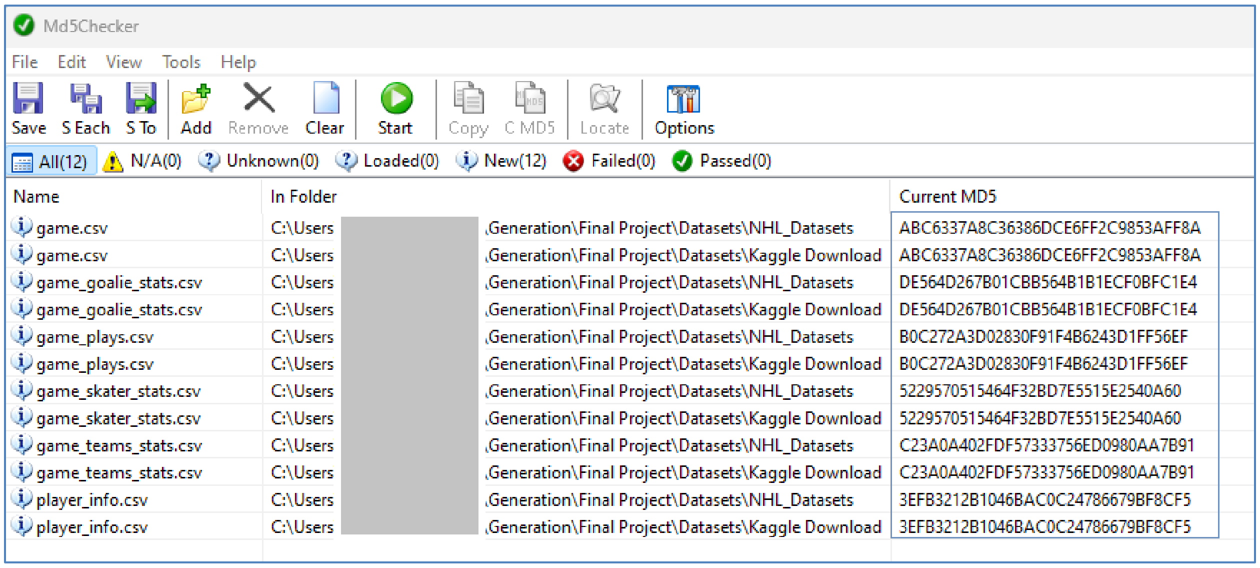
- **Direct Download from Kaggle Website**
- **Download via Python Script Using Kaggle API**

MD5 Hash Validation Process

- **MD5 Hash Computation:**
    - The **MD5 hashing function** generates a **128-bit unique hash** for each file.
    - Even a **1-byte change** in a file results in a **completely different hash value**.
    - This helps detect any corruption, truncation, or modification during file transfers.
- **Comparison of MD5 Hashes:**
    - After downloading, the **MD5 hashes of both datasets** (Kaggle direct download vs. Python script) are computed.
    - The hashes **must match** to confirm file integrity.
    - Any mismatch indicates possible errors in download, extraction, or storage.

Verification Results

- The dataset was validated using **MD5Checker**, as shown in the screenshot.
- The **MD5 hashes of the files from both sources matched**, confirming the integrity and correctness of the data.
- The verified dataset was then **uploaded to Azure Data Lake** for further transformation and analysis.

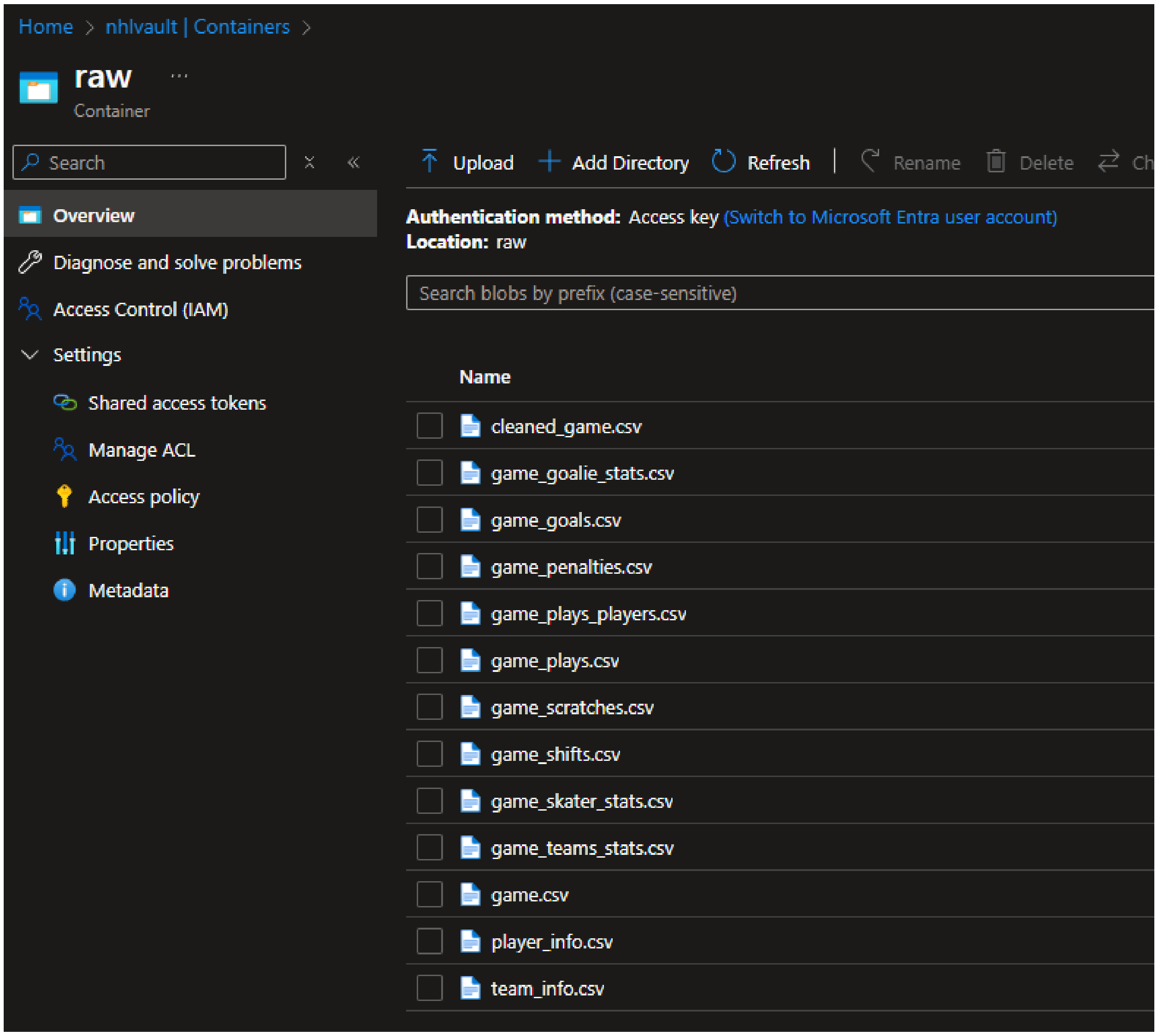

Md5Checker
File Edit View Tools Help
Save S Each S To Add Remove Clear Start Copy C MD5 Locate Options
All(12) N/A(0) Unknown(0) Loaded(0) New(12) Failed(0) Passed(0)

Name	In Folder	Current MD5
game.csv	C:\Users [redacted] \Generation\Final Project\Datasets\NHL_Datasets	ABC6337A8C36386DCE6FF2C9853AFF8A
game.csv	C:\Users [redacted] \Generation\Final Project\Datasets\Kaggle Download	ABC6337A8C36386DCE6FF2C9853AFF8A
game_goalie_stats.csv	C:\Users [redacted] \Generation\Final Project\Datasets\NHL_Datasets	DE564D267B01CBB564B1B1ECF0BFC1E4
game_goalie_stats.csv	C:\Users [redacted] \Generation\Final Project\Datasets\Kaggle Download	DE564D267B01CBB564B1B1ECF0BFC1E4
game_plays.csv	C:\Users [redacted] \Generation\Final Project\Datasets\NHL_Datasets	B0C272A3D02830F91F4B6243D1FF56EF
game_plays.csv	C:\Users [redacted] \Generation\Final Project\Datasets\Kaggle Download	B0C272A3D02830F91F4B6243D1FF56EF
game_skater_stats.csv	C:\Users [redacted] \Generation\Final Project\Datasets\NHL_Datasets	5229570515464F32BD7E5515E2540A60
game_skater_stats.csv	C:\Users [redacted] \Generation\Final Project\Datasets\Kaggle Download	5229570515464F32BD7E5515E2540A60
game_teams_stats.csv	C:\Users [redacted] \Generation\Final Project\Datasets\NHL_Datasets	C23A0A402FDF57333756ED0980AA7B91
game_teams_stats.csv	C:\Users [redacted] \Generation\Final Project\Datasets\Kaggle Download	C23A0A402FDF57333756ED0980AA7B91
player_info.csv	C:\Users [redacted] \Generation\Final Project\Datasets\NHL_Datasets	3EFB3212B1046BAC0C24786679BF8CF5
player_info.csv	C:\Users [redacted] \Generation\Final Project\Datasets\Kaggle Download	3EFB3212B1046BAC0C24786679BF8CF5

By implementing **MD5 hash validation**, the team ensures that the extracted data remains **unchanged, accurate, and reliable** throughout the ETL process.

After the **extraction phase**, the raw NHL game data is **manually uploaded** to **Azure Data Lake Storage Gen 2** (**raw** container) for further processing in the ETL pipeline. This storage solution provides a secure, scalable, and high-performance environment for managing structured and unstructured data.

The Azure portal is used to navigate to the **nhlvault storage account** and access the **raw container**. The **"Upload"** option is used to transfer files from the local system to the cloud.



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

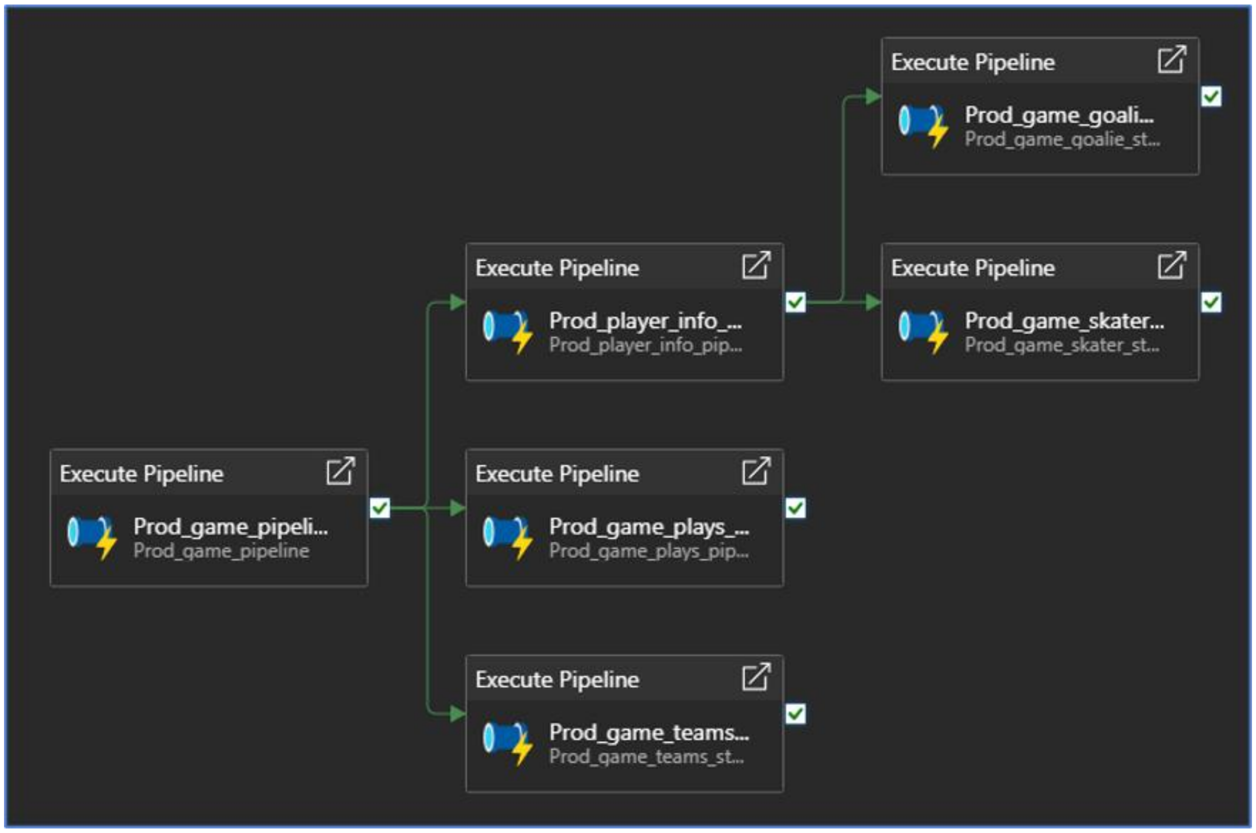
Search blobs by prefix (case-sensitive)

Name
cleaned_game.csv
game_goalie_stats.csv
game_goals.csv
game_penalties.csv
game_plays_players.csv
game_plays.csv
game_scratches.csv
game_shifts.csv
game_skater_stats.csv
game_teams_stats.csv
game.csv
player_info.csv
team_info.csv

The **raw data** is now accessible in the **raw container**, as shown in the screenshot.

Data Transformation Using Azure Data Flows

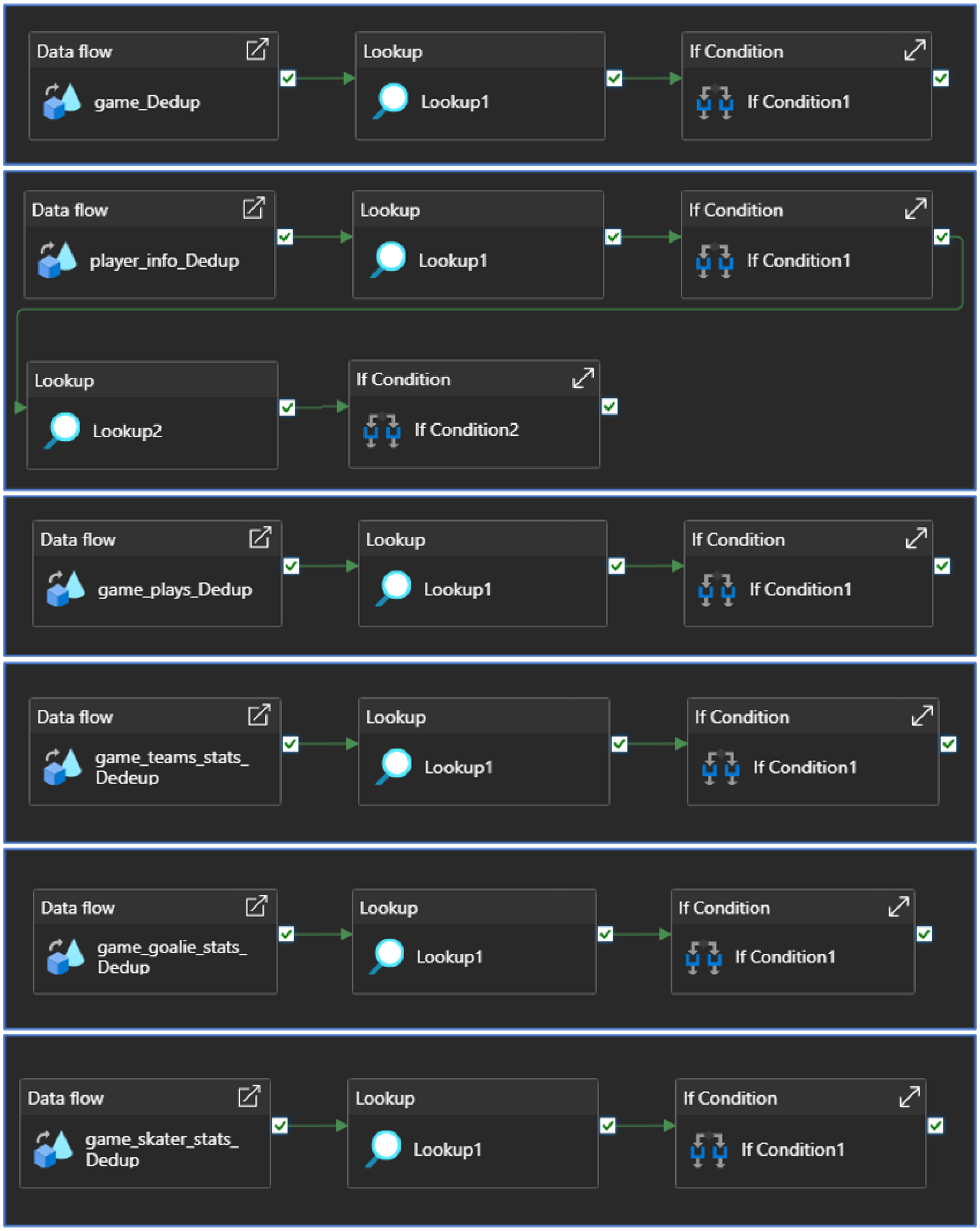
The team leverages **Azure Data Factory (ADF) Data Flow** as part of the **ETL process** to transform raw NHL game data into a structured format suitable for analysis and reporting. The transformation is orchestrated through a series of **Azure Data Factory pipelines**, as shown in the attached pipeline flow.

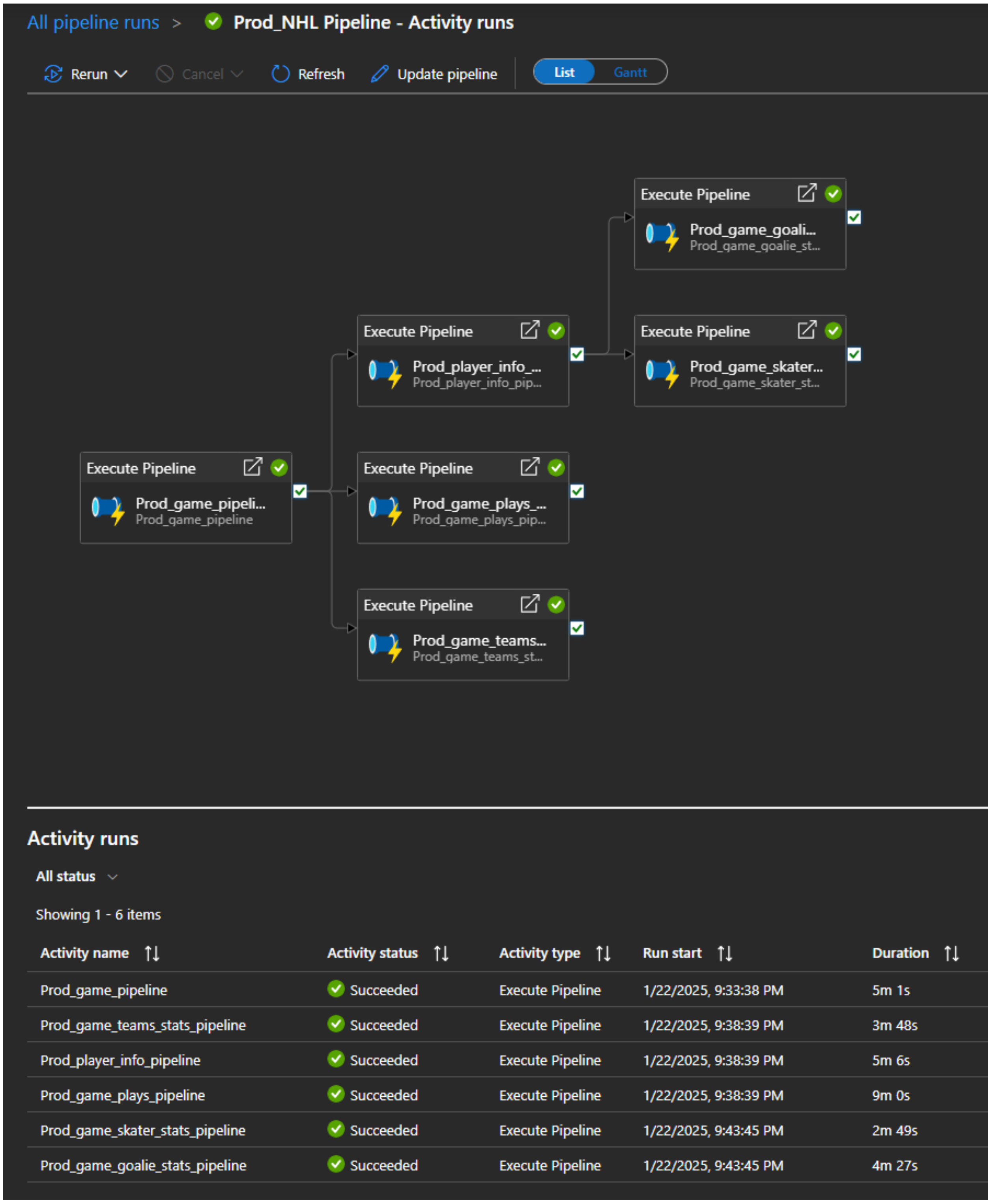


The **main execution pipeline (Prod_game_pipeline)** triggers multiple downstream pipelines for processing different aspects of the game data:

1. **Prod_game_pipeline (Main Pipeline)**
   - This serves as the **master pipeline** that initiates the entire data transformation process.
   - It executes multiple **sub-pipelines** that transform specific datasets.
2. **Sub-Pipelines for Data Transformation**
   - **Prod_player_info_pipeline –** Processes player information, ensuring clean and structured player metadata.
   - **Prod_game_plays_pipeline –** Handles game play-by-play data, applying transformations to standardize events and actions.
   - **Prod_game_teams_stats_pipeline –** Aggregates team-based statistics for analytical use.
   - **Prod_game_goalie_stats_pipeline –** Cleans and transforms goalie-related performance metrics.

- **Prod_game_skater_stats_pipeline –** Processes statistics related to skaters, such as shots, assists, and penalties.





Activity runs

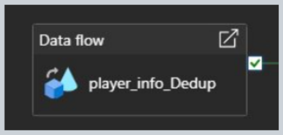
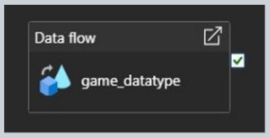
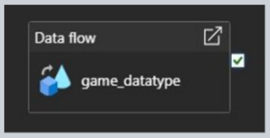
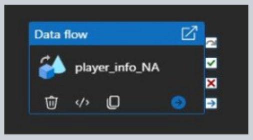
All status

Showing 1 - 6 items

Activity name	Activity status	Activity type	Run start	Duration
Prod_game_pipeline	✓ Succeeded	Execute Pipeline	1/22/2025, 9:33:38 PM	5m 1s
Prod_game_teams_stats_pipeline	✓ Succeeded	Execute Pipeline	1/22/2025, 9:38:39 PM	3m 48s
Prod_player_info_pipeline	✓ Succeeded	Execute Pipeline	1/22/2025, 9:38:39 PM	5m 6s
Prod_game_plays_pipeline	✓ Succeeded	Execute Pipeline	1/22/2025, 9:38:39 PM	9m 0s
Prod_game_skater_stats_pipeline	✓ Succeeded	Execute Pipeline	1/22/2025, 9:43:45 PM	2m 49s
Prod_game_goalie_stats_pipeline	✓ Succeeded	Execute Pipeline	1/22/2025, 9:43:45 PM	4m 27s

The **Prod_NHL Pipeline** successfully executed all six sub-pipelines, including game, player, team, and stats-related processes, with completion times ranging from 2m 49s to 9m 0s.

Each of these pipelines performs **deduplication, missing/null values treatment, and datatype conversion** before loading the transformed data into **Azure SQL Database**.

Deduplication	Imputation	Datatype Conversion and Derived Column	Drop NaN Rows
			
• This data flow is specifically designed to identify and remove duplicate records from the dataset. • This is the **FIRST** flow in the pipeline, ensuring that all subsequent processes work with a dataset with no duplicates.	• This data flow focuses on managing null or missing values in the dataset using imputation technique. • By imputing missing values, this data flow ensures that gaps in the data are filled.	• The same data flow used for imputation also manages datatype conversion and derived new column fields. • This data flow ensures consistent data formats with Azure SQL database	• This data flow is specifically designed to drop rows that contain NaN or missing values which have minimal impact to downstream analysis. • Depending on use case, this data flow does not apply to all datasets.

Data Cleaning in Azure Data Factory

To ensure high-quality and reliable data**, data cleaning and transformation** are key processes within the ETL pipeline. The team leverages **Azure Data Factory Data Flows** to apply multiple data cleaning techniques, including:

1. Deduplication

- Identifies and removes **duplicate records** to ensure data integrity.
- This is the **first step** in the data flow, ensuring all subsequent transformations work with unique data.

2. Imputation (Handling Missing Data)

- Addresses missing or null values using imputation techniques.
- This step ensures that gaps in the dataset are filled, improving completeness.

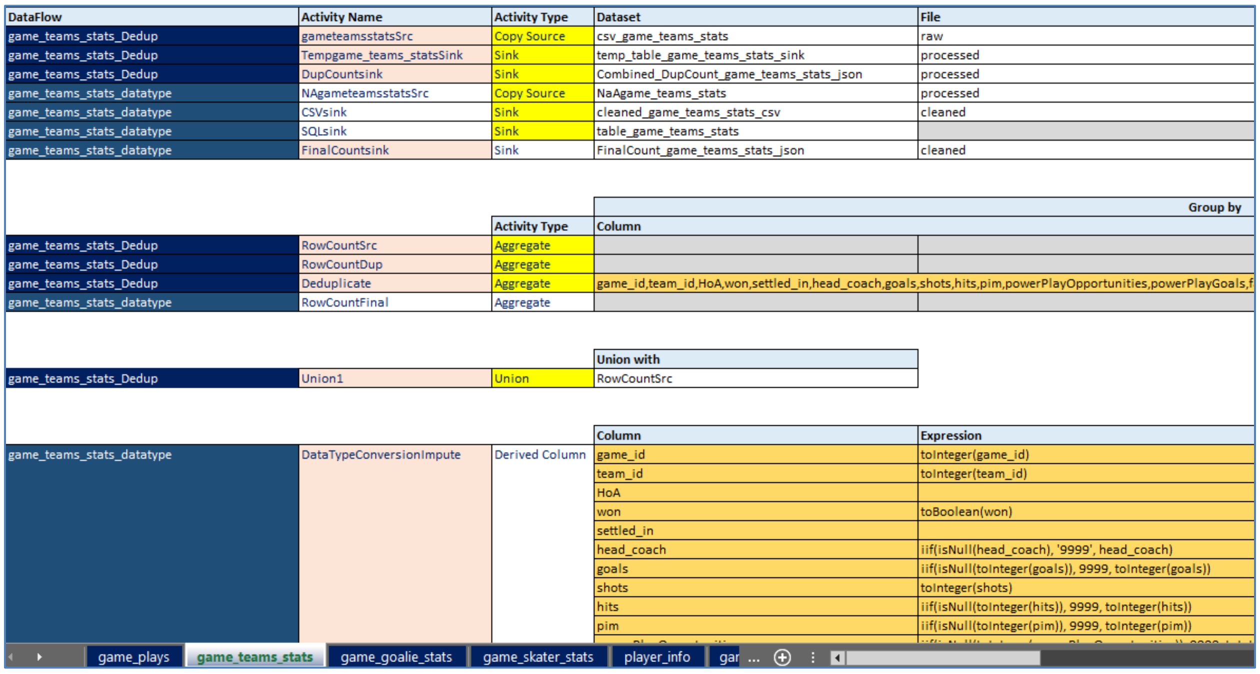
3. Datatype Conversion & Derived Columns

- Standardizes data types to match database requirements.
- Generates new calculated columns for enhanced data insights.
- Ensures consistency with Azure SQL database formats.

4. Dropping NaN Rows

- Removes rows containing NaN (null) values when necessary.
- Applied selectively to avoid removing essential data.
- Helps maintain data quality while preserving analytical value.

Data Flow Dictionary



DataFlow	Activity Name	Activity Type	Dataset	File
game_teams_stats_Dedup	gameteamsstatsSrc	Copy Source	csv_game_teams_stats	raw
game_teams_stats_Dedup	Tempgame_teams_statsSink	Sink	temp_table_game_teams_stats_sink	processed
game_teams_stats_Dedup	DupCountsink	Sink	Combined_DupCount_game_teams_stats_json	processed
game_teams_stats_datatype	NAgameteamsstatsSrc	Copy Source	NaAgame_teams_stats	processed
game_teams_stats_datatype	CSVsink	Sink	cleaned_game_teams_stats_csv	cleaned
game_teams_stats_datatype	SQLsink	Sink	table_game_teams_stats	
game_teams_stats_datatype	FinalCountsink	Sink	FinalCount_game_teams_stats_json	cleaned

			Group by	
		Activity Type	Column	
game_teams_stats_Dedup	RowCountSrc	Aggregate		
game_teams_stats_Dedup	RowCountDup	Aggregate		
game_teams_stats_Dedup	Deduplicate	Aggregate	game_id,team_id,HoA,won,settled_in,head_coach,goals,shots,hits,pim,powerPlayOpportunities,powerPlayGoals,f	
game_teams_stats_datatype	RowCountFinal	Aggregate		

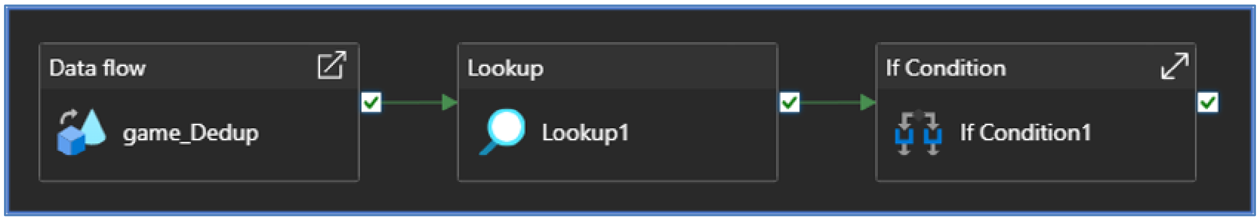
			Union with
game_teams_stats_Dedup	Union1	Union	RowCountSrc

			Column	Expression
game_teams_stats_datatype	DataTypeConversionImpute	Derived Column	game_id	toInteger(game_id)
			team_id	toInteger(team_id)
			HoA	
			won	toBoolean(won)
			settled_in	
			head_coach	iif(isNull(head_coach), '9999', head_coach)
			goals	iif(isNull(toInteger(goals)), 9999, toInteger(goals))
			shots	toInteger(shots)
			hits	iif(isNull(toInteger(hits)), 9999, toInteger(hits))
			pim	iif(isNull(toInteger(pim)), 9999, toInteger(pim))

The **Data Flow Dictionary** serves as a comprehensive reference guide for understanding the structure, transformations, and processing steps applied to all the datasets within the ETL pipeline in **Azure Data Factory (ADF)**. It documents the various activities, data sources, transformations, and final outputs to ensure transparency, consistency, and data integrity.

This document can be found in Appendix.

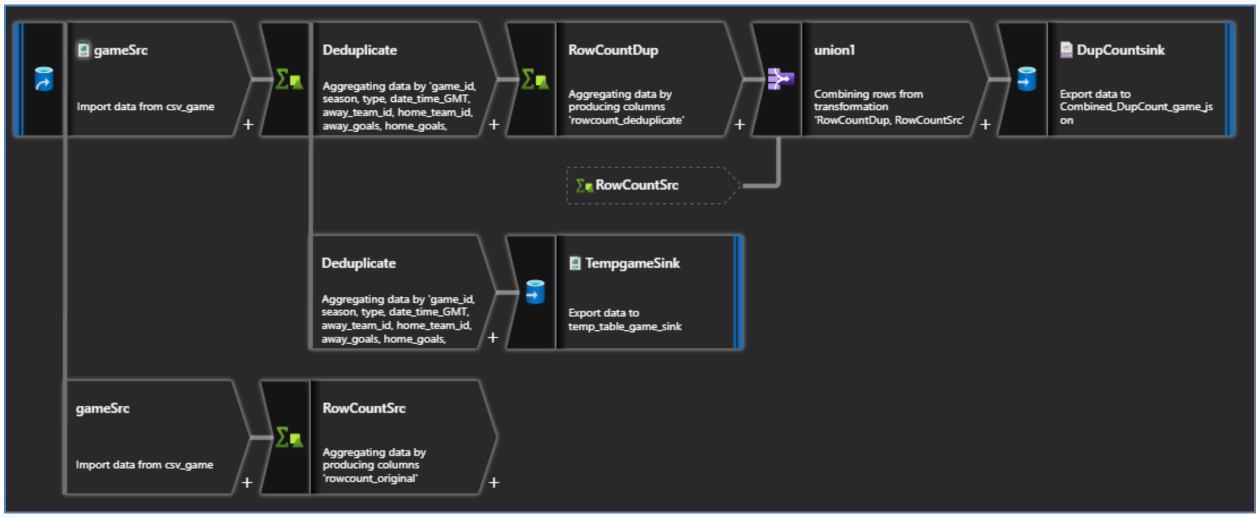
Shared Data Flow Design



The **Prod_game_pipeline**, **Prod_game_plays_pipeline**, **Prod_game_teams_stats_pipeline** all share similar main data flow designs comprising of:

A. **<dataset name>_Dedup**
B. **Lookup1**
C. **If Condition 1**

A. <dataset name>_Dedup Dataflow


gameSrc
Import data from csv_game
Deduplicate
Aggregating data by 'game_id, season, type, date_time_GMT, away_team_id, home_team_id, away_goals, home_goals,
RowCountDup
Aggregating data by producing columns 'rowcount_deduplicate'
union1
Combining rows from transformation 'RowCountDup, RowCountSrc'
DupCountsink
Export data to Combined_DupCount_game_json
RowCountSrc
Deduplicate
Aggregating data by 'game_id, season, type, date_time_GMT, away_team_id, home_team_id, away_goals, home_goals,
TempgameSink
Export data to temp_table_game_sink
gameSrc
Import data from csv_game
RowCountSrc
Aggregating data by producing columns 'rowcount_original'

This dataflow is designed to process **game data** by performing deduplication, row count validation, and data export for further analysis.

Step-by-Step Breakdown of the Data Flow

1.**<dataset name>Src (Source Data Import)**

- Imports raw game data from the **CSV file** stored in Azure Data Lake.

2. **Deduplication**

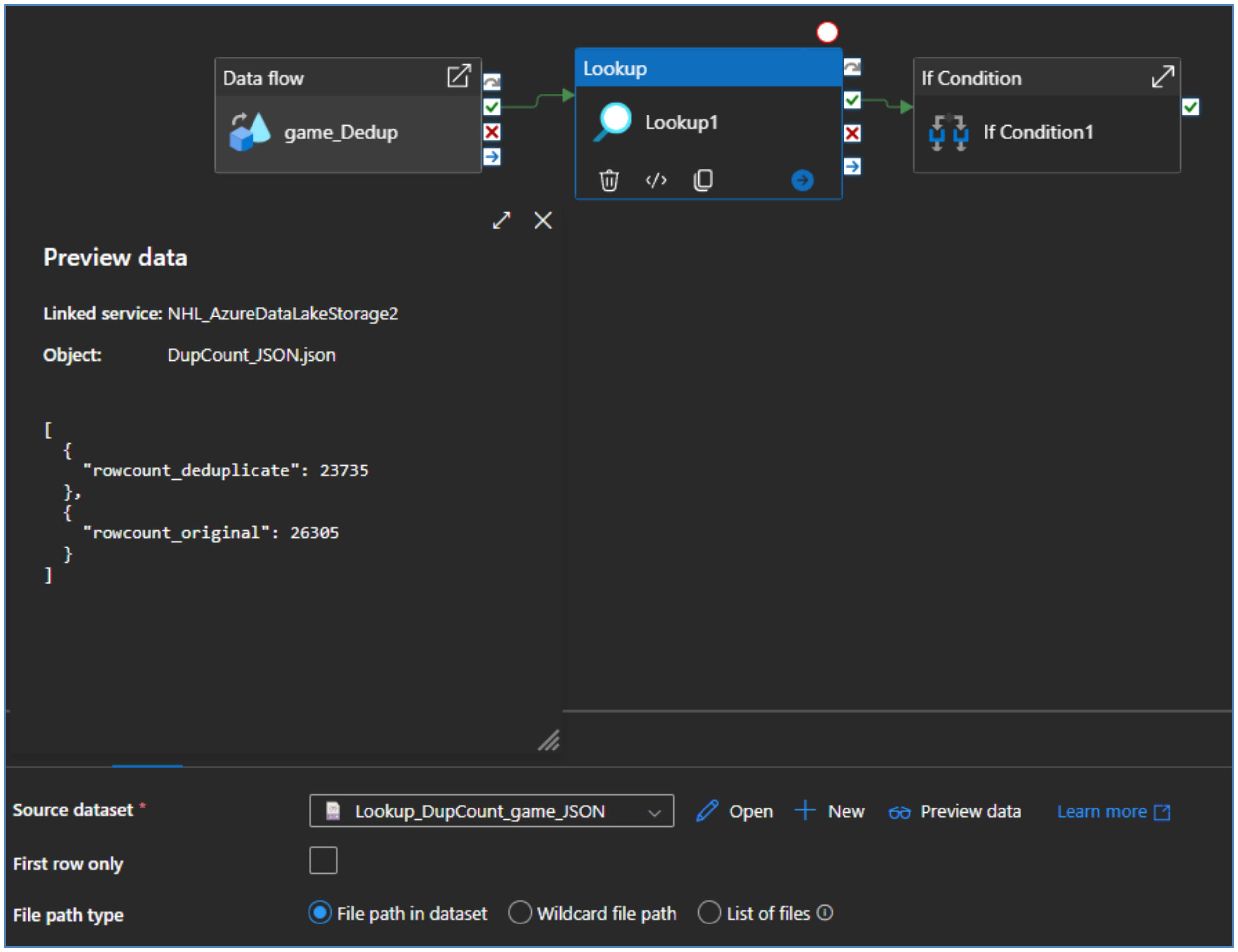
- **Deduplicate transformations** are applied to remove duplicate records based on key attributes.
- Aggregation ensures that only **unique game records** are retained.

3. **Row Count Validation**

- **RowCountDup**: Computes the count of duplicate records removed.
- **RowCountSrc**: Computes the original row count before deduplication.
- These metrics help in tracking data integrity by identifying the number of duplicates.

4. **Union Transformation (union1)**
   - Combines the results from **RowCountDup** and **RowCountSrc** into a unified dataset.
   - This provides a consolidated view of original vs. deduplicated row counts for validation.
5. **Data Export (DupCountSink)**
   - The combined row count results from union1 are exported to a **JSON file (DupCount_JSON.json)** for logging and verification.
6. **Temporary Table Storage (TempgameSink)**
   - The cleaned and deduplicated game data is stored in a temporary sink **(temp_table_game_sink)**.

B. Lookup1 Dataflow



Preview data

Linked service: NHL_AzureDataLakeStorage2

Object: DupCount_JSON.json

```
[
  {
    "rowcount_deduplicate": 23735
  },
  {
    "rowcount_original": 26305
  }
]
```

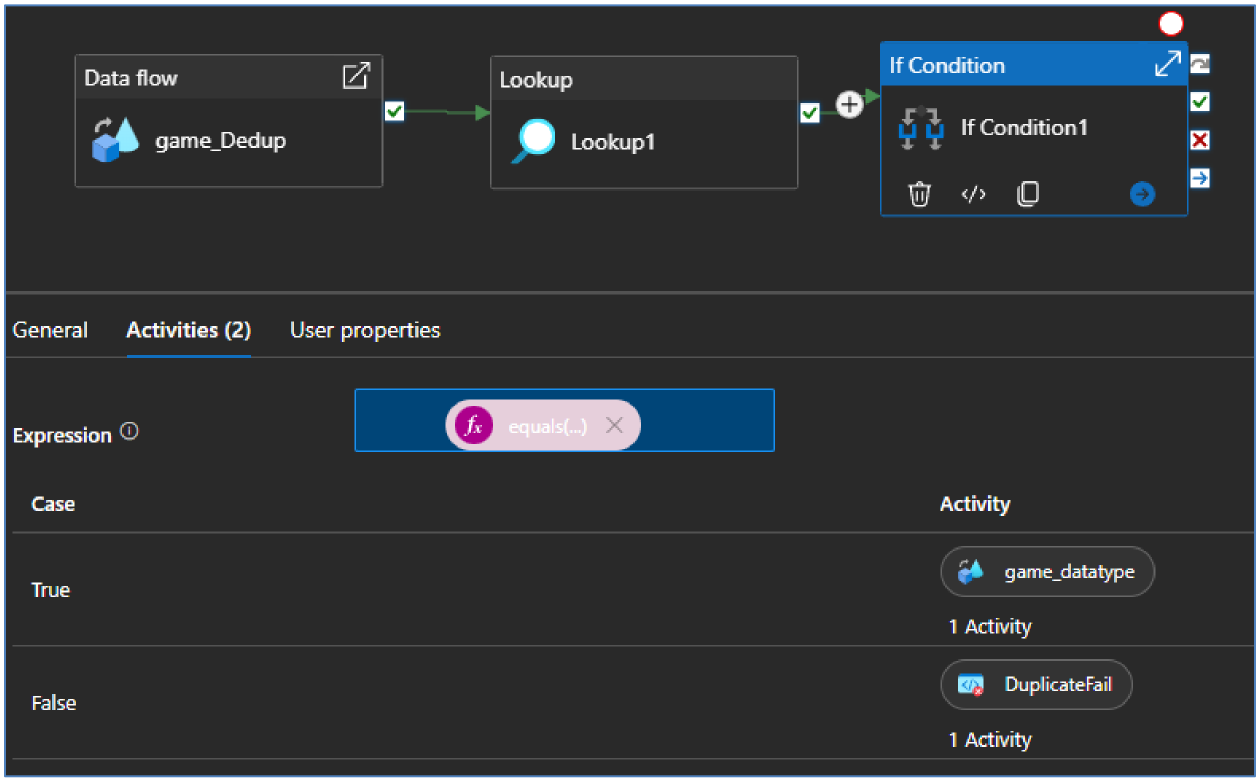
Source dataset * Lookup_DupCount_game_JSON Open New Preview data Learn more

First row only ☐

File path type [✓ File path in dataset] Wildcard file path List of files

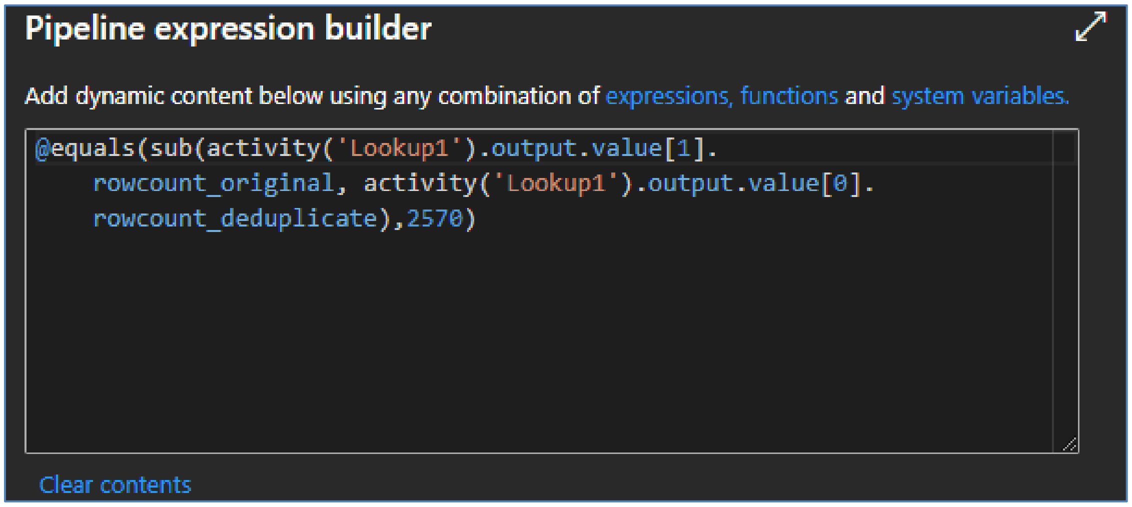
The **Lookup1** activity is designed to retrieve reference data from **DupCount_JSON.json** and passed them to the upstream **If Condition** data flow

C. If Condition1 Dataflow



Case	Activity
True	game_datatype 1 Activity
False	DuplicateFail 1 Activity

The **If Condition1** activity acts as a decision-making step based on the results from the **Lookup1** operation. It ensures that the deduplicated game data aligns with expected duplicate counts before proceeding with further transformations.


Pipeline expression builder

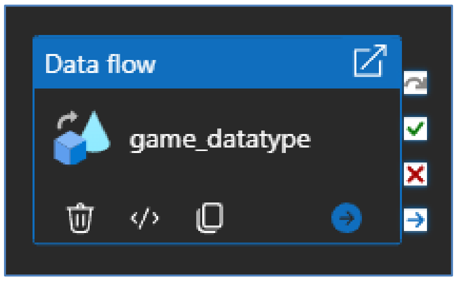
Add dynamic content below using any combination of expressions, functions and system variables.

```
@equals(sub(activity('Lookup1').output.value[1].
    rowcount_original, activity('Lookup1').output.value[0].
    rowcount_deduplicate),2570)
```

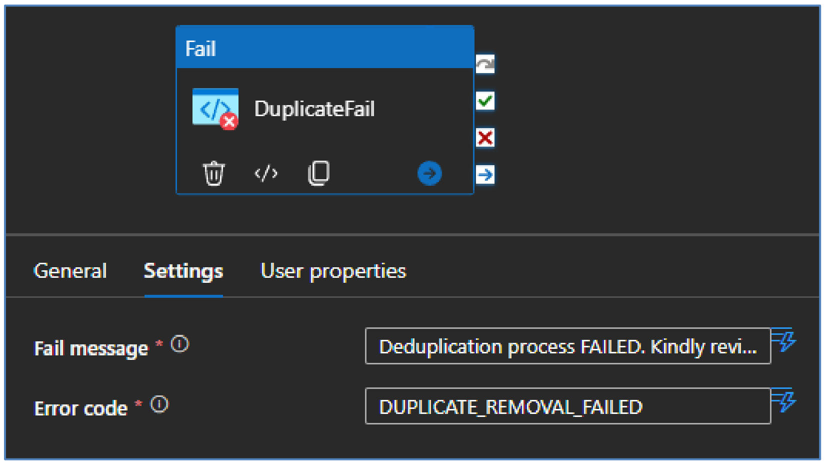
Clear contents

It uses the conditional expression defined to check if the duplicate count from **Lookup1** matches an expected value using the equals(...) function.

If the condition evaluates **TRUE**, it means the deduplication process is successful, The pipeline proceeds to the **<dataset name>_datatype** transformation, where data type conversion are applied.



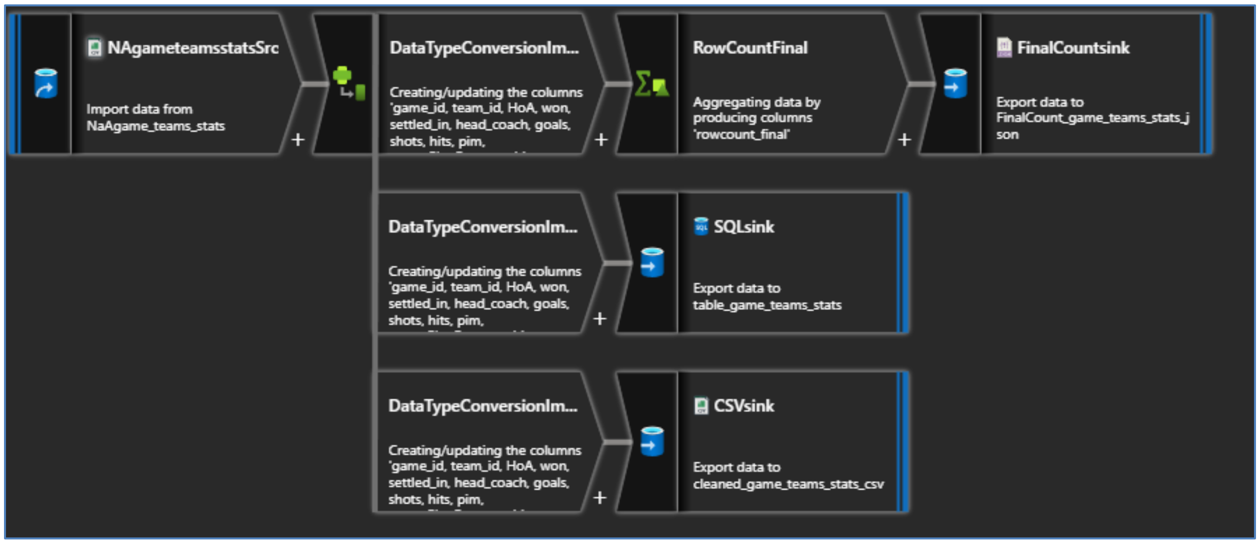
If the condition evaluates **FALSE**, The process moves to the **DuplicateFail** activity, flagging an error message.



Fail message * Deduplication process FAILED. Kindly revi...

Error code * DUPLICATE_REMOVAL_FAILED

<dataset name>_datatype Data Flow


NAgameteamsstatsSrc
Import data from NaAgame_teams_stats
DataTypeConversionIm...
Creating/updating the columns 'game_id, team_id, HoA, won, settled_in, head_coach, goals, shots, hits, pim,
RowCountFinal
Aggregating data by producing columns 'rowcount_final'
FinalCountsink
Export data to FinalCount_game_teams_stats_json
DataTypeConversionIm...
Creating/updating the columns 'game_id, team_id, HoA, won, settled_in, head_coach, goals, shots, hits, pim,
SQLsink
Export data to table_game_teams_stats
DataTypeConversionIm...
Creating/updating the columns 'game_id, team_id, HoA, won, settled_in, head_coach, goals, shots, hits, pim,
CSVsink
Export data to cleaned_game_teams_stats_csv

This data flow ensures that the dataset undergoes data type conversion and imputation (if necessary).

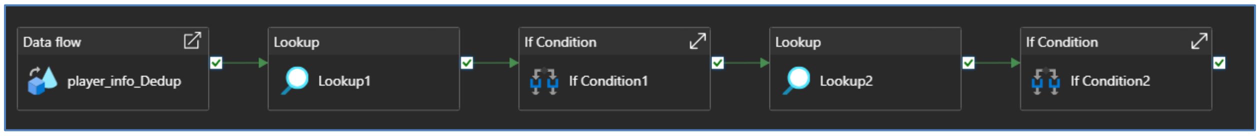
The process begins with importing the deduplicated data from the previous data flow and it undergoes data type conversion and imputation (if there are missing or NaN values). This ensures that the dataset aligns with the required schema for further processing.

To maintain data integrity, a row count aggregation step (**RowCountFinal**) is performed to validate the number of records processed. The cleaned dataset is then exported to multiple destinations, each serving a specific purpose. The **FinalCountsink** stores row count validation results in a JSON file ensuring clear tracking and verification.

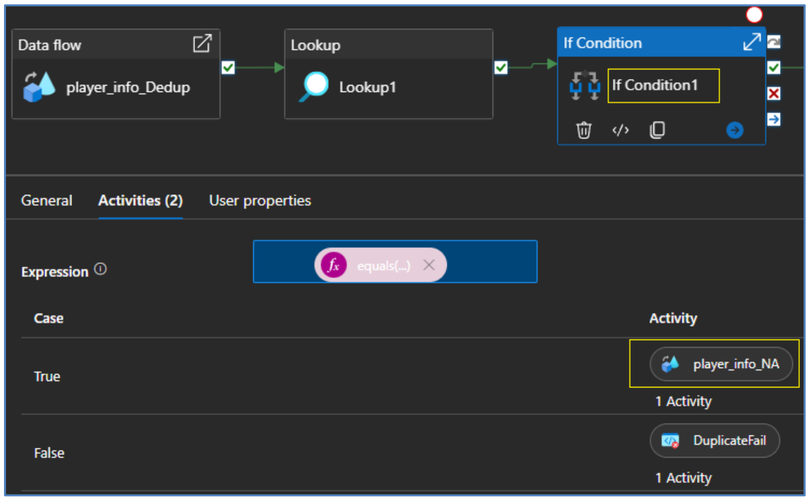
The **SQLsink** saves the transformed dataset into Azure SQL making it available for the analysis phrase

Simultaneously, the **CSVsink** exports the cleaned data as a CSV file serving as a backup in the Azure Data Lake Storage.

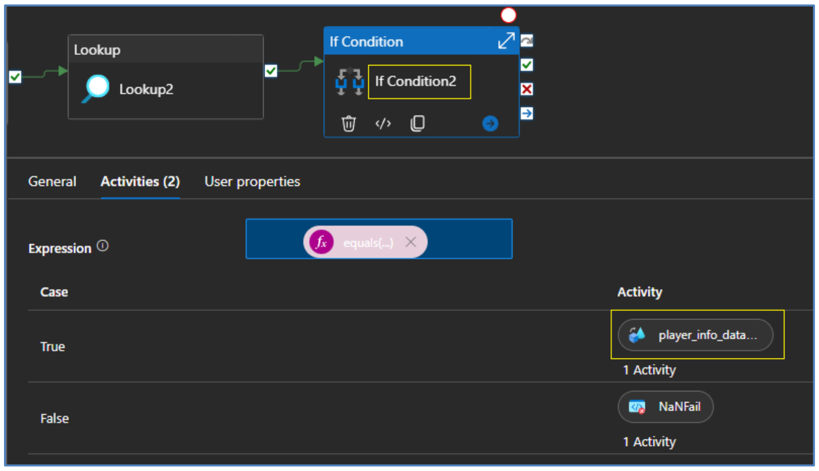
Enhanced Data Flow Design #1



This enhanced data flow is specifically designed for the **Prod_player_info** pipeline to address the need for removing NaN (null) rows from the **player_info** dataset. The process introduces a second If Condition activity to ensure a more refined data cleansing approach.

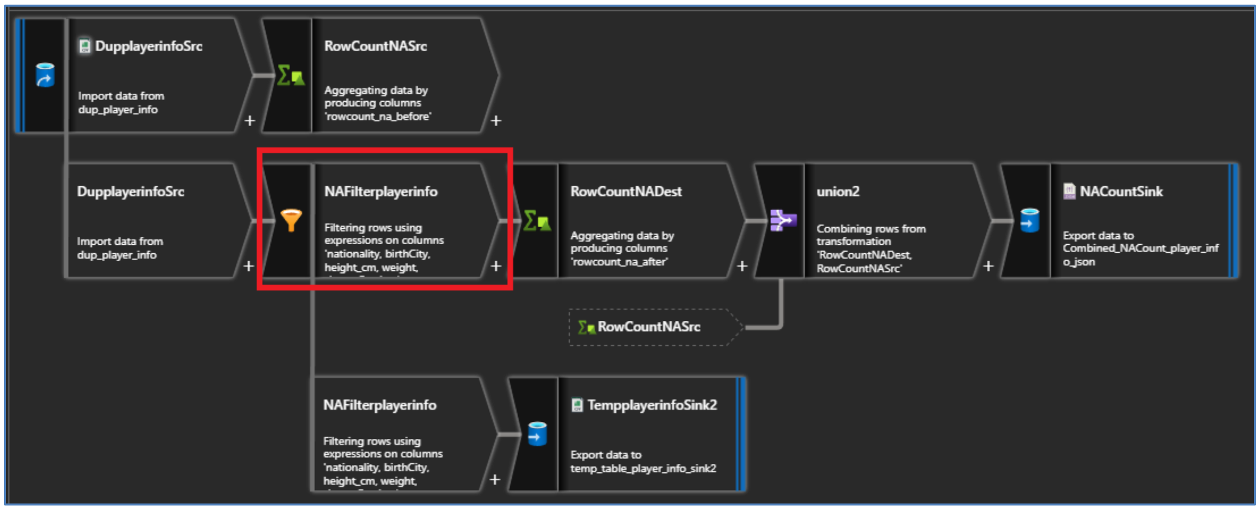


The first If Condition (**If Condition1**) evaluates whether the dataset meets initial validation checks. If true, the pipeline proceeds to the **player_info_NA** data flow, which removes records containing missing values. This ensures that only complete and reliable data moves forward. After this, a Lookup operation (Lookup2) further validates the cleaned dataset before the second If Condition (If Condition2) is applied.

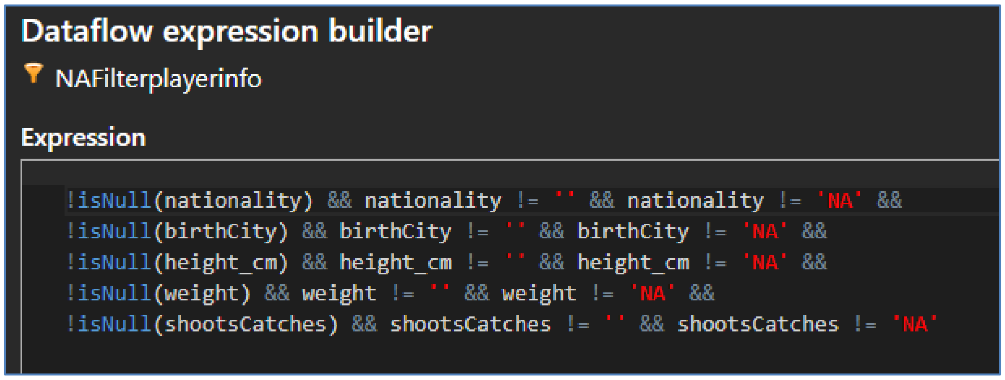


The second If Condition (**If Condition2**) performs a final verification, ensuring that all required transformations, including NaN row removal, have been successfully completed. If the dataset still contains inconsistencies, it is flagged for corrective action.

player_info_NA Data Flow


DupplayerinfoSrc
Import data from dup_player_info
RowCountNASrc
Aggregating data by producing columns 'rowcount_na_before'
DupplayerinfoSrc
Import data from dup_player_info
NAFilterplayerinfo
Filtering rows using expressions on columns 'nationality, birthCity, height_cm, weight,
RowCountNADest
Aggregating data by producing columns 'rowcount_na_after'
union2
Combining rows from transformation 'RowCountNADest, RowCountNASrc'
NACountSink
Export data to Combined_NACount_player_info_json
RowCountNASrc
NAFilterplayerinfo
Filtering rows using expressions on columns 'nationality, birthCity, height_cm, weight,
TempplayerinfoSink2
Export data to temp_table_player_info_sink2

The **NAFilterplayerinfo** activity serves as the heart of this data flow, playing a critical role in ensuring data quality by filtering out records with missing or invalid values.


Dataflow expression builder

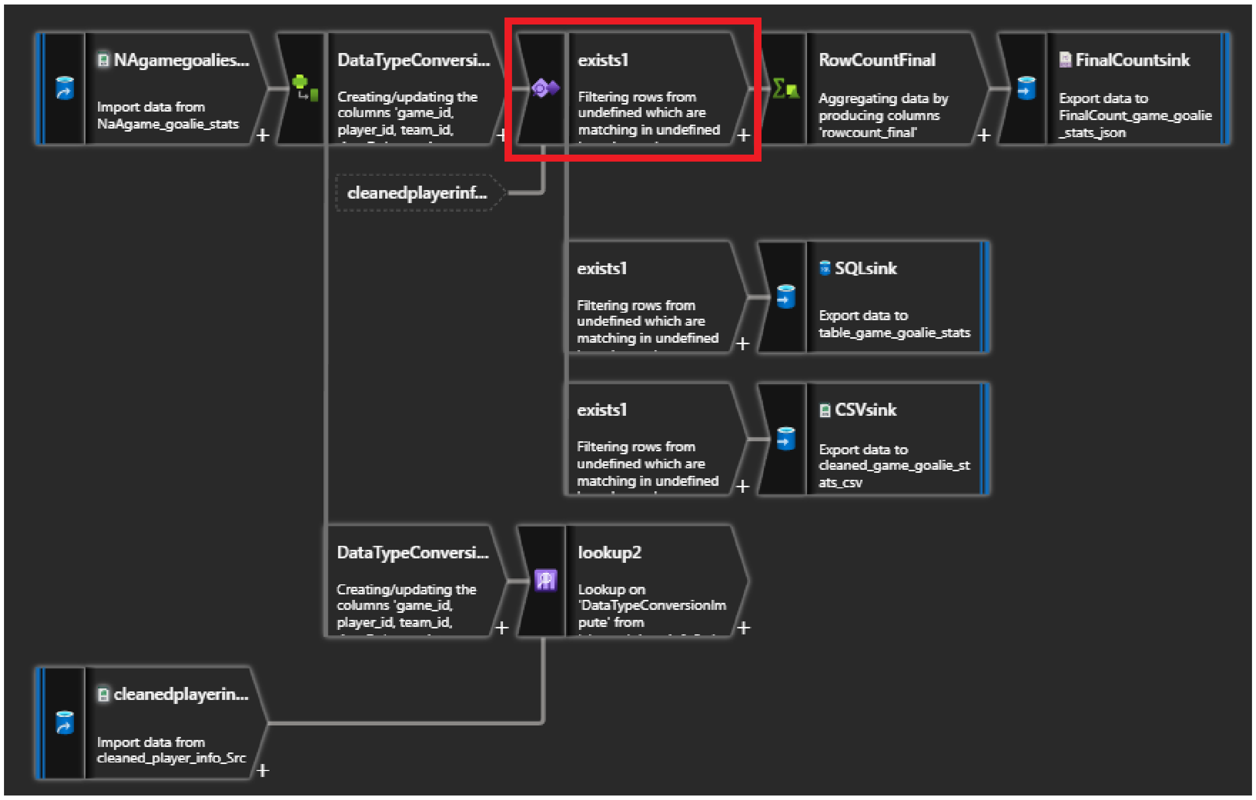
NAFilterplayerinfo

Expression

```
!isNull(nationality) && nationality != '' && nationality != 'NA' &&
!isNull(birthCity) && birthCity != '' && birthCity != 'NA' &&
!isNull(height_cm) && height_cm != '' && height_cm != 'NA' &&
!isNull(weight) && weight != '' && weight != 'NA' &&
!isNull(shootsCatches) && shootsCatches != '' && shootsCatches != 'NA'
```

At its core, **NAFilterplayerinfo** applies a filter expression to exclude rows where key attributes—**nationality**, **birthCity**, **height_cm**, **weight**, and **shootsCatches**—are either null (isNull()), empty (''), or contain the placeholder value 'NA'. The applied logic ensures that only players with valid nationality, birthplace, height, weight, and shooting preferences are retained, enhancing the accuracy and reliability of downstream data processing.

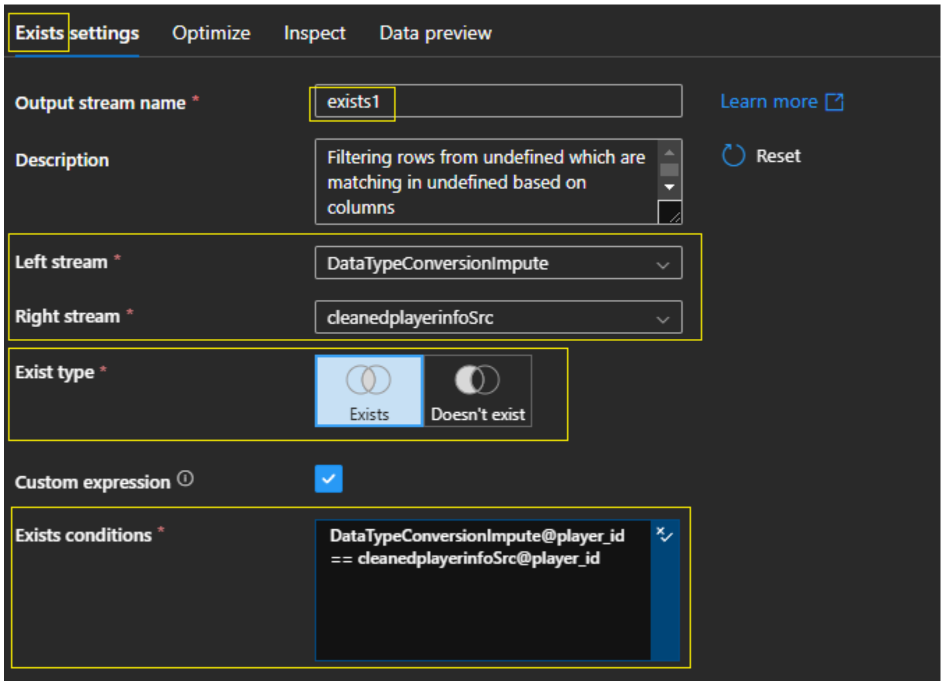
Enhanced Data Flow Design #2


NAgamegoalies...
Import data from NaAgame_goalie_stats
DataTypeConversi...
Creating/updating the columns 'game_id, player_id, team_id,
exists1
Filtering rows from undefined which are matching in undefined
RowCountFinal
Aggregating data by producing columns 'rowcount_final'
FinalCountsink
Export data to FinalCount_game_goalie_stats_json
cleanedplayerinf...
exists1
Filtering rows from undefined which are matching in undefined
SQLsink
Export data to table_game_goalie_stats
exists1
Filtering rows from undefined which are matching in undefined
CSVsink
Export data to cleaned_game_goalie_stats_csv
DataTypeConversi...
Creating/updating the columns 'game_id, player_id, team_id,
lookup2
Lookup on 'DataTypeConversionImpute' from
cleanedplayerin...
Import data from cleaned_player_info_Src

This **data flow** specifically applies to **goalie_stats** and **skater_stats** datasets because these datasets contain **player IDs that are not found in the player_info dataset**. To ensure data integrity and consistency, these unmatched records need to be **filtered out and removed** before further processing.

The pipeline begins by importing the deduplicated dataset from the downstream and applying data type conversions and imputation to standardize key fields such as **game_id**, **player_id**, and **team_id** and missing values treatment.

A filtering step (**exists1**) is used to check whether each **player_id** exists in the cleaned **player_info** dataset. If a **player_id** does not match any record in **player_info**, it is removed from further processing.



Output stream name * exists1

Description: Filtering rows from undefined which are matching in undefined based on columns

Left stream * DataTypeConversionImpute

Right stream * cleanedplayerinfoSrc

Exist type * Exists

Custom expression ✓

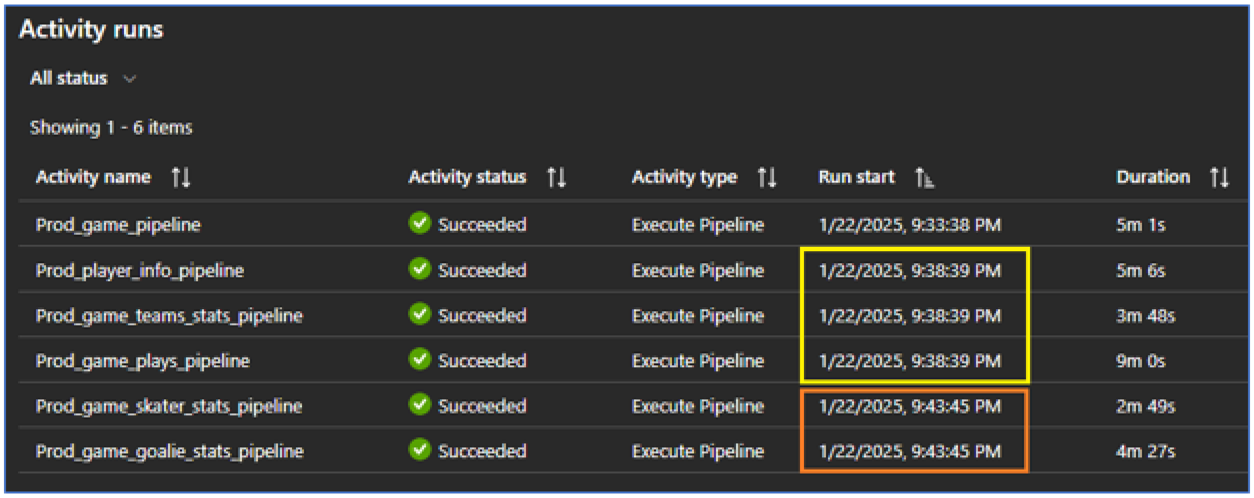
Exists conditions *

```
DataTypeConversionImpute@player_id
== cleanedplayerinfoSrc@player_id
```

After filtering, **valid player records** proceed to **aggregation (RowCountFinal)**, ensuring the final count reflects only relevant records. The cleaned dataset is then stored in **multiple sinks**:

- **FinalCountsink –** Exports row count validation results in a **JSON file (FinalCount_game_goalie_stats.json)** for tracking.
- **SQLsink –** Saves the cleaned goalie stats in **Azure SQL (table_game_goalie_stats)** for structured querying.
- **CSVsink –** Stores the filtered dataset as a **CSV file** for backup and verification.

Pipeline Execution Summary



Activity runs

All status

Showing 1 - 6 items

Activity name	Activity status	Activity type	Run start	Duration
Prod_game_pipeline	✓ Succeeded	Execute Pipeline	1/22/2025, 9:33:38 PM	5m 1s
Prod_player_info_pipeline	✓ Succeeded	Execute Pipeline	1/22/2025, 9:38:39 PM	5m 6s
Prod_game_teams_stats_pipeline	✓ Succeeded	Execute Pipeline	1/22/2025, 9:38:39 PM	3m 48s
Prod_game_plays_pipeline	✓ Succeeded	Execute Pipeline	1/22/2025, 9:38:39 PM	9m 0s
Prod_game_skater_stats_pipeline	✓ Succeeded	Execute Pipeline	1/22/2025, 9:43:45 PM	2m 49s
Prod_game_goalie_stats_pipeline	✓ Succeeded	Execute Pipeline	1/22/2025, 9:43:45 PM	4m 27s

The **Prod_NHL Pipeline** employs **parallel computing** to optimize execution efficiency and minimize total runtime. Based on the activity logs, **Prod_game_pipeline** started first followed by **Prod_game_teams_stats_pipeline, Prod_player_info_pipeline, and Prod_game_plays_pipeline.** But the latter **3** all start at approximately **9:38:39 PM**, indicating that they are executed **concurrently** rather than sequentially.

The next set of concurrent executed pipelines, **Prod_game_skater_stats_pipeline** and **Prod_game_goalie_stats_pipeline**, start at **9:43:45 PM**, shortly after the earlier pipelines have completed, and finish within **2 minutes 49 seconds** and **4 minutes 27 seconds**, respectively.

If executed sequentially, the total duration would have been significantly longer, summing up all individual runtimes:
5m 1s (Prod_game_pipeline) + **3m 48s** (Prod_game_teams_stats_pipeline) + **5m 6s** (Prod_player_info_pipeline) + **9m 0s** (Prod_game_plays_pipeline) + **2m 49s** (Prod_game_skater_stats_pipeline) + **4m 27s** (Prod_game_goalie_stats_pipeline) = **30m 11s**.

However, with parallel execution, the total runtime is dictated by the longest-running pipeline in each batch rather than the sum of all individual durations. In this case:

5m 1s (Prod_game_pipeline) + **9m 0s** (Prod_game_plays_pipeline) + **4m 27s** (Prod_game_goalie_stats_pipeline) = **18m 28s**.

This **parallel execution reduces the total processing time by approximately 60%**, down from **30 minutes 11 seconds** to just **18 minutes 28 seconds**, significantly enhancing efficiency and resource utilization.

Functional Testing

Validating the data before and after transformation in an **Azure Data Factory (ADF) Data Flow** is essential for ensuring data accuracy and consistency. Below are key steps and techniques to validate the transformed data loaded from ADF Data Flow to an Azure SQL Database

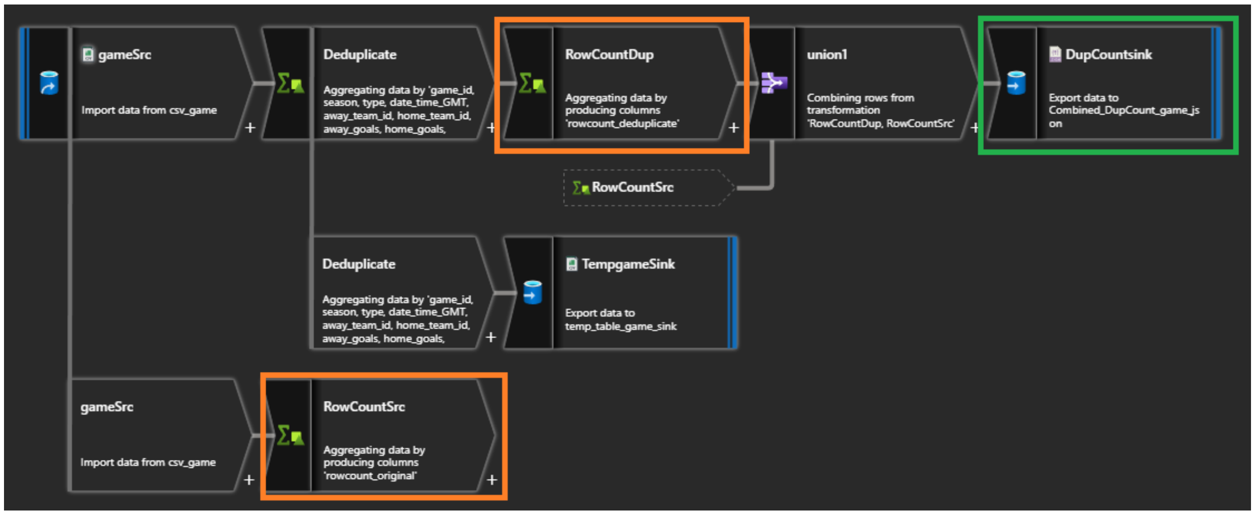
To ensure data accuracy and consistency when loading transformed data from **ADF Data Flow** into **Azure SQL Database**, it is crucial to define clear **data quality rules**. These rules help verify that the data meets expected standards before and after transformation.

1. Record count matching (number of rows before and after).
2. Data type and schema validation.

1. Record count matching (number of rows before and after)

1.1 Within ADF

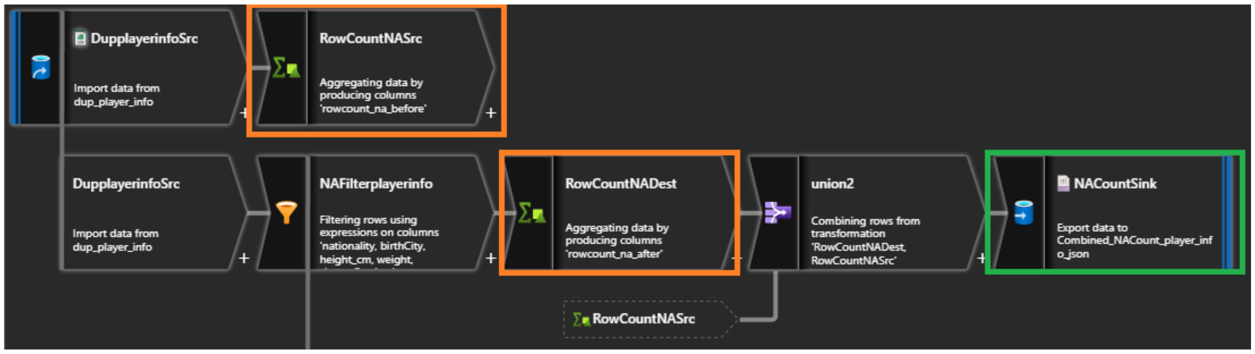
To ensure data integrity and consistency, a **record count matching validation** was conducted before and after the **deduplication** process in the data flow. The objective was to compare the original row count (RowCountSrc) with the deduplicated row count (RowCountDup) and identify the number of duplicate records removed.


gameSrc
Import data from csv_game
Deduplicate
Aggregating data by 'game_id, season, type, date_time_GMT, away_team_id, home_team_id, away_goals, home_goals,
RowCountDup
Aggregating data by producing columns 'rowcount_deduplicate'
union1
Combining rows from transformation 'RowCountDup, RowCountSrc'
DupCountsink
Export data to Combined_DupCount_game_json
RowCountSrc
Deduplicate
Aggregating data by 'game_id, season, type, date_time_GMT, away_team_id, home_team_id, away_goals, home_goals,
TempgameSink
Export data to temp_table_game_sink
gameSrc
Import data from csv_game
RowCountSrc
Aggregating data by producing columns 'rowcount_original'

In the **Data Flow design**, the RowCountSrc transformation was used to capture the initial row count before deduplication, while RowCountDup was used to compute the row count after applying the deduplication logic. These transformations were then merged using a **union operation** and exported to temp sink (DupCountsink) for consolidation.

Dataset	RowCountSrc	RowCountDup	Duplicate Count (ADF)	Duplicate Count (EDA)
game	26305	23735	2570	2570
game_goalie_stats	56656	51163	5493	5493
game_plays	5050529	4217063	833466	833466
game_skater_stats	945830	853404	92426	92426
game_teams_stats	52610	47470	5140	5140
player_info	3925	3925	0	0

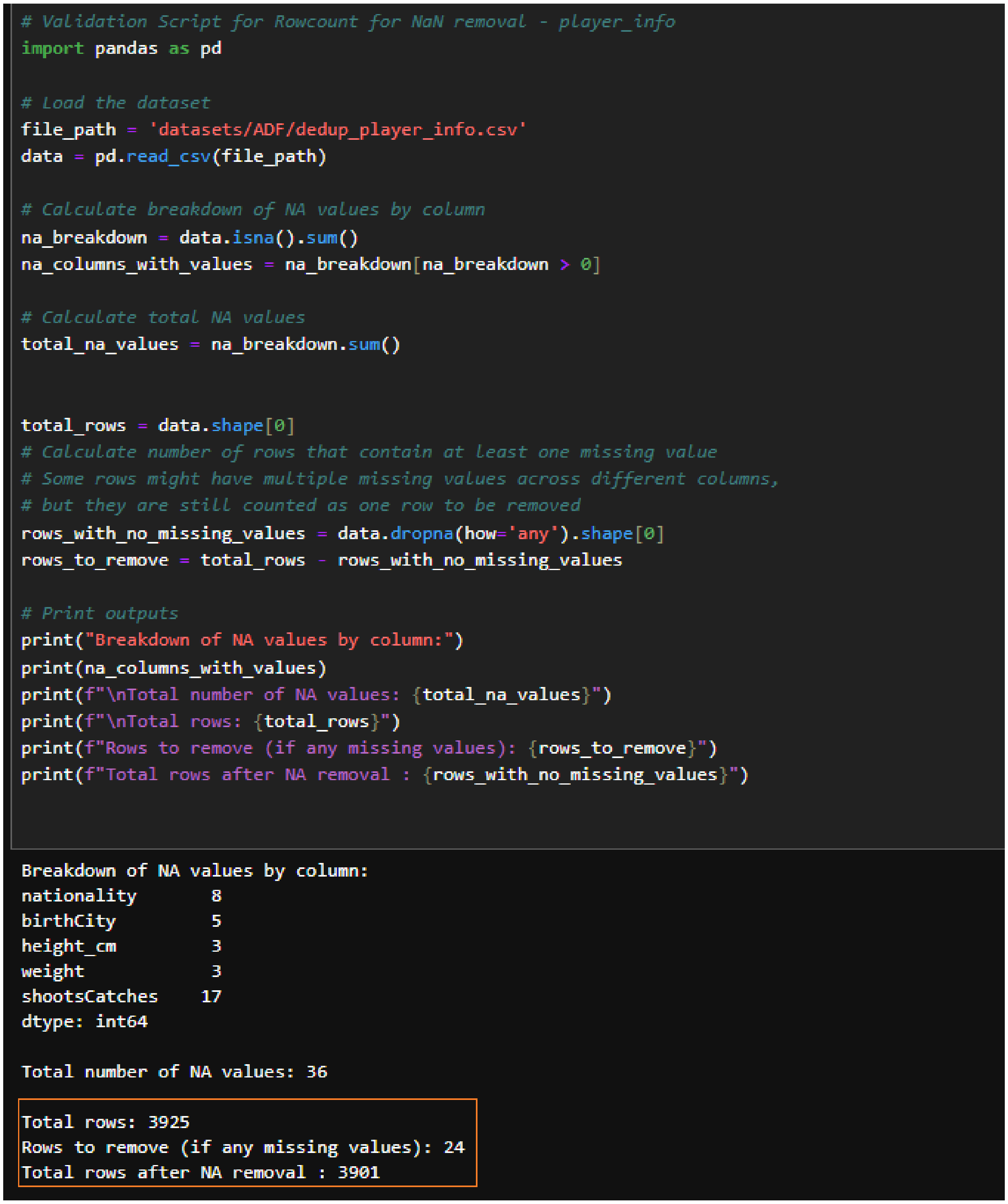
These results confirm that the deduplication process effectively removed redundant records while maintaining data accuracy. The **Duplicate Count from EDA** further validates the number of duplicates detected, ensuring alignment between the expected and actual data cleansing outcomes.


DupplayerinfoSrc
Import data from dup_player_info
RowCountNASrc
Aggregating data by producing columns 'rowcount_na_before'
DupplayerinfoSrc
Import data from dup_player_info
NAFilterplayerinfo
Filtering rows using expressions on columns 'nationality, birthCity, height_cm, weight,
RowCountNADest
Aggregating data by producing columns 'rowcount_na_after'
union2
Combining rows from transformation 'RowCountNADest, RowCountNASrc'
NACountSink
Export data to Combined_NACount_player_info_json
RowCountNASrc

To ensure data completeness and accuracy, an additional **NaN (null) value removal process** was implemented within the **player_info** dataset in **Azure Data Factory (ADF) Data Flow**. The goal was to filter out rows containing missing values in key columns such as **nationality, birthCity, and height_cm** before loading the cleaned data into the final dataset.

Dataset	RowCountNASrc	RowCountNADest	Drop NaN Count (ADF)	Drop NaN Count (Python Script)
player_info	3925	3901	24	24

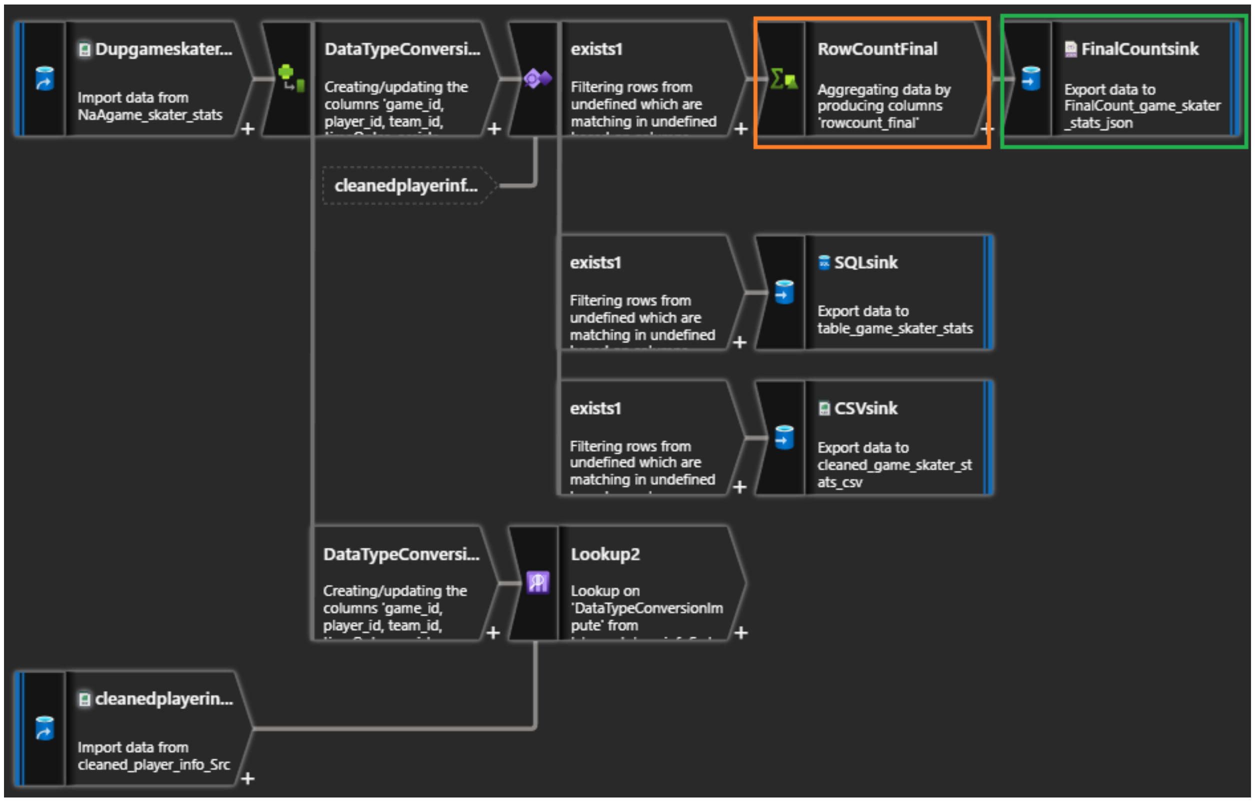
In the Data Flow design, the RowCountNASrc transformation captured the initial row count before applying the NaN removal logic, showing a total of 3,925 records. The NAFilterPlayerInfo transformation was then applied to filter out rows with missing values, and the resulting dataset was aggregated using RowCountNADest, which recorded 3,901 rows after processing. This indicates that 24 rows containing NaN values were successfully removed.



```python
# Validation Script for Rowcount for NaN removal - player_info
import pandas as pd

# Load the dataset
file_path = 'datasets/ADF/dedup_player_info.csv'
data = pd.read_csv(file_path)

# Calculate breakdown of NA values by column
na_breakdown = data.isna().sum()
na_columns_with_values = na_breakdown[na_breakdown > 0]

# Calculate total NA values
total_na_values = na_breakdown.sum()


total_rows = data.shape[0]
# Calculate number of rows that contain at least one missing value
# Some rows might have multiple missing values across different columns,
# but they are still counted as one row to be removed
rows_with_no_missing_values = data.dropna(how='any').shape[0]
rows_to_remove = total_rows - rows_with_no_missing_values

# Print outputs
print("Breakdown of NA values by column:")
print(na_columns_with_values)
print(f"\nTotal number of NA values: {total_na_values}")
print(f"\nTotal rows: {total_rows}")
print(f"Rows to remove (if any missing values): {rows_to_remove}")
print(f"Total rows after NA removal : {rows_with_no_missing_values}")
```

```
Breakdown of NA values by column:
nationality       8
birthCity         5
height_cm         3
weight            3
shootsCatches    17
dtype: int64

Total number of NA values: 36

Total rows: 3925
Rows to remove (if any missing values): 24
Total rows after NA removal : 3901
```

This Python script further validates the number of NaN (missing) values removed in **ADF Data Flow** by independently calculating the total missing values across key columns and determining the number of rows to be dropped. The results confirm that **24 rows** containing at least one missing value were removed, aligning with the count recorded in **ADF**, reinforcing the accuracy of the data cleansing process.


Dupgameskater...
Import data from NaAgame_skater_stats
DataTypeConversi...
Creating/updating the columns 'game_id, player_id, team_id,
exists1
Filtering rows from undefined which are matching in undefined
RowCountFinal
Aggregating data by producing columns 'rowcount_final'
FinalCountsink
Export data to FinalCount_game_skater_stats_json
cleanedplayerinf...
exists1
Filtering rows from undefined which are matching in undefined
SQLsink
Export data to table_game_skater_stats
exists1
Filtering rows from undefined which are matching in undefined
CSVsink
Export data to cleaned_game_skater_stats_csv
DataTypeConversi...
Creating/updating the columns 'game_id, player_id, team_id,
Lookup2
Lookup on 'DataTypeConversionImpute' from
cleanedplayerin...
Import data from cleaned_player_info_Src

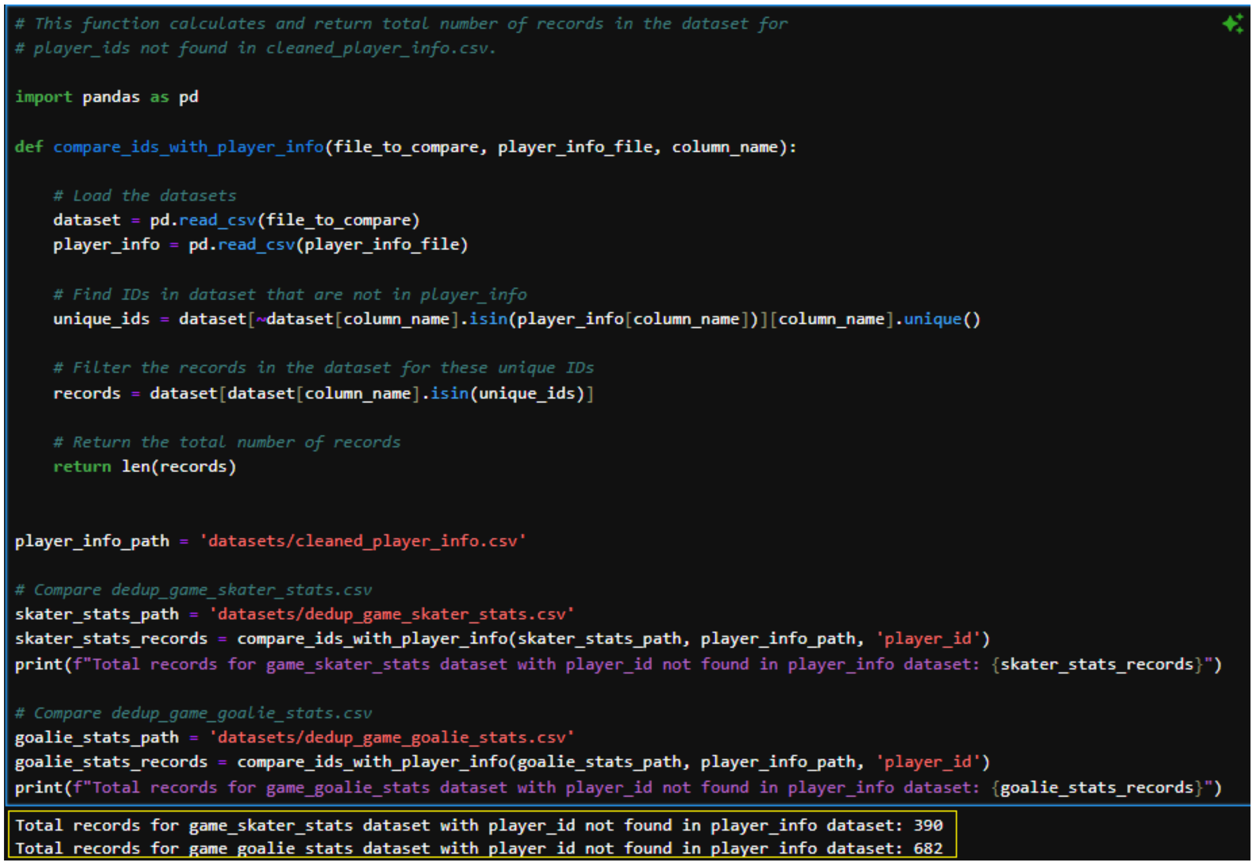
A closer examination of the **game_goalie_stats** and **game_skater_stats** datasets revealed that both contain **player_id values** not found in the cleaned **player_info** dataset. To ensure data integrity and maintain referential consistency, these unmatched records were filtered out as part of the data cleansing process in ADF Data Flow.

Dataset	RowCountDup	RowCountFinal	Drop Count (ADF)	Drop Count (Python Script)
game_goalie_stats	51163	50481	682	682
game_skater_stats	853404	853014	390	390

The **RowCountDup** transformation captured the initial record count before filtering. The **RowCountFinal** transformation then recorded the final row count after this filtering process.

The **game_goalie_stats** dataset originally contained **51,163** records. After filtering out invalid **player_id** values, the final count was **50,481**, confirming that **682 records** were removed.

The **game_skater_stats** dataset initially had **853,404** records, which was reduced to **853,014**, removing **390 records** with unmatched **player_id** values.



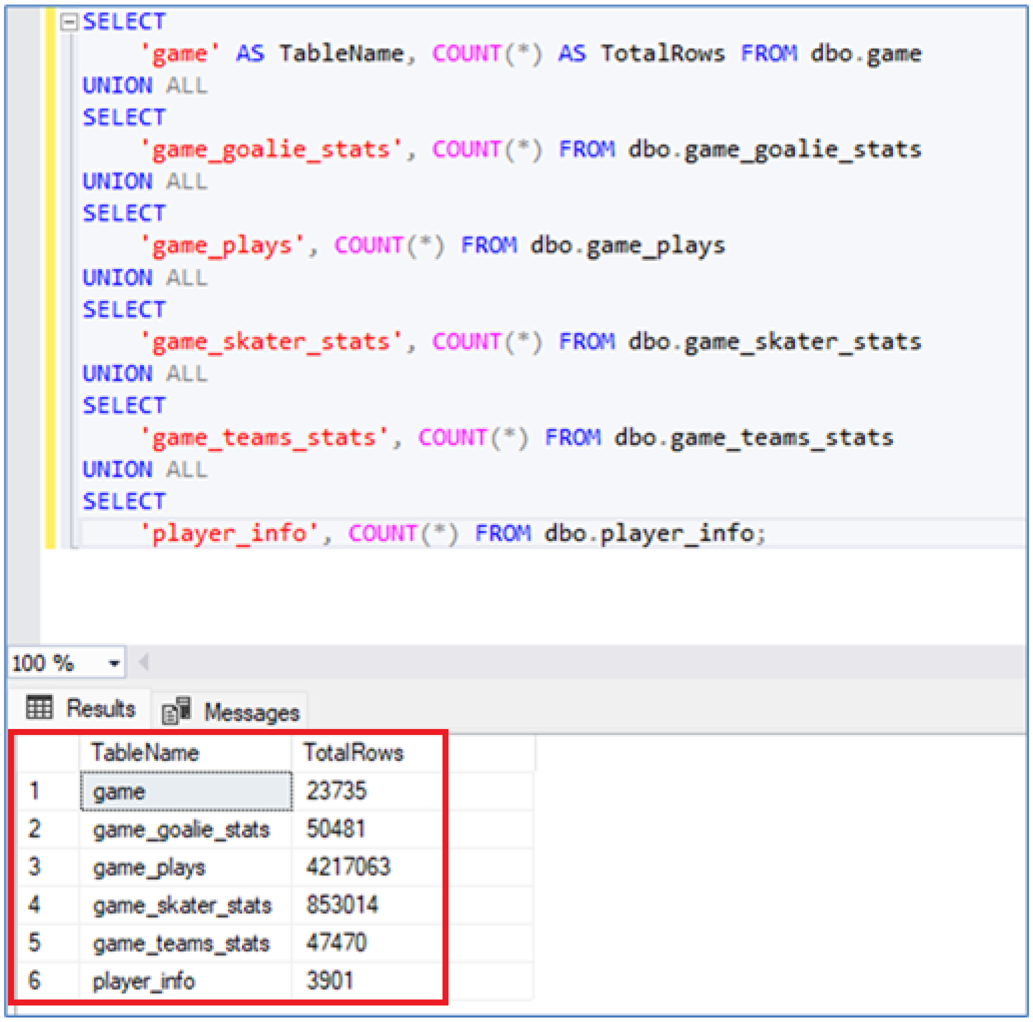
```python
# This function calculates and return total number of records in the dataset for
# player_ids not found in cleaned_player_info.csv.

import pandas as pd

def compare_ids_with_player_info(file_to_compare, player_info_file, column_name):

    # Load the datasets
    dataset = pd.read_csv(file_to_compare)
    player_info = pd.read_csv(player_info_file)

    # Find IDs in dataset that are not in player_info
    unique_ids = dataset[~dataset[column_name].isin(player_info[column_name])][column_name].unique()

    # Filter the records in the dataset for these unique IDs
    records = dataset[dataset[column_name].isin(unique_ids)]

    # Return the total number of records
    return len(records)


player_info_path = 'datasets/cleaned_player_info.csv'

# Compare dedup_game_skater_stats.csv
skater_stats_path = 'datasets/dedup_game_skater_stats.csv'
skater_stats_records = compare_ids_with_player_info(skater_stats_path, player_info_path, 'player_id')
print(f"Total records for game_skater_stats dataset with player_id not found in player_info dataset: {skater_stats_records}")

# Compare dedup_game_goalie_stats.csv
goalie_stats_path = 'datasets/dedup_game_goalie_stats.csv'
goalie_stats_records = compare_ids_with_player_info(goalie_stats_path, player_info_path, 'player_id')
print(f"Total records for game_goalie_stats dataset with player_id not found in player_info dataset: {goalie_stats_records}")
```

```
Total records for game_skater_stats dataset with player_id not found in player_info dataset: 390
Total records for game_goalie_stats dataset with player_id not found in player_info dataset: 682
```

The Drop Count recorded in ADF precisely **matched** the results obtained using a Python validation script, reinforcing the correctness of the data cleansing process.

1.2 In Azure SQL Database

Dataset	RowCountFinal (ADF)	Row Count (Azure SQL)
game	23735	23735
game_goalie_stats	50481	50481
game_plays	4217063	4217063
game_skater_stats	853014	853014
game_teams_stats	47470	47470
player_info	3901	3901


```
SELECT
    'game' AS TableName, COUNT(*) AS TotalRows FROM dbo.game
UNION ALL
SELECT
    'game_goalie_stats', COUNT(*) FROM dbo.game_goalie_stats
UNION ALL
SELECT
    'game_plays', COUNT(*) FROM dbo.game_plays
UNION ALL
SELECT
    'game_skater_stats', COUNT(*) FROM dbo.game_skater_stats
UNION ALL
SELECT
    'game_teams_stats', COUNT(*) FROM dbo.game_teams_stats
UNION ALL
SELECT
    'player_info', COUNT(*) FROM dbo.player_info;
```

```
   TableName           TotalRows
1  game                23735
2  game_goalie_stats   50481
3  game_plays          4217063
4  game_skater_stats   853014
5  game_teams_stats    47470
6  player_info         3901
```

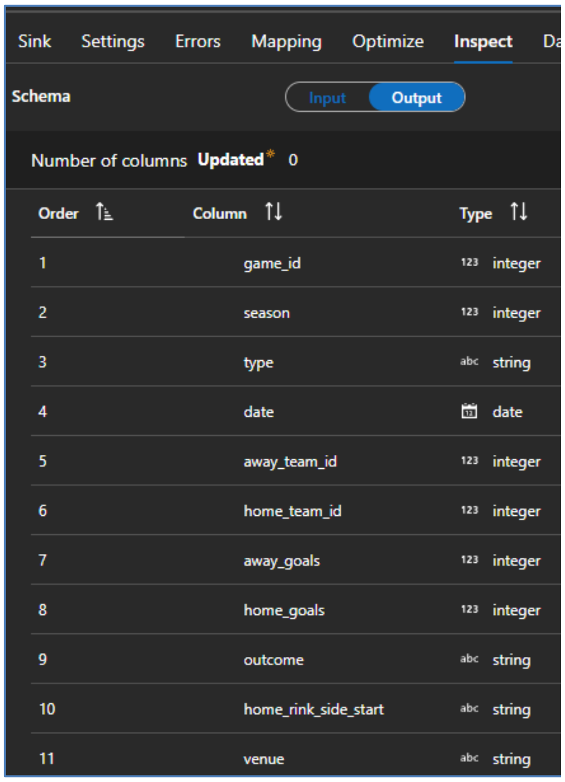
After the data is loaded into Azure SQL Database, it is essential to perform **Record Count Validation** to ensure data completeness.

As shown in the validation table, the row counts from ADF match exactly with the row counts retrieved from Azure SQL using SQL queries. This confirms that the data has been fully transferred without any loss or duplication, ensuring data integrity and reliability in the pipeline.

2. **Data type and schema validation in Azure SQL Database**

game dataset

ADF



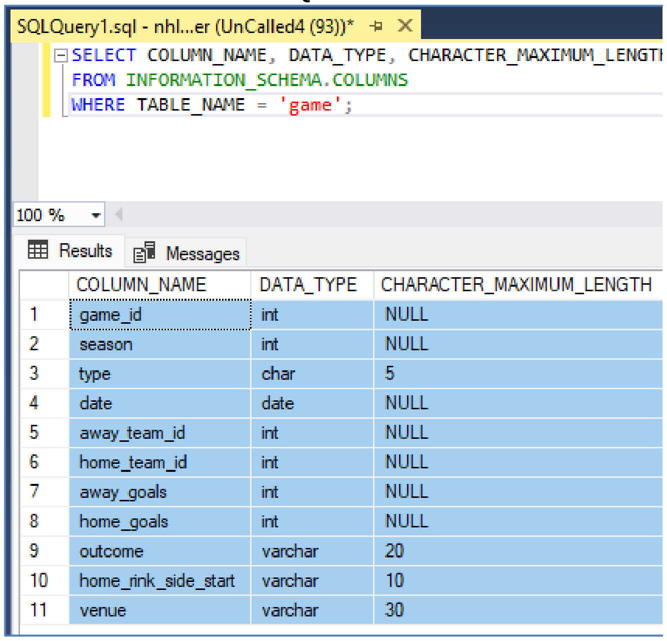
Sink Settings Errors Mapping Optimize Inspect Da

Schema — Input | Output

Number of columns Updated* 0

Order	Column	Type
1	game_id	integer
2	season	integer
3	type	string
4	date	date
5	away_team_id	integer
6	home_team_id	integer
7	away_goals	integer
8	home_goals	integer
9	outcome	string
10	home_rink_side_start	string
11	venue	string

Azure SQL Database



SQLQuery1.sql - nhl...er (UnCalled4 (93))*

```
SELECT COLUMN_NAME, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'game';
```

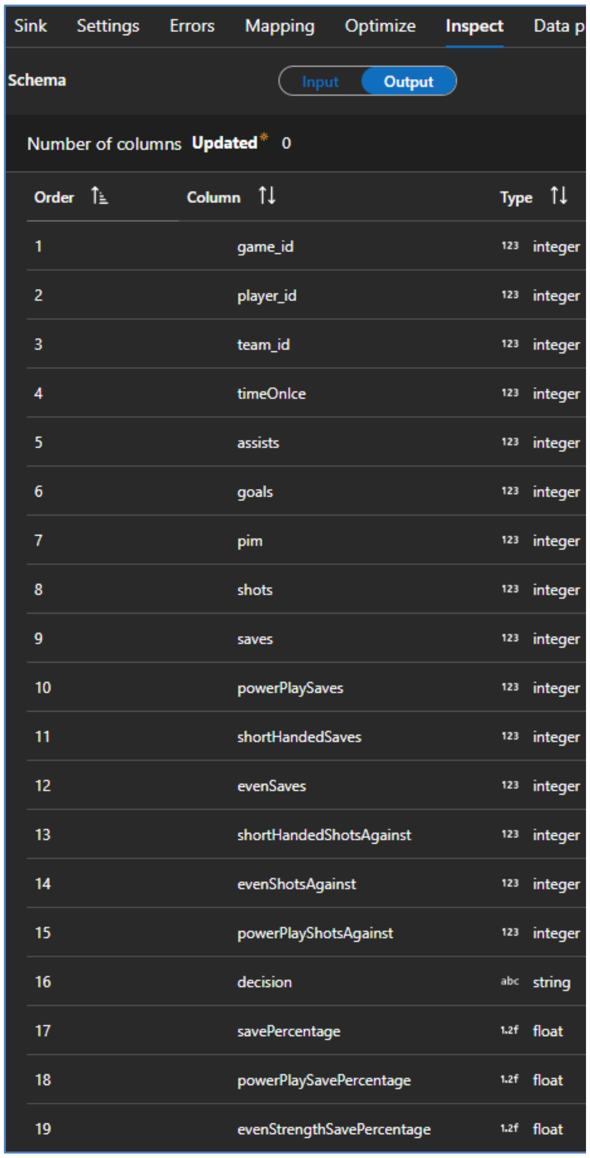
100 %

Results Messages

	COLUMN_NAME	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH
1	game_id	int	NULL
2	season	int	NULL
3	type	char	5
4	date	date	NULL
5	away_team_id	int	NULL
6	home_team_id	int	NULL
7	away_goals	int	NULL
8	home_goals	int	NULL
9	outcome	varchar	20
10	home_rink_side_start	varchar	10
11	venue	varchar	30

game_goalie_stats

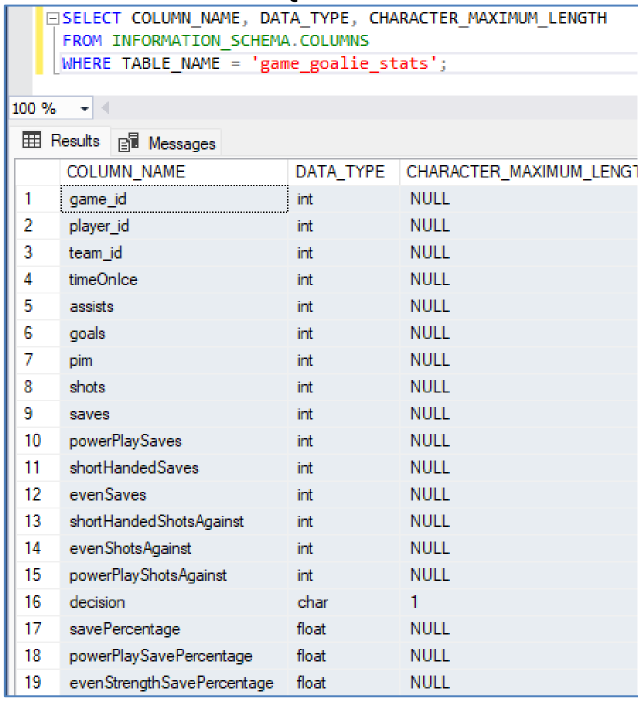
ADF



Number of columns **Updated*** 0

Order	Column	Type
1	game_id	integer
2	player_id	integer
3	team_id	integer
4	timeOnIce	integer
5	assists	integer
6	goals	integer
7	pim	integer
8	shots	integer
9	saves	integer
10	powerPlaySaves	integer
11	shortHandedSaves	integer
12	evenSaves	integer
13	shortHandedShotsAgainst	integer
14	evenShotsAgainst	integer
15	powerPlayShotsAgainst	integer
16	decision	string
17	savePercentage	float
18	powerPlaySavePercentage	float
19	evenStrengthSavePercentage	float

Azure SQL Database

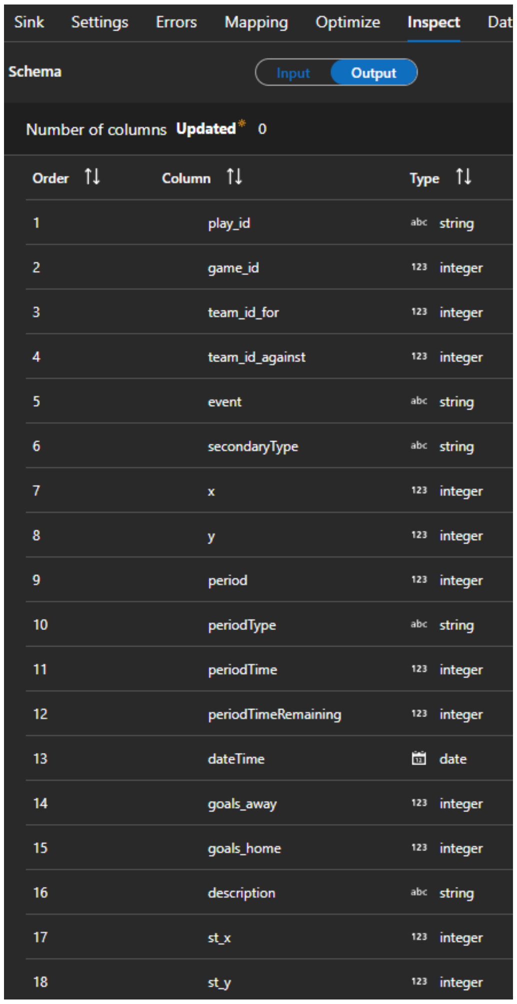


```
SELECT COLUMN_NAME, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'game_goalie_stats';
```

	COLUMN_NAME	DATA_TYPE	CHARACTER_MAXIMUM_LENGT
1	game_id	int	NULL
2	player_id	int	NULL
3	team_id	int	NULL
4	timeOnIce	int	NULL
5	assists	int	NULL
6	goals	int	NULL
7	pim	int	NULL
8	shots	int	NULL
9	saves	int	NULL
10	powerPlaySaves	int	NULL
11	shortHandedSaves	int	NULL
12	evenSaves	int	NULL
13	shortHandedShotsAgainst	int	NULL
14	evenShotsAgainst	int	NULL
15	powerPlayShotsAgainst	int	NULL
16	decision	char	1
17	savePercentage	float	NULL
18	powerPlaySavePercentage	float	NULL
19	evenStrengthSavePercentage	float	NULL

game_plays

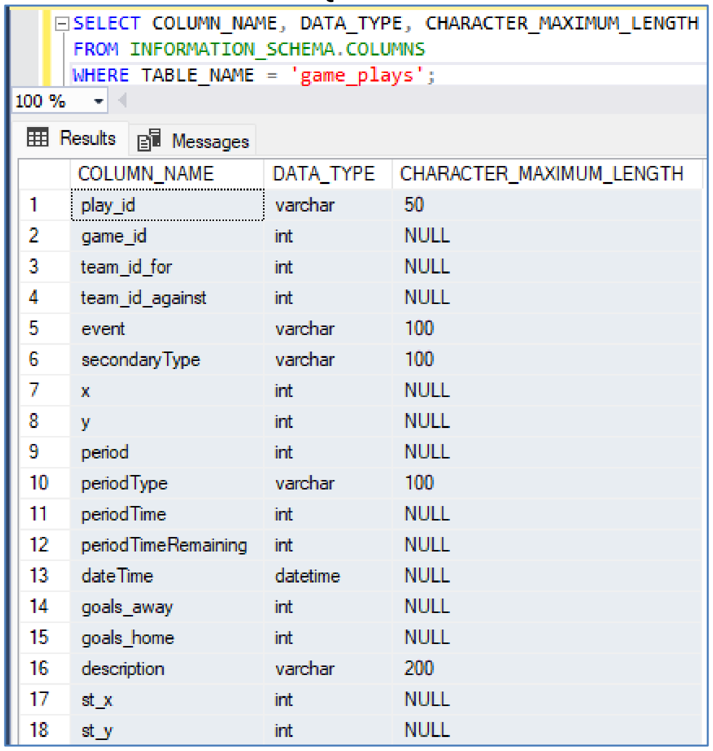
ADF



Number of columns Updated* 0

Order	Column	Type
1	play_id	abc string
2	game_id	123 integer
3	team_id_for	123 integer
4	team_id_against	123 integer
5	event	abc string
6	secondaryType	abc string
7	x	123 integer
8	y	123 integer
9	period	123 integer
10	periodType	abc string
11	periodTime	123 integer
12	periodTimeRemaining	123 integer
13	dateTime	date
14	goals_away	123 integer
15	goals_home	123 integer
16	description	abc string
17	st_x	123 integer
18	st_y	123 integer

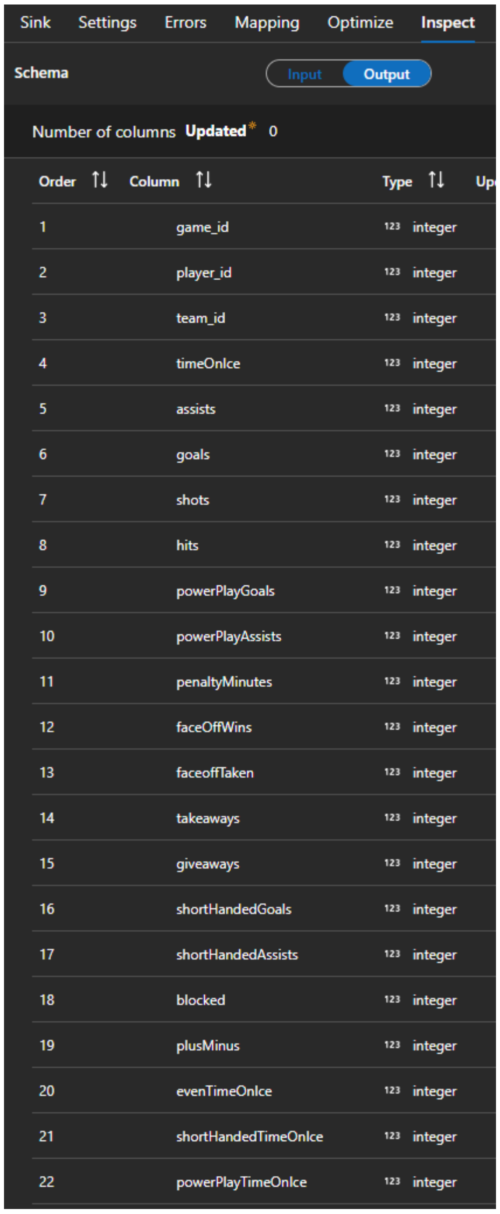
Azure SQL Database


```sql
SELECT COLUMN_NAME, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'game_plays';
```

	COLUMN_NAME	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH
1	play_id	varchar	50
2	game_id	int	NULL
3	team_id_for	int	NULL
4	team_id_against	int	NULL
5	event	varchar	100
6	secondaryType	varchar	100
7	x	int	NULL
8	y	int	NULL
9	period	int	NULL
10	periodType	varchar	100
11	periodTime	int	NULL
12	periodTimeRemaining	int	NULL
13	dateTime	datetime	NULL
14	goals_away	int	NULL
15	goals_home	int	NULL
16	description	varchar	200
17	st_x	int	NULL
18	st_y	int	NULL

game_skater_stats

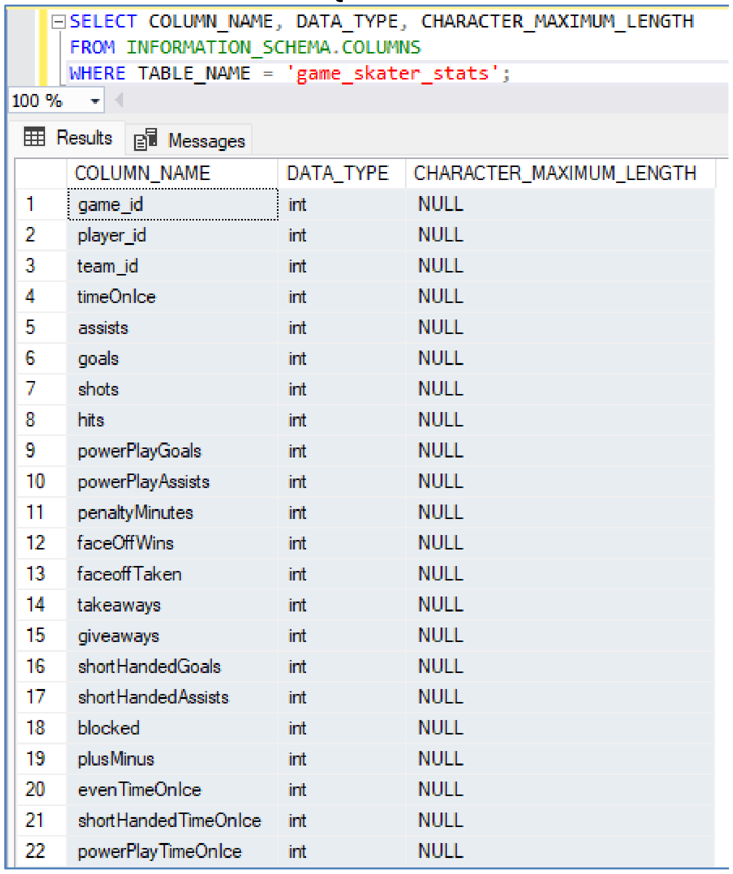
ADF



Number of columns **Updated*** 0

Order	Column	Type
1	game_id	123 integer
2	player_id	123 integer
3	team_id	123 integer
4	timeOnIce	123 integer
5	assists	123 integer
6	goals	123 integer
7	shots	123 integer
8	hits	123 integer
9	powerPlayGoals	123 integer
10	powerPlayAssists	123 integer
11	penaltyMinutes	123 integer
12	faceOffWins	123 integer
13	faceoffTaken	123 integer
14	takeaways	123 integer
15	giveaways	123 integer
16	shortHandedGoals	123 integer
17	shortHandedAssists	123 integer
18	blocked	123 integer
19	plusMinus	123 integer
20	evenTimeOnIce	123 integer
21	shortHandedTimeOnIce	123 integer
22	powerPlayTimeOnIce	123 integer

Azure SQL Database

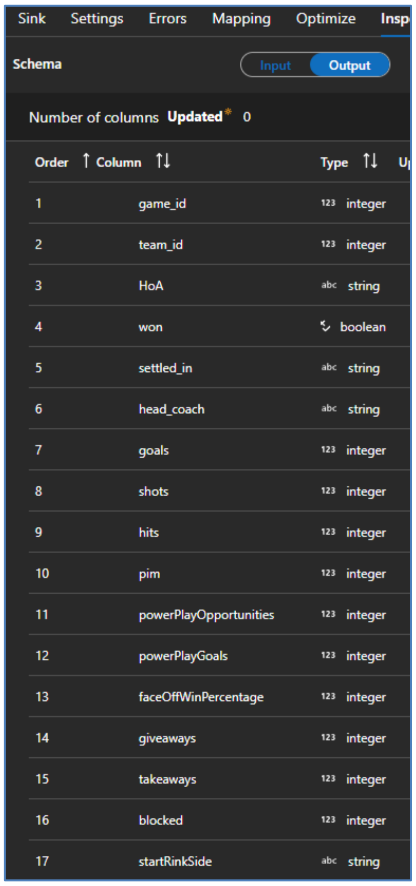


```sql
SELECT COLUMN_NAME, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'game_skater_stats';
```

	COLUMN_NAME	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH
1	game_id	int	NULL
2	player_id	int	NULL
3	team_id	int	NULL
4	timeOnIce	int	NULL
5	assists	int	NULL
6	goals	int	NULL
7	shots	int	NULL
8	hits	int	NULL
9	powerPlayGoals	int	NULL
10	powerPlayAssists	int	NULL
11	penaltyMinutes	int	NULL
12	faceOffWins	int	NULL
13	faceoffTaken	int	NULL
14	takeaways	int	NULL
15	giveaways	int	NULL
16	shortHandedGoals	int	NULL
17	shortHandedAssists	int	NULL
18	blocked	int	NULL
19	plusMinus	int	NULL
20	evenTimeOnIce	int	NULL
21	shortHandedTimeOnIce	int	NULL
22	powerPlayTimeOnIce	int	NULL

game_teams_stats

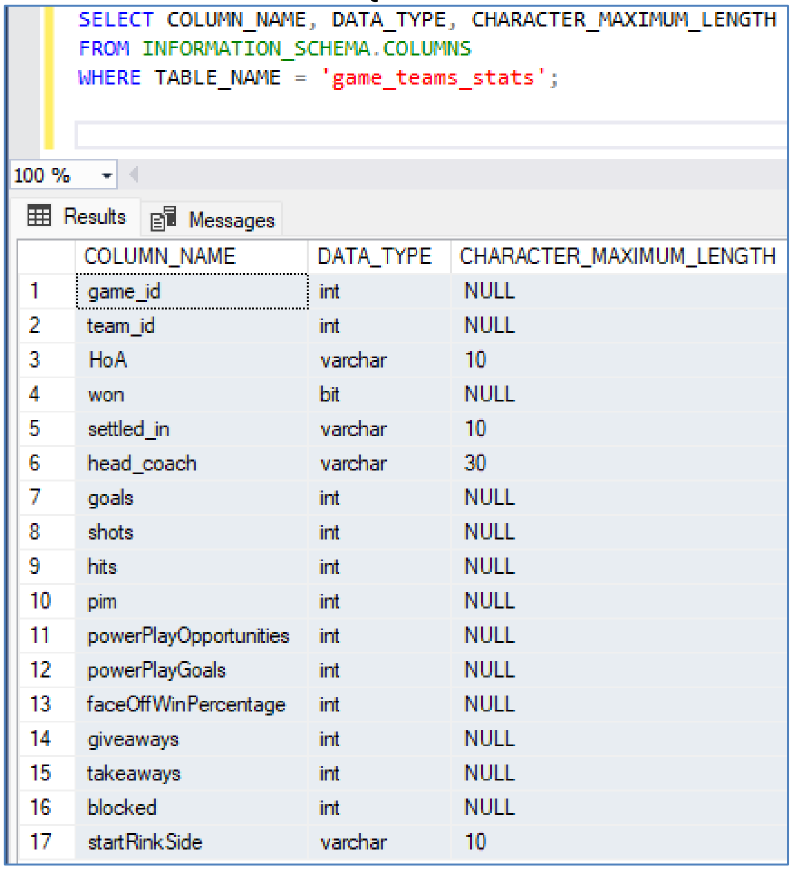
ADF



Number of columns **Updated*** 0

Order	Column	Type
1	game_id	integer
2	team_id	integer
3	HoA	string
4	won	boolean
5	settled_in	string
6	head_coach	string
7	goals	integer
8	shots	integer
9	hits	integer
10	pim	integer
11	powerPlayOpportunities	integer
12	powerPlayGoals	integer
13	faceOffWinPercentage	integer
14	giveaways	integer
15	takeaways	integer
16	blocked	integer
17	startRinkSide	string

Azure SQL Database

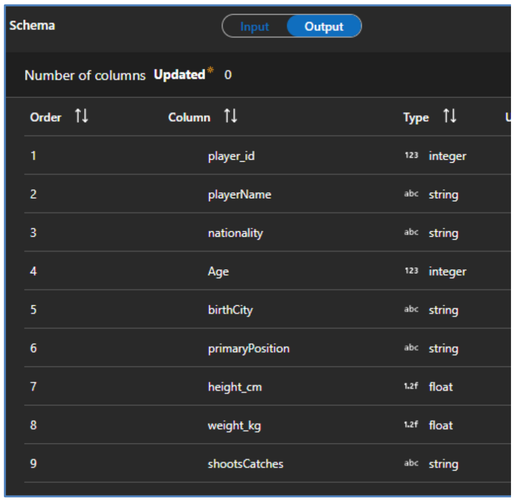


```sql
SELECT COLUMN_NAME, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'game_teams_stats';
```

	COLUMN_NAME	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH
1	game_id	int	NULL
2	team_id	int	NULL
3	HoA	varchar	10
4	won	bit	NULL
5	settled_in	varchar	10
6	head_coach	varchar	30
7	goals	int	NULL
8	shots	int	NULL
9	hits	int	NULL
10	pim	int	NULL
11	powerPlayOpportunities	int	NULL
12	powerPlayGoals	int	NULL
13	faceOffWinPercentage	int	NULL
14	giveaways	int	NULL
15	takeaways	int	NULL
16	blocked	int	NULL
17	startRinkSide	varchar	10

player_info

ADF

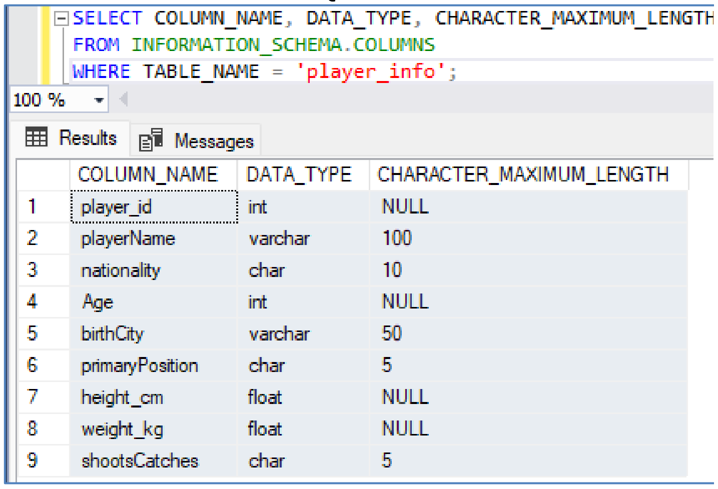


Schema Input Output

Number of columns Updated* 0

Order	Column	Type
1	player_id	123 integer
2	playerName	abc string
3	nationality	abc string
4	Age	123 integer
5	birthCity	abc string
6	primaryPosition	abc string
7	height_cm	1.2f float
8	weight_kg	1.2f float
9	shootsCatches	abc string

Azure SQL Database



```
SELECT COLUMN_NAME, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'player_info';
```

100 %

Results Messages

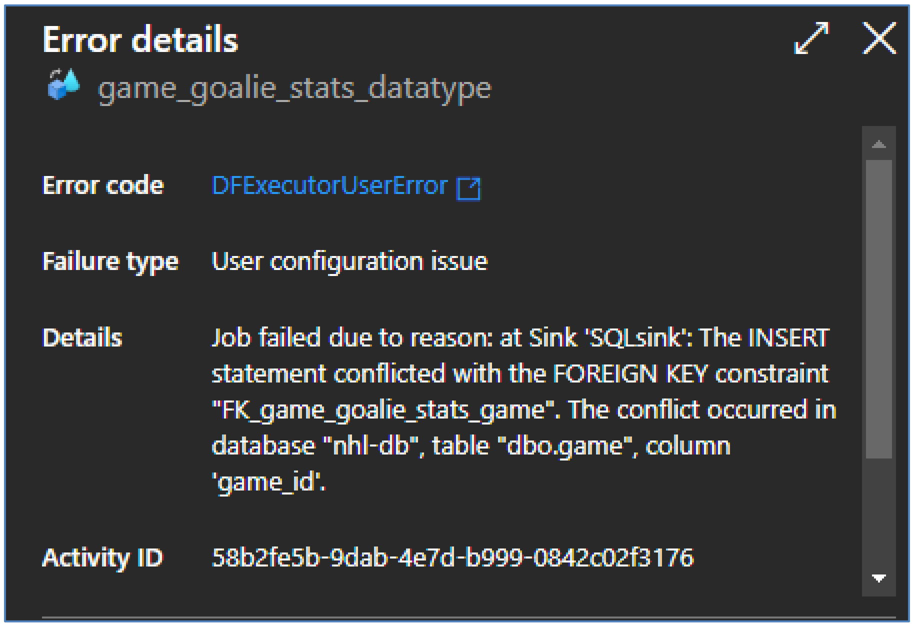
	COLUMN_NAME	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH
1	player_id	int	NULL
2	playerName	varchar	100
3	nationality	char	10
4	Age	int	NULL
5	birthCity	varchar	50
6	primaryPosition	char	5
7	height_cm	float	NULL
8	weight_kg	float	NULL
9	shootsCatches	char	5

Challenges

Issue#1

Insertion failed error while trying to load cleaned data for **game_goalie_stats** pipeline to Azure SQL Database:

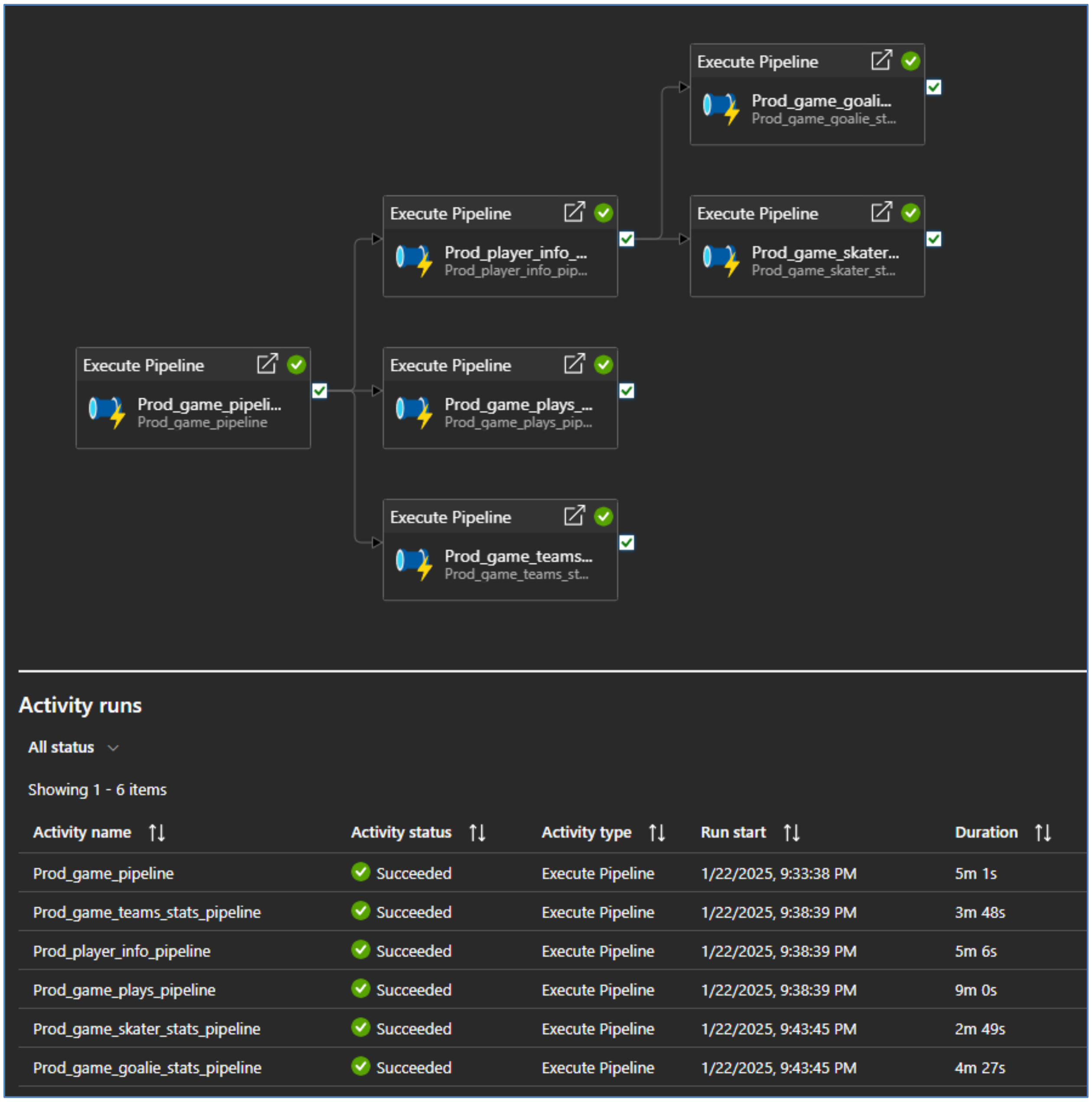
Job failed due to reason: at Sink 'SQLsink': The INSERT statement conflicted with the FOREIGN KEY constraint "FK_game_goalie_stats_game". The conflict occurred in database "nhl-db", table "dbo.game", column 'game_id'.



Error details
game_goalie_stats_datatype

Error code	DFExecutorUserError
Failure type	User configuration issue
Details	Job failed due to reason: at Sink 'SQLsink': The INSERT statement conflicted with the FOREIGN KEY constraint "FK_game_goalie_stats_game". The conflict occurred in database "nhl-db", table "dbo.game", column 'game_id'.
Activity ID	58b2fe5b-9dab-4e7d-b999-0842c02f3176

Reason:
The foreign key constraint ensures referential integrity by requiring that all **game_id** values in game_goalie_stats have a corresponding entry in the game table.
The game_goalie_stats pipeline was triggered first and succeeded BEFORE the game pipeline. Because the parent table (game) does not hold any records, so any attempt to insert records into game_goalie_stats with a game_id will violate the foreign key constraint. Therefore, the insertion fails.



Activity runs

All status

Showing 1 - 6 items

Activity name	Activity status	Activity type	Run start	Duration
Prod_game_pipeline	✓ Succeeded	Execute Pipeline	1/22/2025, 9:33:38 PM	5m 1s
Prod_game_teams_stats_pipeline	✓ Succeeded	Execute Pipeline	1/22/2025, 9:38:39 PM	3m 48s
Prod_player_info_pipeline	✓ Succeeded	Execute Pipeline	1/22/2025, 9:38:39 PM	5m 6s
Prod_game_plays_pipeline	✓ Succeeded	Execute Pipeline	1/22/2025, 9:38:39 PM	9m 0s
Prod_game_skater_stats_pipeline	✓ Succeeded	Execute Pipeline	1/22/2025, 9:43:45 PM	2m 49s
Prod_game_goalie_stats_pipeline	✓ Succeeded	Execute Pipeline	1/22/2025, 9:43:45 PM	4m 27s

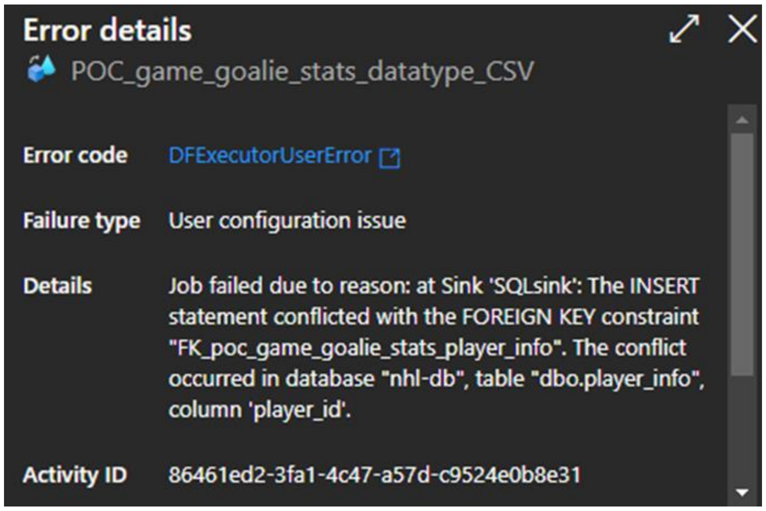
Resolution:
Ensure that the pipelines of parent tables (**game** and **player_info**) are completed first before triggering the dependant pipelines (**game_goalie_stats**)

The **ORDER** of pipeline processing matters.

Issue#2

Insertion failed error while trying to load cleaned data for **game_goalie_stats** pipeline to Azure SQL Database:

Job failed due to reason: at Sink 'SQLsink': The INSERT statement conflicted with the FOREIGN KEY constraint "FK_poc_game_goalie_stats_player_info". The conflict occurred in database "nhl-db", table "dbo.player_info", column 'player_id'.



Error details

POC_game_goalie_stats_datatype_CSV

Error code DFExecutorUserError

Failure type User configuration issue

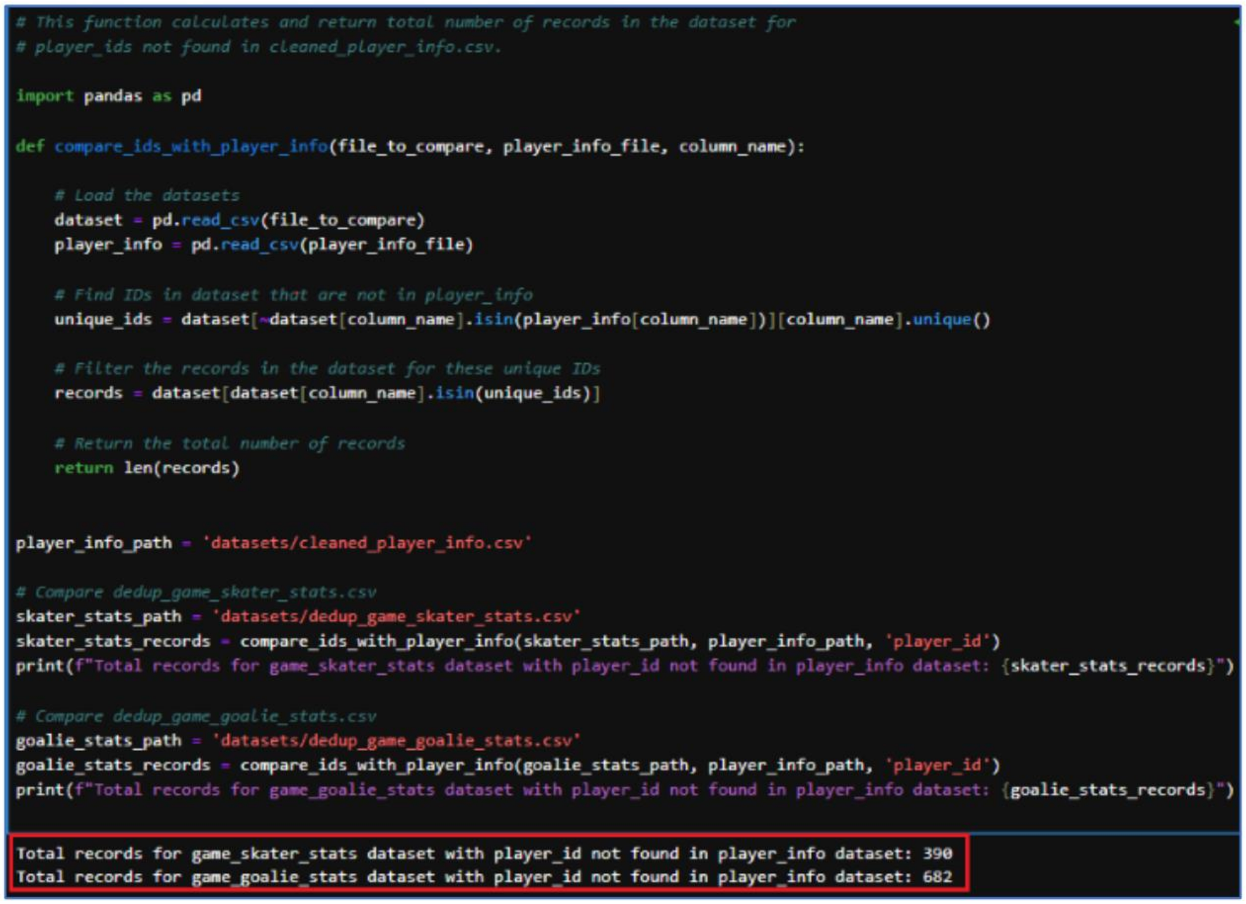
Details Job failed due to reason: at Sink 'SQLsink': The INSERT statement conflicted with the FOREIGN KEY constraint "FK_poc_game_goalie_stats_player_info". The conflict occurred in database "nhl-db", table "dbo.player_info", column 'player_id'.

Activity ID 86461ed2-3fa1-4c47-a57d-c9524e0b8e31

The **player_info** table was already loaded and populated with the cleaned version of player_info dataset. Similar error message was also encountered while loading **game_skater_stats** pipeline.

Reason:
Closer examination of both game_goalie_stats and game_skater_stats datasets revealed that both contain player_id values NOT found in cleaned player_info dataset.
The foreign key constraint ensures referential integrity by requiring that all player_id values in game_goalie_stats and game_skater_stats have corresponding entries in the player_info table.

Total records with player_id values not found in player_info dataset:
For game_skater_stats dataset - **390**
For game_goalie_stats dataset **– 682**


```
# This function calculates and return total number of records in the dataset for
# player_ids not found in cleaned_player_info.csv.

import pandas as pd

def compare_ids_with_player_info(file_to_compare, player_info_file, column_name):

    # Load the datasets
    dataset = pd.read_csv(file_to_compare)
    player_info = pd.read_csv(player_info_file)

    # Find IDs in dataset that are not in player_info
    unique_ids = dataset[~dataset[column_name].isin(player_info[column_name])][column_name].unique()

    # Filter the records in the dataset for these unique IDs
    records = dataset[dataset[column_name].isin(unique_ids)]

    # Return the total number of records
    return len(records)


player_info_path = 'datasets/cleaned_player_info.csv'

# Compare dedup_game_skater_stats.csv
skater_stats_path = 'datasets/dedup_game_skater_stats.csv'
skater_stats_records = compare_ids_with_player_info(skater_stats_path, player_info_path, 'player_id')
print(f"Total records for game_skater_stats dataset with player_id not found in player_info dataset: {skater_stats_records}")

# Compare dedup_game_goalie_stats.csv
goalie_stats_path = 'datasets/dedup_game_goalie_stats.csv'
goalie_stats_records = compare_ids_with_player_info(goalie_stats_path, player_info_path, 'player_id')
print(f"Total records for game_goalie_stats dataset with player_id not found in player_info dataset: {goalie_stats_records}")
```

```
Total records for game_skater_stats dataset with player_id not found in player_info dataset: 390
Total records for game_goalie_stats dataset with player_id not found in player_info dataset: 682
```

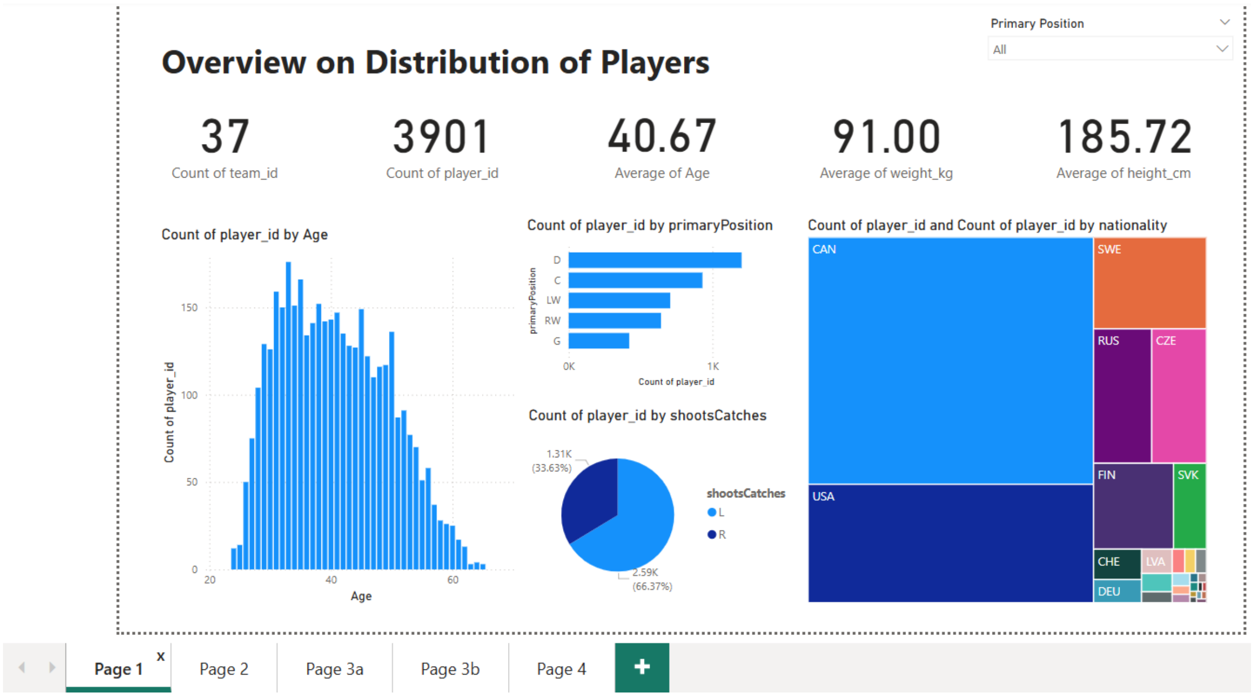
Resolution:
Filter and remove these records from **game_goalie_stats** and **game_skater**_stats through an enhanced data flow (Refers to Section - **Enhanced Data Flow Design #2**)

Insights & Analysis

PowerBI Dashboard & Analysis

Based on the NHL dataset, a comprehensive dashboard has been created to analyze various aspects of the league. Here's an overview of the dashboard's pages and their key insights:

Page 1: Distribution of Players



This visualization effectively summarizes the key demographic and physical characteristics of NHL players, providing valuable insights into the league's composition and player attributes.

<u>League Size and Structure</u>

- The NHL consists of 37 teams with 3,901 registered players
- The average player is 40.67 years old, weighs 91 kg, and stands 185.72 cm tall

<u>Age Distribution</u>

- The age distribution follows a bell curve with a peak around 30-35 years, with a gradual decline as players approach retirement age.

<u>Player Positions</u>

- Defensemen (D) make up the largest portion of players
- Centers (C) and Left/Right Wing (L/R) positions show relatively balanced distributions
- Goalie (G) positions have the smallest representation, which is expected given team composition

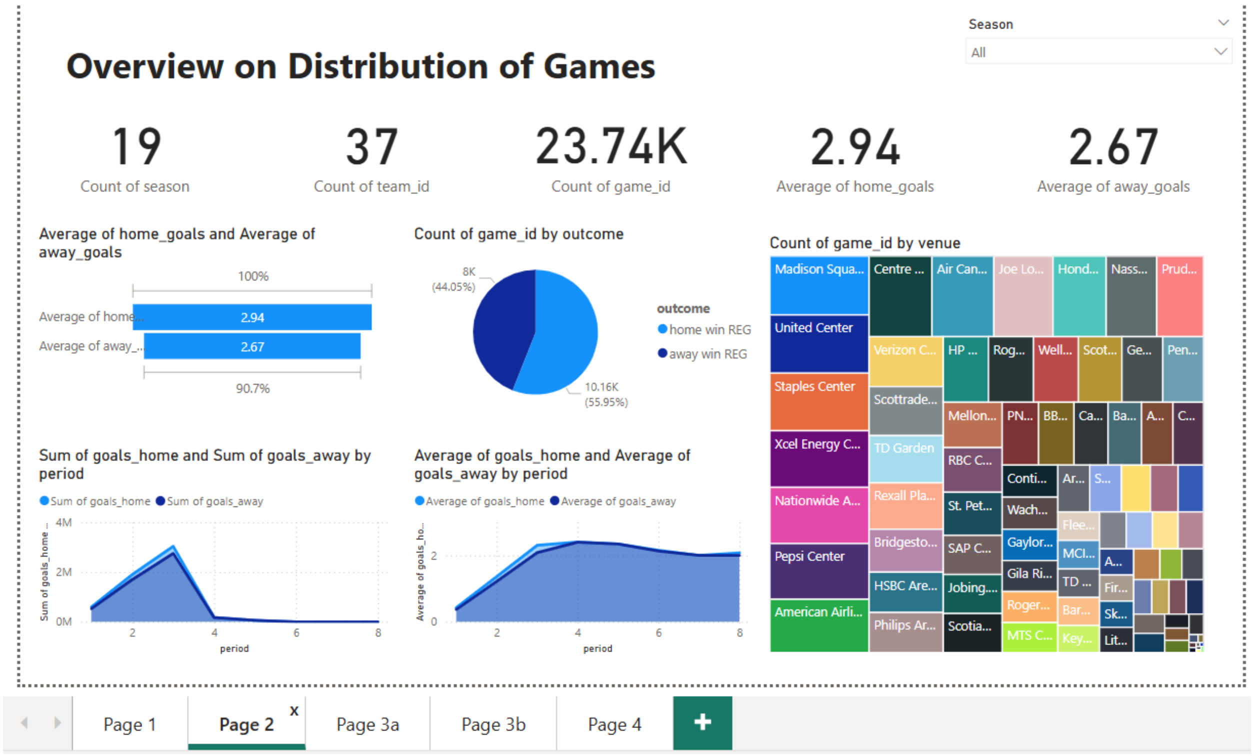
<u>Nationality Distribution</u>

- Canadian (CAN) players form the largest nationality group, while American (USA) players represent the second-largest group
- There's a diverse mix of other nationalities represented in smaller numbers

<u>Playing Style</u>

- The shootsCatches distribution shows approximately a 66.37% to 33.63% split, indicating a clear preference in shooting/catching handedness among players

Page 2: Distribution of Games



The dataset spans 19 seasons across 37 teams, containing 23,740 games and here are the key insights:

<u>Game outcomes</u>

- Home teams tend to have higher scoring rates compared to away teams, with average scores of 2.94 and 2.67 respectively
- The pie chart shows approximately 55.95% home wins versus 44.05% away wins, again demonstrating a significant home-ice advantage in the NHL

<u>Scoring Patterns</u>

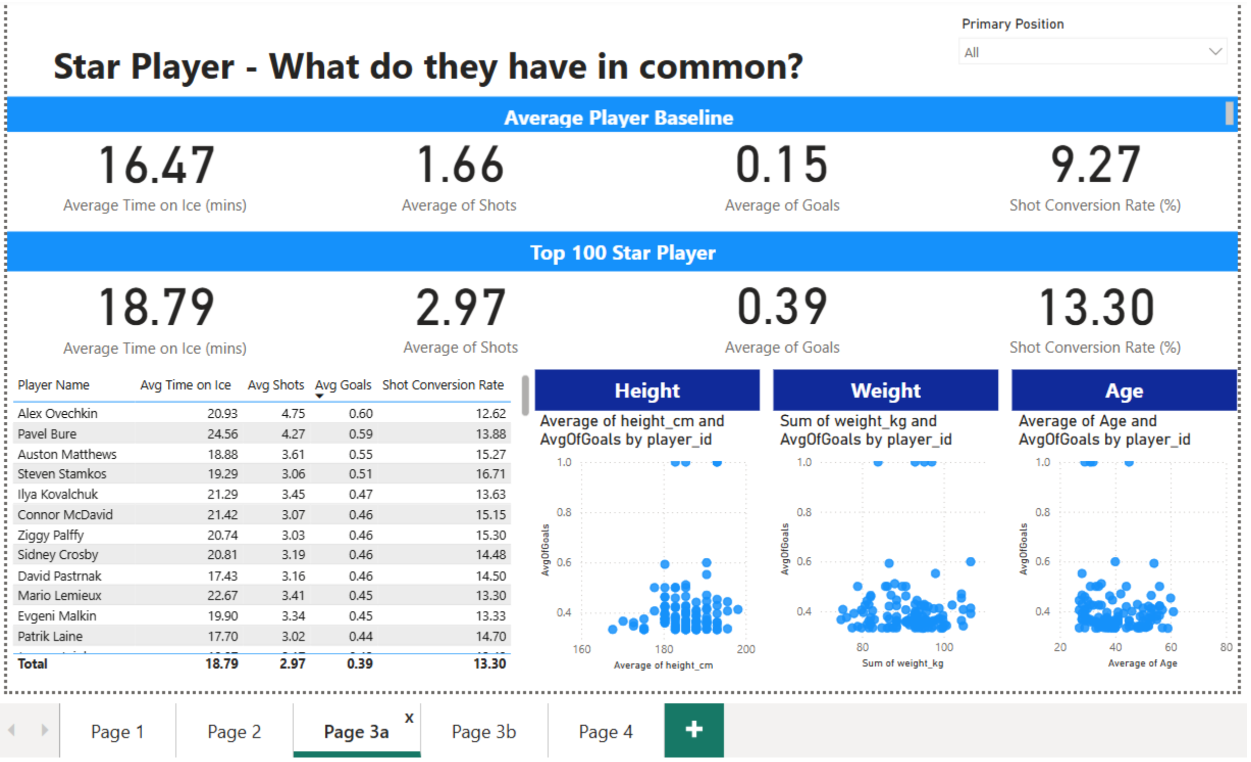
- The period-by-period analysis shows scoring trends fluctuate throughout the game
- Peak scoring appears to occur in middle periods based on the line graph

<u>Venue Analysis</u>

- The treemap shows various NHL venues with different sizes indicating game frequency

- Major venues like United Center, Staples Center, and Madison Square Garden host significant numbers of games

Page 3a: Top 100 Star Players Analysis



Primary Position
All

Star Player - What do they have in common?

Average Player Baseline

16.47	1.66	0.15	9.27
Average Time on Ice (mins)	Average of Shots	Average of Goals	Shot Conversion Rate (%)

Top 100 Star Player

18.79	2.97	0.39	13.30
Average Time on Ice (mins)	Average of Shots	Average of Goals	Shot Conversion Rate (%)

Player Name	Avg Time on Ice	Avg Shots	Avg Goals	Shot Conversion Rate
Alex Ovechkin	20.93	4.75	0.60	12.62
Pavel Bure	24.56	4.27	0.59	13.88
Auston Matthews	18.88	3.61	0.55	15.27
Steven Stamkos	19.29	3.06	0.51	16.71
Ilya Kovalchuk	21.29	3.45	0.47	13.63
Connor McDavid	21.42	3.07	0.46	15.15
Ziggy Palffy	20.74	3.03	0.46	15.30
Sidney Crosby	20.81	3.19	0.46	14.48
David Pastrnak	17.43	3.16	0.46	14.50
Mario Lemieux	22.67	3.41	0.45	13.30
Evgeni Malkin	19.90	3.34	0.45	13.33
Patrik Laine	17.70	3.02	0.44	14.70
Total	**18.79**	**2.97**	**0.39**	**13.30**

Height
Average of height_cm and AvgOfGoals by player_id
AvgOfGoals
Average of height_cm
Weight
Sum of weight_kg and AvgOfGoals by player_id
Sum of weight_kg
Age
Average of Age and AvgOfGoals by player_id
Average of Age
Page 1
Page 2
Page 3a
Page 3b
Page 4
+

Common traits among top performers include:

<u>Performance Metric</u>
Top performers has better performance metrics than average player as shown below:

	Avg min on Ice	Average of Shot	Average of Goal	Shot conversion Rate (%)
Average Player	16.47	1.66	0.15	9.27
Top 100 Player	18.78	3.01	0.41	13.46

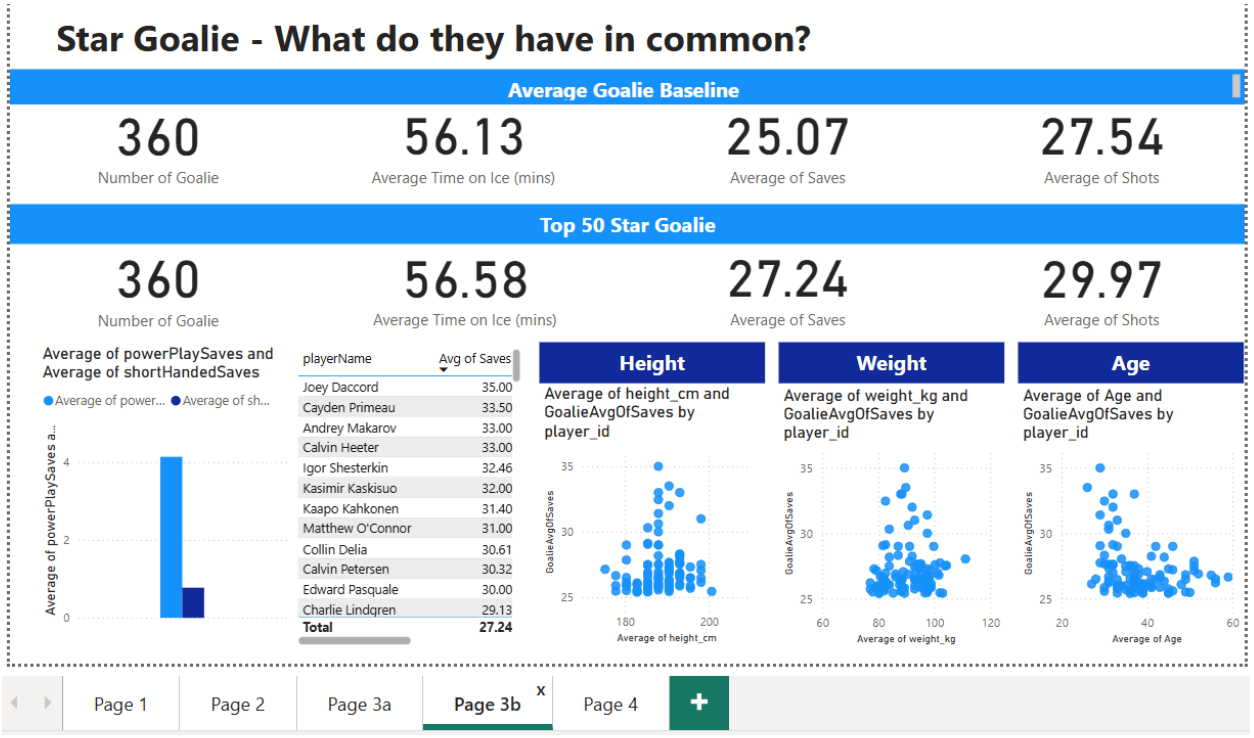
<u>Physical Attributes</u>

- Height Distribution: The scatter plot shows a concentrated distribution where top player heights cluster around 180-190 cm. A positive correlation exists between height and average goals scored.

- Weight Distribution: Top player weights show a balanced distribution. The relationship between weight and goal-scoring ability appears moderate

- Age Distribution: Top player age shows a balanced distribution. Top performers span across different age groups (20-60 years). The scatter plot shows consistent performance across various age groups

Page 3b: Top 50 Goalies Analysis



Star Goalie - What do they have in common?

Average Goalie Baseline

360	56.13	25.07	27.54
Number of Goalie	Average Time on Ice (mins)	Average of Saves	Average of Shots

Top 50 Star Goalie

360	56.58	27.24	29.97
Number of Goalie	Average Time on Ice (mins)	Average of Saves	Average of Shots

playerName	Avg of Saves
Joey Daccord	35.00
Cayden Primeau	33.50
Andrey Makarov	33.00
Calvin Heeter	33.00
Igor Shesterkin	32.46
Kasimir Kaskisuo	32.00
Kaapo Kahkonen	31.40
Matthew O'Connor	31.00
Collin Delia	30.61
Calvin Petersen	30.32
Edward Pasquale	30.00
Charlie Lindgren	29.13
Total	27.24

Average of powerPlaySaves and Average of shortHandedSaves
● Average of power... ● Average of sh...
Average of powerPlaySaves a...
0
2
4
Height
Average of height_cm and GoalieAvgOfSaves by player_id
GoalieAvgOfSaves
25
30
35
180
200
Average of height_cm
Weight
Average of weight_kg and GoalieAvgOfSaves by player_id
60
80
100
120
Average of weight_kg
Age
Average of Age and GoalieAvgOfSaves by player_id
20
40
60
Average of Age
Page 1
Page 2
Page 3a
Page 3b
Page 4
+

We analyzed 360 goalies and found common traits among top goalies include:

<u>Performance Metric</u>
Top 50 goalies has save percentage higher than average goalie, with The top 10 goalies shown all average above 30 saves per game:

	Avg min on Ice	Average of Goal	Average of Shot
Average Goalie	56.13	25.07	27.54
Top 50 Goalie	56.58	27.24	29.97

<u>Physical Attributes</u>

- Height Distribution: The scatter plot shows a concentrated distribution where top player heights cluster around 185-190 cm. A positive correlation exists between height and average save. Optimal height range evident for maximum save efficiency

- Weight Distribution: While data points concentrate between 80-100 kg, it is quite a balanced distribution. The relationship between weight and save ability appears moderate

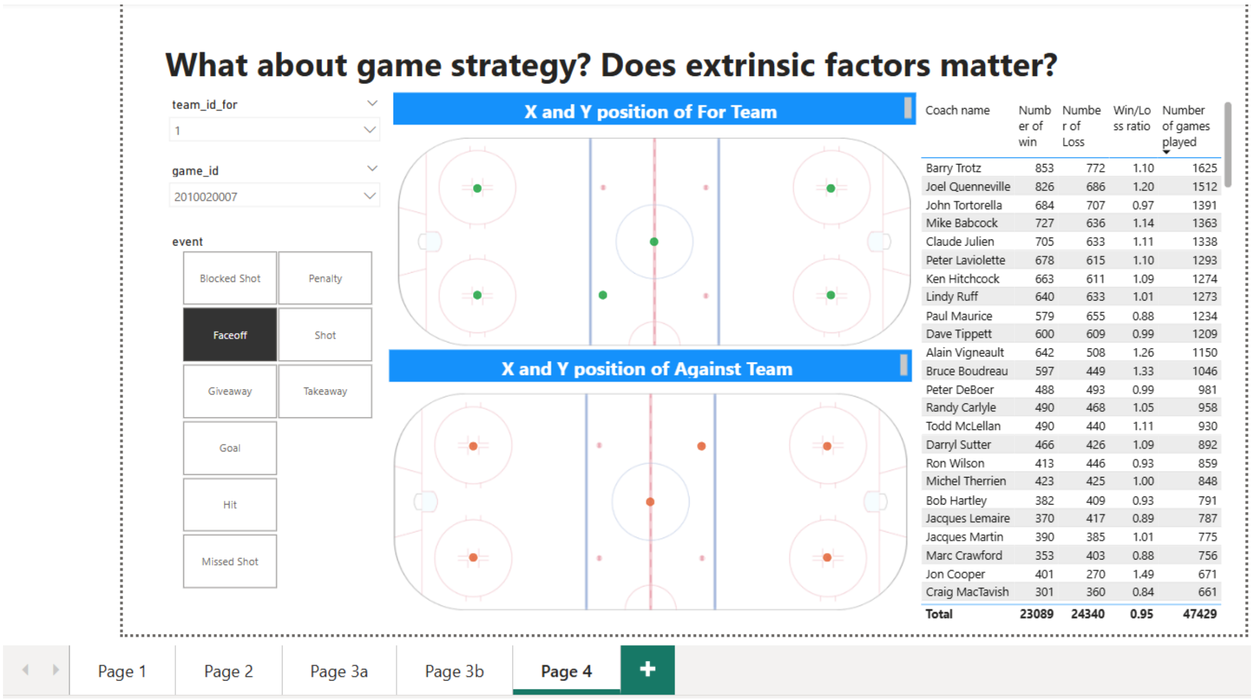
- Age Distribution: Age distribution spans from approximately 20-60 years. Peak performance appears in the 25-35 age range for Goalie. This is perhaps due to better stamina among younger goalie to endure 56 minutes on ice.

<u>Special event</u>

We analyzed performance in high-pressure situations, such as PowerPlay and Shorthand scenarios, to understand consistency of save across different game scenarios.

- Power play saves significantly higher than short-handed saves
- Clear difference in save patterns between different game situations
- This could be due to psychological stress during short-hand period, leading to poorer save rates

Page 4: Game Strategy Analysis



What about game strategy? Does extrinsic factors matter?

team_id_for: 1

game_id: 2010020007

event: Blocked Shot, Penalty, Faceoff, Shot, Giveaway, Takeaway, Goal, Hit, Missed Shot

X and Y position of For Team

X and Y position of Against Team

Coach name	Number of win	Number of Loss	Win/Loss ratio	Number of games played
Barry Trotz	853	772	1.10	1625
Joel Quenneville	826	686	1.20	1512
John Tortorella	684	707	0.97	1391
Mike Babcock	727	636	1.14	1363
Claude Julien	705	633	1.11	1338
Peter Laviolette	678	615	1.10	1293
Ken Hitchcock	663	611	1.09	1274
Lindy Ruff	640	633	1.01	1273
Paul Maurice	579	655	0.88	1234
Dave Tippett	600	609	0.99	1209
Alain Vigneault	642	508	1.26	1150
Bruce Boudreau	597	449	1.33	1046
Peter DeBoer	488	493	0.99	981
Randy Carlyle	490	468	1.05	958
Todd McLellan	490	440	1.11	930
Darryl Sutter	466	426	1.09	892
Ron Wilson	413	446	0.93	859
Michel Therrien	423	425	1.00	848
Bob Hartley	382	409	0.93	791
Jacques Lemaire	370	417	0.89	787
Jacques Martin	390	385	1.01	775
Marc Crawford	353	403	0.88	756
Jon Cooper	401	270	1.49	671
Craig MacTavish	301	360	0.84	661
Total	**23089**	**24340**	**0.95**	**47429**

In this dashboard, we aimed to analyse the strategic elements that impact game outcomes, by looking into the x, y positions of players in order to understand:

- Shot location significantly affects goal-scoring probability
- Teams should focus on optimizing shot placement and angles on the ice
- Adapting strategies for home vs. away games could be crucial, given the scoring differences

- Player positioning and movement patterns play a vital role in creating scoring opportunities

Despite our initial objectives to conduct comprehensive analysis of shot dynamics and game strategies, we encountered technical limitations that constrained our analytical capabilities. The current dashboard presents a simplified version of our intended analysis, focusing primarily on basic X-Y position plotting and coaching statistics. Future iterations of this analysis would benefit from enhanced technical capabilities to fully explore these critical aspects of NHL game strategy.

<u>Position Analysis</u>

For team:

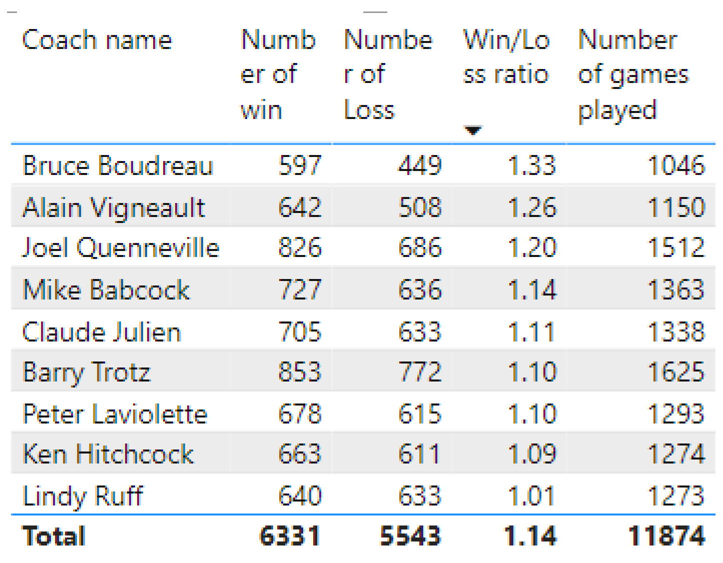
- The X and Y position scatter plot shows clustering for different types of events
- There's a clear pattern in the average X and Y coordinates for each type of event

Against team:

- The scatter plot demonstrates comparable clustering to the "For Team" analysis
- Shot locations appear to be consistent regardless of team position, with similar positioning patterns

<u>Coaching Impact</u>

To get a better understanding of a good coach, we filter coaches that have played more than 1000 games (veteran coach) and have a win-loss ratio of more than 1.00.
The results are shown in the table below:



Coach name	Number of win	Number of Loss	Win/Loss ratio	Number of games played
Bruce Boudreau	597	449	1.33	1046
Alain Vigneault	642	508	1.26	1150
Joel Quenneville	826	686	1.20	1512
Mike Babcock	727	636	1.14	1363
Claude Julien	705	633	1.11	1338
Barry Trotz	853	772	1.10	1625
Peter Laviolette	678	615	1.10	1293
Ken Hitchcock	663	611	1.09	1274
Lindy Ruff	640	633	1.01	1273
Total	**6331**	**5543**	**1.14**	**11874**

- Bruce Boudreau has the highest win rate of 1.33
- Lindy Ruff have the lowest win rate of just 1.01
- Mean win rate is 1.14

Although Bruce has the highest win rate among the others whose teams played over 1000 games, he has the least games played among the others.
However this does not translate to having a good team as there are also other coaches that are less experienced (<1000 games) but have better win ratio than the more experienced.
There could be variables like team cohesion, coaching style for the team, etc.
In conclusion, this dashboard provides a comprehensive view of NHL player demographics, performance metrics, and strategic considerations, offering valuable insights for teams, analysts, and fans alike.

Challenges

Complexity and Volume of Data

The sheer volume and complexity of sports data was overwhelming, presenting challenges like:

- Analysis paralysis, where we struggle to make sense of the 13 dataset to extract meaningful insights
- Difficulty in prioritizing key performance indicators, such as goal, shots, goal conversion rate
- Challenges in presenting data in an intuitive, easily digestible way

Visual Design and User Experience

Crafting effective visualizations posed several challenges:

- Balancing the amount of information presented without overwhelming users
- Designing intuitive and easily understood visualizations for various stakeholders
- Creating interactive elements that allow users to explore data dynamically

Advanced Analytics Implementation

Incorporating advanced analytics features presented challenges:

- Implementing complex calculations and metrics specific to hockey analytics
- Utilizing DAX (Data Analysis Expressions) for custom calculations and measures. For example, the team faced several technical problems while trying to filter out the top 100 players and top 50 goalie based on performance.
- Creating dynamic visuals that update based on user interactions and filters

Solutions

Get Help Early

Reach out to peers with technical backgrounds for guidance on approaching visualizations and coding:

- Consult teammate who have experience with data visualization tools and techniques
- Join online communities or forums dedicated to data visualization to ask questions and get advice

- Participate in local meetups or user groups focused on data analytics and visualization

Start Simple

Begin with basic visualizations and gradually incorporate more complex elements:

- Master fundamental chart types like bar charts, line graphs, and scatter plots before moving to advanced visualizations
- Focus on clearly communicating one or two key insights per visualization initially
- Incrementally add interactive elements or multiple layers of data as your skills improve

Seek Feedback

Regularly solicit feedback to refine your visualizations and improve clarity:

- Share drafts with colleagues and ask for specific input on readability and effectiveness
- Present visualizations to non-technical stakeholders to ensure they are easily understood
- Iterate based on feedback, making incremental improvements to enhance impact

Utilize Resources

Leverage online resources, tutorials, and communities for guidance and support:

- Take advantage of free courses on platforms like DataCamp to learn new visualization techniques
- Explore galleries of example visualizations on sites like the Data Visualization Catalogue for inspiration

Future Considerations

Future Considerations for enhancing the NHL data analysis project include:

Machine Learning Integration

Implement machine learning techniques to extract deeper insights from the NHL dataset:

- Use supervised learning algorithms like Support Vector Machines (SVM) and Random Forests to predict game outcomes and player performance
- Apply unsupervised learning methods to identify tactical patterns in gameplay, similar to the approach used in football analytics
- Develop AI-powered deep neural networks to analyze players' postures, movements, and interactions on the ice from video data

Data Acquisition Improvements

- Implement an HTTP connector in Azure Data Factory (ADF) to interact with the Kaggle API, automating dataset updates and reducing manual intervention
- Explore real-time data acquisition methods, similar to those used in soccer, to capture live player and puck tracking data during games

Pipeline Optimization

- Utilize flowlets and parameterization in ADF to create reusable data flows, improving efficiency and maintainability of the data pipeline
- Implement a machine learning pipeline that includes data preprocessing, feature engineering, model training, and evaluation stages

Advanced Analytics

- Develop an Expected Goals (xG) model for ice hockey, similar to those used in soccer analytics, to better quantify shot quality
- Create a player load monitoring system using machine learning to help prevent injuries and optimize performance

By incorporating these considerations, the NHL data analysis project can evolve into a more sophisticated, automated, and insightful system, providing valuable information for teams, players, and management.

References

Marsh, James H.(2012) *National Hockey League (NHL)*, in Marshall, T. (ed.). Available at: https://www.thecanadianencyclopedia.ca/en/article/national-hockey-league/ (Accessed 20 January 2025).

Brewer. A. (2024) *What's wrong with the NHL's pre-season and ideas to fix it*. Available at: https://www.sportsnet.ca/nhl/article/whats-wrong-with-the-nhls-pre-season-and-ideas-to-fix-it/ (Accessed 21 January 2025)

Cheap Seat Sports (2021) How Seasons, Standings and Playoffs work in the NHL | NHL 101. Available at: https://www.youtube.com/watch?v=IkQ1L3qVNZo/ (Accessed

NHL (n.d). *Playoff format*. Available at: https://records.nhl.com/history/playoff-formats/ (Accessed 20 January 2025)

Ellis, M. (2019, updated in 2021) ‘NHL Game Data’. Available at: https://www.kaggle.com/datasets/martinellis/nhl-game-data/data/ (Accessed 10 December 2024)