

**A MINI PROJECT**

**PYSPARK AND AZURE  
HOCKEY ANALYSIS PROJECT REPORT**

**DOCUMENTATION BY  
IYSHWARYA RAMESH  
RAMITHA R M  
SANJANA B L**

# TABLE OF CONTENTS

S.NO	TITLE
1	<b>Introduction</b>
2	<b>Problem Statement</b>  <b>2.1 Objective</b>  <b>2.2 Abstract of the Project</b>  <b>2.3 CSV Files Used</b>
3	<b>Implementation</b>  <b>3.1 Work Overview</b>  <b>3.2 Project Flow &amp; Data Warehouse Schema</b>  <b>3.3 Data Transformation Process</b>
4	<b>Reports + Output CSV Files</b>
5	<b>Conclusion</b>

## 1. INTRODUCTION

The purpose of this mini-project is to design, develop, and implement a comprehensive data analysis and reporting system aimed at delivering meaningful insights into hockey game statistics using PySpark. This project focuses on extracting operational data from multiple flat file sources, performing systematic data transformation and cleaning processes, and structuring it into an optimized dimensional data warehouse model following a star schema approach.

Through this framework, the system facilitates in-depth analysis of player and team performances across various leagues, seasons, and match types. By leveraging the power of distributed data processing with PySpark, the project efficiently handles large-scale datasets, performs complex transformations, and generates insightful reports that support strategic decision-making for coaches, analysts, and sports management teams.

The project not only enhances data handling and reporting capabilities but also serves as a practical exercise in applying real-world data engineering techniques using PySpark in a structured ETL (Extract, Transform, Load) pipeline, culminating in the generation of comprehensive reports in CSV format for further visualization and interpretation.

## **2. PROBLEM STATEMENT**

### **2.1 OBJECTIVE**

To develop an Analysis & Reporting System for Hockey Games using PySpark within an Azure-supported environment. The system aims to manage, analyze, and report operational hockey game data — covering player performance statistics, club records, and match outcomes — by transforming raw operational data from multiple flat files into structured, analytical reports that support effective decision-making.

### **2.2 ABSTRACT OF THE PROJECT**

This project involves building an end-to-end data processing and reporting pipeline using PySpark to handle diverse operational hockey data, including player profiles, team performances, league records, and match results. The raw data, collected from heterogeneous flat file sources, undergoes a systematic Extract, Load, and Transform (ELT) process to convert it into a well-defined dimensional data warehouse model.

A star schema is designed, comprising various dimension tables (such as Player, Club, Country, League, Match Type, and Time) and fact tables for capturing key performance metrics. The transformed data is then used to generate a set of insightful, business-relevant reports, including top goal scorers, maximum appearances, disciplinary records, and club performance rankings.

The system facilitates efficient, scalable processing of large datasets and demonstrates practical application of PySpark's distributed computing capabilities in building a real-time sports analytics reporting solution.

## **2.3 CSV FILES USED**



Hockey\_Src1.csv

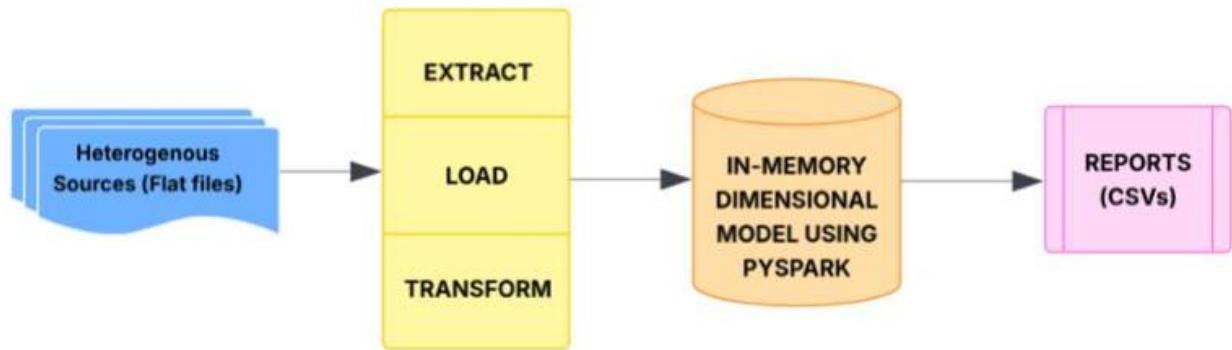


Hockey\_Src2.csv



DIM.Date.Table.csv

### 3. IMPLEMENTATION



#### 3.1 WORK OVERVIEW:

1. Data Collection & Loading
2. Data Cleaning & Preprocessing
3. Data Analysis
4. Reporting & Visualization
5. Final Summary

**The source files are:**

#### SOURCE FILE 1: PLAYER STATISTICS

- ❖ Year
- ❖ Match
- ❖ Player Name
- ❖ Club Name
- ❖ Country Name
- ❖ Appearances, Goals scored, Goals assist, Shots, Fouls, Cards, Salary, etc.

## SOURCE FILE 2: TEAM STATISTICS

- ❖ Year
- ❖ Match Name
- ❖ League Name
- ❖ Club Name
- ❖ Country Name
- ❖ Appearances, Wins, Losses, Drawn, Clean Sheets, Net Worth
- ❖ Date Dimension.

## 3.2 PROJECT FLOW & DATA WAREHOUSE SCHEMA

### PROJECT WORKFLOW:

- ❖ Operational data from heterogeneous sources (CSV files)
- ❖ Extraction, Loading, and Transformation using PySpark
- ❖ In-memory dimensional model creation
- ❖ Report generation in CSV format

### DATA WAREHOUSE MODEL:

#### DIMENSION TABLES

##### ❖ League:

League\_ID  
League\_Name.

##### ❖ Club:

Club\_ID  
Club\_Name  
League\_ID  
Manager  
Owner

❖ **Country:**

Country\_ID

Country\_Name

Coach

❖ **Match Type:**

Match\_Type\_ID

Match\_Name

❖ **Player:**

Player\_ID

Player\_Name

Club\_ID

Country\_ID

Position

Jersey No

DOB

Nationality

❖ **Time:**

Time\_ID

Year

Quarter

Month

## FACT TABLES

### Fact\_Player\_Statistics:

- Fact\_ID
- Time\_ID
- Match\_Type\_ID
- Player\_ID

- Club\_ID
- Appearances
- Goals
- Shots
- Fouls
- Salary, etc.

#### **Fact\_Team\_Statistics:**

- Fact\_ID
- Time\_ID
- League\_ID
- Club\_ID
- Wins
- Losses
- Drawn
- Clean Sheets
- Net Worth, etc.

### **3.3 DATA TRANSFORMATION PROCESS**

#### **STEPS:**

- ❖ Load source CSV files into PySpark DataFrames
- ❖ Clean and preprocess the data (null handling, trimming, datatype conversion)
- ❖ Create unique IDs for each dimension table using ranking functions
- ❖ Populate Dimension tables first
- ❖ Generate Fact tables by joining respective dimension IDs
- ❖ Create analytical queries to produce the required reports
- ❖ Export reports as CSV files

## **STEPS PERFORMED:**

### **STEP 1: AZURE RESOURCE SETUP**

- Create Azure Storage Account
- Create Containers

### **STEP 2: UPLOAD FILES TO BLOB STORAGE**

### **STEP 3: INGEST SOURCE FILES USING AZURE DATA FACTORY**

#### **(ADF)**

- Create Azure Data factory
- Open ADF Studio
- Create Linked services
- Create datasets
- Create a pipeline to copy data
- Run the pipeline

### **STEP 4: CREATE AZURE SQL DATABASE**

- Create SQL Database on Azure
- Configure firewall
- Connect to SQL database
- Create table structure

## **STEP 5: REGISTERED AN APP MOUNTED THE AZURE DATA LAKE**

### **STORAGE (ADLS) CONTAINER TO AZURE DATABRICKS**

## **STEP 6: CREATE AZURE DATABRICKS WORKSPACE**

## **STEP 7: TRANSFORMATION IN DATABRICKS**

- Read CSVs
- Create dimensions
- Create fact tables
- Write transformed tables to output

## **STEP 8: LOAD TO SQL USING AZURE DATA FRACTORY (ADF)**

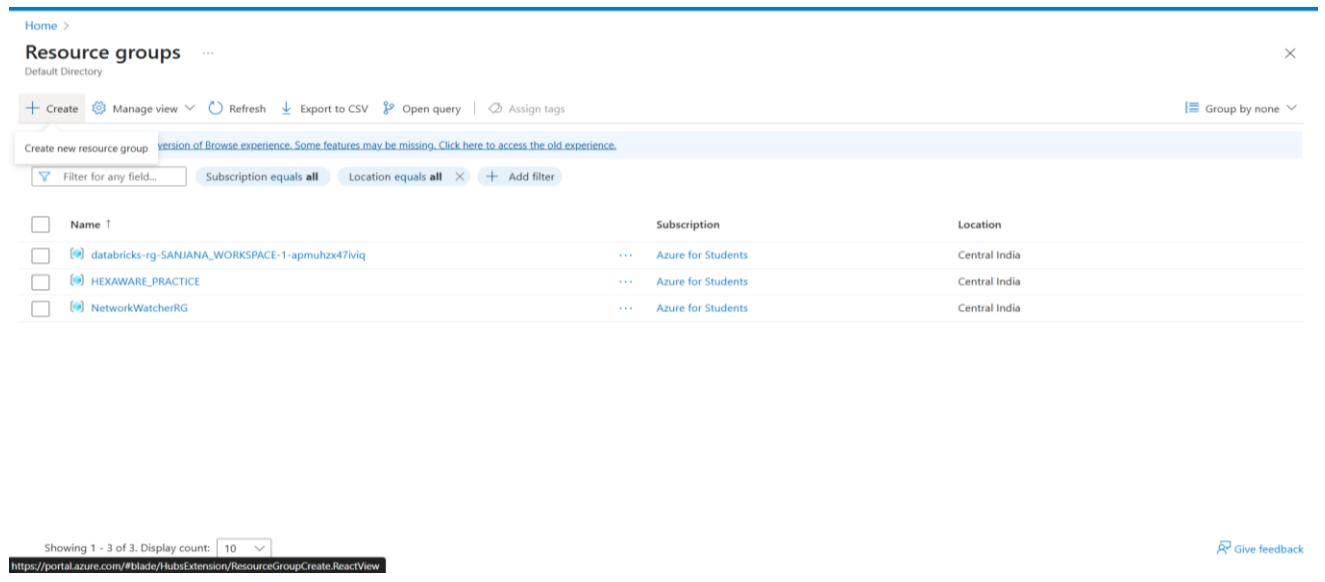
- Create linked services
- Create datasets (Input + Output)
- Build the copy data pipeline
- Run the pipeline
- Verify in Azure SQL

## **STEP 9: SQL REPORTING QUERIES**

## **STEP 10: CSV OUTPUT**

# SCREENSHOTS

## STEP 1: AZURE RESOURCE SETUP

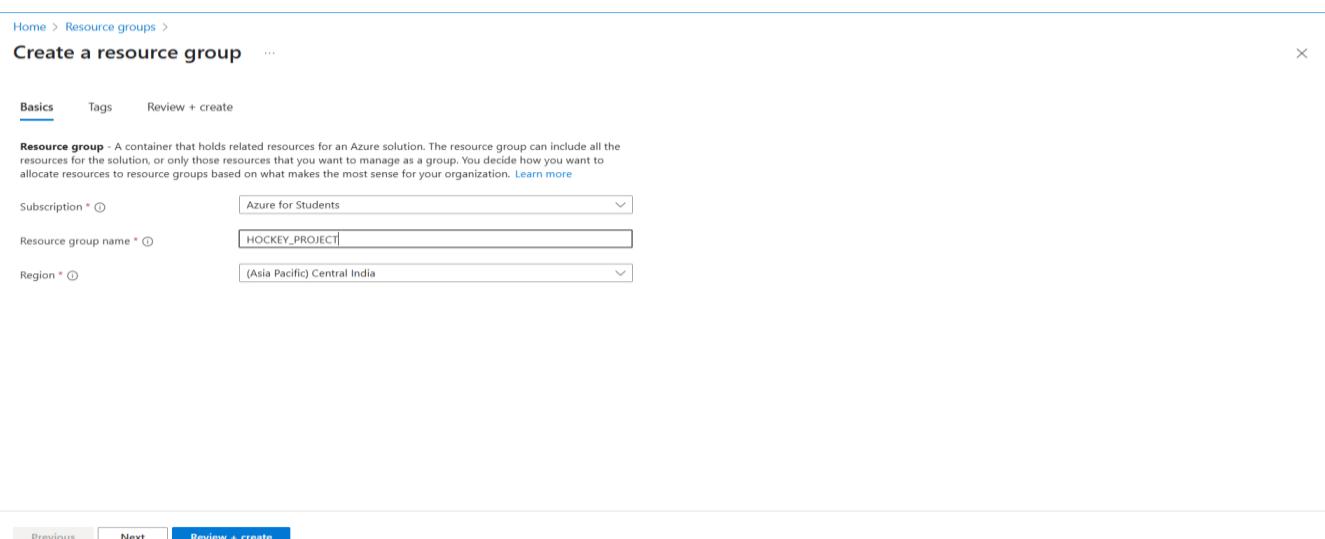


Showing 1 - 3 of 3. Display count: 10

<https://portal.azure.com/#blade/HubsExtension/ResourceGroupCreateReactView>

[Give feedback](#)

---



Basics Tags Review + create

Resource group - A container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. [Learn more](#)

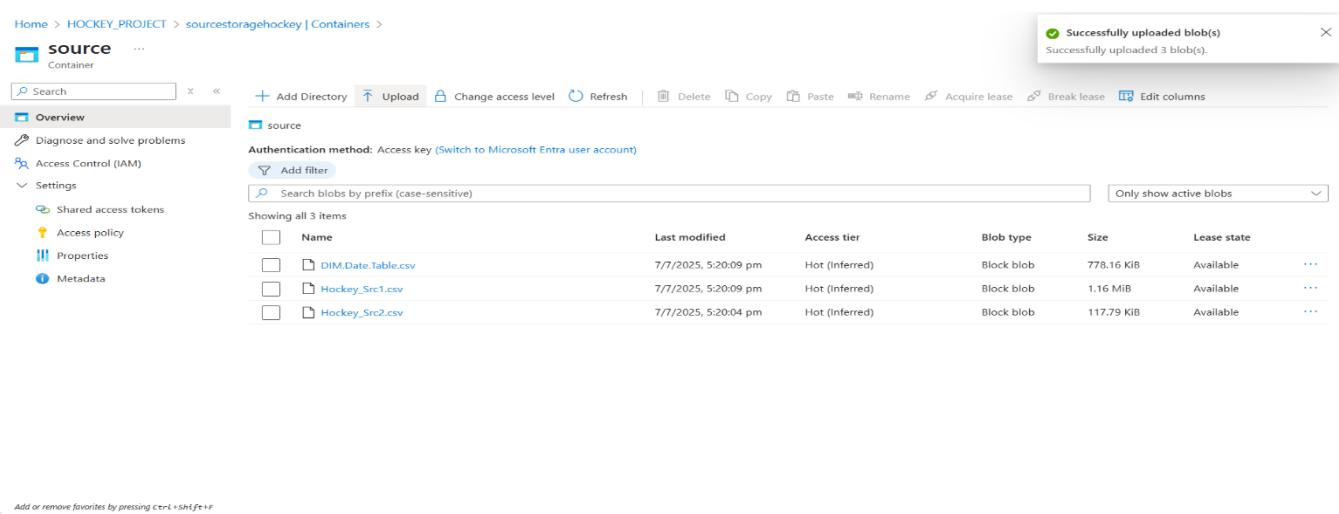
Subscription \* (Azure for Students)

Resource group name \* (HOCKEY\_PROJECT)

Region \* (Asia Pacific) Central India

Previous Next Review + create

## STEP 2: UPLOAD FILES TO BLOB STORAGE



Home > HOCKEY\_PROJECT > sourcestoragehockey | Containers >

source ... Container

Search Add Directory Upload Change access level Refresh Delete Copy Paste Rename Acquire lease Break lease Edit columns

Overview

Diagnose and solve problems

Access Control (IAM)

Settings

Shared access tokens

Access policy

Properties

Metadata

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

Add filter

Search blobs by prefix (case-sensitive)

Showing all 3 items

Name	Last modified	Access tier	Block type	Size	Lease state
DLM.Date.Table.csv	7/7/2025, 5:20:09 pm	Hot (Inferred)	Block blob	778.16 KiB	Available
Hockey_src1.csv	7/7/2025, 5:20:09 pm	Hot (Inferred)	Block blob	1.16 MiB	Available
Hockey_src2.csv	7/7/2025, 5:20:04 pm	Hot (Inferred)	Block blob	117.79 KiB	Available

Successfully uploaded blob(s)  
Successfully uploaded 3 blob(s).

Add or remove favorites by pressing **Ctrl+L+Shift+F+T**

# STEP 3: INGEST SOURCE FILES USING AZURE DATA FACTORY (ADF)

## Create Azure Data factory

The screenshot shows the Azure Data Factory Studio Overview page for a factory named "ADF-HOCKEY-ANALYSIS". The page includes a search bar, a delete button, and a JSON View link. The "Essentials" section displays resource group (HOCKEY\_PROJECT), status (Succeeded), location (Central India), and subscription information (Azure for Students). Below this is a large blue icon of a factory building, followed by the text "Azure Data Factory Studio" and a "Launch studio" button. At the bottom, there are four cards: "Quick Starts" (cloud icon), "Tutorials" (book icon), "Template Gallery" (document icon), and "Training Modules" (certificate icon).

## Create Linked Services

The screenshot shows the "New linked service" dialog in the Azure Data Factory studio. The "Data store" tab is selected, showing a grid of connection options. The "Azure" tab is highlighted. Options include Azure AI Search, Azure Blob Storage, Azure Cosmos DB for MongoDB, Azure Cosmos DB for NoSQL, Azure Data Explorer (Kusto), Azure Data Lake Storage Gen2, and Azure Synapse Analytics. Below the grid are "Continue" and "Cancel" buttons.

The screenshot shows the "New linked service" dialog with specific details filled in. The "Name" field is set to "AzureBlobStorage". The "Connect via integration runtime" dropdown is set to "AutoResolveIntegrationRuntime". The "Authentication type" dropdown is set to "Account key". The "Azure subscription" dropdown is set to "Azure for Students (d349bf7d-203a-43a4-a4d1-47f12f24bc01)". The "Storage account name" dropdown is set to "sourcestoragehockey". At the bottom, there are "Create", "Back", "Test connection", and "Cancel" buttons.

**New linked service**

Data store Compute

All Azure Database File Generic protocol

Azure AI Search	Azure Blob Storage	Azure Cosmos DB for MongoDB
Azure Cosmos DB for NoSQL	Azure Data Explorer (Kusto)	Azure Data Lake Storage Gen2
My		A

Continue Cancel

**New linked service**

Azure Data Lake Storage Gen2 Learn more

Name \* AzureDataLakeStorage

Description

Connect via integration runtime  AutoResolveIntegrationRuntime

Authentication type Account key

Account selection method  From Azure subscription  Enter manually

Azure subscription Azure for Students (d349bf7d-203a-43a4-a4d1-47f1f24bc01)

Storage account name \* sinkstoragehockey

Test connection  To linked service  To file path

Annotations

Create Back Test connection Cancel

## Create datasets

**DS\_Hockey\_Src1 (Similarly For DS\_Hockey\_Src2, DS\_Date\_Table)**

**Set properties**

Name DS\_Hockey\_Src1

Linked service \* AzureBlobStorage

File path source / Directory / Hockey\_src1.csv

First row as header

Import schema  From connection/store  From sample file  None

OK Back Cancel

### 3 Destination Datasets (For Raw Container)

**DS\_Hockey\_Src1\_ADLS** (Similarly For DS\_Hockey\_Src2\_ADLS, DS\_Date\_Table\_ADLS)

The screenshot shows the Azure Data Factory Author interface. On the left sidebar, under 'Factory Resources', 'Datasets' is selected, showing four datasets: DS\_Date\_Table, DS\_Hockey\_Src1, DS\_Hockey\_Src1\_ADLS, and DS\_Hockey\_Src2. The 'DS\_Hockey\_Src1\_ADLS' dataset is currently selected. The main pane displays its properties: it is a 'DelimitedText' type with a 'CSV' icon. The 'Connection' tab shows it is connected to an 'AzureDataLakeStorage' linked service. The 'File path' is set to 'raw/Directory/File'. Other settings include 'No compression' for compression type, 'Comma (,) for column delimiter, 'Default (\r\n or \n\r)' for row delimiter, 'Default(UTF-8)' for encoding, 'Double quote (")' for quote character, and 'Backslash (\)' for escape character.

Create and run the Pipeline to Copy Data

The screenshot shows the Azure Data Factory Author interface. On the left sidebar, under 'Activities', 'Pipelines' is selected, showing one pipeline named 'PI\_Copy\_Hockey\_Data'. The main pane shows the pipeline's structure: it has three activities: 'Copy data' (Copy\_Src1), 'Copy data' (Copy\_Src2), and 'Copy data' (Copy\_Date). The Copy\_Src1 and Copy\_Src2 activities are grouped together and connected to the Copy\_Date activity.

The screenshot shows the Azure Data Factory Author interface. On the left sidebar, under 'Activities', 'Pipelines' is selected, showing one pipeline named 'PI\_Copy\_Hockey\_Data'. The main pane displays the execution results: the pipeline ran successfully with three items. The table shows the following details:

Activity name	Activity st...	Activit...	Run start	Duration	Integra...
Copy_Date	Succeeded	Copy data	7/7/2025, 6:53:30 PM	13s	AutoRe...
Copy_Src1	Succeeded	Copy data	7/7/2025, 6:53:30 PM	12s	AutoRe...
Copy_Src2	Succeeded	Copy data	7/7/2025, 6:53:30 PM	13s	AutoRe...

## STEP 4: CREATE AZURE SQL DATABASE

The screenshot shows the 'Create SQL Database Server' page. In the 'Server details' section, 'Server name' is set to 'hockey-sever' and 'Location' is '(Asia Pacific) Central India'. Under 'Authentication', it says 'Azure Active Directory (Azure AD) is now Microsoft Entra ID.' Below this, there are three authentication method options: 'Use Microsoft Entra-only authentication', 'Use both SQL and Microsoft Entra authentication', and 'Use SQL authentication', which is selected. 'Server admin login' is 'sanjana', and 'Password' and 'Confirm password' are both '\*\*\*\*\*'. At the bottom right, there is an 'OK' button and a 'Feedback' link.

## Configure firewall

The screenshot shows the 'Networking' blade for the 'hockey-sever' SQL server. Under 'Public access', 'Selected networks' is selected. In the 'Firewall rules' section, a new rule is being added for 'ClientIPAddress\_2025-7-9-8-11-4' with 'Start IPv4 address' '157.49.106.250' and 'End IPv4 address' '157.49.106.250'. A success message at the top right says 'Successfully updated server firewall rules'.

## Connect to SQL Database

The screenshot shows the 'Query editor (preview)' for the 'HOCKEY\_DATABASE' database. It displays a login screen with two authentication methods: 'SQL server authentication' (Login: sanjana, Password: \*\*\*\*\*) and 'Microsoft Entra authentication' (Continue as sanjanatalalp12@gmail.com). There are 'OK' buttons for both methods.

## Create table structures

```
20 CREATE TABLE dim_match_type (
21     Match_Type_ID INT PRIMARY KEY,
22     Match_Name NVARCHAR(100)
23 );
24
25
26 -- PLAYER TABLE
27 CREATE TABLE dim_player (
28     Player_ID INT PRIMARY KEY,
29     Player_Name NVARCHAR(100),
30     Club_Name NVARCHAR(100),
31     Country_Name NVARCHAR(100),
32     Position NVARCHAR(50),
33     Jersey_Number INT,
34     DOB DATE,
35     Nationality NVARCHAR(50),
36     Club_ID INT,
37     Country_ID INT
38 );
39
40
41 -- TIME TABLE
42 CREATE TABLE dim_time (
43     Time_ID INT PRIMARY KEY,
44     Date_Key NVARCHAR(50),
45     Year INT,
46     Quarter NVARCHAR(10),
47     Month INT
48 );
49
50
51 -- CREATING THE FACT TABLES
52 -- FACT_PLAYER TABLE
53 CREATE TABLE fact_player_statistics (
54     Fact_ID INT PRIMARY KEY,
55     Player_ID INT,
56     Time_ID INT,
57     Score INT,
58     Assists INT,
59     Goals INT
60 );
```

Add or remove favorites by pressing **Ctrl+Shift+F**

## STEP 5: USING ACCESS KEY MOUNTED THE AZURE DATA LAKE STORAGE (ADLS) CONTAINER TO AZURE DATABRICKS

Access keys authenticate your applications' requests to this storage account. Keep your keys in a secure location like Azure Key Vault, and replace them often with new keys. The two keys allow you to replace one while still using the other.

Remember to update the keys with any Azure resources and apps that use this storage account.  
Learn more about managing storage account access keys [↗](#)

Storage account name: sinkstoragehockey

key1 Rotate key  
Last rotated: 7/7/2025 (1 days ago)  
Key: niSy4z8lDc1tDxbpzsmHssGJ7JUFqM6VX7gOTuqUKPw8pKPrwyQjeo1c2LyGekbel... Copied

Connection string:

key2 Rotate key  
Last rotated: 7/7/2025 (1 days ago)  
Key:   
Connection string:

Add or remove favorites by pressing **Ctrl+Shift+F**

## STEP 6: CREATE AZURE DATABRICKS WORKSPACE

Home > HOCKEY\_PROJECT\_HOCKEY\_ANALYSIS\_WORKSPACE | Overview >

**HOCKEY\_ANALYSIS\_WORKSPACE** ...

Azure Databricks Service

Overview

Status: Active

Resource group: HOCKEY\_PROJECT

Location: Central India

Subscription: Azure for Students

Subscription ID: d349bf7d-203a-43a4-a4d1-47f12f24bc01

Tags (edit): Add tags

Managed Resource Group: databricks-rg-HOCKEY\_ANALYSIS\_WORKSPACE-6sqbmt056thf0

URL: <https://adb-4496951497978655.15.azuredatabricks.net>

Pricing Tier: Standard (Apache Spark, Secure with Microsoft Entra ID) (Click to...)

Enable No Public IP: Yes

Essentials

Launch Workspace   
Upgrade to Premium

Documentation Getting Started Import Data from File Import Data from Azure Storage

Add or remove favorites by pressing **Ctrl+Shift+F**

## STEP 7: TRANSFORMATION IN DATABRICKS

### Read CSVs

```

# OUTPUT CONTAINER MOUNT POINT
configs = {
    "fs.azure.account.key.sinkstoragehockey.blob.core.windows.net": "niSy4z8IDCtDxbpzmsissG77UfQh6VX7gOTuqUKP68pKpmeyOjeo1c2LyGekbeIWJ9xvsGhx2+AtqlWf5jg=="
}
dbutils.fs.mount(
    source = "wasbs://output@sinkstoragehockey.blob.core.windows.net/",
    mount_point = "/mnt/mount",
    extra_configs=configs
)

# Read CSV files
df_src1 = spark.read.option("header", True).csv("/mnt/mount/Hockey_Src1.csv")
df_src2 = spark.read.option("header", True).csv("/mnt/mount/Hockey_Src2.csv")
df_date = spark.read.option("header", True).csv("/mnt/mount/DIM.Date.Table.csv")

df_src1.show()
df_src2.show()
df_date.show()

# Display data
df_date.pyspark.sql.DataFrame = [date_key:string, full_date:string ... 22 more fields]
df_src1.pyspark.sql.DataFrame = [Year:string, Month:string ... 22 more fields]
df_src2.pyspark.sql.DataFrame = [Year:string, Match_Name:string ... 12 more fields]
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|2010| Club| LA Liga| Athletico Madrid| NA| N|Alvin Lucas|Dean Green| 8| 2| 1| 5| 1|$745,634 |
|2010| Club| LA Liga| Athletico Madrid| NA| N|Alvin Lucas|Dean Green| 8| 6| 1| 1| 3|$745,634 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 3 rows
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|date_key| full_date|day_of_week|day_num_in_month|day_num_overall|day_name|day_abbr|weekday_flag|week_num_in_year|week_num_overall| week_begin_date|week_begin_date_key|
fb_month|month_num_overall|month_name|month_abbr|quarter|fb_year|yearmo|fiscal_month|fiscal_quarter|fiscal_year|last_day_in_month_flag|same_day_year_ago_date|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

## Create dimension tables

This screenshot shows a Databricks notebook titled "HOCKEY\_DATA\_ETL\_PROCESS\_1" in Python mode. The notebook contains two sections of code:

```
# LEAGUE DIMENSION
df_league = df_src2.select("League_Name").distinct()

window_spec = Window.orderBy("League_Name")
dim_league = df_league.withColumn("League_ID", row_number().over(window_spec))

dim_league.write.option("header", True).mode("overwrite").csv("/mnt/mount2/dim_league")

# Club Dimension
# Unique Club + League + Manager + Owner
df_club = df_src2.select("Club_Name", "League_Name", "Manager", "Owner").distinct()

# Join League to get League_ID
dim_club = df_club.join(dim_league, on="League_Name", how="left")

# Add Club ID
window_spec = Window.orderBy("Club_Name")
```

This screenshot shows the same notebook with the code expanded to show more details about the data types and schema:

```
# CLUB DIMENSION
# UNIQUE CLUB + LEAGUE + MANAGER + OWNER
df_club = df_src2.select("Club_Name", "League_Name", "Manager", "Owner").distinct()

# JOIN LEAGUE TO GET League_ID
dim_club = df_club.join(dim_league, on="League_Name", how="left")

# ADD Club_ID
window_spec = Window.orderBy("Club_Name")
dim_club = dim_club.withColumn("Club_ID", row_number().over(window_spec))

dim_club.write.option("header", True).mode("overwrite").csv("/mnt/mount2/dim_club")
```

## Create fact tables

This screenshot shows a Databricks notebook titled "HOCKEY\_DATA\_ETL\_PROCESS\_2" in Python mode. The notebook contains complex SQL code for generating fact tables:

```
# JOIN WITH TIME
fact_player = df_src1.join(dim_time.select("Time_ID", "Year"), on="Year", how="left")

# JOIN WITH MATCH TYPE
fact_player = fact_player.join(dim_match_type.select("Match_Type_ID", dim_match_type["Match_Name"].alias("Match")), on="Match", how="left")

# JOIN WITH PLAYER
fact_player = fact_player.join(dim_player.select("Player_ID", "Player_Name"), on="Player_Name", how="left")

# JOIN WITH CLUB
fact_player = fact_player.join(dim_club.select("Club_ID", dim_club["Club_Name"].alias("Club Name")), on="Club Name", how="left")

# JOIN WITH COUNTRY
fact_player = fact_player.join(dim_country.select("Country_ID", dim_country["Country_Name"].alias("Country Name")), on="Country Name", how="left")

# SELECT REQUIRED COLUMNS + GENERATE Fact_ID
window_spec = Window.orderBy("Player_ID")

fact_player = fact_player.withColumn("Fact_ID", row_number().over(window_spec)).select(
    "Fact_ID", "Time_ID", "Match_Type_ID", "Player_ID", "Club_ID", "Country_ID",
    col("Goals_Scored").alias("Goals_Scored"),
    col("Goals_Assist").alias("Goals_Assist"),
    col("Total_Shots").alias("Total_Shots"),
    col("Shots_on_Goal").alias("Shots_on_Goal"),
    col("Shots_on_Target").alias("Shots_on_Target"),
    col("Fouls_Made").alias("Fouls_Made"),
    col("Fouls_Suffered").alias("Fouls_Suffered"),
    col("Yellow_Card").alias("Yellow_Card"),
    col("Goals_Saved").alias("Goals_Saved"),
    col("Goals_Conceded").alias("Goals_Conceded"),
    col("Total_Penalty").alias("Total_Penalty"),
    col("Successful_Penalty").alias("Successful_Penalty"),
    col("Salary").alias("Salary")
)

# WRITE TO ADS
fact_player.write.option("header", True).mode("overwrite").csv("/mnt/mount2/fact_player_statistics")
```

## Write transformed tables to output

The screenshot shows the Azure Storage Explorer interface. A container named 'output' is selected. Inside, there are nine blobs with names like '\$azuretmpfolder\$', 'dim\_club', 'dim\_country', etc., all modified on 9/7/2025. The interface includes standard file operations like Add Directory, Upload, Refresh, Delete, Copy, Paste, Rename, Acquire lease, Break lease, and Edit columns.

## STEP 8: LOAD TO SQL USING AZURE DATA FACTORY (ADF)

### Create linked services

The screenshot shows the 'Linked services' blade in Azure Data Factory. It lists existing connections to 'AzureBlobStorage' and 'AzureDataLakeStorage'. A new connection is being created for 'Azure Data Lake Storage Gen2'. The 'Name' field is set to 'AzureDataLakeStorage2'. The 'Connect via integration runtime' dropdown is set to 'AutoResolveIntegrationRuntime'. The 'Authentication type' is set to 'Account key'. Under 'Account selection method', the radio button is selected for 'From Azure subscription'. The 'Azure subscription' dropdown is set to 'Azure for Students (d349bf7d-203a-43a4-a4d1-47f12f24bc01)'. The 'Storage account name' dropdown is set to 'sinkstoragehockey'. A 'Test connection' button is shown as successful. The 'Create' button is visible at the bottom.

### Create datasets (Input + Output)

The screenshot shows the 'Factory Resources' blade in Azure Data Factory. It lists various resources including Pipelines, Datasets, and Data flows. A dataset named 'DS\_dim\_league\_input' is selected. The 'Connection' tab is active, showing it is connected to 'AzureDataLakeStorage2'. The 'File path' is set to 'output/dim\_league/part-00000-tid-1713500449...'. Other settings include 'No compression', 'Comma (,) as column delimiter', 'Default (\r,\n, or \v\n) as row delimiter', 'Default(UTF-8) as encoding', 'Double quote (") as quote character', and 'Backslash (\) as escape character'. The 'Preview experience' toggle is off.

**NOTE:** Similarly created for other files

## Build the copy data pipeline and Run the pipeline

Activity name	Activity state	Activity type	Run start	Duration	Integration runtime	User
LEAGUE	Succeeded	Copy data	7/9/2025, 9:15:32 AM	14s	AutoResolveIntegrationRuntime (Central India)	
MATCH_TYPE	Succeeded	Copy data	7/9/2025, 9:15:32 AM	14s	AutoResolveIntegrationRuntime (Central India)	
PLAYER	Succeeded	Copy data	7/9/2025, 9:15:32 AM	14s	AutoResolveIntegrationRuntime (Central India)	
CLUB	Succeeded	Copy data	7/9/2025, 9:15:32 AM	14s	AutoResolveIntegrationRuntime (Central India)	
FACT_PLAYER	Succeeded	Copy data	7/9/2025, 9:15:32 AM	10m 33s	AutoResolveIntegrationRuntime (Central India)	
COUNTRY	Succeeded	Copy data	7/9/2025, 9:15:32 AM	14s	AutoResolveIntegrationRuntime (Central India)	
FACT_TEAM	Succeeded	Copy data	7/9/2025, 9:15:32 AM	36s	AutoResolveIntegrationRuntime (Central India)	
TIME	Succeeded	Copy data	7/9/2025, 9:15:32 AM	14s	AutoResolveIntegrationRuntime (Central India)	

NAME	CHANGE	EXISTING
Pipelines	Load_Transformed_Data... (New)	-
Datasets	DS_dim_league_input (New)	-
	DS_dim_club_input (New)	-
	DS_dim_country_input (New)	-
	DS_dim_match_type_input (New)	-
	DS_dim_player_input (New)	-
	DS_dim_time_input (New)	-
	fact_player_statistics_input (New)	-
	fact_team_statistics_input (New)	-
	DS_dim_league_output (New)	-
	DS_dim_club_output (New)	-
	DS_dim_country_output (New)	-
	DS_dim_match_type_out... (New)	-

## STEP 9: SQL REPORTING QUERIES

## STEP 10: CSV OUTPUT

### QUERY-1: Player report with country and club info

```
-- REPORT 1: PLAYER REPORT WITH COUNTRY & CLUB INFO
SELECT
    p.Player_ID,
    p.Player_Name,
    c.Country_Name,
    cl.Club_Name
FROM
    dim_player p
LEFT JOIN
    dim_country c ON p.Country_ID = c.Country_ID
LEFT JOIN
    dim_club cl ON p.Club_ID = cl.Club_ID;
```

Player_ID	Player_Name	Country_Name	Club_Name
1	Adrian	NA	Real Madrid
2	Adrian	Netherlands	NA
3	Adrian	NA	Real Sociedad
4	Aguero	Italy	NA
5	Aguero	NA	Sevilla FC
6	Aguero	NA	Valencia
7	Alberto M	Brasil	NA
8	Alberto M	NA	Schalke

### QUERY-2: Player with max Appearances in international matches

```
-- REPORT 2: PLAYER WITH MAX APPEARANCES IN INTERNATIONAL MATCHES
WITH fact_with_match AS (
    SELECT
        fps.Player_ID,
        fps.Appearances,
        mt.Match_Name
    FROM
        dbo.fact_player_statistics fps
    LEFT JOIN
        dbo.dim_match_type mt ON fps.Match_Type_ID = mt.Match_Type_ID
    WHERE
        mt.Match_Name = 'International'
)
SELECT
    Player_ID,
    Player_Name,
    Total_Appearances
FROM
    fact_with_match
ORDER BY
    Total_Appearances DESC;
```

Player_ID	Player_Name	Total_Appearances
122	Ezequiel Lavezzi	105120
199	Javier Manquillo	105120
216	Jonas	105120
310	Nani	105120
335	Pedro	105120
439	Ulysses Blair	105120
471	Willian	105120
97	Diego Costa	105120

### QUERY-3: Top 10 goal sources of 2010

```
-- REPORT 3: TOP 10 GOAL SCORES OF 2010
SELECT
    TOP 10
    fp.Player_ID,
    dp.Player_Name,
    SUM(fp.Goals_Scored) AS Total_Goals
FROM
    dbo.fact_player_statistics fp
INNER JOIN
    dbo.dim_time dt ON fp.Time_ID = dt.Time_ID
LEFT JOIN
    dbo.dim_player dp ON fp.Player_ID = dp.Player_ID
WHERE
    dt.Year = 2010
GROUP BY
```

Player_ID	Player_Name	Total_Goals
205	Jerome Boateng	58035
206	Jerome Boateng	58035
207	Jerome Boateng	58035
152	Frederica Cleveland	57670
180	Hugo Lloris	57670
154	Frederica Cleveland	57670
153	Frederica Cleveland	57670

## QUERY-4: Top 5 players with most red cards

The screenshot shows the Microsoft SQL Database Query editor interface. The left sidebar contains navigation links like Home, Overview, Activity log, Tags, Diagnose and solve problems, Query editor (preview), Resource visualizer, Settings, Data management, Integrations, Power Platform, Security, Intelligent performance, Monitoring, Automation, and Help. The main area has tabs for Query 1 through Query 5, with Query 5 selected. Below the tabs are buttons for Run, Cancel query, Save query, Export data as, and Show only Editor. The code for Query 4 is displayed in the editor:

```
1 -- REPORT 4: TOP 5 PLAYERS WITH MOST RED CARDS
2 SELECT TOP 5
3     fps.Player_ID,
4     dp.Player_Name,
5     SUM(fps.Red_Card) AS Total_Red_Cards
6 FROM
7     dbo.fact_player_statistics fps
8 LEFT JOIN
9     dbo.dim_player dp ON fps.Player_ID = dp.Player_ID
10 GROUP BY
11     fps.Player_ID, dp.Player_Name
12 ORDER BY
13     Total_Red_Cards DESC;
```

The Results pane shows the output of the query:

Player_ID	Player_Name	Total_Red_Cards
42	Barthez	32485
41	Barthez	32485
40	Barthez	32485
185	Hulk	31025
184	Hulk	31025

A message at the bottom indicates "Query succeeded | 2s".

## QUERY-5: Top 5 Successful clubs(by wins)

The screenshot shows the Microsoft SQL Database Query editor interface, identical to the one above but with a different query selected. The left sidebar and tabs are the same. The code for Query 5 is displayed in the editor:

```
1 -- REPORT 5: TOP 5 SUCCESSFUL CLUBS (BY WINS)
2 SELECT TOP 5
3     fts.Club_ID,
4     dc.Club_Name,
5     SUM(fts.Wins) AS Total_Wins
6 FROM
7     dbo.fact_team_statistics fts
8 LEFT JOIN
9     dbo.dim_club dc ON fts.Club_ID = dc.Club_ID
10 GROUP BY
11     fts.Club_ID, dc.Club_Name
12 ORDER BY
13     Total_Wins DESC;
```

The Results pane shows the output of the query:

Club_ID	Club_Name	Total_Wins
19	NA	515745
15	Liverpool	62050
26	Sunderland	57670
7	Bayern Munich	57305
17	Man City	56940

A message at the bottom indicates "Query succeeded | 0s".

## **4. REPORTS**

**REPORT-1: Player report with country and club info**



REPORT-1.csv

**REPORT-2: Player with max Appearances in international matches**



REPORT-2.csv

**REPORT-3: Top 10 goal sources of 2010**



REPORT-3.csv

**REPORT-4: Top 5 players with most red cards**



REPORT-4.csv

**REPORT-5: Top 5 Successful clubs(by wins)**



REPORT-5.csv

## **5.CONCLUSION**

This mini-project successfully demonstrates the design and implementation of a scalable, end-to-end data analysis and reporting system for hockey game statistics using PySpark within the Azure ecosystem. By efficiently handling large volumes of flat file data, the system transforms raw operational datasets into a well-structured dimensional data warehouse using a star schema model. This model provides a clear, analytical view of various aspects of the game, including player performance, club statistics, and match outcomes.

The use of PySpark's distributed processing capabilities significantly enhanced the speed and scalability of data transformation, allowing for complex ETL operations to be performed seamlessly. The implementation of an ELT pipeline ensured that the data was cleaned, enriched, and integrated before being used for reporting. The dimensional model supported the generation of multiple performance-based reports such as top scorers, most appearances, and club rankings, offering valuable insights for coaches, analysts, and management teams.

In conclusion, this project not only enhanced our understanding of PySpark and big data processing but also reinforced the significance of well-designed data pipelines in modern analytics. The solution built through this project is flexible, scalable, and adaptable, capable of supporting future enhancements such as real-time data integration, machine learning-based performance predictions, and integration with cloud-based reporting tools.