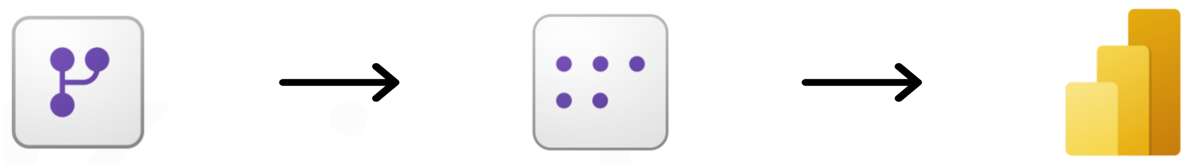
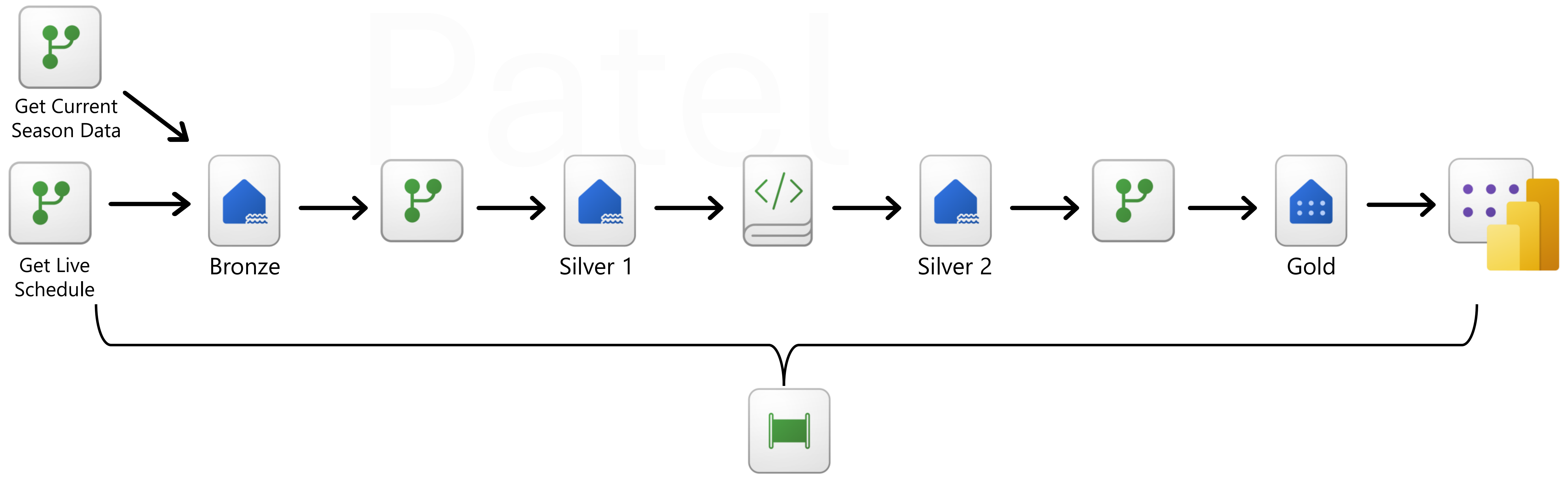
Why I Moved My Power BI Solution to Microsoft Fabric

Power BI Solution



Fabric System



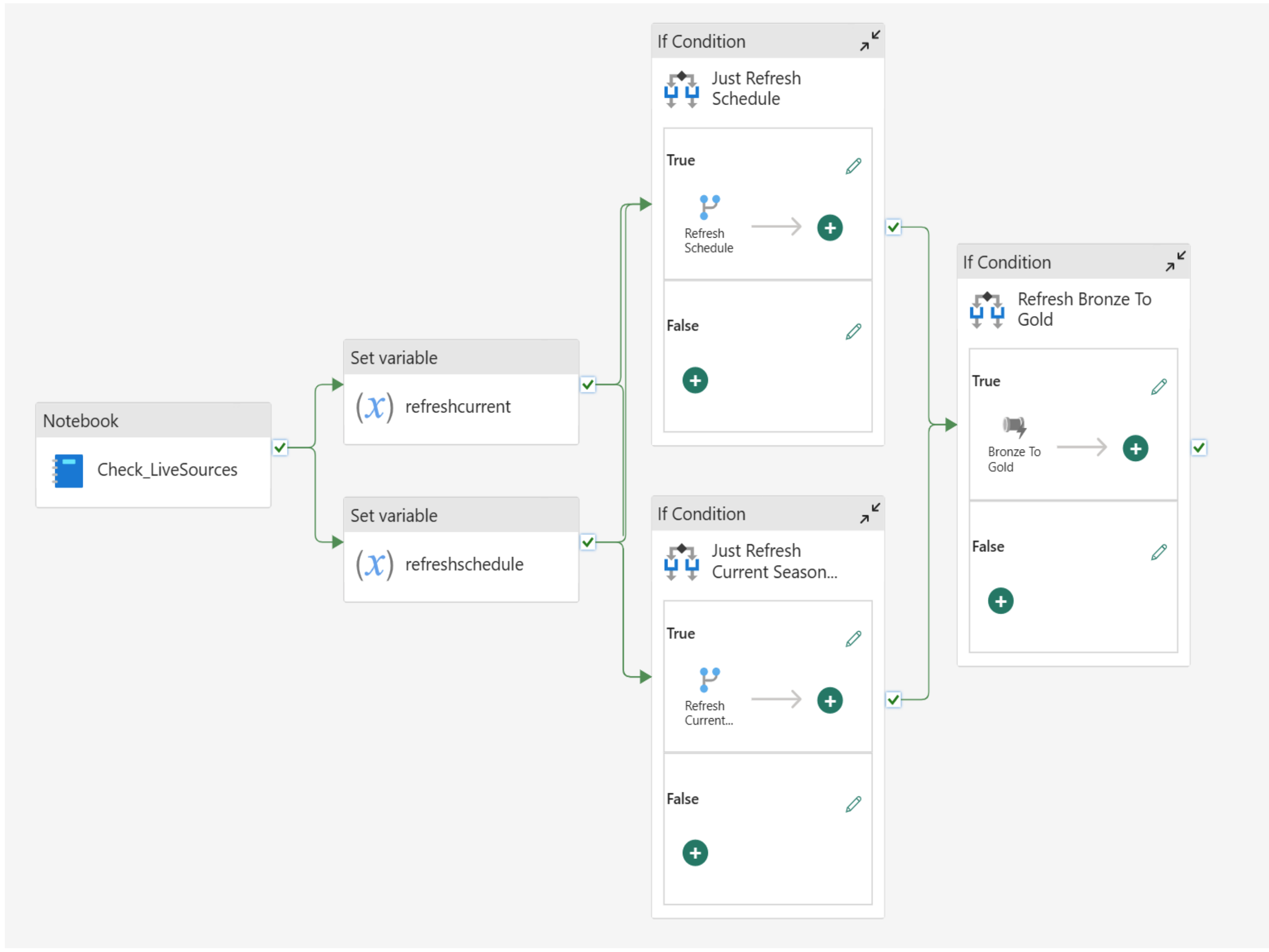
1. Resilience & Robustness

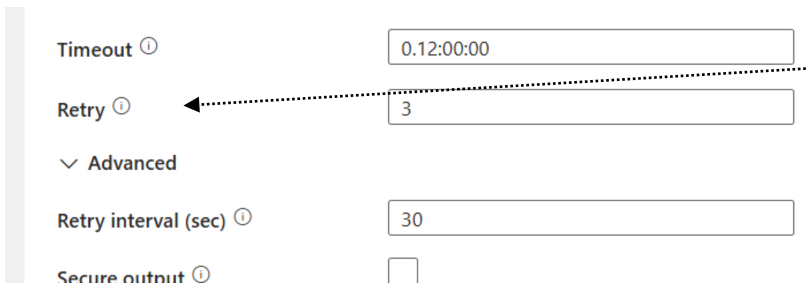
I am ingesting data from two web sources.

Key Problem:
What if one or both data sources are unavailable due to downtime or maintenance?

Solution:

- I can use a **notebook** to return two **exit values** indicating whether each web source is accessible and providing valid data.
- If a web source fails validation, the exit value is "No." If it passes validation, the exit value is "Yes."
- Using **Fabric Pipelines**, these exit values are stored as **variables** and applied in "If Condition" activities to **orchestrate the pipeline**. This ensures that it skips refreshing any source that is down or providing invalid data.
- This ensures the pipeline remains **resilient** and continues processing available sources while defaulting any problematic ones.





Timeout	0.12:00:00
Retry	3
Advanced	
Retry interval (sec)	30
Secure output	☐

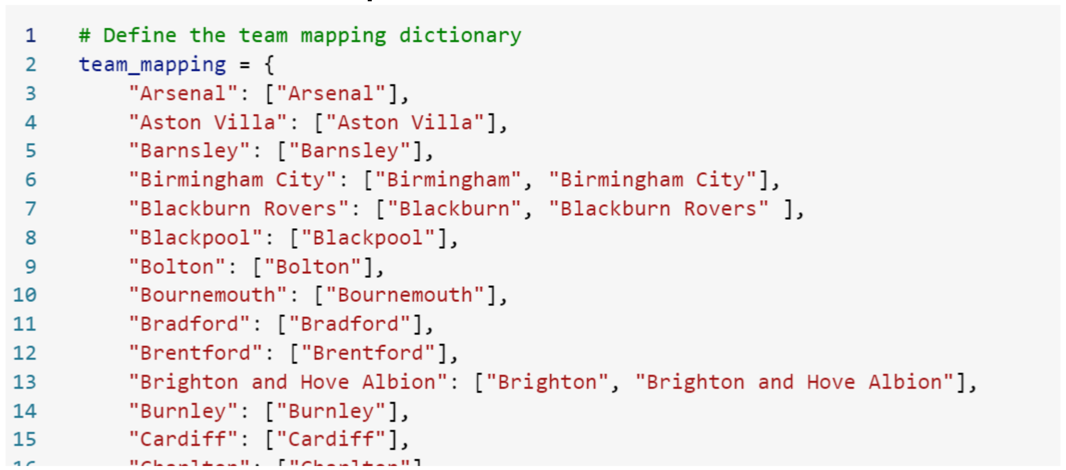
Additionally, I enabled the **retry property** in the general settings of pipeline activities to handle temporary server errors (like HTTP 500). This allows the pipeline to automatically attempt to fetch data again if it encounters transient errors, enhancing overall **robustness**.

2. Validation & Quality Checks

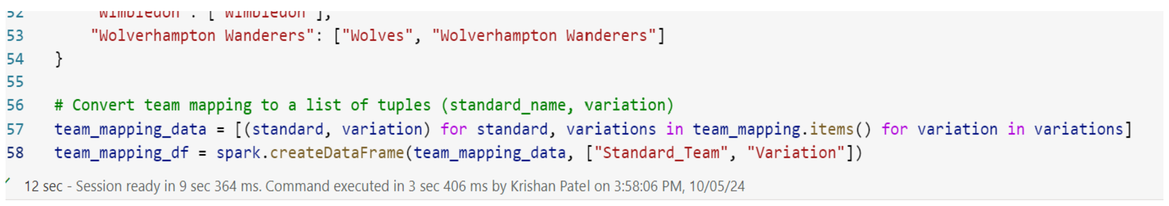
Key Problem:
Team Names are inconsistent between the two sources (i.e. one source could identify Manchester United as "Man United" whilst the other as "Man Utd").
Furthermore, we need safeguards to prevent unexpected or unrecognized team names from appearing

Solution:
We can use **notebooks** to **validate** and standardize team names between the two sources.

(1): Create a mapping dictionary where each standardized team name is the key, and the value is a list of possible variations.



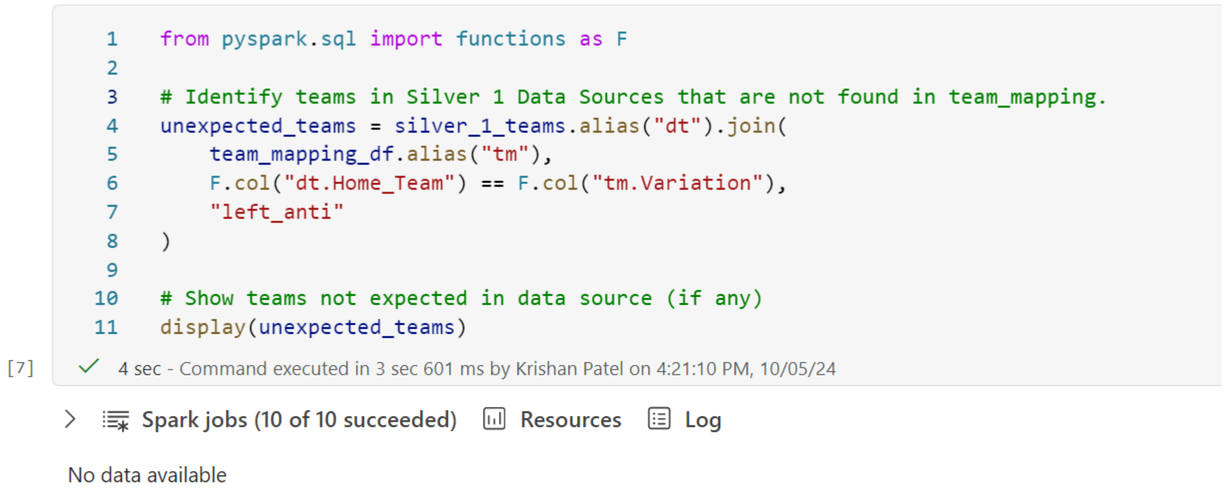
```python
# Define the team mapping dictionary
team_mapping = {
    "Arsenal": ["Arsenal"],
    "Aston Villa": ["Aston Villa"],
    "Barnsley": ["Barnsley"],
    "Birmingham City": ["Birmingham", "Birmingham City"],
    "Blackburn Rovers": ["Blackburn", "Blackburn Rovers" ],
    "Blackpool": ["Blackpool"],
    "Bolton": ["Bolton"],
    "Bournemouth": ["Bournemouth"],
    "Bradford": ["Bradford"],
    "Brentford": ["Brentford"],
    "Brighton and Hove Albion": ["Brighton", "Brighton and Hove Albion"],
    "Burnley": ["Burnley"],
    "Cardiff": ["Cardiff"],
    "Charlton": ["Charlton"]
```



```python
    "Wimbledon": ["Wimbledon"],
    "Wolverhampton Wanderers": ["Wolves", "Wolverhampton Wanderers"]
}

# Convert team mapping to a list of tuples (standard_name, variation)
team_mapping_data = [(standard, variation) for standard, variations in team_mapping.items() for variation in variations]
team_mapping_df = spark.createDataFrame(team_mapping_data, ["Standard_Team", "Variation"])
```

12 sec - Session ready in 9 sec 364 ms. Command executed in 3 sec 406 ms by Krishan Patel on 3:58:06 PM, 10/05/24

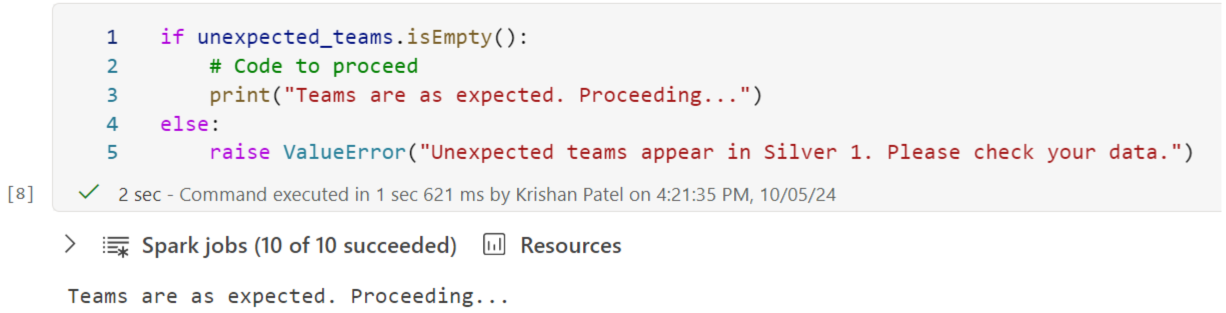
(2): Implement a **validation step** to check if the team names from both sources match the standardized dictionary. Flag any unrecognized names for review..



```python
from pyspark.sql import functions as F

# Identify teams in Silver 1 Data Sources that are not found in team_mapping.
unexpected_teams = silver_1_teams.alias("dt").join(
    team_mapping_df.alias("tm"),
    F.col("dt.Home_Team") == F.col("tm.Variation"),
    "left_anti"
)

# Show teams not expected in data source (if any)
display(unexpected_teams)
```

[7] ✓ 4 sec - Command executed in 3 sec 601 ms by Krishan Patel on 4:21:10 PM, 10/05/24

```
No data available
```



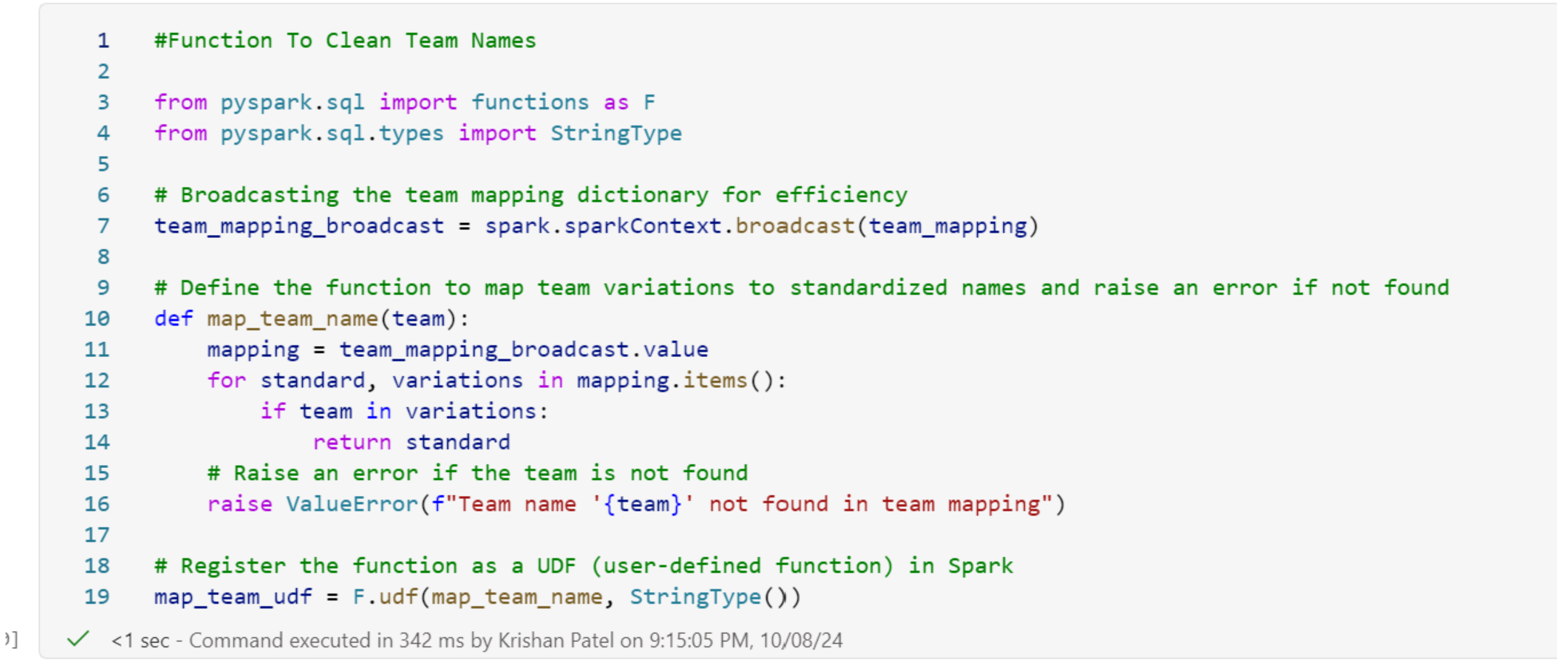
```python
if unexpected_teams.isEmpty():
    # Code to proceed
    print("Teams are as expected. Proceeding...")
else:
    raise ValueError("Unexpected teams appear in Silver 1. Please check your data.")
```

[8] ✓ 2 sec - Command executed in 1 sec 621 ms by Krishan Patel on 4:21:35 PM, 10/05/24

```
Teams are as expected. Proceeding...
```

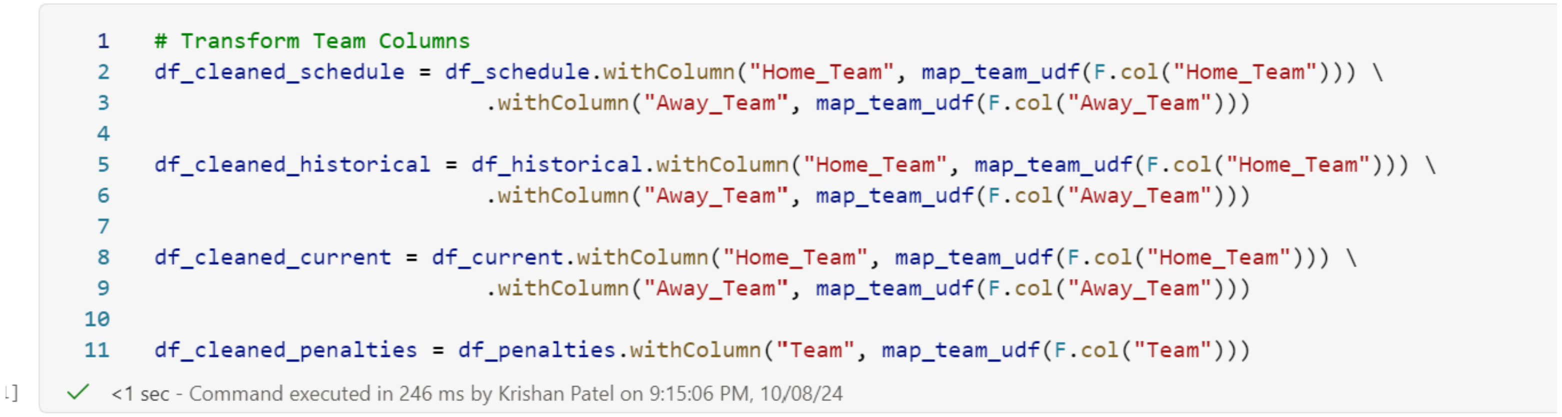
2. Validation & Quality Checks

(3): If all teams are recognized, then team names in sources from the Silver 1 Lakehouse are standardized according to the mapping dictionary. If any team is unrecognized, the process is halted, and the issue is flagged for investigation.



```python
#Function To Clean Team Names

from pyspark.sql import functions as F
from pyspark.sql.types import StringType

# Broadcasting the team mapping dictionary for efficiency
team_mapping_broadcast = spark.sparkContext.broadcast(team_mapping)

# Define the function to map team variations to standardized names and raise an error if not found
def map_team_name(team):
    mapping = team_mapping_broadcast.value
    for standard, variations in mapping.items():
        if team in variations:
            return standard
    # Raise an error if the team is not found
    raise ValueError(f"Team name '{team}' not found in team mapping")

# Register the function as a UDF (user-defined function) in Spark
map_team_udf = F.udf(map_team_name, StringType())
```

✓ <1 sec - Command executed in 342 ms by Krishan Patel on 9:15:05 PM, 10/08/24



```python
# Transform Team Columns
df_cleaned_schedule = df_schedule.withColumn("Home_Team", map_team_udf(F.col("Home_Team"))) \
                         .withColumn("Away_Team", map_team_udf(F.col("Away_Team")))

df_cleaned_historical = df_historical.withColumn("Home_Team", map_team_udf(F.col("Home_Team"))) \
                         .withColumn("Away_Team", map_team_udf(F.col("Away_Team")))

df_cleaned_current = df_current.withColumn("Home_Team", map_team_udf(F.col("Home_Team"))) \
                         .withColumn("Away_Team", map_team_udf(F.col("Away_Team")))

df_cleaned_penalties = df_penalties.withColumn("Team", map_team_udf(F.col("Team")))
```

✓ <1 sec - Command executed in 246 ms by Krishan Patel on 9:15:06 PM, 10/08/24

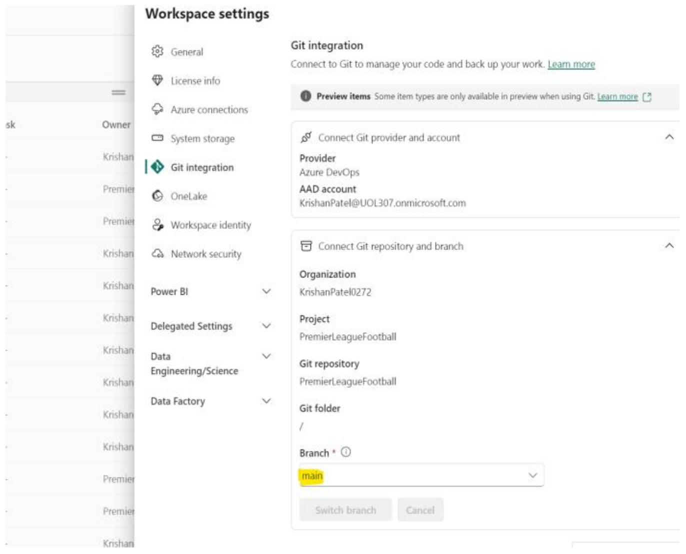
3. Enhanced Collaboration

Key Problem:

Without an integrated platform, collaboration on an analytic solution becomes challenging, leading to difficulties in managing version control, tracking changes, and maintaining code quality.

Solution:

I enabled **Git integration** using Microsoft Fabric by connecting to an **Azure DevOps repository**. This setup enhanced collaboration on the analytic solution by streamlining version control, facilitating efficient change tracking, and enabling code reviews, ensuring consistent code quality and effective teamwork throughout the project.



Workspace settings

Git integration

Connect to Git to manage your code and back up your work. Learn more

Preview items Some item types are only available in preview when using Git. Learn more

Connect Git provider and account

Provider: Azure DevOps

AAD account: KrishanPatel@UOL307.onmicrosoft.com

Connect Git repository and branch

Organization: KrishanPatel0272

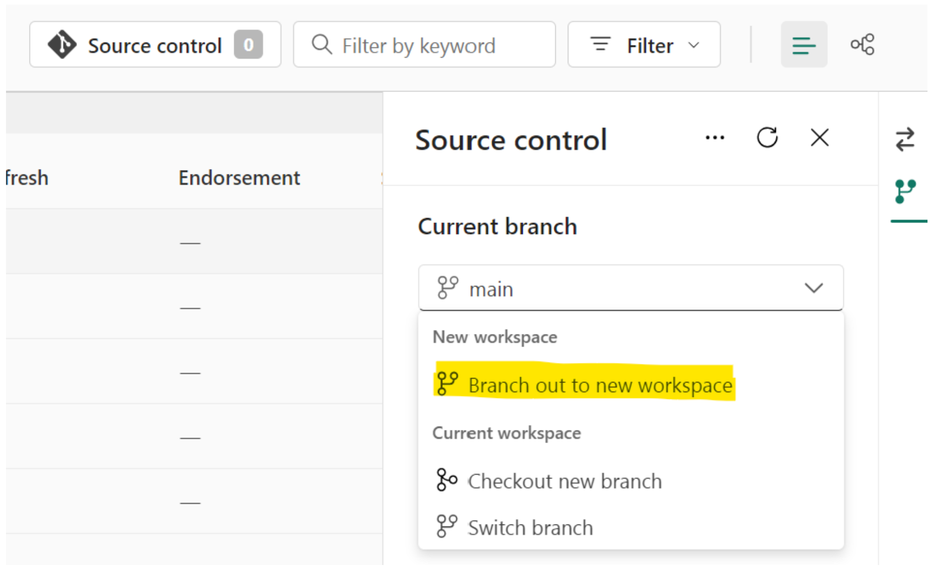
Project: PremierLeagueFootball

Git repository: PremierLeagueFootball

Git folder: /

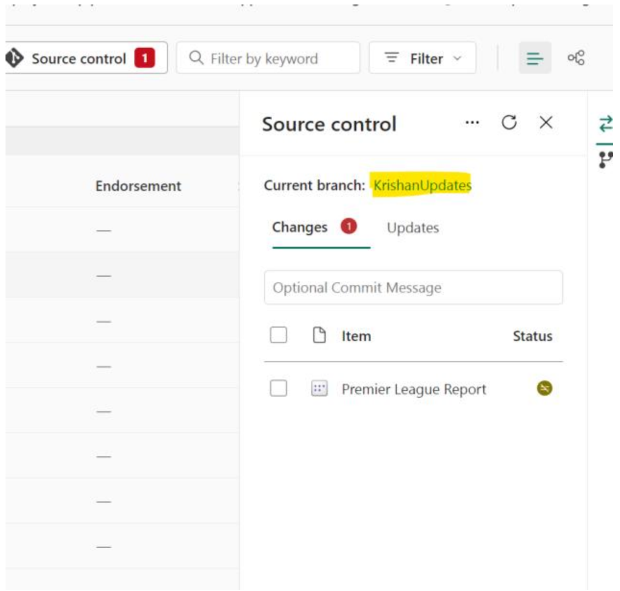
Branch *: main

An example of how Git Integration works in Fabric:



1:

To prevent interfering with others, each team member should select "Branch out to new workspace" to work in an isolated workspace until ready to share changes.



Current branch: KrishanUpdates

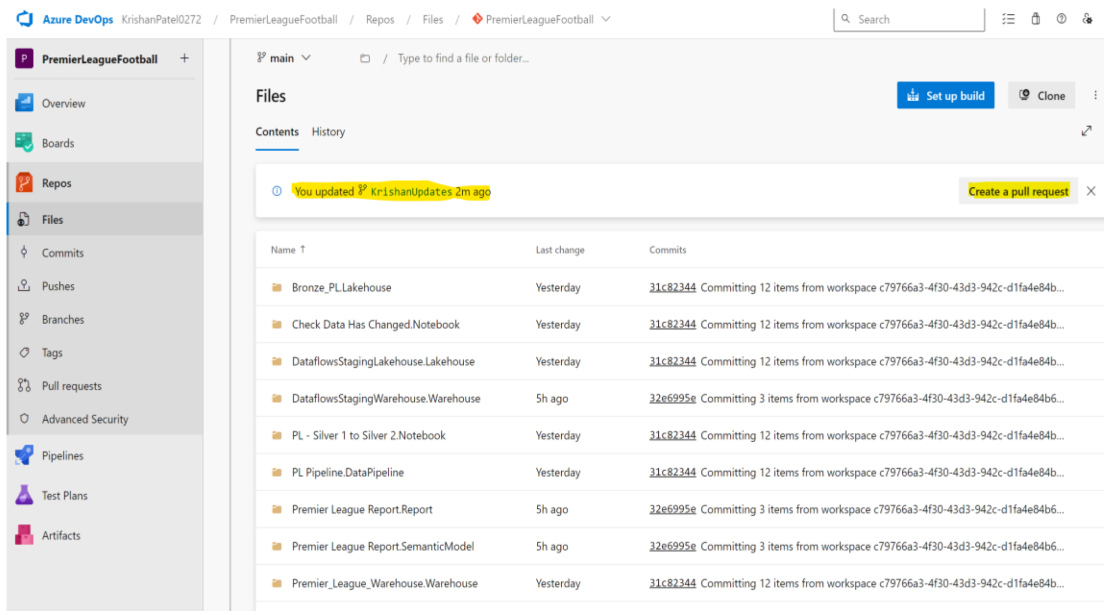
Item	Status
Premier League Report	●

2:

Once I'm satisfied with my changes, I can commit them to a separate branch, "KrishanUpdates."

3. Enhanced Collaboration

Continued: An example of how Git Integration works in Fabric:



Azure DevOps KrishanPatel0272 / PremierLeagueFootball / Repos / Files / PremierLeagueFootball

PremierLeagueFootball

Overview · Boards · Repos · Files · Commits · Pushes · Branches · Tags · Pull requests · Advanced Security · Pipelines · Test Plans · Artifacts

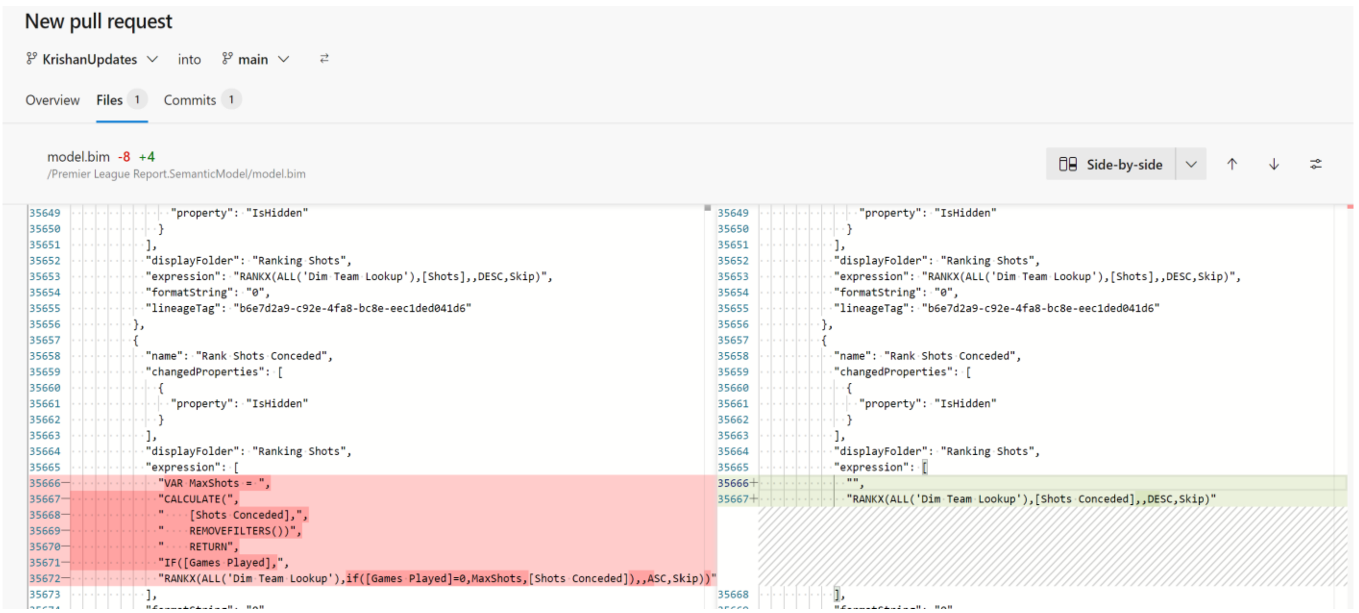
main / Type to find a file or folder...

Files — Set up build — Clone

Contents · History

You updated KrishanUpdates 2m ago — Create a pull request

Name ↑	Last change	Commits
Bronze_PL.Lakehouse	Yesterday	31c82344 Committing 12 items from workspace c79766a3-4f30-43d3-942c-d1fa4e84b...
Check Data Has Changed.Notebook	Yesterday	31c82344 Committing 12 items from workspace c79766a3-4f30-43d3-942c-d1fa4e84b...
DataflowsStagingLakehouse.Lakehouse	Yesterday	31c82344 Committing 12 items from workspace c79766a3-4f30-43d3-942c-d1fa4e84b...
DataflowsStagingWarehouse.Warehouse	5h ago	32e6995e Committing 3 items from workspace c79766a3-4f30-43d3-942c-d1fa4e84b6...
PL - Silver 1 to Silver 2.Notebook	Yesterday	31c82344 Committing 12 items from workspace c79766a3-4f30-43d3-942c-d1fa4e84b...
PL Pipeline.DataPipeline	Yesterday	31c82344 Committing 12 items from workspace c79766a3-4f30-43d3-942c-d1fa4e84b...
Premier League Report.Report	5h ago	32e6995e Committing 3 items from workspace c79766a3-4f30-43d3-942c-d1fa4e84b6...
Premier League Report.SemanticModel	5h ago	32e6995e Committing 3 items from workspace c79766a3-4f30-43d3-942c-d1fa4e84b6...
Premier_League_Warehouse.Warehouse	Yesterday	31c82344 Committing 12 items from workspace c79766a3-4f30-43d3-942c-d1fa4e84b...



New pull request

KrishanUpdates into main

Overview · Files 1 · Commits 1

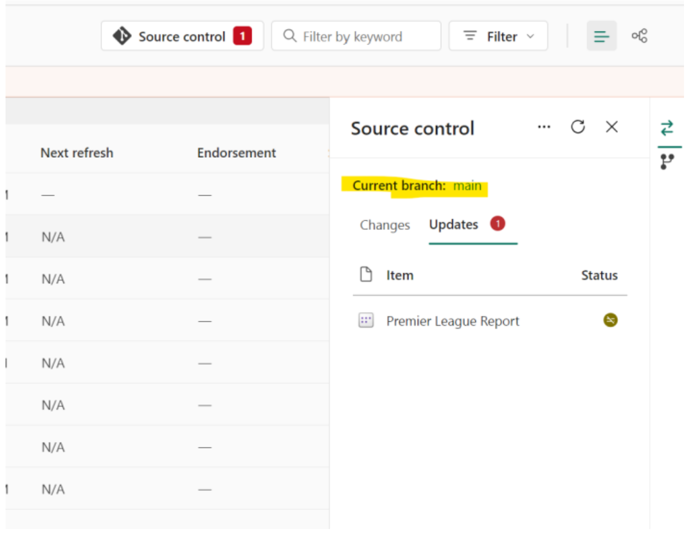
model.bim -8 +4
/Premier League Report.SemanticModel/model.bim

Side-by-side

```
            "property": "IsHidden"
          }
        ],
        "displayFolder": "Ranking Shots",
        "expression": "RANKX(ALL('Dim Team Lookup'),[Shots],,DESC,Skip)",
        "formatString": "0",
        "lineageTag": "b6e7d2a9-c92e-4fa8-bc8e-eec1ded041d6"
      },
      {
        "name": "Rank Shots Conceded",
        "changedProperties": [
          {
            "property": "IsHidden"
          }
        ],
        "displayFolder": "Ranking Shots",
        "expression": [
-         "VAR MaxShots = ",
-         "CALCULATE(",
-         "    [Shots Conceded],",
-         "    REMOVEFILTERS())",
-         "    RETURN",
-         "IF([Games Played],",
-         "RANKX(ALL('Dim Team Lookup'),if([Games Played]=0,MaxShots,[Shots Conceded]),,ASC,Skip))"
+         "",
+         "RANKX(ALL('Dim Team Lookup'),[Shots Conceded],,DESC,Skip)"
        ],
```

3:
In the Git repo, I created a **pull request** to merge the KrishanUpdates branch with the main branch.

4:
A **pull request** enables code reviews, protects the main branch with reviewer approval, promotes collaboration, and ensures traceability, allowing safe integration of changes without disrupting the shared workspace.



Source control 1 — Filter by keyword — Filter

Next refresh	Endorsement
—	—
N/A	—
N/A	—
N/A	—
N/A	—
N/A	—
N/A	—
N/A	—

Source control

Current branch: main

Changes · Updates 1

Item	Status
Premier League Report	●

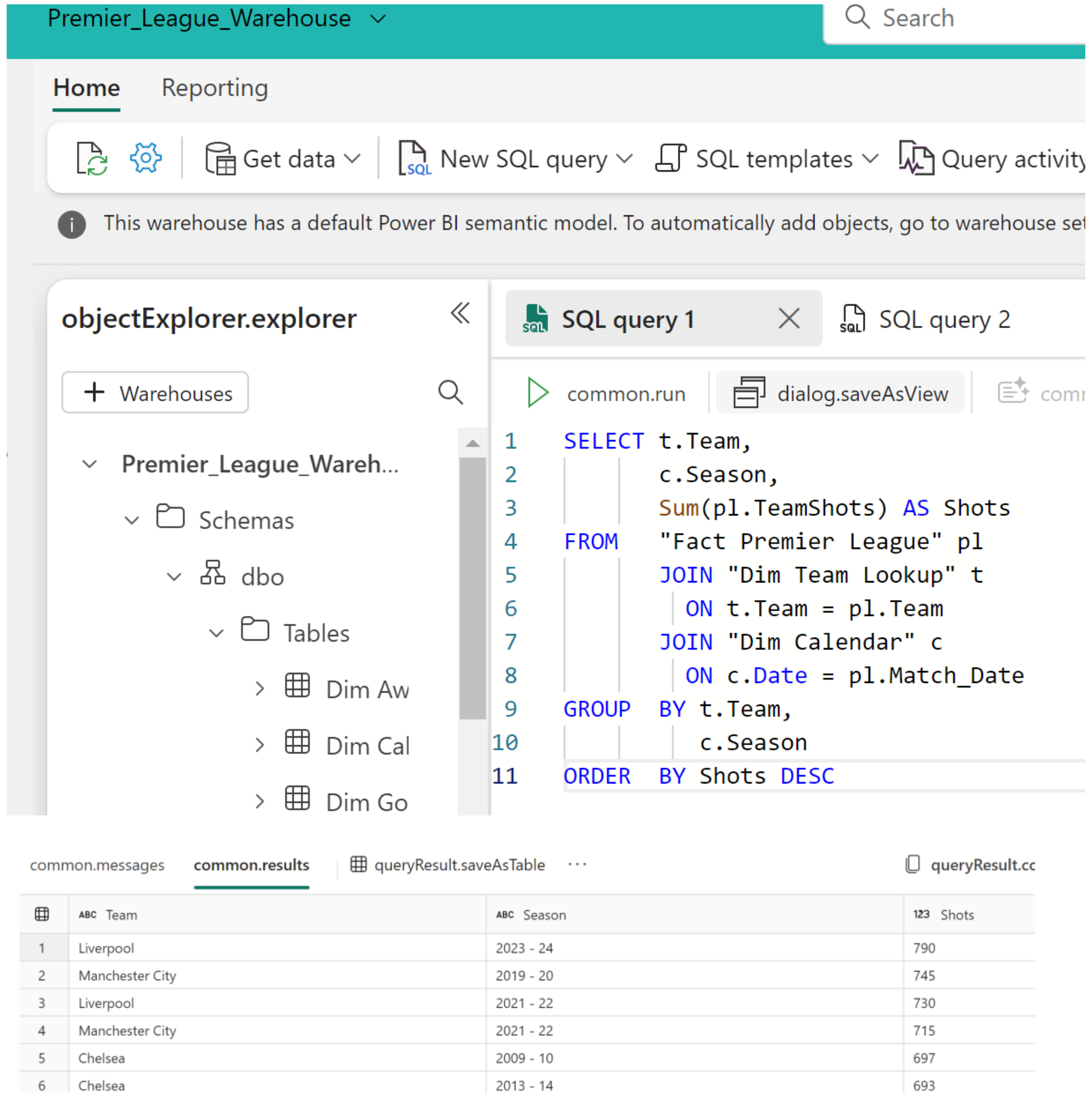
5:
After merging my changes, I sync my workspace with the main branch to stay updated with the team's progress.

4. Enhanced Analytical Capabilities

Power BI is great for visualization and reporting, but by adopting Microsoft Fabrics, we can leverage:

- Notebooks for interactive, explorative analysis, which can extend to using advanced statistical techniques
- T-SQL (Warehouse) offers more flexibility and control for complex data analysis and exploration.

Querying Gold-Layer Warehouse using T-SQL:

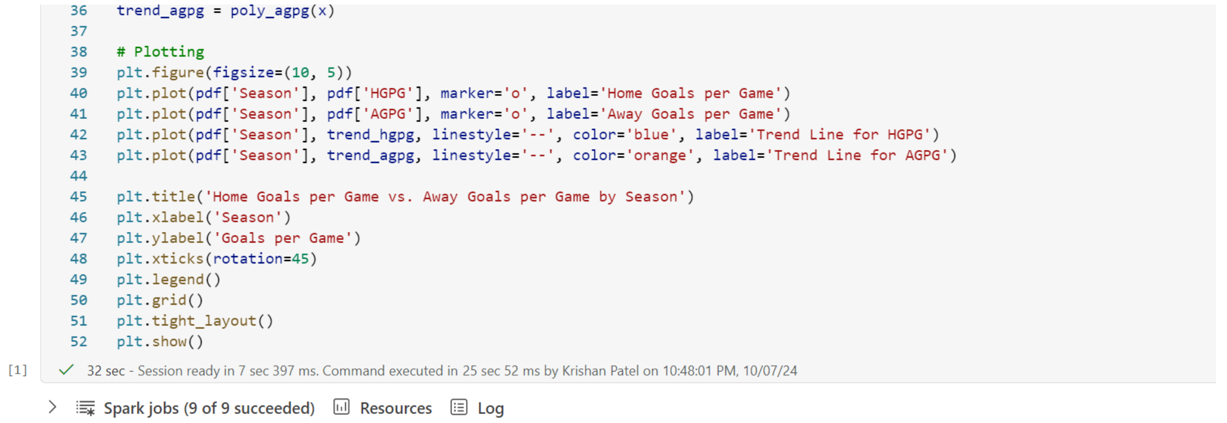


This warehouse has a default Power BI semantic model. To automatically add objects, go to warehouse se

```sql
SELECT t.Team,
       c.Season,
       Sum(pl.TeamShots) AS Shots
FROM   "Fact Premier League" pl
       JOIN "Dim Team Lookup" t
         ON t.Team = pl.Team
       JOIN "Dim Calendar" c
         ON c.Date = pl.Match_Date
GROUP  BY t.Team,
          c.Season
ORDER  BY Shots DESC
```

	Team	Season	Shots
1	Liverpool	2023 - 24	790
2	Manchester City	2019 - 20	745
3	Liverpool	2021 - 22	730
4	Manchester City	2021 - 22	715
5	Chelsea	2009 - 10	697
6	Chelsea	2013 - 14	693

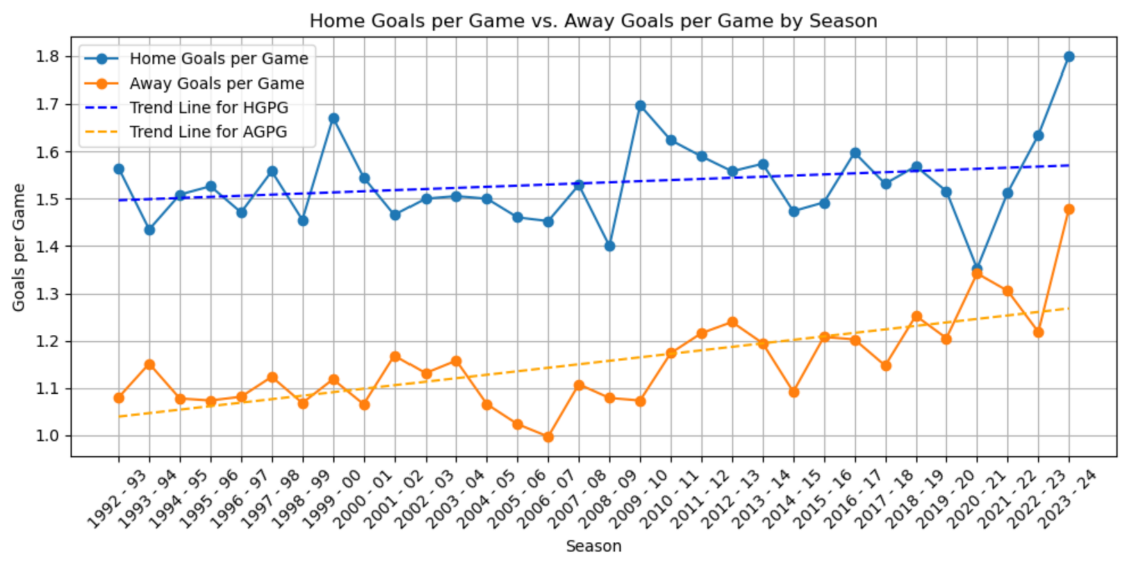
Visualize data in a Spark notebook



```python
trend_agpg = poly_agpg(x)

# Plotting
plt.figure(figsize=(10, 5))
plt.plot(pdf['Season'], pdf['HGPG'], marker='o', label='Home Goals per Game')
plt.plot(pdf['Season'], pdf['AGPG'], marker='o', label='Away Goals per Game')
plt.plot(pdf['Season'], trend_hgpg, linestyle='--', color='blue', label='Trend Line for HGPG')
plt.plot(pdf['Season'], trend_agpg, linestyle='--', color='orange', label='Trend Line for AGPG')

plt.title('Home Goals per Game vs. Away Goals per Game by Season')
plt.xlabel('Season')
plt.ylabel('Goals per Game')
plt.xticks(rotation=45)
plt.legend()
plt.grid()
plt.tight_layout()
plt.show()
```

```
[1]  ✓ 32 sec - Session ready in 7 sec 397 ms. Command executed in 25 sec 52 ms by Krishan Patel on 10:48:01 PM, 10/07/24
```

Spark jobs (9 of 9 succeeded)
Resources
Log


Home Goals per Game vs. Away Goals per Game by Season
Home Goals per Game
Away Goals per Game
Trend Line for HGPG
Trend Line for AGPG
Goals per Game
Season