In [124…
```
# Importing required libraries
import pandas as pd
import sqlite3
import matplotlib.pyplot as plt
import seaborn as sns

# Loading CSV Data
df_csv = pd.read_csv('nba_games_cleaned_csv.csv')
print("\nCSV Data:")
df_csv.head()
```

```
CSV Data:
/var/folders/fj/1y30smkn19d0x1r6g3zr71d40000gn/T/ipykernel_80841/895792958.py:8: DtypeWarning: Columns (6,7,9) have mixed types. Specify dtype option on import or set low_memory=False.
  df_csv = pd.read_csv('nba_games_cleaned_csv.csv')
```

Out[124…

	GAME_ID	TEAM_ID	TEAM_ABBREVIATION	TEAM_CITY	PLAYER_ID	PLAYER_NA
0	22200477	1610612759	SAS	San Antonio	1629641	Ror Lang
1	22200477	1610612759	SAS	San Antonio	1631110	Jeremy Soc
2	22200477	1610612759	SAS	San Antonio	1627751	Jakob Pc
3	22200477	1610612759	SAS	San Antonio	1630170	Devin Vas
4	22200477	1610612759	SAS	San Antonio	1630200	Tre Jo

5 rows × 29 columns

In [122…
```
# Loading wiki data
df_wiki = pd.read_csv('nba_games_cleaned_wiki.csv')
print("\nWiki Data:")
df_wiki.head()
```

```
Wiki Data:
```

Out[122…

	Team	Wins	Losses	Win_Percentage	Made_Playoffs
0	Boston Celtics	57	25	0.695	1
1	Philadelphia 76ers	54	28	0.659	1
2	New York Knicks	47	35	0.573	1
3	Brooklyn Nets	45	37	0.549	1
4	Toronto Raptors	41	41	0.500	0

In [128…
```
# Loading API data
df_api = pd.read_csv('nba_games_cleaned_api.csv')
print("API Data:")
df_api.head()
```

```
API Data:
```

Out[128…

	id	date	season	status	period	time	postseason	home_score	away_sco
0	1038045	2024-01-01	2023	Final	4	Final	False	136	1
1	1038043	2024-01-01	2023	Final	4	Final	False	112	1
2	1038046	2024-01-01	2023	Final	4	Final	False	113	1
3	1038044	2024-01-01	2023	Final	4	Final	False	124	1
4	1038050	2024-01-01	2023	Final	4	Final	False	121	1

5 rows × 25 columns

In [136…

```
# Converting Team abbreviations to full Team names for data merging and easi
abbrev_to_full = {
    'ATL': 'Atlanta Hawks', 'BOS': 'Boston Celtics', 'BKN': 'Brooklyn Nets',
    'CHA': 'Charlotte Hornets', 'CHI': 'Chicago Bulls', 'CLE': 'Cleveland Ca
    'DAL': 'Dallas Mavericks', 'DEN': 'Denver Nuggets', 'DET': 'Detroit Pist
    'GSW': 'Golden State Warriors', 'HOU': 'Houston Rockets', 'IND': 'Indiar
    'LAL': 'Los Angeles Lakers', 'MEM': 'Memphis Grizzlies', 'MIA': 'Miami H
    'MIL': 'Milwaukee Bucks', 'MIN': 'Minnesota Timberwolves', 'NOP': 'New O
    'NYK': 'New York Knicks', 'OKC': 'Oklahoma City Thunder', 'ORL': 'Orlanc
    'PHI': 'Philadelphia 76ers', 'PHX': 'Phoenix Suns', 'POR': 'Portland Tra
    'SAC': 'Sacramento Kings', 'SAS': 'San Antonio Spurs', 'TOR': 'Toronto R
    'UTA': 'Utah Jazz', 'WAS': 'Washington Wizards'
}

# Adding full name and normalized version
df_csv['TEAM_FULL_NAME'] = df_csv['TEAM_ABBREVIATION'].map(abbrev_to_full)
df_csv['team_lower'] = df_csv['TEAM_FULL_NAME'].str.strip().str.lower()
df_api['team_lower'] = df_api['home_team'].str.strip().str.lower()
df_wiki['team_lower'] = df_wiki['Team'].str.strip().str.lower()


# Creating SQLite DB and Storing three Tables
conn = sqlite3.connect('nba_milestone5.db')

# Adding data to SQL tables.
df_api.to_sql('api_games', conn, if_exists='replace', index=False)
df_csv.to_sql('csv_stats', conn, if_exists='replace', index=False)
df_wiki.to_sql('wiki_standings', conn, if_exists='replace', index=False)
```

Out[136… 30

In [146…

```
# Grouping csv_stats table stats to get Total/Average POINTS per TEAM
team_avg_points = df_csv.groupby('team_lower', as_index=False)['POINTS'].mea
team_avg_points.rename(columns={'POINTS': 'avg_team_points'}, inplace=True)

# Replacing the csv_stats table with the aggregated one
team_avg_points.to_sql("csv_stats_agg", conn, if_exists="replace", index=Fal
```

```
# Creating the join query to aggregate all three tables
query = """
SELECT
    api.date,
    api.home_team,
    api.away_team,
    api.home_score,
    api.away_score,
    csv.avg_team_points,
    wiki.Win_Percentage AS home_win_pct
FROM api_games api
JOIN csv_stats_agg csv ON LOWER(api.home_team) = csv.team_lower
JOIN wiki_standings wiki ON LOWER(api.home_team) = wiki.team_lower
"""

# Runing the join query
merged_df = pd.read_sql_query(query, conn)
merged_df['date'] = pd.to_datetime(merged_df['date'])

# Preview final deduplicated joined data
merged_df.head(10)
```

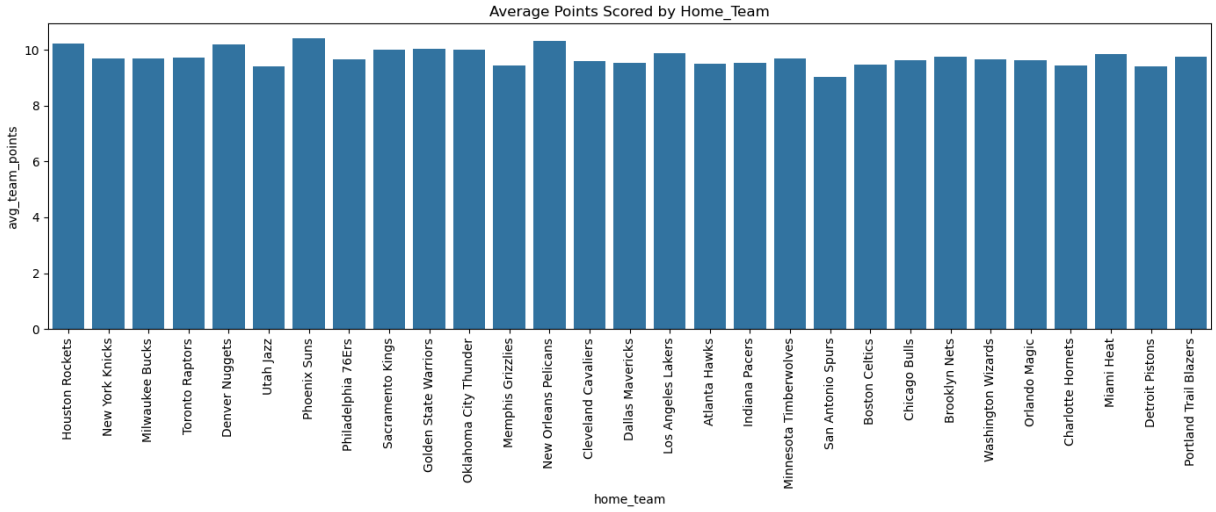
Out[146…

	date	home_team	away_team	home_score	away_score	avg_team_points	hom
0	2024-01-01	Houston Rockets	Detroit Pistons	136	113	10.231804	
1	2024-01-01	New York Knicks	Minnesota Timberwolves	112	106	9.678052	
2	2024-01-01	Milwaukee Bucks	Indiana Pacers	113	122	9.682365	
3	2024-01-01	Toronto Raptors	Cleveland Cavaliers	124	121	9.734643	
4	2024-01-01	Denver Nuggets	Charlotte Hornets	111	93	10.191317	
5	2024-01-01	Utah Jazz	Dallas Mavericks	127	90	9.396751	
6	2024-01-01	Phoenix Suns	Portland Trail Blazers	109	88	10.414740	
7	2024-01-02	Philadelphia 76Ers	Chicago Bulls	110	97	9.669075	
8	2024-01-02	Sacramento Kings	Charlotte Hornets	104	111	10.002922	
9	2024-01-02	Golden State Warriors	Orlando Magic	121	115	10.045124	

In [162…

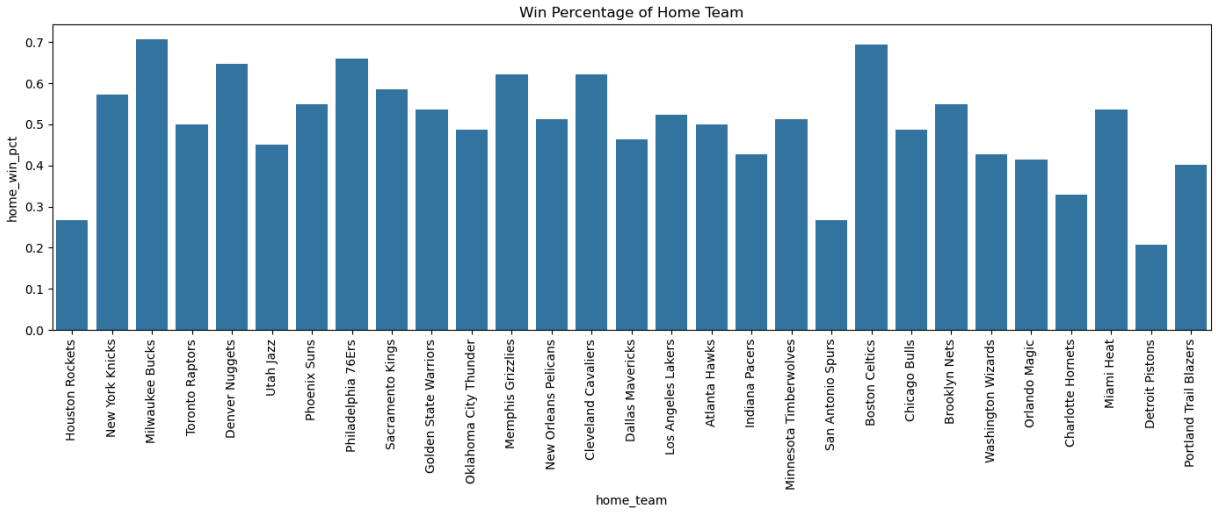
```
# Plot 1: Points Scored by Teams (from CSV_stats Table)
plt.figure(figsize=(14,6))
```

```
sns.barplot(data=merged_df, x='home_team', y='avg_team_points')
plt.xticks(rotation=90)
plt.title("Average Points Scored by Home_Team")
plt.tight_layout()
plt.show()
```



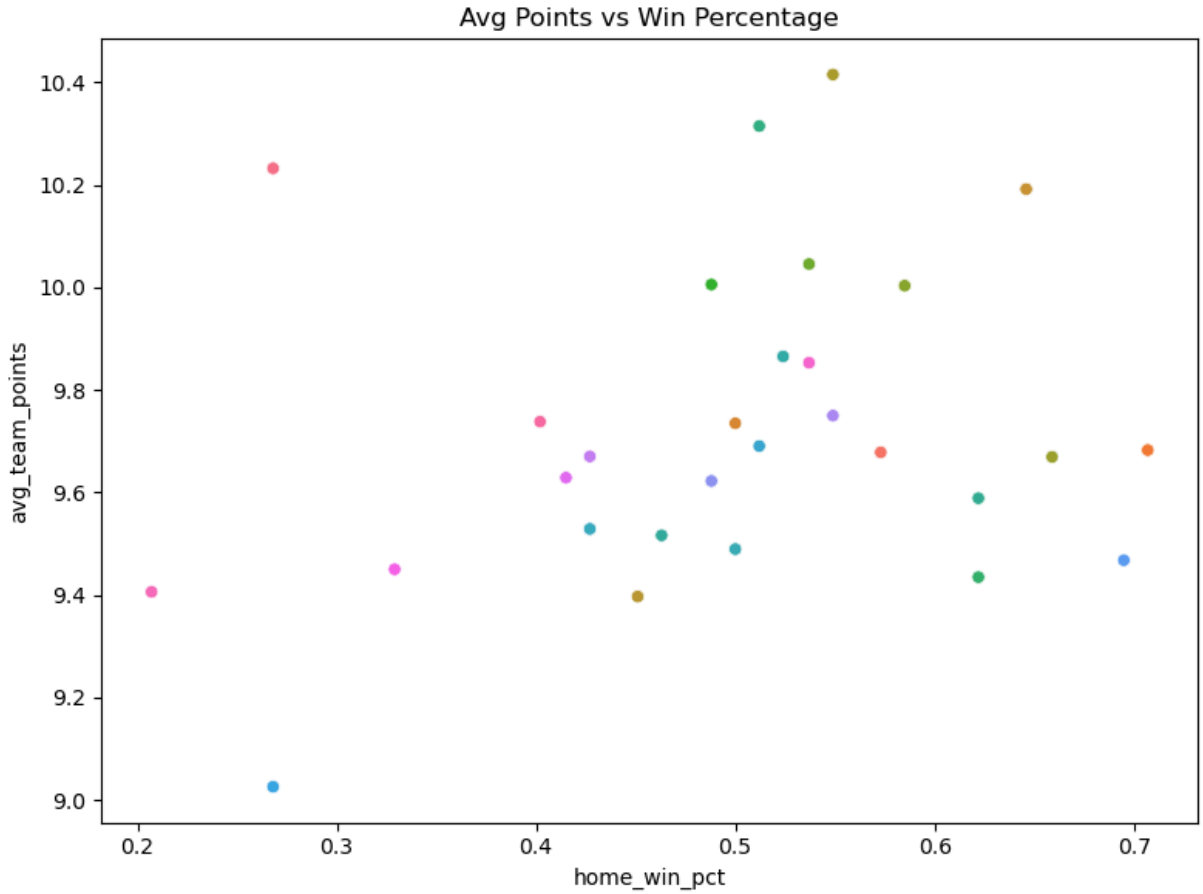
In [164…

```
# Plot 2: Win Percentage from Wiki Standings Table
plt.figure(figsize=(14,6))
sns.barplot(data=merged_df, x='home_team', y='home_win_pct')
plt.xticks(rotation=90)
plt.title("Win Percentage of Home Team")
plt.tight_layout()
plt.show()
```

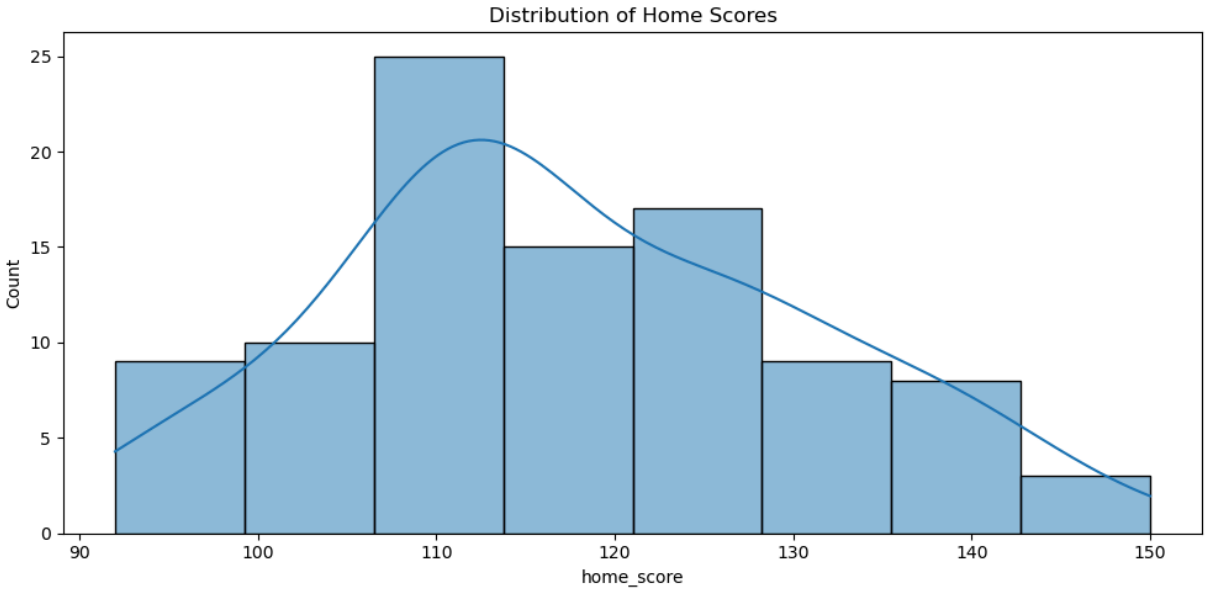


In [168…

```
# Plot 3: Points vs. Win % (Its from the joined data)
plt.figure(figsize=(8,6))
sns.scatterplot(data=merged_df, x='home_win_pct', y='avg_team_points', hue='
plt.title("Avg Points vs Win Percentage")
plt.tight_layout()
plt.show()
```

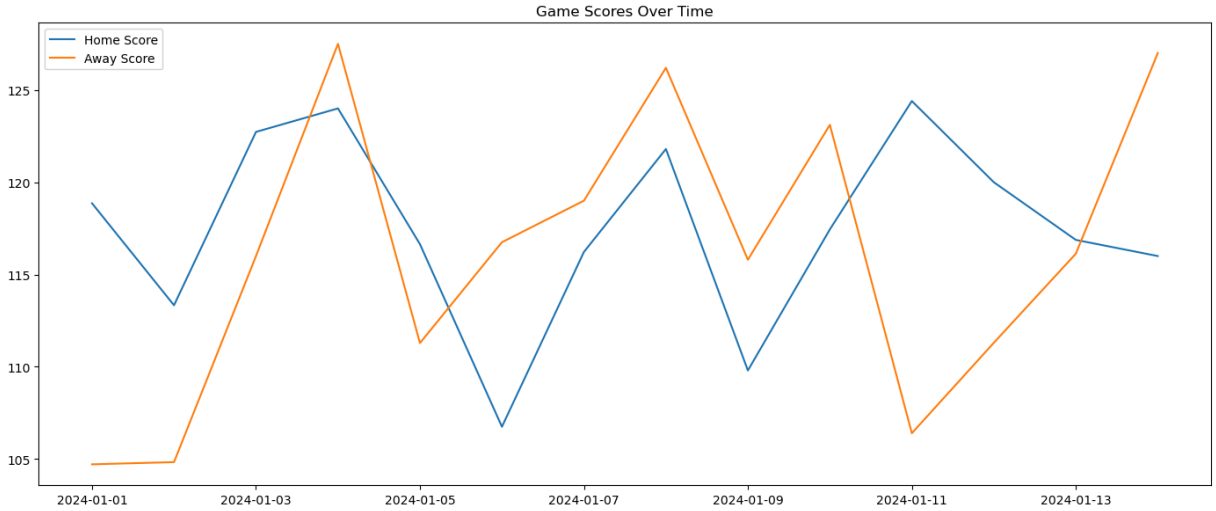


In [176…
```
# Plot 4: Plotting Home Score Distribution
plt.figure(figsize=(10,5))
sns.histplot(data=merged_df, x='home_score', kde=True)
plt.title("Distribution of Home Scores")
plt.tight_layout()
plt.show()
```



In [174…
```
# Plot 5: Plotting Score Trends Over Time
score_over_time = merged_df.groupby('date')[['home_score', 'away_score']].me
plt.figure(figsize=(14,6))
```

```
plt.plot(score_over_time['date'], score_over_time['home_score'], label='Home
plt.plot(score_over_time['date'], score_over_time['away_score'], label='Away
plt.title("Game Scores Over Time")
plt.legend()
plt.tight_layout()
plt.show()
```



In [178…

```
# Printing Final Dataset
print("\nFinal Dataset:")
merged_df.head(20)
```

```
Final Dataset:
```

Out[178...

	date	home_team	away_team	home_score	away_score	avg_team_points	hor
0	2024-01-01	Houston Rockets	Detroit Pistons	136	113	10.231804	
1	2024-01-01	New York Knicks	Minnesota Timberwolves	112	106	9.678052	
2	2024-01-01	Milwaukee Bucks	Indiana Pacers	113	122	9.682365	
3	2024-01-01	Toronto Raptors	Cleveland Cavaliers	124	121	9.734643	
4	2024-01-01	Denver Nuggets	Charlotte Hornets	111	93	10.191317	
5	2024-01-01	Utah Jazz	Dallas Mavericks	127	90	9.396751	
6	2024-01-01	Phoenix Suns	Portland Trail Blazers	109	88	10.414740	
7	2024-01-02	Philadelphia 76Ers	Chicago Bulls	110	97	9.669075	
8	2024-01-02	Sacramento Kings	Charlotte Hornets	104	111	10.002922	
9	2024-01-02	Golden State Warriors	Orlando Magic	121	115	10.045124	
10	2024-01-02	Oklahoma City Thunder	Boston Celtics	127	123	10.005269	
11	2024-01-02	Memphis Grizzlies	San Antonio Spurs	106	98	9.434671	
12	2024-01-02	New Orleans Pelicans	Brooklyn Nets	112	85	10.314296	
13	2024-01-03	Cleveland Cavaliers	Washington Wizards	140	101	9.588548	
14	2024-01-03	Dallas Mavericks	Portland Trail Blazers	126	97	9.516083	
15	2024-01-03	Sacramento Kings	Orlando Magic	138	135	10.002922	
16	2024-01-03	Los Angeles Lakers	Miami Heat	96	110	9.865018	
17	2024-01-03	Houston Rockets	Brooklyn Nets	112	101	10.231804	
18	2024-01-03	Memphis Grizzlies	Toronto Raptors	111	116	9.434671	

	date	home_team	away_team	home_score	away_score	avg_team_points	hor
19	2024-01-03	Phoenix Suns	La Clippers	122	131	10.414740	

In [182…

```
# Ethical consideration:
# 250-500 word summary of what you learned and a summary of the ethical impl

"""
In this milestone 5 project, I merged three datasets related to NBA games: 1
 3. Wiki dataset: Team-level standings. After that, I cleaned & normalized t
string stripping to allow consistent SQL joins. The CSV player stats were ag
and then joined with game scores and win percentage. Data was loaded into ar
consolidated dataset. Using this merged data. At last, I created five visual
    - Average player points per team
    - Win percentage by team
    - Relationship between average points and win percentage
    - Score distribution
    - Score trends over time.

    First of all, I normalized Team Names, removed null values & duplicates.
I have also done multiple-level cleanup on each dataset before merging. The
identifiable personal information, ensuring compliance with data use policie
or bring out bias if Teams have disproportionate game counts.
    I believe Team Names in different sources referred to the same entity, a
The API was well-documented and sourced from a reputable sports open-source
All sources were free, open, and non-sensitive in nature. All transformatior
transparency. Future users can retrace steps or may audit transformations.

     This project helped me practice real data merging, SQL operations, plot
joins, verified sources, and visual storytelling with cross-source metrics.
"""
```

Out[182…

```
'\nIn this milestone 5 project, I merged three datasets related to NBA game
s: 1. API dataset: Game logs, 2. CSV dataset: Player-level stats, \n 3. Wik
i dataset: Team-level standings. After that, I cleaned & normalized the tea
m names across datasets using lowercase mapping,\nstring stripping to allow
consistent SQL joins. The CSV player stats were aggregated by team to calcu
late average points scored, \nand then joined with game scores and win perc
entage. Data was loaded into an SQLite database, and SQL joins were perform
ed to generate a final\nconsolidated dataset. Using this merged data. At la
st, I created five visualizations using Seaborn and Matplotlib, includin
g:\n    - Average player points per team\n    - Win percentage by team\n
- Relationship between average points and win percentage\n    - Score distr
ibution\n    - Score trends over time.\n\n    First of all, I normalized Te
am Names, removed null values & duplicates. I also merged player-level data
to the Team level. \nI have also done multiple-level cleanup on each datase
t before merging. The datasets were publicly available, open source, with n
o \nidentifiable personal information, ensuring compliance with data use po
licies. Aggregation may mask individual performance variations\nor bring ou
t bias if Teams have disproportionate game counts.\n    I believe Team Name
s in different sources referred to the same entity, and that missing rows c
ould be dropped without biasing trends.\nThe API was well-documented and so
urced from a reputable sports open-source database. Wikipedia data was clea
ned and verified against standings.\nAll sources were free, open, and non-s
ensitive in nature. All transformation steps were documented. The code is a
vailable in a GitHub repo for \ntransparency. Future users can retrace step
s or may audit transformations.\n\n     This project helped me practice rea
l data merging, SQL operations, plots, and ethical data handling. It highli
ghted the importance of clean \njoins, verified sources, and visual storyte
lling with cross-source metrics.\n'
```

In [ ]: