ETL Project

Jess Fett

MLB Payroll vs. MLB Outcomes

Project Outline

In 2020, the World Series featured one of the highest team payrolls, the LA Dodgers, and one of the lowest team payrolls, Tampa Bay Rays.

This made me question, does high payroll actually create a better outcome for baseball teams? Using the 2019 season, this project will look at the payroll data for each team, as well as the statistical outputs including but not limited to: wins, losses, runs, homeruns, postseason wins, and more.

E: Extract

Data Source #1: Baseball Databank

Baseball Databank is a compilation of historical baseball data in a convenient, tidy format, distributed under Open Data terms.

Format: CSV

Link: https://github.com/chadwickbureau/baseballdatabank

Data Source #2: CBS Sports

CBS Sports did an article outlining the opening day payrolls for each team during the 2019 series.

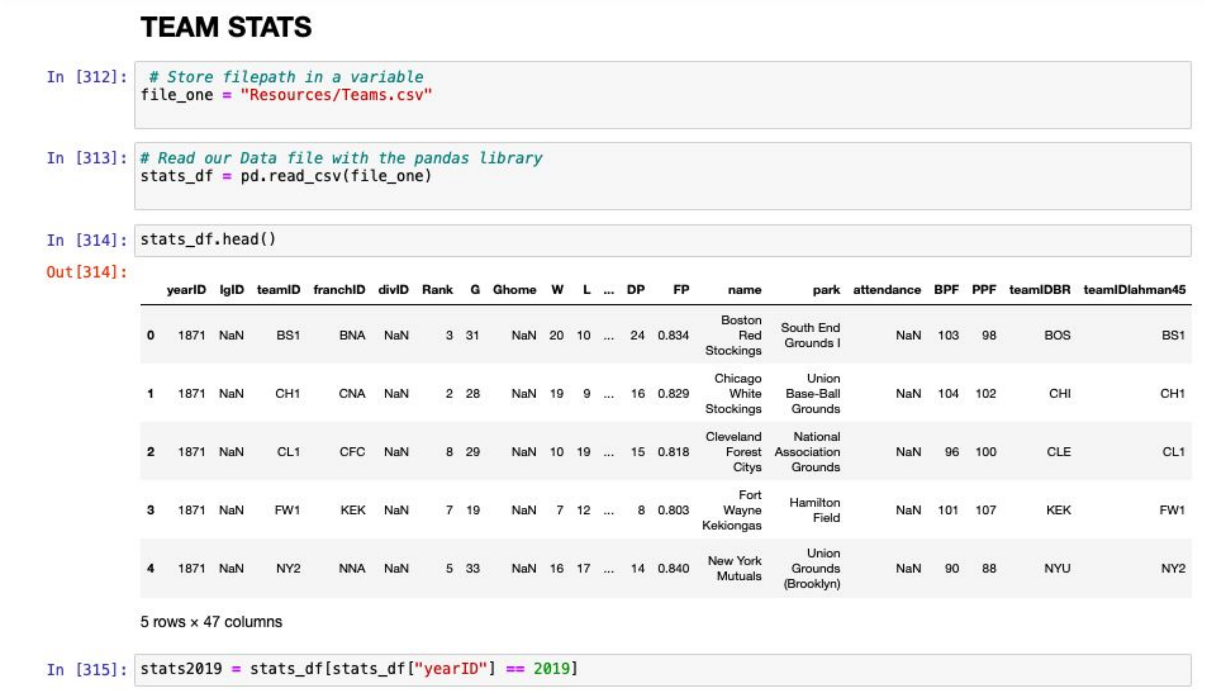
Format: WebScrape

Link: https://www.cbssports.com/mlb/news/2019-mlb-opening-day-payrolls-red-sox-cubs-yankees-open-season-above-competitive-balance-tax-threshold/

E: Extract DataSource #1

The CSV was very thorough going all the way back to the 1871 season. I only required data for the 2019 season, so after reading in the CSV file, I created a subset of the Dataframe for only 2019.

This CSV did require a lot of cleaning to make it easy to use. It included a lot of stats I did not require, as well as unique header names that needed to be changed.



TEAM STATS

```
In [312]: # Store filepath in a variable
          file_one = "Resources/Teams.csv"
```

```
In [313]: # Read our Data file with the pandas library
          stats_df = pd.read_csv(file_one)
```

```
In [314]: stats_df.head()
```

Out[314]:

	yearID	lgID	teamID	franchID	divID	Rank	G	Ghome	W	L	...	DP	FP	name	park	attendance	BPF	PPF	teamIDBR	teamIDlahman45
0	1871	NaN	BS1	BNA	NaN	3	31	NaN	20	10	...	24	0.834	Boston Red Stockings	South End Grounds I	NaN	103	98	BOS	BS1
1	1871	NaN	CH1	CNA	NaN	2	28	NaN	19	9	...	16	0.829	Chicago White Stockings	Union Base-Ball Grounds	NaN	104	102	CHI	CH1
2	1871	NaN	CL1	CFC	NaN	8	29	NaN	10	19	...	15	0.818	Cleveland Forest Citys	National Association Grounds	NaN	96	100	CLE	CL1
3	1871	NaN	FW1	KEK	NaN	7	19	NaN	7	12	...	8	0.803	Fort Wayne Kekiongas	Hamilton Field	NaN	101	107	KEK	FW1
4	1871	NaN	NY2	NNA	NaN	5	33	NaN	16	17	...	14	0.840	New York Mutuals	Union Grounds (Brooklyn)	NaN	90	88	NYU	NY2

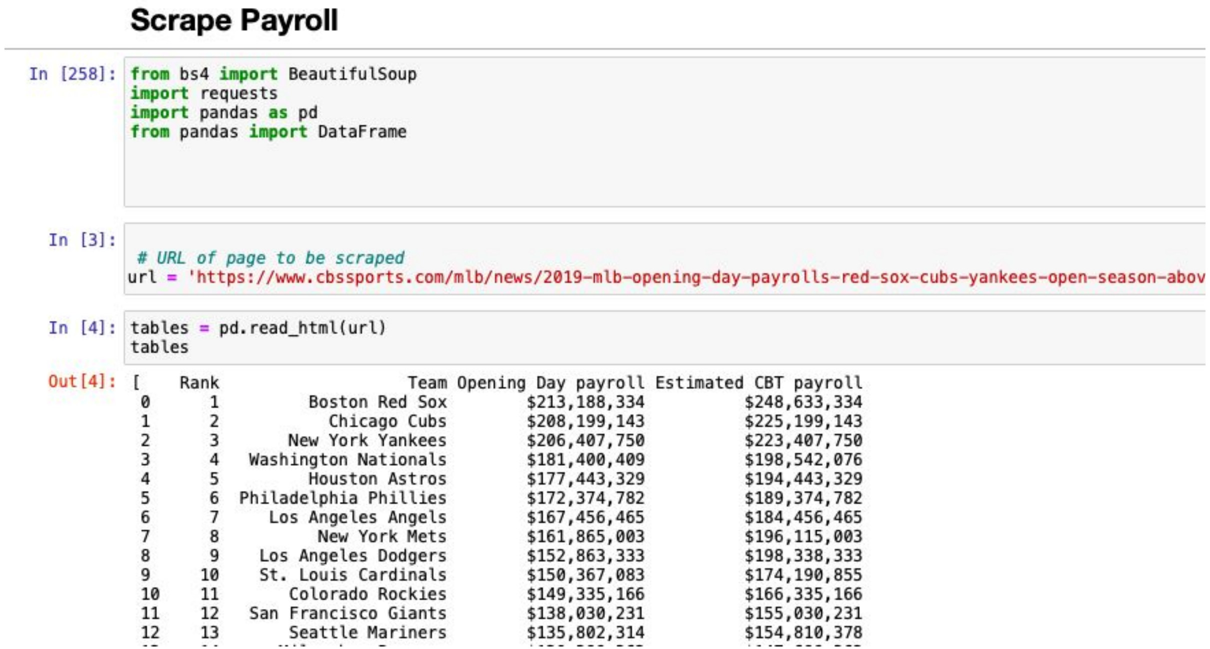
5 rows × 47 columns

```
In [315]: stats2019 = stats_df[stats_df["yearID"] == 2019]
```

E: Extract DataSource #2

For the web scrape of CBS Sport's article, it was simply done utilizing the table html scrape.

tables = pd.read_html(url)



Scrape Payroll

```
In [258]: from bs4 import BeautifulSoup
          import requests
          import pandas as pd
          from pandas import DataFrame
```

```
In [3]:    # URL of page to be scraped
          url = 'https://www.cbssports.com/mlb/news/2019-mlb-opening-day-payrolls-red-sox-cubs-yankees-open-season-abov
```

```
In [4]:   tables = pd.read_html(url)
          tables
```

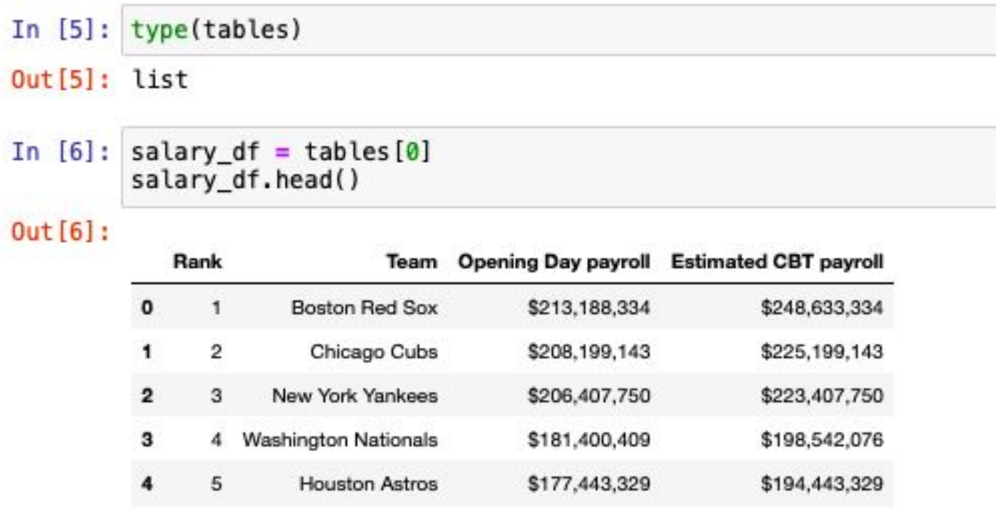
```
Out[4]: [    Rank                  Team Opening Day payroll Estimated CBT payroll
         0      1       Boston Red Sox        $213,188,334          $248,633,334
         1      2         Chicago Cubs        $208,199,143          $225,199,143
         2      3     New York Yankees        $206,407,750          $223,407,750
         3      4  Washington Nationals       $181,400,409          $198,542,076
         4      5       Houston Astros        $177,443,329          $194,443,329
         5      6 Philadelphia Phillies       $172,374,782          $189,374,782
         6      7    Los Angeles Angels       $167,456,465          $184,456,465
         7      8        New York Mets        $161,865,003          $196,115,003
         8      9   Los Angeles Dodgers       $152,863,333          $198,338,333
         9     10  St. Louis Cardinals        $150,367,083          $174,190,855
         10    11     Colorado Rockies        $149,335,166          $166,335,166
         11    12  San Francisco Giants       $138,030,231          $155,030,231
         12    13     Seattle Mariners        $135,802,314          $154,810,378
```

E: Extract DataSource #2

For the web scrape of CBS Sport's article, it was simply done utilizing the table html scrape.

tables = pd.read_html(url)

I then was able to turn it into a dataframe to work with in Pandas.



```
In [5]: type(tables)

Out[5]: list

In [6]: salary_df = tables[0]
        salary_df.head()

Out[6]:
```

	Rank	Team	Opening Day payroll	Estimated CBT payroll
0	1	Boston Red Sox	$213,188,334	$248,633,334
1	2	Chicago Cubs	$208,199,143	$225,199,143
2	3	New York Yankees	$206,407,750	$223,407,750
3	4	Washington Nationals	$181,400,409	$198,542,076
4	5	Houston Astros	$177,443,329	$194,443,329

T: Transform

Data Source #1: Baseball Databank

The baseball dataframe required a lot of cleaning such as:

Tasks:

- Rename 20 Column Headers
- Deleting 26 Columns

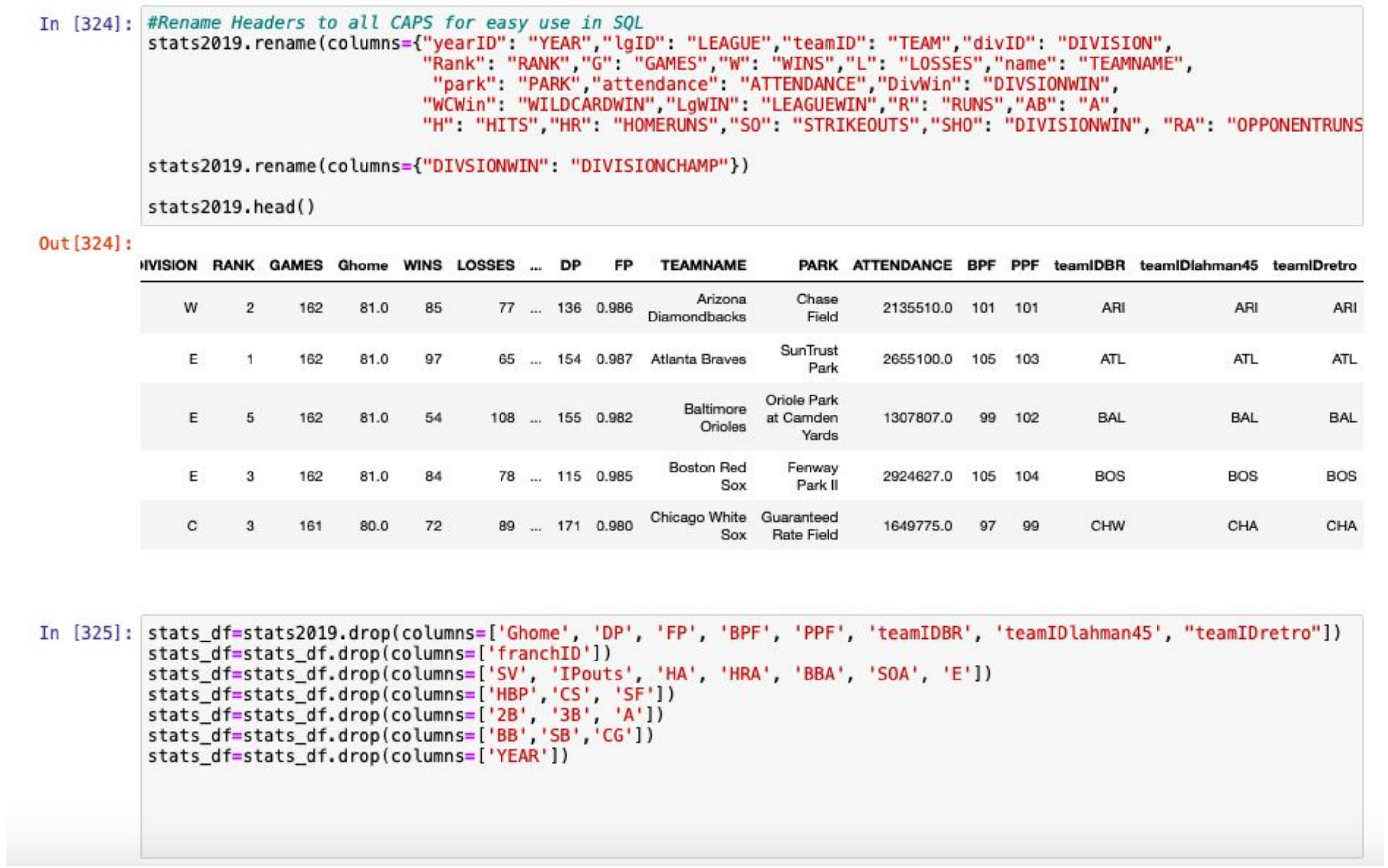
Data Source #2: CBS Sports

Transforming the Payroll Web Scrape was simple as it already was in a nice, easy format.

Tasks:

- Sort Team Name (A-Z)
- Rename Headers
- Add TEAMID column

T: Transform



In [324]:
```python
#Rename Headers to all CAPS for easy use in SQL
stats2019.rename(columns={"yearID": "YEAR","lgID": "LEAGUE","teamID": "TEAM","divID": "DIVISION",
                         "Rank": "RANK","G": "GAMES","W": "WINS","L": "LOSSES","name": "TEAMNAME",
                          "park": "PARK","attendance": "ATTENDANCE","DivWin": "DIVSIONWIN",
                         "WCWin": "WILDCARDWIN","LgWIN": "LEAGUEWIN","R": "RUNS","AB": "A",
                         "H": "HITS","HR": "HOMERUNS","SO": "STRIKEOUTS","SHO": "DIVISIONWIN", "RA": "OPPONENTRUNS

stats2019.rename(columns={"DIVSIONWIN": "DIVISIONCHAMP"})

stats2019.head()
```

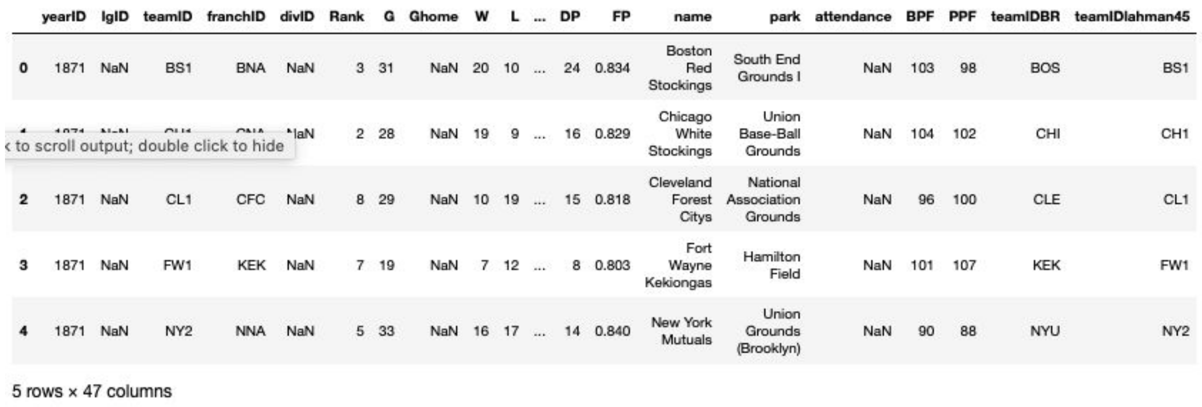
Out[324]:

IVISION	RANK	GAMES	Ghome	WINS	LOSSES	...	DP	FP	TEAMNAME	PARK	ATTENDANCE	BPF	PPF	teamIDBR	teamIDlahman45	teamIDretro
W	2	162	81.0	85	77	...	136	0.986	Arizona Diamondbacks	Chase Field	2135510.0	101	101	ARI	ARI	ARI
E	1	162	81.0	97	65	...	154	0.987	Atlanta Braves	SunTrust Park	2655100.0	105	103	ATL	ATL	ATL
E	5	162	81.0	54	108	...	155	0.982	Baltimore Orioles	Oriole Park at Camden Yards	1307807.0	99	102	BAL	BAL	BAL
E	3	162	81.0	84	78	...	115	0.985	Boston Red Sox	Fenway Park II	2924627.0	105	104	BOS	BOS	BOS
C	3	161	80.0	72	89	...	171	0.980	Chicago White Sox	Guaranteed Rate Field	1649775.0	97	99	CHW	CHA	CHA

In [325]:
```python
stats_df=stats2019.drop(columns=['Ghome', 'DP', 'FP', 'BPF', 'PPF', 'teamIDBR', 'teamIDlahman45', "teamIDretro"])
stats_df=stats_df.drop(columns=['franchID'])
stats_df=stats_df.drop(columns=['SV', 'IPouts', 'HA', 'HRA', 'BBA', 'SOA', 'E'])
stats_df=stats_df.drop(columns=['HBP','CS', 'SF'])
stats_df=stats_df.drop(columns=['2B', '3B', 'A'])
stats_df=stats_df.drop(columns=['BB','SB','CG'])
stats_df=stats_df.drop(columns=['YEAR'])
```

T: Transform

Before

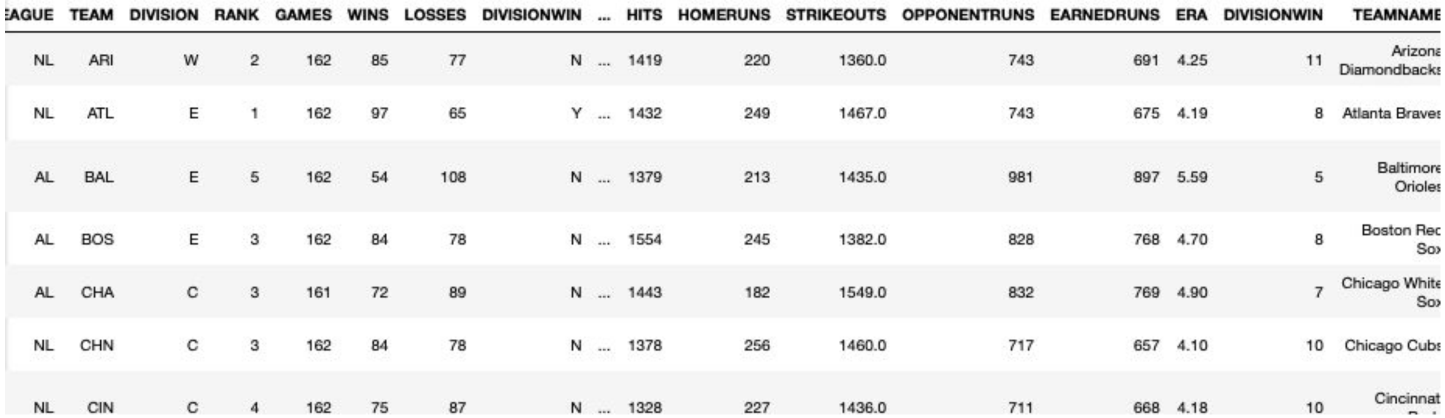


	yearID	lgID	teamID	franchID	divID	Rank	G	Ghome	W	L	...	DP	FP	name	park	attendance	BPF	PPF	teamIDBR	teamIDlahman45
0	1871	NaN	BS1	BNA	NaN	3	31	NaN	20	10	...	24	0.834	Boston Red Stockings	South End Grounds I	NaN	103	98	BOS	BS1
1	1871	NaN	CH1	CNA	NaN	2	28	NaN	19	9	...	16	0.829	Chicago White Stockings	Union Base-Ball Grounds	NaN	104	102	CHI	CH1
2	1871	NaN	CL1	CFC	NaN	8	29	NaN	10	19	...	15	0.818	Cleveland Forest Citys	National Association Grounds	NaN	96	100	CLE	CL1
3	1871	NaN	FW1	KEK	NaN	7	19	NaN	7	12	...	8	0.803	Fort Wayne Kekiongas	Hamilton Field	NaN	101	107	KEK	FW1
4	1871	NaN	NY2	NNA	NaN	5	33	NaN	16	17	...	14	0.840	New York Mutuals	Union Grounds (Brooklyn)	NaN	90	88	NYU	NY2

k to scroll output; double click to hide

5 rows × 47 columns

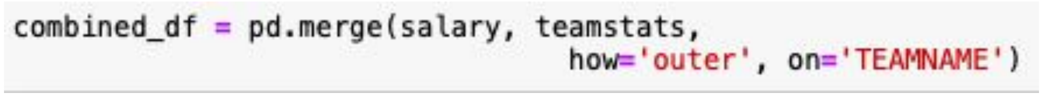
After



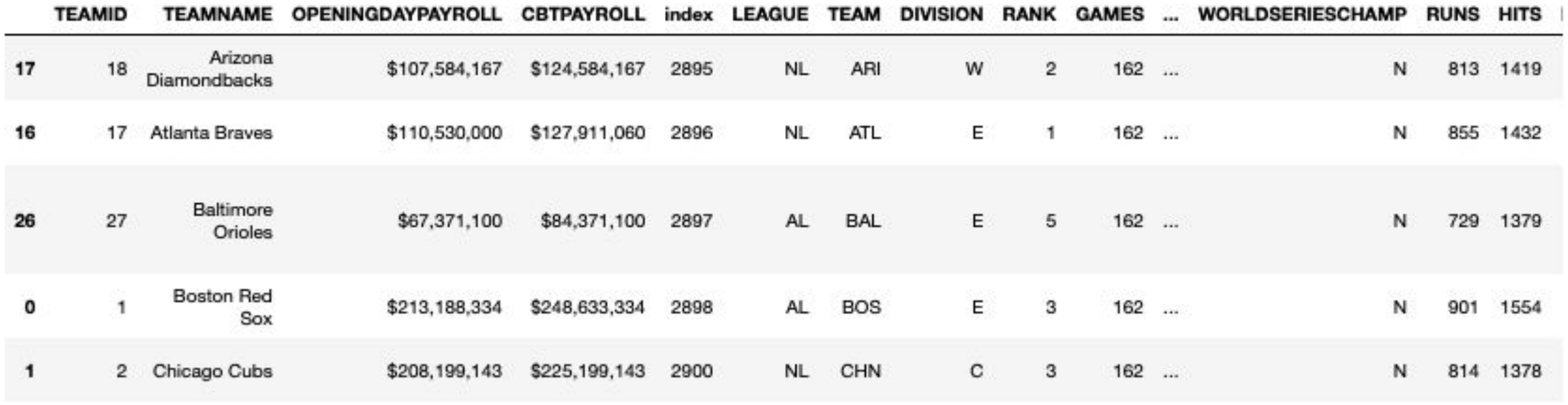
EAGUE	TEAM	DIVISION	RANK	GAMES	WINS	LOSSES	DIVISIONWIN	...	HITS	HOMERUNS	STRIKEOUTS	OPPONENTRUNS	EARNEDRUNS	ERA	DIVISIONWIN	TEAMNAME
NL	ARI	W	2	162	85	77	N	...	1419	220	1360.0	743	691	4.25	11	Arizona Diamondbacks
NL	ATL	E	1	162	97	65	Y	...	1432	249	1467.0	743	675	4.19	8	Atlanta Braves
AL	BAL	E	5	162	54	108	N	...	1379	213	1435.0	981	897	5.59	5	Baltimore Orioles
AL	BOS	E	3	162	84	78	N	...	1554	245	1382.0	828	768	4.70	8	Boston Red Sox
AL	CHA	C	3	161	72	89	N	...	1443	182	1549.0	832	769	4.90	7	Chicago White Sox
NL	CHN	C	3	162	84	78	N	...	1378	256	1460.0	717	657	4.10	10	Chicago Cubs
NL	CIN	C	4	162	75	87	N	...	1328	227	1436.0	711	668	4.18	10	Cincinnat

T: Transform Merge

After cleaning the dataframes, I was able to complete a merge of the payroll dataframe and the statistics dataframe. I completed this with an outer join on the column “TEAMNAME”



```
combined_df = pd.merge(salary, teamstats,
                                        how='outer', on='TEAMNAME')
```



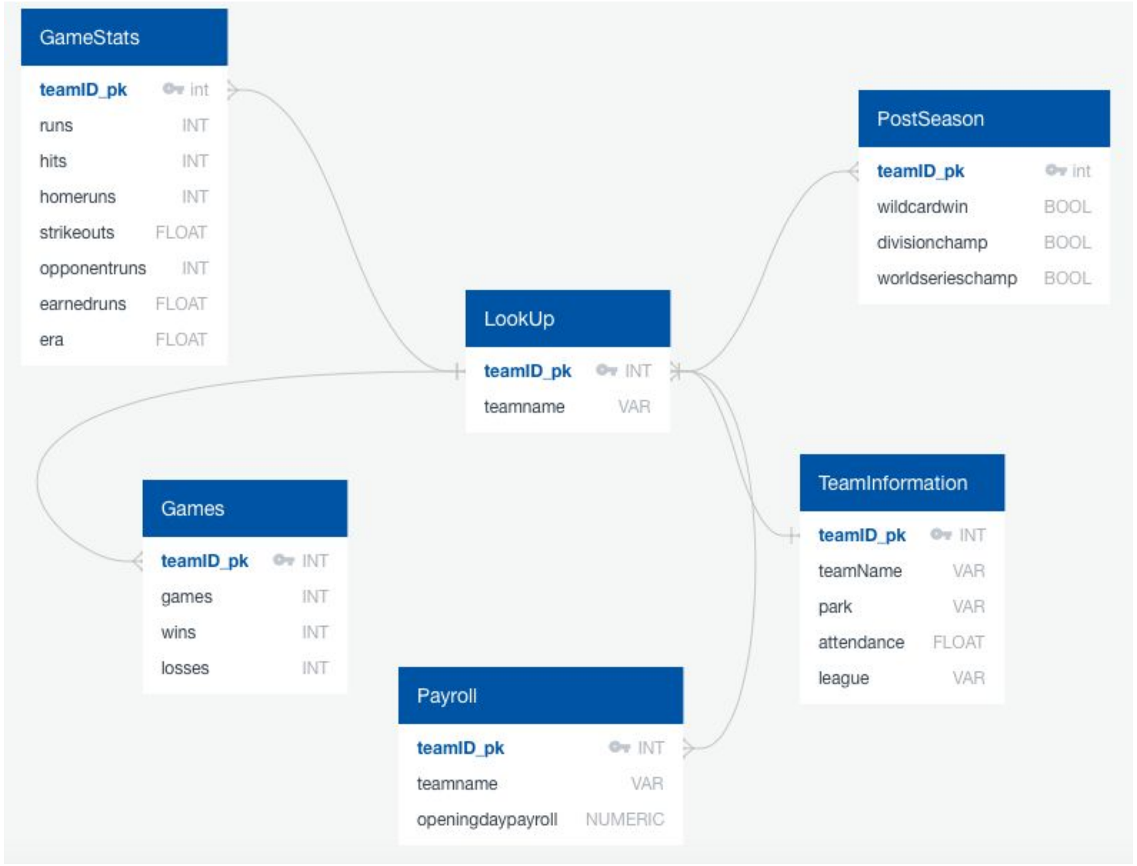
	TEAMID	TEAMNAME	OPENINGDAYPAYROLL	CBTPAYROLL	index	LEAGUE	TEAM	DIVISION	RANK	GAMES	...	WORLDSERIESCHAMP	RUNS	HITS
17	18	Arizona Diamondbacks	$107,584,167	$124,584,167	2895	NL	ARI	W	2	162	...	N	813	1419
16	17	Atlanta Braves	$110,530,000	$127,911,060	2896	NL	ATL	E	1	162	...	N	855	1432
26	27	Baltimore Orioles	$67,371,100	$84,371,100	2897	AL	BAL	E	5	162	...	N	729	1379
0	1	Boston Red Sox	$213,188,334	$248,633,334	2898	AL	BOS	E	3	162	...	N	901	1554
1	2	Chicago Cubs	$208,199,143	$225,199,143	2900	NL	CHN	C	3	162	...	N	814	1378

L: Load

Task #1: Create ERD

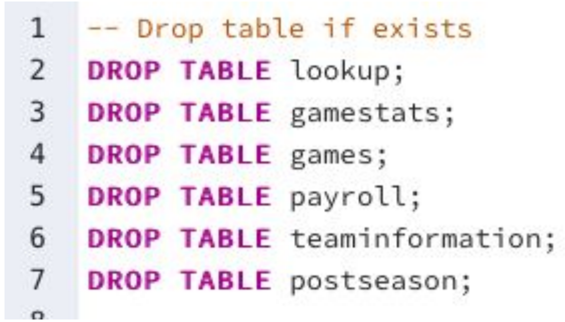
I created relational tables based off of a primary key of teamID_pk, which all tables would include.

teamID_pk works as a PK because it is an integer, avoiding any text PK.

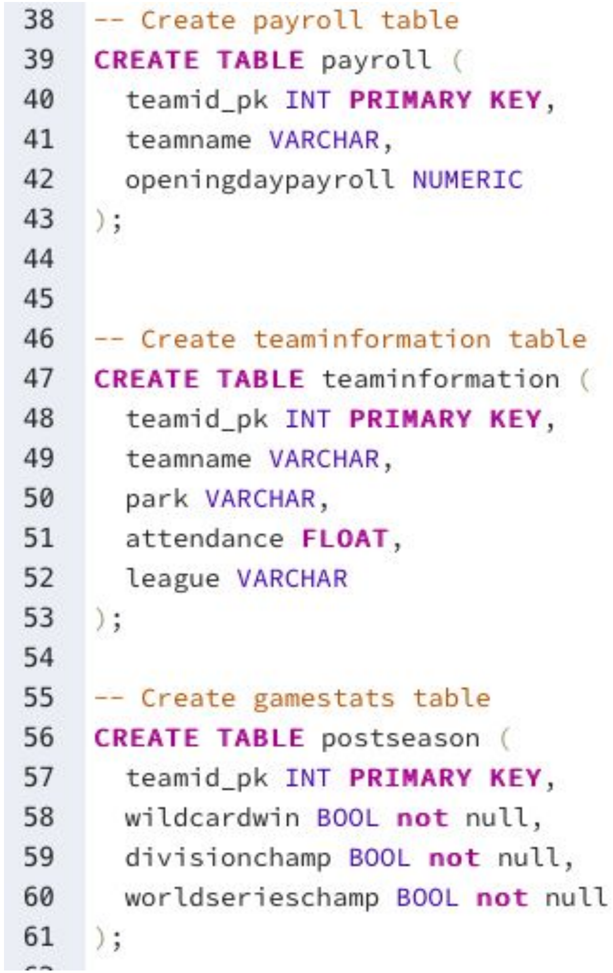


L: Load

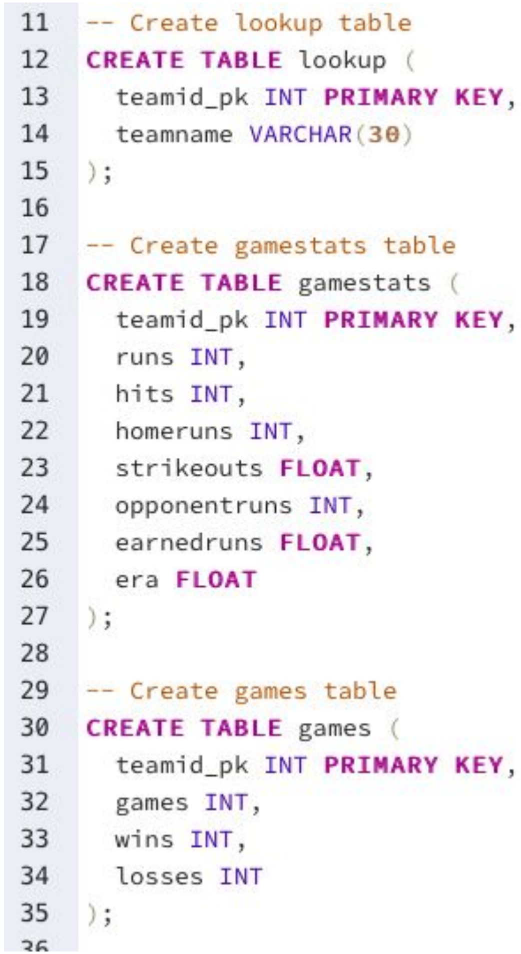
Task #2: Create SQL Tables



```sql
-- Drop table if exists
DROP TABLE lookup;
DROP TABLE gamestats;
DROP TABLE games;
DROP TABLE payroll;
DROP TABLE teaminformation;
DROP TABLE postseason;
```



```sql
-- Create payroll table
CREATE TABLE payroll (
  teamid_pk INT PRIMARY KEY,
  teamname VARCHAR,
  openingdaypayroll NUMERIC
);


-- Create teaminformation table
CREATE TABLE teaminformation (
  teamid_pk INT PRIMARY KEY,
  teamname VARCHAR,
  park VARCHAR,
  attendance FLOAT,
  league VARCHAR
);

-- Create gamestats table
CREATE TABLE postseason (
  teamid_pk INT PRIMARY KEY,
  wildcardwin BOOL not null,
  divisionchamp BOOL not null,
  worldserieschamp BOOL not null
);
```



```sql
-- Create lookup table
CREATE TABLE lookup (
  teamid_pk INT PRIMARY KEY,
  teamname VARCHAR(30)
);

-- Create gamestats table
CREATE TABLE gamestats (
  teamid_pk INT PRIMARY KEY,
  runs INT,
  hits INT,
  homeruns INT,
  strikeouts FLOAT,
  opponentruns INT,
  earnedruns FLOAT,
  era FLOAT
);

-- Create games table
CREATE TABLE games (
  teamid_pk INT PRIMARY KEY,
  games INT,
  wins INT,
  losses INT
);
```

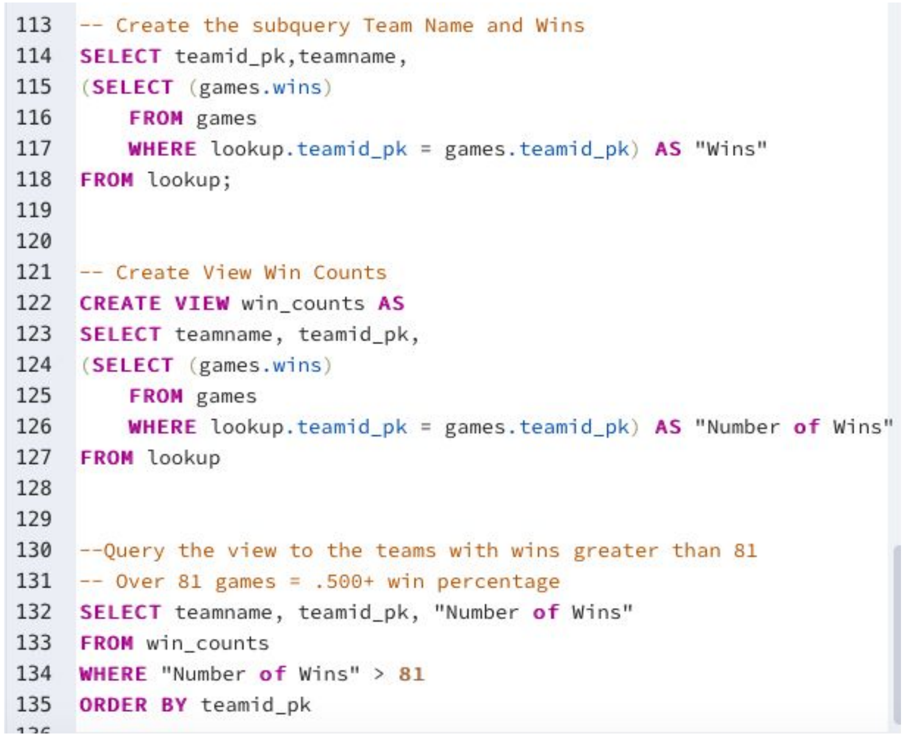
Proof Of Concept

Proof of Concept #1: Teams with 81+ wins

An 81 win season means the team had a win percentage of .500 or higher.

Tasks:

- Create a subquery
- Create a view
- Query the View



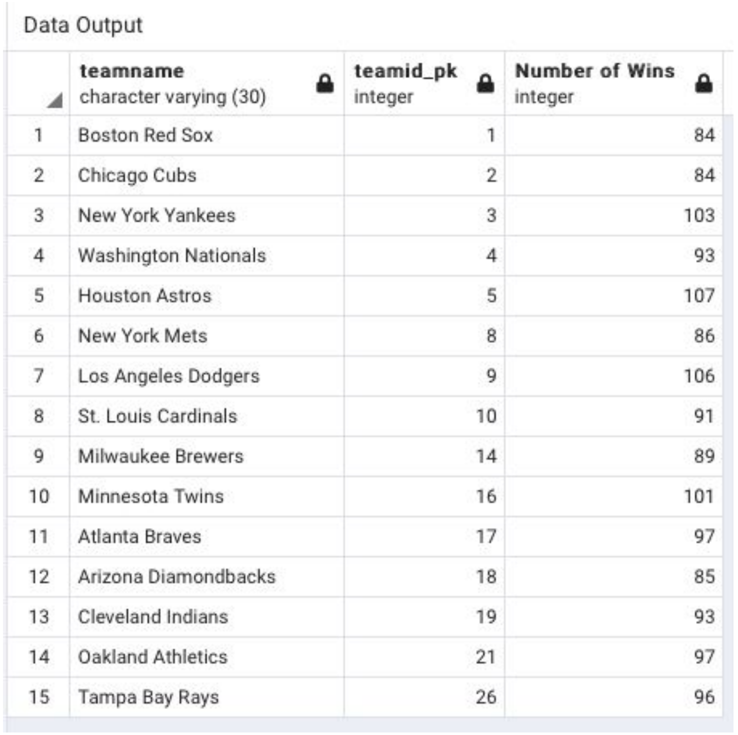
```sql
-- Create the subquery Team Name and Wins
SELECT teamid_pk,teamname,
(SELECT (games.wins)
    FROM games
    WHERE lookup.teamid_pk = games.teamid_pk) AS "Wins"
FROM lookup;


-- Create View Win Counts
CREATE VIEW win_counts AS
SELECT teamname, teamid_pk,
(SELECT (games.wins)
    FROM games
    WHERE lookup.teamid_pk = games.teamid_pk) AS "Number of Wins"
FROM lookup


--Query the view to the teams with wins greater than 81
-- Over 81 games = .500+ win percentage
SELECT teamname, teamid_pk, "Number of Wins"
FROM win_counts
WHERE "Number of Wins" > 81
ORDER BY teamid_pk
```

Proof Of Concept

Proof of Concept #1: Teams with 81+ wins

An 81 win season means the team had a win percentage of .500 or higher.

The query results in 15 teams with 81+ wins and with the teamid_pk (payroll rank) sorted you can see that 9 out of the 10 highest payrolls eclipsed this value.



Data Output

	teamname character varying (30)	teamid_pk integer	Number of Wins integer
1	Boston Red Sox	1	84
2	Chicago Cubs	2	84
3	New York Yankees	3	103
4	Washington Nationals	4	93
5	Houston Astros	5	107
6	New York Mets	8	86
7	Los Angeles Dodgers	9	106
8	St. Louis Cardinals	10	91
9	Milwaukee Brewers	14	89
10	Minnesota Twins	16	101
11	Atlanta Braves	17	97
12	Arizona Diamondbacks	18	85
13	Cleveland Indians	19	93
14	Oakland Athletics	21	97
15	Tampa Bay Rays	26	96

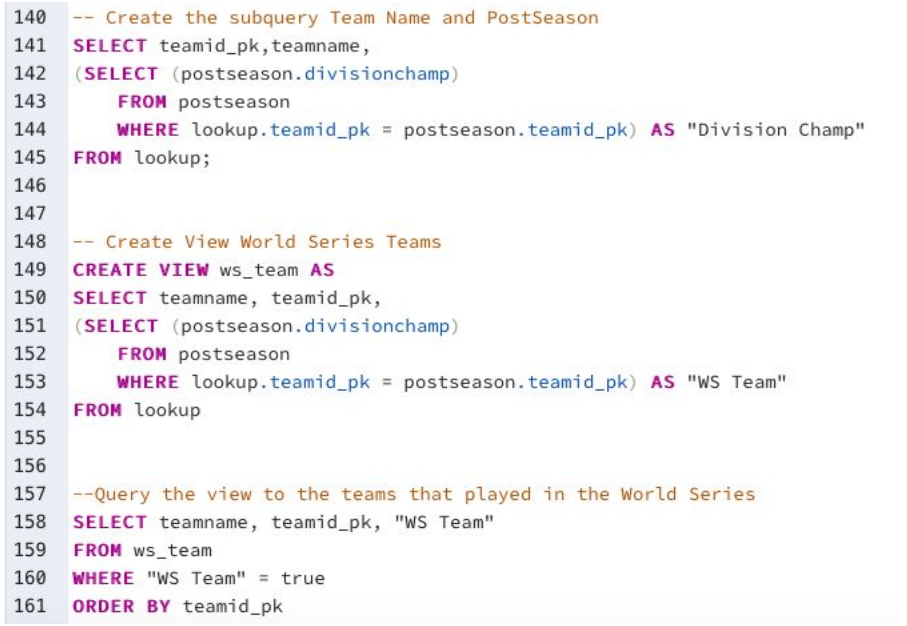
Proof Of Concept

Proof of Concept #2: World Series Teams

The teams in the World Series would conclude they won their League/Wildcard, Division Series, and Championship Series.

Tasks:

- Create a subquery
- Create a view
- Query the View



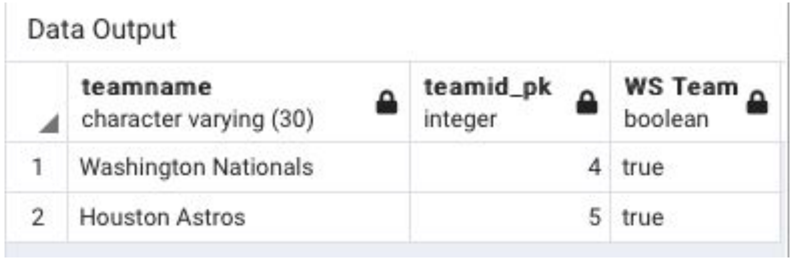
```
-- Create the subquery Team Name and PostSeason
SELECT teamid_pk,teamname,
(SELECT (postseason.divisionchamp)
    FROM postseason
    WHERE lookup.teamid_pk = postseason.teamid_pk) AS "Division Champ"
FROM lookup;


-- Create View World Series Teams
CREATE VIEW ws_team AS
SELECT teamname, teamid_pk,
(SELECT (postseason.divisionchamp)
    FROM postseason
    WHERE lookup.teamid_pk = postseason.teamid_pk) AS "WS Team"
FROM lookup


--Query the view to the teams that played in the World Series
SELECT teamname, teamid_pk, "WS Team"
FROM ws_team
WHERE "WS Team" = true
ORDER BY teamid_pk
```

Proof Of Concept

Proof of Concept #2: World Series Teams

The teams in the World Series would conclude they won their League/Wildcard, Division Series, and Championship Series.

The query shows the World Series Teams for 2019 as the Washington Nationals and Houston Astros. These teams each have the 4 and 5 highest payroll, respectfully.

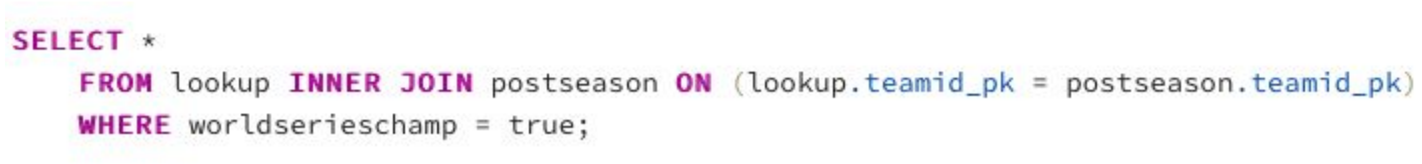


Data Output

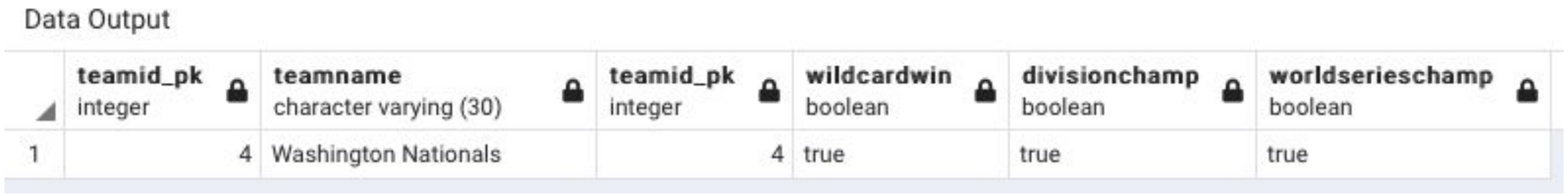
	teamname character varying (30)	teamid_pk integer	WS Team boolean
1	Washington Nationals	4	true
2	Houston Astros	5	true

Additional Query Examples

Locating the Winning Team of Championships in respect to their Payroll



```
SELECT *
    FROM lookup INNER JOIN postseason ON (lookup.teamid_pk = postseason.teamid_pk)
    WHERE worldserieschamp = true;
```

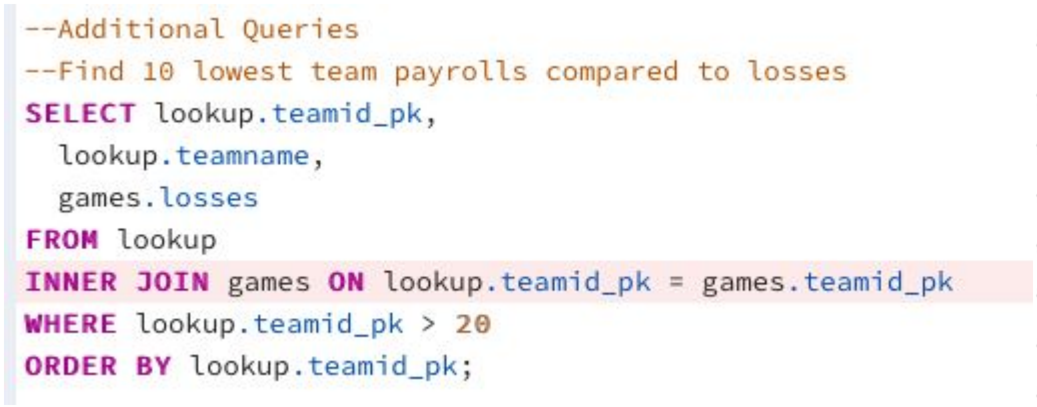


Data Output

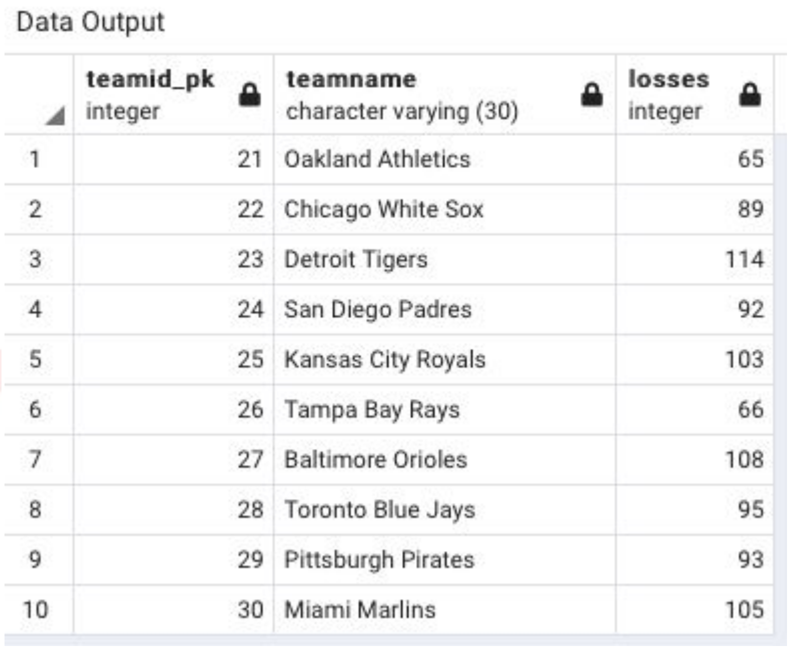
	teamid_pk integer	teamname character varying (30)	teamid_pk integer	wildcardwin boolean	divisionchamp boolean	worldserieschamp boolean
1	4	Washington Nationals	4	true	true	true

Additional Query Examples

Query 10 lowest team payrolls compared to loss totals



```
--Additional Queries
--Find 10 lowest team payrolls compared to losses
SELECT lookup.teamid_pk,
  lookup.teamname,
  games.losses
FROM lookup
INNER JOIN games ON lookup.teamid_pk = games.teamid_pk
WHERE lookup.teamid_pk > 20
ORDER BY lookup.teamid_pk;
```



Data Output

	teamid_pk integer	teamname character varying (30)	losses integer
1	21	Oakland Athletics	65
2	22	Chicago White Sox	89
3	23	Detroit Tigers	114
4	24	San Diego Padres	92
5	25	Kansas City Royals	103
6	26	Tampa Bay Rays	66
7	27	Baltimore Orioles	108
8	28	Toronto Blue Jays	95
9	29	Pittsburgh Pirates	93
10	30	Miami Marlins	105

Conclusion

Extracting data from various sources (CSV tables and Web Scrapes) allowed for the creation of a master dataframe comparing 2019 MLB team payrolls with their season outcomes. The season outcomes include statistics, game outcomes, postseason accolades, which can be compared to the team payroll.

Overall, utilizing a relational database to create table relationships allowed for a multitude of queries to be completed on the data creating additional datasets that can be used for producing correlations between payroll and various statistical measures for MLB teams in 2019.