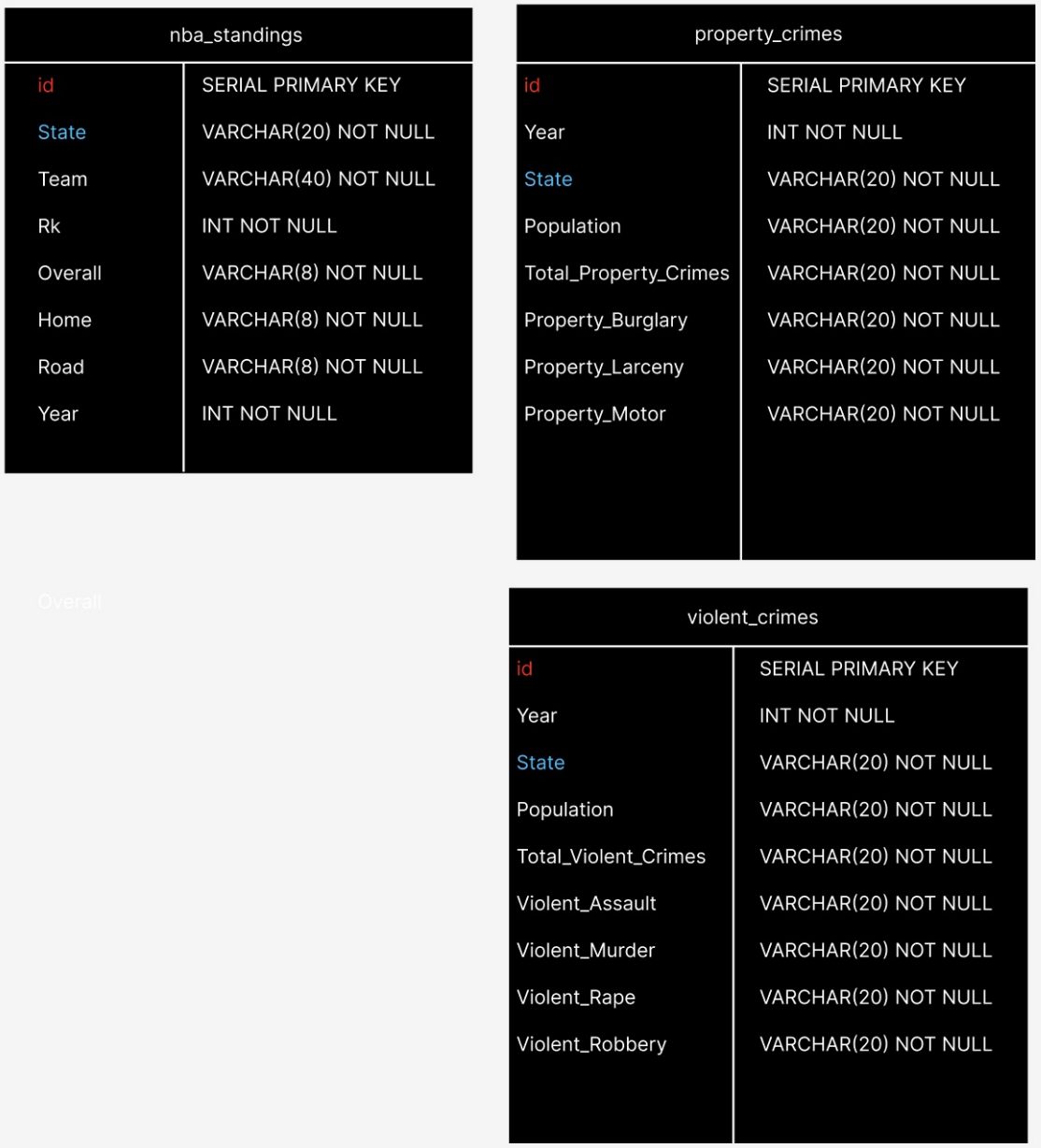
DB SCHEMA



primary key in RED color &

foreign key in BLUE color

What is your goal?

My goal is to load in nba standings data and crime rate data into a relational database.

What types of questions will you be able to answer?

Overall, I will be able to answer questions relating to how any NBA team's success can benefit their state in terms of crime. Or if a team is doing poorly does crime increase? I will be looking at crime data and NBA records from 2010-2019

What can your RDB be used for, who would the end user be?

The RDB could be used by city officials and NBA teams to look at the potential impact an NBA team's success could have on a state as a whole. In general, it highlights the broader societal implications the NBA can have on an area

EXTRACT & EXPLORE

```
#imports
import pandas as pd
import numpy as np
from google.colab import files
import matplotlib.pyplot as plt
import seaborn as sns


"""
NBA standings data from amazon s3 - Standings from 2010/11 Season to 2018/19 for all 30 NBA teams

Links to datasets:
https://www.basketball-reference.com/leagues/NBA_2011_standings.html
https://www.basketball-reference.com/leagues/NBA_2012_standings.html
https://www.basketball-reference.com/leagues/NBA_2013_standings.html
https://www.basketball-reference.com/leagues/NBA_2014_standings.html
https://www.basketball-reference.com/leagues/NBA_2015_standings.html
https://www.basketball-reference.com/leagues/NBA_2016_standings.html
https://www.basketball-reference.com/leagues/NBA_2017_standings.html
https://www.basketball-reference.com/leagues/NBA_2018_standings.html
https://www.basketball-reference.com/leagues/NBA_2019_standings.html
"""
NBA_2010_11 = pd.read_csv('https://ista322finalproject.s3.amazonaws.com/2010-11_NBA_Standings.csv', index_col = 1, skiprows = 1)
NBA_2011_12 = pd.read_csv('https://ista322finalproject.s3.amazonaws.com/2011-12_NBA_Standings.csv', index_col = 1, skiprows = 1)
NBA_2012_13 = pd.read_csv('https://ista322finalproject.s3.amazonaws.com/2012-13_NBA_Standings.csv', index_col = 1, skiprows = 1)
NBA_2013_14 = pd.read_csv('https://ista322finalproject.s3.amazonaws.com/2013-14_NBA_Standings.csv', index_col = 1, skiprows = 1)
NBA_2014_15 = pd.read_csv('https://ista322finalproject.s3.amazonaws.com/2014-15_NBA_Standings.csv', index_col = 1, skiprows = 1)
NBA_2015_16 = pd.read_csv('https://ista322finalproject.s3.amazonaws.com/2015-16_NBA_Standings.csv', index_col = 1, skiprows = 1)
NBA_2016_17 = pd.read_csv('https://ista322finalproject.s3.amazonaws.com/2016-17_NBA_Standings.csv', index_col = 1, skiprows = 1)
NBA_2017_18 = pd.read_csv('https://ista322finalproject.s3.amazonaws.com/2017-18_NBA_Standings.csv', index_col = 1, skiprows = 1)
NBA_2018_19 = pd.read_csv('https://ista322finalproject.s3.amazonaws.com/2018-19_NBA_Standings.csv', index_col = 1, skiprows = 1)


"""
Crime rates data from amazon s3

Link to datatset:
https://corgis-edu.github.io/corgis/csv/state_crime/
"""
crimes = pd.read_csv('https://ista322finalproject.s3.amazonaws.com/state_crime.csv')


#lets take a look at one of the standings df. They all follow this same formart
NBA_2010_11
```

Team																				
Chicago Bulls	1	62-20	36-5	26-15	39-13	23-7	10-8	15-1	14-4	7-3	...	24-4	9-5	31-5	1-1	8-5	12-4	12-4	8-3	13-3	8-0
San Antonio Spurs	2	61-21	36-5	25-16	23-7	38-14	6-4	9-1	8-2	14-4	...	15-11	7-4	32-9	1-1	14-1	13-2	12-3	9-3	8-8	4-3
Miami Heat	3	58-24	30-11	28-13	38-14	20-10	13-5	12-6	13-3	6-4	...	17-9	2-8	35-6	3-1	7-7	15-1	9-5	9-3	9-6	6-1
Dallas Mavericks	4	57-25	29-12	28-13	22-8	35-17	9-1	4-6	9-1	12-6	...	17-9	10-6	22-8	2-1	11-3	11-3	8-8	11-1	10-6	4-3
Los Angeles Lakers	5	57-25	30-11	27-14	21-9	36-16	9-1	6-4	6-4	12-6	...	19-6	4-4	30-9	3-0	10-5	10-5	10-5	9-4	12-1	3-5
Boston Celtics	6	56-26	33-8	23-18	37-15	19-11	13-3	13-5	11-7	8-2	...	16-12	8-8	30-7	2-1	11-3	11-3	12-4	7-4	9-7	4-4
Oklahoma City Thunder	7	55-27	30-11	25-16	22-8	33-19	6-4	8-2	8-2	13-3	...	20-8	10-5	22-8	2-1	10-5	11-5	7-6	6-5	14-2	5-3
Orlando Magic	8	52-30	29-12	23-18	36-16	16-14	12-6	13-5	11-5	4-6	...	16-9	6-7	33-9	1-1	12-3	8-8	10-6	7-4	9-6	5-2
Denver Nuggets	9	50-32	33-8	17-24	20-10	30-22	5-5	8-2	7-3	9-7	...	18-7	7-8	29-10	2-1	8-5	8-7	10-7	7-6	10-3	5-3
Portland Trail Blazers	10	48-34	30-11	18-23	18-12	30-22	5-5	8-2	5-5	10-6	...	16-10	7-4	19-15	3-0	5-9	9-7	8-6	8-4	10-6	5-2
Memphis Grizzlies	11	46-36	30-11	16-25	16-14	30-22	5-5	7-3	4-6	11-7	...	15-10	9-10	23-12	2-1	6-9	6-8	11-6	8-4	9-5	4-3
New Orleans Hornets	12	46-36	28-13	18-23	19-11	27-25	4-6	6-4	9-1	9-9	...	13-11	8-6	16-17	3-0	9-5	7-9	12-4	4-8	8-6	3-4
Atlanta Hawks	13	44-38	24-17	20-21	31-21	13-17	10-8	12-6	9-7	5-5	...	10-17	7-1	21-21	3-0	8-7	10-7	9-4	6-6	7-8	1-6

crimes

	State	Year	Data.Population	Data.Rates.Property.All	Data.Rates.Property.Burglary	Data.Rates.Property.Larceny	Data
0	Alabama	1960	3266740	1035.4	355.9	592.1	
1	Alabama	1961	3302000	985.5	339.3	569.4	
2	Alabama	1962	3358000	1067.0	349.1	634.5	
3	Alabama	1963	3347000	1150.9	376.9	683.4	
4	Alabama	1964	3407000	1358.7	466.6	784.1	
...	...	...	...	...	...	...	...
3110	Wyoming	2015	586107	1902.6	300.6	1500.9	
3111	Wyoming	2016	585501	1957.3	302.5	1518.2	
3112	Wyoming	2017	579315	1830.4	275.0	1421.0	
3113	Wyoming	2018	577737	1785.1	264.0	1375.9	
3114	Wyoming	2019	578759	1571.1	241.2	1206.7	

3115 rows × 21 columns

TRANSFORM

```
# select columns we want from crimes df
crime = crimes[['State', 'Year', 'Data.Population', 'Data.Totals.Property.All', 'Data.Totals.Property.Burglary', 'Data.Totals.Pr
crime
```

	State	Year	Data.Population	Data.Totals.Property.All	Data.Totals.Property.Burglary	Data.Totals.Property.Larceny	D
0	Alabama	1960	3266740	33823	11626	19344	
1	Alabama	1961	3302000	32541	11205	18801	
2	Alabama	1962	3358000	35829	11722	21306	
3	Alabama	1963	3347000	38521	12614	22874	
4	Alabama	1964	3407000	46290	15898	26713	
...	...	...	...	...	...	...	
3110	Wyoming	2015	586107	11151	1762	8797	
3111	Wyoming	2016	585501	11460	1771	8889	
3112	Wyoming	2017	579315	10604	1593	8232	

```
# Renaming columns for clarity
crime.columns = ['State', 'Year', 'Population', 'Total_Property_Crimes', 'Property_Burglary', 'Propert_Larceny', 'Property_Motor
crime
```

	State	Year	Population	Total_Property_Crimes	Property_Burglary	Propert_Larceny	Property_Motor	Total_Violent_Crime
0	Alabama	1960	3266740	33823	11626	19344	2853	60
1	Alabama	1961	3302000	32541	11205	18801	2535	55
2	Alabama	1962	3358000	35829	11722	21306	2801	52
3	Alabama	1963	3347000	38521	12614	22874	3033	61
4	Alabama	1964	3407000	46290	15898	26713	3679	72
...	...	...	...	...	...	...	...	
3110	Wyoming	2015	586107	11151	1762	8797	592	13
3111	Wyoming	2016	585501	11460	1771	8889	800	14
3112	Wyoming	2017	579315	10604	1593	8232	779	13
3113	Wyoming	2018	577737	10313	1525	7949	839	12
3114	Wyoming	2019	578759	9093	1396	6984	713	12

3115 rows × 12 columns

```
# Add a comma as a a thounsands seperator for readability
crime['Population'] = crime['Population'].map('{:,}'.format)
crime['Total_Property_Crimes'] = crime['Total_Property_Crimes'].map('{:,}'.format)
crime['Total_Violent_Crimes'] = crime['Total_Violent_Crimes'].map('{:,}'.format)
crime['Property_Burglary'] = crime['Property_Burglary'].map('{:,}'.format)
crime['Propert_Larceny'] = crime['Propert_Larceny'].map('{:,}'.format)
crime['Property_Motor'] = crime['Property_Motor'].map('{:,}'.format)
crime['Violent_Assault'] = crime['Violent_Assault'].map('{:,}'.format)
crime['Violent_Murder'] = crime['Violent_Murder'].map('{:,}'.format)
crime['Violent_Rape'] = crime['Violent_Rape'].map('{:,}'.format)
crime['Violent_Robbery'] = crime['Violent_Robbery'].map('{:,}'.format)

crime
```

```
<ipython-input-7-0c3fce69ef7e>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view
  crime['Population'] = crime['Population'].map('{:,}'.format)
<ipython-input-7-0c3fce69ef7e>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view
  crime['Total_Property_Crimes'] = crime['Total_Property_Crimes'].map('{:,}'.format)
<ipython-input-7-0c3fce69ef7e>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view
  crime['Total_Violent_Crimes'] = crime['Total_Violent_Crimes'].map('{:,}'.format)
<ipython-input-7-0c3fce69ef7e>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view
  crime['Property_Burglary'] = crime['Property_Burglary'].map('{:,}'.format)
<ipython-input-7-0c3fce69ef7e>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view
  crime['Propert_Larceny'] = crime['Propert_Larceny'].map('{:,}'.format)
<ipython-input-7-0c3fce69ef7e>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view
  crime['Property_Motor'] = crime['Property_Motor'].map('{:,}'.format)
<ipython-input-7-0c3fce69ef7e>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view
  crime['Violent_Assault'] = crime['Violent_Assault'].map('{:,}'.format)
<ipython-input-7-0c3fce69ef7e>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view
  crime['Violent_Murder'] = crime['Violent_Murder'].map('{:,}'.format)
<ipython-input-7-0c3fce69ef7e>:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view
  crime['Violent_Rape'] = crime['Violent_Rape'].map('{:,}'.format)
<ipython-input-7-0c3fce69ef7e>:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view
  crime['Violent_Robbery'] = crime['Violent_Robbery'].map('{:,}'.format)
```

	State	Year	Population	Total_Property_Crimes	Property_Burglary	Propert_Larceny	Property_Motor	Total_Violent_Crime
0	Alabama	1960	3,266,740	38,803	11,606	19,244	8,953	6,0

```
# Only need year from 2011 on because these are the years of the NBA data being looked at, so grabbing rows from 2011-2019
crime_2011_to_2019 = crime.loc[(crime['Year'] >= 2011) & (crime['Year'] <= 2019)]
crime_2011_to_2019
```

	State	Year	Population	Total_Property_Crimes	Property_Burglary	Propert_Larceny	Property_Motor	Total_Violent_Crime
51	Alabama	2011	4,803,689	173,192	51,119	111,411	10,662	20,1
52	Alabama	2012	4,822,023	168,878	47,481	111,523	9,874	21,6
53	Alabama	2013	4,833,722	161,993	42,429	108,993	10,571	20,8

```
#Which states actually have NBA teams... Not all 50 states in the crime df are needed

"""
1. Arizona:
  Phoenix Suns

2. California:
  Golden State Warriors
  Los Angeles Clippers
  Los Angeles Lakers
  Sacramento Kings

3. Colorado:
  Denver Nuggets

4. Florida:
  Miami Heat
  Orlando Magic

5. Georgia:
  Atlanta Hawks

6. Illinois:
  Chicago Bulls

7. Indiana:
  Indiana Pacers

8. Louisiana:
  New Orleans Pelicans

9. Massachusetts:
  Boston Celtics

10. Michigan:
  Detroit Pistons

11. Minnesota:
  Minnesota Timberwolves

12. New York:
  Brooklyn Nets
  New York Knicks

13. North Carolina:
  Charlotte Hornets

14. Ohio:
  Cleveland Cavaliers

15. Oklahoma:
  Oklahoma City Thunder

16. Oregon:
  Portland Trail Blazers

17. Pennsylvania:
  Philadelphia 76ers

18. Tennessee:
  Memphis Grizzlies

19. Texas:
  Dallas Mavericks
  Houston Rockets
  San Antonio Spurs

20. Utah:
  Utah Jazz

21. Wisconsin:
  Milwaukee Bucks

22. Washington DC*:
  Washington Wizards
"""
```

```
'\n1. Arizona:\n  Phoenix Suns\n\n2. California:\n  Golden State Warriors\n  Los Angeles Clippers\n  Los Angeles Lakers\n
Sacramento Kings\n\n3. Colorado:\n  Denver Nuggets\n\n4. Florida:\n  Miami Heat\n  Orlando Magic\n\n5. Georgia:\n  Atlanta
Hawks\n\n6. Illinois:\n  Chicago Bulls\n\n7. Indiana:\n  Indiana Pacers\n\n8. Louisiana:\n  New Orleans Pelicans\n\n9. Mass
achusetts:\n  Boston Celtics\n\n10. Michigan:\n  Detroit Pistons\n\n11. Minnesota:\n  Minnesota Timberwolves\n\n12. New Yor
k:\n  Brooklyn Nets\n  New York Knicks\n\n13. North Carolina:\n  Charlotte Hornets\n\n14. Ohio:\n  Cleveland Cavaliers\n\n1
5. Oklahoma:\n  Oklahoma City Thunder\n\n16. Oregon:\n  Portland Trail Blazers\n\n17. Pennsylvania:\n  Philadelphia 76ers\n
```

```
# List of states with NBA teams
nba_states = ['Arizona', 'California', 'Colorado', 'Florida', 'Georgia', 'Illinois', 'Indiana', 'Louisiana',
              'Massachusetts', 'Michigan', 'Minnesota', 'New York', 'North Carolina', 'Ohio', 'Oklahoma',
              'Oregon', 'Pennsylvania', 'Tennessee', 'Texas', 'Utah', 'Wisconsin', 'District of Columbia']

# Filter the crime DataFrame based on the states with NBA teams
nba_crime_states = crime_2011_to_2019[crime_2011_to_2019['State'].isin(nba_states)]

# Show that we have the correct states
nba_crime_states['State'].unique()
```

```
array(['Arizona', 'California', 'Colorado', 'District of Columbia',
       'Florida', 'Georgia', 'Illinois', 'Indiana', 'Louisiana',
       'Massachusetts', 'Michigan', 'Minnesota', 'New York',
       'North Carolina', 'Ohio', 'Oklahoma', 'Oregon', 'Pennsylvania',
       'Tennessee', 'Texas', 'Utah', 'Wisconsin'], dtype=object)
```

```
nba_crime_states
```

	State	Year	Population	Total_Property_Crimes	Property_Burglary	Propert_Larceny	Property_Motor	Total_Violent_Crim
171	Arizona	2011	6,467,315	229,896	54,695	155,400	19,801	26,7
172	Arizona	2012	6,553,255	231,930	52,934	159,838	19,158	28,1
173	Arizona	2013	6,626,624	225,243	48,533	159,272	17,438	27,5
174	Arizona	2014	6,731,484	215,240	43,562	154,091	17,587	26,9
175	Arizona	2015	6,828,065	207,107	37,957	152,365	16,785	28,0
...	...	...	...	...	...	...	...	
3050	Wisconsin	2015	5,771,337	113,924	19,554	83,385	10,985	17,6
3051	Wisconsin	2016	5,778,708	111,720	19,425	82,337	9,958	17,6
3052	Wisconsin	2017	5,795,483	104,802	17,599	77,735	9,468	18,5
3053	Wisconsin	2018	5,813,568	90,686	14,099	67,953	8,634	17,1
3054	Wisconsin	2019	5,822,434	85,672	12,667	65,620	7,385	17,0

198 rows × 12 columns

```
# create list of all nba standings df, so we can combine them all
all_seasons = [NBA_2010_11, NBA_2011_12, NBA_2012_13, NBA_2013_14, NBA_2014_15, NBA_2015_16, NBA_2016_17, NBA_2017_18, NBA_2018_

# Combine each seasons standings into a single DataFrame
all_seasons_df = pd.concat(all_seasons)
all_seasons_df
```

	Rk	Overall	Home	Road	E	W	A	C	SE	NW	...	Post	≤3	≥10	Oct	Nov	Dec	Jan	Feb	Mar	Apr

```
# Do not need all these columns, only intrested in Team, Rk, Home, Road
NBA_2011_to_2019 = all_seasons_df[['Rk', 'Overall', 'Home', 'Road']]
NBA_2011_to_2019
```

Team	Rk	Overall	Home	Road
Chicago Bulls	1	62-20	36-5	26-15
San Antonio Spurs	2	61-21	36-5	25-16
Miami Heat	3	58-24	30-11	28-13
Dallas Mavericks	4	57-25	29-12	28-13
Los Angeles Lakers	5	57-25	30-11	27-14
...	...	...	...	...
Atlanta Hawks	26	29-53	17-24	12-29
Chicago Bulls	27	22-60	9-32	13-28
Cleveland Cavaliers	28	19-63	13-28	6-35
Phoenix Suns	29	19-63	12-29	7-34
New York Knicks	30	17-65	9-32	8-33

270 rows × 4 columns

```
 # Adding year column for clarity. 30 NBA teams, so every 30 rows will be a new year

# Creating a list of years corresponding for 30 teams
years = ['2011'] * 30 + ['2012'] * 30 + ['2013'] * 30 + ['2014'] * 30 + ['2015'] * 30 + ['2016'] * 30 + ['2017'] * 30 + ['2018']

# Adding a 'Year' column to the df
NBA_2011_to_2019['Year'] = years

NBA_2011_to_2019
```

```
<ipython-input-14-1951f4289b86>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view
  NBA_2011_to_2019['Year'] = years
```

Team	Rk	Overall	Home	Road	Year
Chicago Bulls	1	62-20	36-5	26-15	2011
San Antonio Spurs	2	61-21	36-5	25-16	2011
Miami Heat	3	58-24	30-11	28-13	2011
Dallas Mavericks	4	57-25	29-12	28-13	2011
Los Angeles Lakers	5	57-25	30-11	27-14	2011
...	...	...	...	...	...
Atlanta Hawks	26	29-53	17-24	12-29	2019
Chicago Bulls	27	22-60	9-32	13-28	2019
Cleveland Cavaliers	28	19-63	13-28	6-35	2019
Phoenix Suns	29	19-63	12-29	7-34	2019
New York Knicks	30	17-65	9-32	8-33	2019

270 rows × 5 columns

```
# need to get rid of Team column as the index so I can reference it in the next step
NBA_2011_to_2019.reset_index(inplace=True)
NBA_2011_to_2019
```

	Team	Rk	Overall	Home	Road	Year
0	Chicago Bulls	1	62-20	36-5	26-15	2011
1	San Antonio Spurs	2	61-21	36-5	25-16	2011
2	Miami Heat	3	58-24	30-11	28-13	2011
3	Dallas Mavericks	4	57-25	29-12	28-13	2011
4	Los Angeles Lakers	5	57-25	30-11	27-14	2011
...	...	...	...	...	...	...
265	Atlanta Hawks	26	29-53	17-24	12-29	2019
266	Chicago Bulls	27	22-60	9-32	13-28	2019
267	Cleveland Cavaliers	28	19-63	13-28	6-35	2019
268	Phoenix Suns	29	19-63	12-29	7-34	2019
269	New York Knicks	30	17-65	9-32	8-33	2019

```
team_state_mapping = {
    'Phoenix Suns': 'Arizona',
    'Golden State Warriors': 'California',
    'Los Angeles Clippers': 'California',
    'Los Angeles Lakers': 'California',
    'Sacramento Kings': 'California',
    'Denver Nuggets': 'Colorado',
    'Miami Heat': 'Florida',
    'Orlando Magic': 'Florida',
    'Atlanta Hawks': 'Georgia',
    'Chicago Bulls': 'Illinois',
    'Indiana Pacers': 'Indiana',
    'New Orleans Pelicans': 'Louisiana',
    'Boston Celtics': 'Massachusetts',
    'Detroit Pistons': 'Michigan',
    'Minnesota Timberwolves': 'Minnesota',
    'Brooklyn Nets': 'New York',
    'New York Knicks': 'New York',
    'Charlotte Hornets': 'North Carolina',
    'Cleveland Cavaliers': 'Ohio',
    'Oklahoma City Thunder': 'Oklahoma',
    'Portland Trail Blazers': 'Oregon',
    'Philadelphia 76ers': 'Pennsylvania',
    'Memphis Grizzlies': 'Tennessee',
    'Dallas Mavericks': 'Texas',
    'Houston Rockets': 'Texas',
    'San Antonio Spurs': 'Texas',
    'Utah Jazz': 'Utah',
    'Milwaukee Bucks': 'Wisconsin',
    'Washington Wizards': 'Washington DC',
}

# Adding the 'State' column based on the 'Team' column
NBA_2011_to_2019['State'] = NBA_2011_to_2019['Team'].map(team_state_mapping)
NBA_2011_to_2019
```

```
<ipython-input-16-48ad94a85996>:34: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view
  NBA_2011_to_2019['State'] = NBA_2011_to_2019['Team'].map(team_state_mapping)
```

	Team	Rk	Overall	Home	Road	Year	State

```
# Convert 'Year' column to int64 in both dataframes so we can merge on this column
NBA_2011_to_2019['Year'] = NBA_2011_to_2019['Year'].astype('int64')
nba_crime_states['Year'] = nba_crime_states['Year'].astype('int64')

# Merge the DataFrames on 'Year' and 'State'
nba_records_vs_crime = pd.merge(NBA_2011_to_2019, nba_crime_states, on=['Year', 'State'])
```

```
<ipython-input-17-50775d45bcdf>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view
  NBA_2011_to_2019['Year'] = NBA_2011_to_2019['Year'].astype('int64')
<ipython-input-17-50775d45bcdf>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view
  nba_crime_states['Year'] = nba_crime_states['Year'].astype('int64')
```

```
# now we have our full df with all the columns we want from the NBA standings data as well as the crime data. It has also been c
nba_records_vs_crime
```

	Team	Rk	Overall	Home	Road	Year	State	Population	Total_Property_Crimes	Property_Burglary	Propert_Larceny	Pr
0	Chicago Bulls	1	62-20	36-5	26-15	2011	Illinois	12,859,752	344,468	77,719	237,362	
1	San Antonio Spurs	2	61-21	36-5	25-16	2011	Texas	25,631,778	892,810	215,755	613,131	
2	Dallas Mavericks	4	57-25	29-12	28-13	2011	Texas	25,631,778	892,810	215,755	613,131	
3	Houston Rockets	14	43-39	25-16	18-23	2011	Texas	25,631,778	892,810	215,755	613,131	
4	Miami Heat	3	58-24	30-11	28-13	2011	Florida	19,082,262	671,200	170,171	461,408	
...	...	...	...	...	...	...	...	...	...	...	...	
238	New Orleans Pelicans	24	33-49	19-22	14-27	2019	Louisiana	4,648,794	146,993	26,918	109,359	
239	Atlanta Hawks	26	29-53	17-24	12-29	2019	Georgia	10,617,423	252,249	39,506	188,967	
240	Chicago Bulls	27	22-60	9-32	13-28	2019	Illinois	12,671,821	233,984	34,433	180,776	
241	Cleveland Cavaliers	28	19-63	13-28	6-35	2019	Ohio	11,689,100	240,291	43,894	177,725	
242	Phoenix Suns	29	19-63	12-29	7-34	2019	Arizona	7,278,717	177,638	28,699	130,788	

243 rows × 17 columns

INDIVIDUAL TEAMS

[ ] *3 cells hidden*

TIMBERWOLVES VISUALIZATION

[ ] *5 cells hidden*

JAZZ VISUALIZATION

[ ] ↳ 5 cells hidden

CONNECT

```
"""
Data is going to be in a SQL relational database
Using AWS

Creating three tables for relational database
1. nba_standings table
2. property_crimes table
3. violent_crimes table

CODE WILL DROP RDB TABLES BEFORE CREATING THEM

In this load step I will be creating the primary and foreign key.
The Primary key for each table is 'id'.
The foreign key that connects each table is 'state_id'

conn.close() AT END!
"""

# AWS - Connecting to my database
#import PostgreSQL
import psycopg2
def get_conn_cur():
 # UPDATE WITH DATABASE INFO!

#connect and store as conn
 conn = psycopg2.connect(
    host="ista322finalproject.ccbfbxwein0q.us-east-1.rds.amazonaws.com",
    database="rl_ista322final",
    user="rl_ista322final",
    password="rleviton",
    port='5432'
    )

 cur = conn.cursor()
 return(conn, cur)


# create databse connection and cursor
conn, cur = get_conn_cur()


# show that it is a connection object with my credentials
conn
```

```
<connection object at 0x7e42a8285440; dsn: 'user=rl_ista322final password=xxx dbname=rl_ista322final
host=ista322finalproject.ccbfbxwein0q.us-east-1.rds.amazonaws.com port=5432', closed: 0>
```

```
# functions in case we want to check our data

# run_query function
def run_query(query_string):

 conn, cur = get_conn_cur() # get connection and cursor

 cur.execute(query_string) # executing string as before

 my_data = cur.fetchall() # fetch query data as before

 # here we're extracting the 0th element for each item in cur.description
 colnames = [desc[0] for desc in cur.description]

 cur.close() # close
 conn.close() # close

 return(colnames, my_data) # return column names AND data

# Column name function for checking out what's in a table
def get_column_names(table_name): # arguement of table_name
 conn, cur = get_conn_cur() # get connection and cursor

 # Now select column names while inserting the table name into the WERE
 column_name_query = """SELECT column_name FROM information_schema.columns
    WHERE table_name = '%s' """ %table_name

 cur.execute(column_name_query) # exectue
 my_data = cur.fetchall() # store

 cur.close() # close
 conn.close() # close

 return(my_data) # return

# Check table_names
def get_table_names():
  conn, cur = get_conn_cur() # get connection and cursor

  # query to get table names
  table_name_query = """SELECT table_name FROM information_schema.tables
       WHERE table_schema = 'public' """

  cur.execute(table_name_query) # execute
  my_data = cur.fetchall() # fetch results

  cur.close() #close cursor
  conn.close() # close connection

  return(my_data) # return your fetched results

# make sql_head function
def sql_head(table_name):
 conn, cur = get_conn_cur() # get connection and cursor

 # Now select column names while inserting the table name into the WERE
 head_query = """SELECT * FROM %s LIMIT 5; """ %table_name

 cur.execute(head_query) # exectue
 colnames = [desc[0] for desc in cur.description] # get column names
 my_data = cur.fetchall() # store first five rows

 cur.close() # close
 conn.close() # close

 df = pd.DataFrame(data = my_data, columns = colnames) # make into df

 return(df) # return

# drop a table from your rdb (if you try to create a table that already exists, it'll throw an error)
def my_drop_table(table_name):
  conn, cur = get_conn_cur()
  tq = """DROP TABLE IF EXISTS %s CASCADE;""" %table_name
  cur.execute(tq)
  conn.commit()
```

```
# I want to create and push tables to AWS.. So I am going to split up the merged df, nba_records_vs_crimed, into three smaller t
# first lets inspect the full df
nba_records_vs_crime
```

	Team	Rk	Overall	Home	Road	Year	State	Population	Total_Property_Crimes	Property_Burglary	Propert_Larceny	Pr
0	Chicago Bulls	1	62-20	36-5	26-15	2011	Illinois	12,859,752	344,468	77,719	237,362	
1	San Antonio Spurs	2	61-21	36-5	25-16	2011	Texas	25,631,778	892,810	215,755	613,131	
2	Dallas Mavericks	4	57-25	29-12	28-13	2011	Texas	25,631,778	892,810	215,755	613,131	
3	Houston Rockets	14	43-39	25-16	18-23	2011	Texas	25,631,778	892,810	215,755	613,131	
4	Miami Heat	3	58-24	30-11	28-13	2011	Florida	19,082,262	671,200	170,171	461,408	
...	...	...	...	...	...	...	...	...	...	...	...	
238	New Orleans Pelicans	24	33-49	19-22	14-27	2019	Louisiana	4,648,794	146,993	26,918	109,359	
239	Atlanta Hawks	26	29-53	17-24	12-29	2019	Georgia	10,617,423	252,249	39,506	188,967	
240	Chicago Bulls	27	22-60	9-32	13-28	2019	Illinois	12,671,821	233,984	34,433	180,776	
241	Cleveland Cavaliers	28	19-63	13-28	6-35	2019	Ohio	11,689,100	240,291	43,894	177,725	
242	Phoenix Suns	29	19-63	12-29	7-34	2019	Arizona	7,278,717	177,638	28,699	130,788	

243 rows × 17 columns

```
# each table must have a primary key so I will create primary key called id that starts at 1
nba_records_vs_crime.insert(0, 'id', range(1, len(nba_records_vs_crime) + 1))
nba_records_vs_crime
```

	id	Team	Rk	Overall	Home	Road	Year	State	Population	Total_Property_Crimes	Property_Burglary	Propert_Larcen
0	1	Chicago Bulls	1	62-20	36-5	26-15	2011	Illinois	12,859,752	344,468	77,719	237,36
1	2	San Antonio Spurs	2	61-21	36-5	25-16	2011	Texas	25,631,778	892,810	215,755	613,13
2	3	Dallas Mavericks	4	57-25	29-12	28-13	2011	Texas	25,631,778	892,810	215,755	613,13
3	4	Houston Rockets	14	43-39	25-16	18-23	2011	Texas	25,631,778	892,810	215,755	613,13
4	5	Miami Heat	3	58-24	30-11	28-13	2011	Florida	19,082,262	671,200	170,171	461,40
...	...	...	...	...	...	...	...	...	...	...	...	.
238	239	New Orleans Pelicans	24	33-49	19-22	14-27	2019	Louisiana	4,648,794	146,993	26,918	109,35
239	240	Atlanta Hawks	26	29-53	17-24	12-29	2019	Georgia	10,617,423	252,249	39,506	188,96
240	241	Chicago Bulls	27	22-60	9-32	13-28	2019	Illinois	12,671,821	233,984	34,433	180,77
241	242	Cleveland Cavaliers	28	19-63	13-28	6-35	2019	Ohio	11,689,100	240,291	43,894	177,72
242	243	Phoenix Suns	29	19-63	12-29	7-34	2019	Arizona	7,278,717	177,638	28,699	130,78

243 rows × 18 columns

LOAD - TABLE 1: nba_standings

```
'''
I have everything needed in the merged df, so now it is time to start spliting the df up.
first I will create the nba_standings table, so I need to create a new df to model this table after. so I will select necessary
'''
nba_standings_df = nba_records_vs_crime[['id','State', 'Team', 'Rk', 'Overall', 'Home', 'Road', 'Year']]


print(nba_standings_df.head())
```

```
   id     State              Team  Rk Overall   Home   Road  Year
0   1  Illinois      Chicago Bulls   1   62-20   36-5  26-15  2011
1   2     Texas  San Antonio Spurs   2   61-21   36-5  25-16  2011
2   3     Texas   Dallas Mavericks   4   57-25  29-12  28-13  2011
3   4     Texas    Houston Rockets  14   43-39  25-16  18-23  2011
4   5   Florida         Miami Heat   3   58-24  30-11  28-13  2011
```

```
# if the table is already created, lets call our drop function
my_drop_table(table_name = 'nba_standings')



get_table_names()
```

```
[]
```

```
#now I will create the nba_standings table
conn, curr = get_conn_cur()
cur = conn.cursor()

q = """ CREATE TABLE nba_standings (
    id SERIAL PRIMARY KEY,
    State VARCHAR(20) NOT NULL,
    Team VARCHAR(40) NOT NULL,
    Rk INT NOT NUll,
    Overall VARCHAR(8) NOT NULL,
    Home VARCHAR(8) NOT NULL,
    Road VARCHAR(8) NOT NULL,
    Year INT NOT NULL) """

cur.execute(q)
conn.commit()


#We should now have one table created called nba_standings
get_table_names()
```

```
[('nba_standings',)]
```

```
#we can also the columns we just created
get_column_names(table_name = 'nba_standings')
```

```
[('id',),
 ('state',),
 ('team',),
 ('rk',),
 ('overall',),
 ('home',),
 ('road',),
 ('year',)]
```

```
#convery df into numpy array in order to insert values into table
data_tups = [tuple(x) for x in nba_standings_df.to_numpy()]


#insert into table query
iq =    '''
        INSERT INTO nba_standings(id, state, team, rk, overall, home, road, year) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)

        '''
```

```
'''
lets push this table with all the standings to our relational database. an analysist could use this data to find teams that have
'''

#execute and commit query
cur.executemany(iq, data_tups)
conn.commit()


sql_head(table_name = 'nba_standings')
```

	id	state	team	rk	overall	home	road	year
0	1	Illinois	Chicago Bulls	1	62-20	36-5	26-15	2011
1	2	Texas	San Antonio Spurs	2	61-21	36-5	25-16	2011
2	3	Texas	Dallas Mavericks	4	57-25	29-12	28-13	2011
3	4	Texas	Houston Rockets	14	43-39	25-16	18-23	2011
4	5	Florida	Miami Heat	3	58-24	30-11	28-13	2011

```
#simple query to extract the data about the Chicago Bulls records

q = '''
    SELECT year, team, overall
    FROM nba_standings
    WHERE team = 'Chicago Bulls'
    '''

run_query(q)

     (['year', 'team', 'overall'],
      [(2011, 'Chicago Bulls', '62-20'),
       (2012, 'Chicago Bulls', '50-16'),
       (2013, 'Chicago Bulls', '45-37'),
       (2014, 'Chicago Bulls', '48-34'),
       (2015, 'Chicago Bulls', '50-32'),
       (2016, 'Chicago Bulls', '42-40'),
       (2017, 'Chicago Bulls', '41-41'),
       (2018, 'Chicago Bulls', '27-55'),
       (2019, 'Chicago Bulls', '22-60')])
```

LOAD - TABLE 2: property_crimes

```
'''
now lets create the PROPERTY crimes table
'''

#take a look at the merged df to see which columns we want for our property crimes table
nba_records_vs_crime
```

	id	Team	Rk	Overall	Home	Road	Year	State	Population	Total_Property_Crimes	Property_Burglary	Propert_Larcen
0	1	Chicago Bulls	1	62-20	36-5	26-15	2011	Illinois	12,859,752	344,468	77,719	237,36
1	2	San Antonio Spurs	2	61-21	36-5	25-16	2011	Texas	25,631,778	892,810	215,755	613,13
2	3	Dallas Mavericks	4	57-25	29-12	28-13	2011	Texas	25,631,778	892,810	215,755	613,13

```
# selecting columns I want for the property crimes table, so selecting all property related columns
prop_crime_df = nba_records_vs_crime[['id', 'Year', 'State', 'Population', 'Total_Property_Crimes', 'Property_Burglary', 'Proper
prop_crime_df
```

	id	Year	State	Population	Total_Property_Crimes	Property_Burglary	Propert_Larceny	Property_Motor
0	1	2011	Illinois	12,859,752	344,468	77,719	237,362	29,387
1	2	2011	Texas	25,631,778	892,810	215,755	613,131	63,924
2	3	2011	Texas	25,631,778	892,810	215,755	613,131	63,924
3	4	2011	Texas	25,631,778	892,810	215,755	613,131	63,924
4	5	2011	Florida	19,082,262	671,200	170,171	461,408	39,621
...	...	...	...	...	...	...	...	...
238	239	2019	Louisiana	4,648,794	146,993	26,918	109,359	10,716
239	240	2019	Georgia	10,617,423	252,249	39,506	188,967	23,776
240	241	2019	Illinois	12,671,821	233,984	34,433	180,776	18,775
241	242	2019	Ohio	11,689,100	240,291	43,894	177,725	18,672
242	243	2019	Arizona	7,278,717	177,638	28,699	130,788	18,151

243 rows × 8 columns

```
# we can see that we have duplicated states for each year, but they all display the same information, so lets get rid of the sta
prop_crime_df = prop_crime_df.drop_duplicates(subset=['Year', 'State'])
prop_crime_df
```

	id	Year	State	Population	Total_Property_Crimes	Property_Burglary	Propert_Larceny	Property_Motor
0	1	2011	Illinois	12,859,752	344,468	77,719	237,362	29,387
1	2	2011	Texas	25,631,778	892,810	215,755	613,131	63,924
4	5	2011	Florida	19,082,262	671,200	170,171	461,408	39,621
6	7	2011	California	37,683,933	973,822	230,075	596,905	146,842
10	11	2011	Massachusetts	6,607,003	148,829	36,403	101,644	10,782
...	...	...	...	...	...	...	...	...
238	239	2019	Louisiana	4,648,794	146,993	26,918	109,359	10,716
239	240	2019	Georgia	10,617,423	252,249	39,506	188,967	23,776
240	241	2019	Illinois	12,671,821	233,984	34,433	180,776	18,775
241	242	2019	Ohio	11,689,100	240,291	43,894	177,725	18,672
242	243	2019	Arizona	7,278,717	177,638	28,699	130,788	18,151

182 rows × 8 columns

```
my_drop_table(table_name = 'property_crimes')
```

```
q = """ CREATE TABLE property_crimes (
    id SERIAL PRIMARY KEY,
    Year INT NOT NULL,
    State VARCHAR(20) NOT NULL,
    Population VARCHAR(20) NOT NULL,
    Total_Property_Crimes VARCHAR(20) NOT NULL,
    Property_Burglary VARCHAR(20) NOT NULL,
    Property_Larceny VARCHAR(20) NOT NULL,
    Property_Motor VARCHAR(20) NOT NULL
```

```
    ) """

cur.execute(q)
conn.commit()


#check to see if the new crimes table added
get_table_names()

    [('nba_standings',), ('property_crimes',)]


get_column_names(table_name = 'property_crimes')

    [('id',),
     ('year',),
     ('state',),
     ('population',),
     ('total_property_crimes',),
     ('property_burglary',),
     ('property_larceny',),
     ('property_motor',)]


data_tups = [tuple(x) for x in prop_crime_df.to_numpy()]


iq = """INSERT INTO property_crimes(id, year, state, population, total_property_crimes, property_burglary, property_larceny, prop
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"""


cur.executemany(iq, data_tups)
conn.commit()


sql_head(table_name = 'property_crimes')
```

	id	year	state	population	total_property_crimes	property_burglary	property_larceny	property_motor
0	1	2011	Illinois	12,859,752	344,468	77,719	237,362	29,387
1	2	2011	Texas	25,631,778	892,810	215,755	613,131	63,924
2	5	2011	Florida	19,082,262	671,200	170,171	461,408	39,621
3	7	2011	California	37,683,933	973,822	230,075	596,905	146,842
4	11	2011	Massachusetts	6,607,003	148,829	36,403	101,644	10,782

```
# simple test query to see the years where minnesota's property crimes were the highest

q = """ SELECT year, population, total_property_crimes
        FROM property_crimes
        WHERE state = 'Minnesota'
        ORDER BY total_property_crimes DESC
        LIMIT 3
    """

run_query(q)

    (['year', 'population', 'total_property_crimes'],
     [(2012, '5,379,139', '138,152'),
      (2011, '5,347,299', '136,183'),
      (2013, '5,420,380', '131,195')])
```

LOAD - TABLE 3: violent_crimes

```
# now creating violent_crimes table - lets see which columns we want to grab
nba_records_vs_crime.head()
```

	id	Team	Rk	Overall	Home	Road	Year	State	Population	Total_Property_Crimes	Property_Burglary	Propert_Larceny	Pr
0	1	Chicago Bulls	1	62-20	36-5	26-15	2011	Illinois	12,859,752	344,468	77,719	237,362	
1	2	San Antonio Spurs	2	61-21	36-5	25-16	2011	Texas	25,631,778	892,810	215,755	613,131	
2	3	Dallas Mavericks	4	57-25	29-12	28-13	2011	Texas	25,631,778	892,810	215,755	613,131	

```
# creating violent crimes df to model SQL table after
violent_crimes_df = nba_records_vs_crime[['id', 'Year', 'State', 'Population', 'Total_Violent_Crimes', 'Violent_Assault',   'Vic
violent_crimes_df
```

	id	Year	State	Population	Total_Violent_Crimes	Violent_Assault	Violent_Murder	Violent_Rape	Violent_Robbery
0	1	2011	Illinois	12,859,752	54,523	30,495	781	3,030	20,217
1	2	2011	Texas	25,631,778	104,734	67,498	1,130	7,486	28,620
2	3	2011	Texas	25,631,778	104,734	67,498	1,130	7,486	28,620
3	4	2011	Texas	25,631,778	104,734	67,498	1,130	7,486	28,620
4	5	2011	Florida	19,082,262	98,198	66,319	984	5,273	25,622
...	...	...	...	...	...	...	...	...	...
238	239	2019	Louisiana	4,648,794	25,537	18,695	544	2,273	4,025
239	240	2019	Georgia	10,617,423	36,170	24,633	654	2,922	7,961
240	241	2019	Illinois	12,671,821	51,561	32,187	832	6,078	12,464
241	242	2019	Ohio	11,689,100	34,269	19,154	538	5,731	8,846
242	243	2019	Arizona	7,278,717	33,141	22,704	365	3,662	6,410

243 rows × 9 columns

```
# we can see that we have texas three times once again in 2011, so need to drop these duplicate values again like we did in prop
violent_crimes_df = violent_crimes_df.drop_duplicates(subset=['Year', 'State'])
violent_crimes_df
```

	id	Year	State	Population	Total_Violent_Crimes	Violent_Assault	Violent_Murder	Violent_Rape	Violent_Robbery
0	1	2011	Illinois	12,859,752	54,523	30,495	781	3,030	20,217
1	2	2011	Texas	25,631,778	104,734	67,498	1,130	7,486	28,620
4	5	2011	Florida	19,082,262	98,198	66,319	984	5,273	25,622
6	7	2011	California	37,683,933	154,943	91,195	1,792	7,665	54,291
10	11	2011	Massachusetts	6,607,003	28,232	19,626	184	1,654	6,768
...	...	...	...	...	...	...	...	...	...
238	239	2019	Louisiana	4,648,794	25,537	18,695	544	2,273	4,025
239	240	2019	Georgia	10,617,423	36,170	24,633	654	2,922	7,961
240	241	2019	Illinois	12,671,821	51,561	32,187	832	6,078	12,464
241	242	2019	Ohio	11,689,100	34,269	19,154	538	5,731	8,846
242	243	2019	Arizona	7,278,717	33,141	22,704	365	3,662	6,410

182 rows × 9 columns

```
# drop table if it already exists
my_drop_table(table_name='violent_crimes')
```

```
# check to ensure table has not been created yet, but the other two have
get_table_names()
```

```
[('nba_standings',), ('property_crimes',)]
```

```
# create violent crimes tale
q = """ CREATE TABLE violent_crimes (
    id SERIAL PRIMARY KEY,
    Year INT NOT NULL,
    State VARCHAR(20) NOT NULL,
    Population VARCHAR(20) NOT NULL,
    Total_Violent_Crimes VARCHAR(20) NOT NULL,
    Violent_Assault VARCHAR(20) NOT NULL,
    Violent_Murder VARCHAR(20) NOT NULL,
    Violent_Rape VARCHAR(20) NOT NULL,
    Violent_Robbery VARCHAR(20) NOT NULL
    ) """

cur.execute(q)
conn.commit()


get_column_names(table_name = 'violent_crimes')
```

```
    [('id',),
     ('year',),
     ('state',),
     ('population',),
     ('total_violent_crimes',),
     ('violent_assault',),
     ('violent_murder',),
     ('violent_rape',),
     ('violent_robbery',)]
```

```
# create tuples of data
data_tups = [tuple(x) for x in violent_crimes_df.to_numpy()]


# insert data into violent crimes table
iq = """INSERT INTO violent_crimes(id, year, state, population, total_violent_crimes, violent_assault, violent_murder, violent_r
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)"""


# takes (query, tuples) - makes an insert string for every tuple and therefore inserts each row
cur.executemany(iq, data_tups)
conn.commit()


sql_head(table_name = 'violent_crimes')
```

	id	year	state	population	total_violent_crimes	violent_assault	violent_murder	violent_rape	violent_robbery
0	1	2011	Illinois	12,859,752	54,523	30,495	781	3,030	20,217
1	2	2011	Texas	25,631,778	104,734	67,498	1,130	7,486	28,620
2	5	2011	Florida	19,082,262	98,198	66,319	984	5,273	25,622
3	7	2011	California	37,683,933	154,943	91,195	1,792	7,665	54,291
4	11	2011	Massachusetts	6,607,003	28,232	19,626	184	1,654	6,768

```
conn.close()
```

QUERIES

```
'''
Selecting Total Property Crimes and Total Violent Crimes by State for each Year:
'''


q = """ SELECT pc.year, pc.state, pc.population, pc.total_property_crimes, vc.total_violent_crimes
        FROM property_crimes AS pc
        JOIN violent_crimes AS vc ON pc.Year = vc.Year AND pc.State = vc.State;
    """


run_query(q)
```

```
        (2011, 'Texas', '25,631,778', '892,810', '104,734'),
        (2011, 'Florida', '19,082,262', '671,200', '98,198'),
        (2011, 'California', '37,683,933', '973,822', '154,943'),
        (2011, 'Massachusetts', '6,607,003', '148,829', '28,232'),
        (2011, 'Oklahoma', '3,784,163', '127,618', '17,311'),
        (2011, 'Colorado', '5,116,302', '132,781', '16,085'),
        (2011, 'Oregon', '3,868,229', '121,869', '9,643'),
        (2011, 'Tennessee', '6,399,787', '230,900', '38,895'),
        (2011, 'Georgia', '9,812,460', '357,235', '36,762'),
        (2011, 'New York', '19,501,616', '371,837', '77,463'),
        (2011, 'Pennsylvania', '12,743,948', '283,442', '46,189'),
        (2011, 'Arizona', '6,467,315', '229,896', '26,789'),
        (2011, 'Utah', '2,814,347', '84,084', '5,547'),
        (2011, 'Indiana', '6,516,353', '206,016', '21,619'),
        (2011, 'Wisconsin', '5,709,843', '139,912', '14,268'),
        (2011, 'Michigan', '9,876,801', '251,329', '43,731'),
        (2011, 'Ohio', '11,541,007', '380,572', '35,218'),
        (2011, 'Minnesota', '5,347,299', '136,183', '12,323'),
        (2012, 'Illinois', '12,875,255', '332,013', '53,403'),
        (2012, 'Texas', '26,059,203', '876,059', '106,476'),
        (2012, 'Oklahoma', '3,814,820', '129,743', '17,902'),
        (2012, 'Florida', '19,317,568', '632,988', '94,087'),
        (2012, 'Indiana', '6,537,334', '198,032', '22,602'),
        (2012, 'California', '38,041,430', '1,049,465', '160,944'),
        (2012, 'Tennessee', '6,456,243', '217,664', '41,550'),
        (2012, 'Georgia', '9,919,945', '338,329', '37,591'),
        (2012, 'Massachusetts', '6,646,144', '143,089', '26,953'),
        (2012, 'Colorado', '5,187,582', '139,270', '16,023'),
        (2012, 'New York', '19,570,261', '376,140', '79,610'),
        (2012, 'Utah', '2,855,287', '85,424', '5,876'),
        (2012, 'Pennsylvania', '12,763,536', '276,496', '44,503'),
        (2012, 'Arizona', '6,553,255', '231,930', '28,108'),
        (2012, 'Wisconsin', '5,726,398', '140,513', '16,064'),
        (2012, 'Oregon', '3,899,353', '125,723', '9,653'),
        (2012, 'Minnesota', '5,379,139', '138,152', '12,419'),
        (2012, 'Michigan', '9,883,360', '250,101', '44,922'),
        (2012, 'Ohio', '11,544,225', '359,883', '34,595'),
        (2013, 'Florida', '19,552,860', '607,172', '91,986'),
        (2013, 'Oklahoma', '3,850,568', '126,057', '16,989'),
        (2013, 'Texas', '26,448,193', '861,734', '107,998'),
        (2013, 'Colorado', '5,268,367', '140,057', '16,226'),
        (2013, 'California', '38,332,521', '1,018,907', '154,129'),
        (2013, 'Tennessee', '6,495,978', '206,629', '38,364'),
        (2013, 'New York', '19,651,127', '358,598', '77,372'),
        (2013, 'Indiana', '6,570,902', '187,536', '23,487'),
        (2013, 'Illinois', '12,882,135', '292,983', '48,974'),
        (2013, 'Georgia', '9,992,167', '334,399', '36,541'),
        (2013, 'Utah', '2,900,872', '85,586', '6,498'),
        (2013, 'Massachusetts', '6,692,824', '137,285', '27,667'),
        (2013, 'Wisconsin', '5,742,713', '125,688', '15,961'),
        (2013, 'Pennsylvania', '12,773,801', '263,240', '42,849'),
        (2013, 'Oregon', '3,930,065', '124,737', '9,984'),
        (2013, 'Minnesota', '5,420,380', '131,195', '12,705'),
        (2013, 'Michigan', '9,895,622', '230,334', '44,523'),
        (2013, 'Arizona', '6,626,624', '225,243', '27,599'),
        (2013, 'Ohio', '11,570,808', '338,731', '33,121'),

#nba standings with property crimes by year and state

q = """ SELECT ns.State, ns.Team, ns.Overall, ns.Home, ns.Year, pc.Total_Property_Crimes
        FROM nba_standings AS ns
        JOIN property_crimes AS pc ON ns.Year = pc.Year AND ns.State = pc.State;"""

run_query(q)

    (['state', 'team', 'overall', 'home', 'year', 'total_property_crimes'],
     [('Illinois', 'Chicago Bulls', '62–20', '36–5', 2011, '344,468'),
      ('Texas', 'San Antonio Spurs', '61–21', '36–5', 2011, '892,810'),
      ('Texas', 'Dallas Mavericks', '57–25', '29–12', 2011, '892,810'),
      ('Texas', 'Houston Rockets', '43–39', '25–16', 2011, '892,810'),
      ('Florida', 'Miami Heat', '58–24', '30–11', 2011, '671,200'),
      ('Florida', 'Orlando Magic', '52–30', '29–12', 2011, '671,200'),
      ('California', 'Los Angeles Lakers', '57–25', '30–11', 2011, '973,822'),
      ('California', 'Golden State Warriors', '36–46', '26–15', 2011, '973,822'),
      ('California', 'Los Angeles Clippers', '32–50', '23–18', 2011, '973,822'),
      ('California', 'Sacramento Kings', '24–58', '11–30', 2011, '973,822'),
      ('Massachusetts', 'Boston Celtics', '56–26', '33–8', 2011, '148,829'),
      ('Oklahoma', 'Oklahoma City Thunder', '55–27', '30–11', 2011, '127,618'),
      ('Colorado', 'Denver Nuggets', '50–32', '33–8', 2011, '132,781'),
      ('Oregon', 'Portland Trail Blazers', '48–34', '30–11', 2011, '121,869'),
      ('Tennessee', 'Memphis Grizzlies', '46–36', '30–11', 2011, '230,900'),
      ('Georgia', 'Atlanta Hawks', '44–38', '24–17', 2011, '357,235'),
      ('New York', 'New York Knicks', '42–40', '23–18', 2011, '371,837'),
      ('Pennsylvania', 'Philadelphia 76ers', '41–41', '26–15', 2011, '283,442'),
```

```
('Arizona', 'Phoenix Suns', '40–42', '23–18', 2011, '229,896'),
('Utah', 'Utah Jazz', '39–43', '21–20', 2011, '84,084'),
('Indiana', 'Indiana Pacers', '37–45', '24–17', 2011, '206,016'),
('Wisconsin', 'Milwaukee Bucks', '35–47', '22–19', 2011, '139,912'),
('Michigan', 'Detroit Pistons', '30–52', '21–20', 2011, '251,329'),
('Ohio', 'Cleveland Cavaliers', '19–63', '12–29', 2011, '380,572'),
('Minnesota', 'Minnesota Timberwolves', '17–65', '12–29', 2011, '136,183'),
('Illinois', 'Chicago Bulls', '50–16', '26–7', 2012, '332,013'),
('Texas', 'San Antonio Spurs', '50–16', '28–5', 2012, '876,059'),
('Texas', 'Dallas Mavericks', '36–30', '23–10', 2012, '876,059'),
('Texas', 'Houston Rockets', '34–32', '22–11', 2012, '876,059'),
('Oklahoma', 'Oklahoma City Thunder', '47–19', '26–7', 2012, '129,743'),
('Florida', 'Miami Heat', '46–20', '28–5', 2012, '632,988'),
('Florida', 'Orlando Magic', '37–29', '21–12', 2012, '632,988'),
('Indiana', 'Indiana Pacers', '42–24', '23–10', 2012, '198,032'),
('California', 'Los Angeles Lakers', '41–25', '26–7', 2012, '1,049,465'),
('California', 'Los Angeles Clippers', '40–26', '24–9', 2012, '1,049,465'),
('California', 'Golden State Warriors', '23–43', '12–21', 2012, '1,049,465'),
('California', 'Sacramento Kings', '22–44', '16–17', 2012, '1,049,465'),
('Tennessee', 'Memphis Grizzlies', '41–25', '26–7', 2012, '217,664'),
('Georgia', 'Atlanta Hawks', '40–26', '23–10', 2012, '338,329'),
('Massachusetts', 'Boston Celtics', '39–27', '24–9', 2012, '143,089'),
('Colorado', 'Denver Nuggets', '38–28', '20–13', 2012, '139,270'),
('New York', 'New York Knicks', '36–30', '22–11', 2012, '376,140'),
('Utah', 'Utah Jazz', '36–30', '25–8', 2012, '85,424'),
('Pennsylvania', 'Philadelphia 76ers', '35–31', '19–14', 2012, '276,496'),
('Arizona', 'Phoenix Suns', '33–33', '19–14', 2012, '231,930'),
('Wisconsin', 'Milwaukee Bucks', '31–35', '17–16', 2012, '140,513'),
('Oregon', 'Portland Trail Blazers', '28–38', '20–13', 2012, '125,723'),
('Minnesota', 'Minnesota Timberwolves', '26–40', '13–20', 2012, '138,152'),
('Michigan', 'Detroit Pistons', '25–41', '18–15', 2012, '250,101'),
('Ohio', 'Cleveland Cavaliers', '21–45', '11–22', 2012, '359,883'),
```