In [1]:
```
#Codes Provided in the Class for Lab-4
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from unicodedata import normalize
```

In [2]:
```
import os, ssl
if (not os.environ.get('PYTHONHTTPSVERIFY', '') and getattr(ssl, '_create_unverifie
    ssl._create_default_https_context = ssl._create_unverified_context
```

In [3]:
```
table_MN = pd.read_html('https://en.wikipedia.org/wiki/Minnesota')
table_MN
```

Out[3]:
```
[                                            Minnesota  \
 0                                               State
 1                                   State of Minnesota
 2   .mw-parser-output .ib-settlement-cols{text-ali...
 3   Nickname(s): Land of 10,000 Lakes;North Star S...
 4   Motto(s): L'Étoile du Nord (French: The Star o...
 5                            Anthem: "Hail! Minnesota"
 6   Map of the United States with Minnesota highli...
 7                                             Country
 8                                    Before statehood
 9                               Admitted to the Union
 10                                            Capital
 11                                       Largest city
 12                   Largest metro and urban areas
 13                                         Government
 14                                         • Governor
 15                              • Lieutenant Governor
 16                                        Legislature
 17                                      • Upper house
```

In [4]:
```

print(f'Total tables: {len(table_MN)}')
```

```
Total tables: 29
```

In [5]:
```
table_MN[0]
```

Out[5]:

	Minnesota	Minnesota.1
0	State	State
1	State of Minnesota	State of Minnesota
2	.mw-parser-output .ib-settlement-cols{text-ali...	.mw-parser-output .ib-settlement-cols{text-ali...
3	Nickname(s): Land of 10,000 Lakes;North Star S...	Nickname(s): Land of 10,000 Lakes;North Star S...
4	Motto(s): L'Étoile du Nord (French: The Star o...	Motto(s): L'Étoile du Nord (French: The Star o...
5	Anthem: "Hail! Minnesota"	Anthem: "Hail! Minnesota"
6	Map of the United States with Minnesota highli...	Map of the United States with Minnesota highli...
7	Country	United States
8	Before statehood	Minnesota Territory
9	Admitted to the Union	May 11, 1858 (32nd State in the Union)
10	Capital	Saint Paul
11	Largest city	Minneapolis

	Minnesota	Minnesota.1
12	Largest metro and urban areas	Minneapolis–Saint Paul
13	Government	Government
14	• Governor	Tim Walz (DFL)
15	• Lieutenant Governor	Peggy Flanagan (DFL)
16	Legislature	Minnesota Legislature
17	• Upper house	Senate
18	• Lower house	House of Representatives
19	Judiciary	Minnesota Supreme Court
20	U.S. senators	Amy Klobuchar (DFL)Tina Smith (DFL)
21	U.S. House delegation	4 Democrats3 Republicans1 vacancy (list)
22	Area	Area
23	• Total	86,935.83 sq mi (225,163 km2)
24	• Land	79,626.74 sq mi (206,232 km2)
25	• Water	7,309.09 sq mi (18,930 km2) 8.40%
26	• Rank	12th
27	Dimensions	Dimensions
28	• Length	about 400 mi (640 km)
29	• Width	200–350 mi (320–560 km)
30	Elevation	1,200 ft (370 m)
31	Highest elevation (Eagle Mountain[1][2])	2,301 ft (701 m)
32	Lowest elevation (Lake Superior[1][2][3])	602 ft (183 m)
33	Population (2021)	Population (2021)
34	• Total	5,707,390[4]
35	• Rank	22nd
36	• Density	68.9/sq mi (26.6/km2)
37	• Rank	30th (2015 estimate)
38	• Median household income	$74,593[5]
39	• Income rank	13th
40	Demonym(s)	Minnesotan
41	Language	Language
42	• Official language	None
43	• Spoken language	English 88.9% Spanish Somali Hmong[6]
44	Time zone	UTC−06:00 (Central)
45	• Summer (DST)	UTC−05:00 (CDT)
46	USPS abbreviation	MN
47	ISO 3166 code	US-MN
48	Traditional abbreviation	Minn.
49	Latitude	43°30′ N to 49°23′ N
50	Longitude	89°29′ W to 97°14′ W

In [6]:
```
table_MN[13]
```

Out[6]:

	0	1
0	Parks	Voyageurs
1	Monuments	Grand Portage Pipestone
2	Rivers	Mississippi National River and Recreation Area...
3	Scenic Trails	North Country Trail
4	WildlifeRefuges	Agassiz Big Stone Crane Meadows Glacial Ridge ...
5	WetlandManagementDistricts	Big Stone Detroit Lakes Fergus Falls Litchfiel...
6	Forests	Chippewa Superior
7	NaturalLandmarks	Ancient River Warren Channel Cedar Creek Ecosy...
8	Wilderness	Agassiz Boundary Waters Canoe Area Tamarac

In [7]:
```
table_MN = pd.read_html('https://en.wikipedia.org/wiki/Minnesota', match='Election
```

In [8]:
```
len(table_MN)
```

Out[8]: 1

In [9]:
```
table_MN[0]
```

Out[9]:

	Year	Office	GOP	DFL	Others
0	2020	President	45.3%	52.4%	2.3%
1	2020	Senator	43.5%	48.8%	7.7%
2	2018	Governor	42.4%	53.9%	3.7%
3	2018	Senator	36.2%	60.3%	3.4%
4	2018	Senator	42.4%	53.0%	4.6%
5	2016	President	44.9%	46.4%	8.6%
6	2014	Governor	44.5%	50.1%	5.4%
7	2014	Senator	42.9%	53.2%	3.9%
8	2012	President	45.1%	52.8%	2.1%
9	2012	Senator	30.6%	65.3%	4.1%
10	2010	Governor	43.2%	43.7%	13.1%
11	2008	President	43.8%	54.1%	2.1%
12	2008	Senator	42.0%	42.0%	16.0%
13	2006	Governor	46.7%	45.7%	7.6%
14	2006	Senator	37.9%	58.1%	4.0%
15	2004	President	47.6%	51.1%	1.3%
16	2002	Governor	44.4%	33.5%	22.1%
17	2002	Senator	49.5%	47.3%	1.0%
18	2000	President	45.5%	47.9%	6.6%
19	2000	Senator	43.3%	48.8%	7.9%

	Year	Office	GOP	DFL	Others
20	1998	Governor	34.3%	28.1%	37.6%
21	1996	President	35.0%	51.1%	13.9%
22	1996	Senator	41.3%	50.3%	8.4%
23	1994	Governor	63.3%	34.1%	2.6%

In [10]:
```
df=table_MN[0]
```

Out[10]:

	Year	Office	GOP	DFL	Others
0	2020	President	45.3%	52.4%	2.3%
1	2020	Senator	43.5%	48.8%	7.7%
2	2018	Governor	42.4%	53.9%	3.7%
3	2018	Senator	36.2%	60.3%	3.4%
4	2018	Senator	42.4%	53.0%	4.6%
5	2016	President	44.9%	46.4%	8.6%
6	2014	Governor	44.5%	50.1%	5.4%
7	2014	Senator	42.9%	53.2%	3.9%
8	2012	President	45.1%	52.8%	2.1%
9	2012	Senator	30.6%	65.3%	4.1%
10	2010	Governor	43.2%	43.7%	13.1%
11	2008	President	43.8%	54.1%	2.1%
12	2008	Senator	42.0%	42.0%	16.0%
13	2006	Governor	46.7%	45.7%	7.6%
14	2006	Senator	37.9%	58.1%	4.0%
15	2004	President	47.6%	51.1%	1.3%
16	2002	Governor	44.4%	33.5%	22.1%
17	2002	Senator	49.5%	47.3%	1.0%
18	2000	President	45.5%	47.9%	6.6%
19	2000	Senator	43.3%	48.8%	7.9%
20	1998	Governor	34.3%	28.1%	37.6%
21	1996	President	35.0%	51.1%	13.9%
22	1996	Senator	41.3%	50.3%	8.4%
23	1994	Governor	63.3%	34.1%	2.6%
24	1994	Senator	49.1%	44.1%	6.8%
25	1992	President	31.9%	43.5%	24.6%

In [11]:

In [12]:
```
#changing value from integer to float, first replace the integer with nothing
df['GOP'].replace({'%':''}, regex=True).astype('float')
```

Out[12]:

```
0     45.3
1     43.5
2     42.4
3     36.2
4     42.4
5     44.9
6     44.5
7     42.9
8     45.1
9     30.6
10    43.2
11    43.8
12    42.0
13    46.7
14    37.9
15    47.6
16    44.4
17    49.5
18    45.5
19    43.3
20    34.3
21    35.0
22    41.3
23    63.3
```

In [13]:

Out[13]:

	Year	Office	GOP	DFL	Others
0	2020	President	45.3%	52.4%	2.3%
1	2020	Senator	43.5%	48.8%	7.7%
2	2018	Governor	42.4%	53.9%	3.7%
3	2018	Senator	36.2%	60.3%	3.4%
4	2018	Senator	42.4%	53.0%	4.6%
5	2016	President	44.9%	46.4%	8.6%
6	2014	Governor	44.5%	50.1%	5.4%
7	2014	Senator	42.9%	53.2%	3.9%
8	2012	President	45.1%	52.8%	2.1%
9	2012	Senator	30.6%	65.3%	4.1%
10	2010	Governor	43.2%	43.7%	13.1%
11	2008	President	43.8%	54.1%	2.1%
12	2008	Senator	42.0%	42.0%	16.0%
13	2006	Governor	46.7%	45.7%	7.6%
14	2006	Senator	37.9%	58.1%	4.0%
15	2004	President	47.6%	51.1%	1.3%
16	2002	Governor	44.4%	33.5%	22.1%
17	2002	Senator	49.5%	47.3%	1.0%
18	2000	President	45.5%	47.9%	6.6%
19	2000	Senator	43.3%	48.8%	7.9%
20	1998	Governor	34.3%	28.1%	37.6%
21	1996	President	35.0%	51.1%	13.9%

	Year	Office	GOP	DFL	Others
22	1996	Senator	41.3%	50.3%	8.4%
23	1994	Governor	63.3%	34.1%	2.6%
24	1994	Senator	49.1%	44.1%	6.8%

In [14]:
```
df['GOP']=df['GOP'].replace({'%':''}, regex=True)
```

In [15]:

Out[15]:

	Year	Office	GOP	DFL	Others
0	2020	President	45.3	52.4%	2.3%
1	2020	Senator	43.5	48.8%	7.7%
2	2018	Governor	42.4	53.9%	3.7%
3	2018	Senator	36.2	60.3%	3.4%
4	2018	Senator	42.4	53.0%	4.6%
5	2016	President	44.9	46.4%	8.6%
6	2014	Governor	44.5	50.1%	5.4%
7	2014	Senator	42.9	53.2%	3.9%
8	2012	President	45.1	52.8%	2.1%
9	2012	Senator	30.6	65.3%	4.1%
10	2010	Governor	43.2	43.7%	13.1%
11	2008	President	43.8	54.1%	2.1%
12	2008	Senator	42.0	42.0%	16.0%
13	2006	Governor	46.7	45.7%	7.6%
14	2006	Senator	37.9	58.1%	4.0%
15	2004	President	47.6	51.1%	1.3%
16	2002	Governor	44.4	33.5%	22.1%
17	2002	Senator	49.5	47.3%	1.0%
18	2000	President	45.5	47.9%	6.6%
19	2000	Senator	43.3	48.8%	7.9%
20	1998	Governor	34.3	28.1%	37.6%
21	1996	President	35.0	51.1%	13.9%
22	1996	Senator	41.3	50.3%	8.4%
23	1994	Governor	63.3	34.1%	2.6%
24	1994	Senator	49.1	44.1%	6.8%
25	1992	President	31.9	43.5%	24.6%

In [16]:
```
df['GOP']=df['GOP'].astype('float')
```

In [17]:

Out[17]:

	Year	Office	GOP	DFL	Others
0	2020	President	45.3	52.4%	2.3%

	Year	Office	GOP	DFL	Others
1	2020	Senator	43.5	48.8%	7.7%
2	2018	Governor	42.4	53.9%	3.7%
3	2018	Senator	36.2	60.3%	3.4%
4	2018	Senator	42.4	53.0%	4.6%
5	2016	President	44.9	46.4%	8.6%
6	2014	Governor	44.5	50.1%	5.4%
7	2014	Senator	42.9	53.2%	3.9%
8	2012	President	45.1	52.8%	2.1%
9	2012	Senator	30.6	65.3%	4.1%
10	2010	Governor	43.2	43.7%	13.1%
11	2008	President	43.8	54.1%	2.1%
12	2008	Senator	42.0	42.0%	16.0%
13	2006	Governor	46.7	45.7%	7.6%
14	2006	Senator	37.9	58.1%	4.0%
15	2004	President	47.6	51.1%	1.3%
16	2002	Governor	44.4	33.5%	22.1%
17	2002	Senator	49.5	47.3%	1.0%
18	2000	President	45.5	47.9%	6.6%
19	2000	Senator	43.3	48.8%	7.9%
20	1998	Governor	34.3	28.1%	37.6%
21	1996	President	35.0	51.1%	13.9%
22	1996	Senator	41.3	50.3%	8.4%
23	1994	Governor	63.3	34.1%	2.6%
24	1994	Senator	49.1	44.1%	6.8%

In [18]:
```
df=df.replace({'%':''},regex=True)
```

In [19]:
```
df.head()
```

Out[19]:

	Year	Office	GOP	DFL	Others
0	2020	President	45.3	52.4	2.3
1	2020	Senator	43.5	48.8	7.7
2	2018	Governor	42.4	53.9	3.7
3	2018	Senator	36.2	60.3	3.4
4	2018	Senator	42.4	53.0	4.6

In [20]:
```
df[['GOP','DFL','Others']]=df[['GOP','DFL','Others']].apply(pd.to_numeric)
```

In [21]:

Out[21]:

	Year	Office	GOP	DFL	Others
0	2020	President	45.3	52.4	2.3

	Year	Office	GOP	DFL	Others
1	2020	Senator	43.5	48.8	7.7
2	2018	Governor	42.4	53.9	3.7
3	2018	Senator	36.2	60.3	3.4
4	2018	Senator	42.4	53.0	4.6
5	2016	President	44.9	46.4	8.6
6	2014	Governor	44.5	50.1	5.4
7	2014	Senator	42.9	53.2	3.9
8	2012	President	45.1	52.8	2.1
9	2012	Senator	30.6	65.3	4.1
10	2010	Governor	43.2	43.7	13.1
11	2008	President	43.8	54.1	2.1
12	2008	Senator	42.0	42.0	16.0
13	2006	Governor	46.7	45.7	7.6
14	2006	Senator	37.9	58.1	4.0
15	2004	President	47.6	51.1	1.3
16	2002	Governor	44.4	33.5	22.1
17	2002	Senator	49.5	47.3	1.0
18	2000	President	45.5	47.9	6.6
19	2000	Senator	43.3	48.8	7.9
20	1998	Governor	34.3	28.1	37.6
21	1996	President	35.0	51.1	13.9
22	1996	Senator	41.3	50.3	8.4
23	1994	Governor	63.3	34.1	2.6
24	1994	Senator	49.1	44.1	6.8

In [22]:
```
table_GDP = pd.read_html('https://en.wikipedia.org/wiki/Economy_of_the_United_State
```

In [23]:
```
df_GDP=table_GDP[3]
df_GDP
```

Out[23]:

	Year	GDP (in Bil. US$PPP)	GDP per capita (in US$ PPP)	GDP (in Bil. US$nominal)	GDP per capita (in US$ nominal)	GDP growth (real)	Inflation rate (in Percent)	Unemployment (in Percent)	Government debt (in % of GDP)
0	1980	2857.3	12552.9	2857.3	12552.9	-0.3%	13.5%	7.2%	NaN
1	1981	3207.0	13948.7	3207.0	13948.7	2.5%	10.4%	7.6%	NaN
2	1982	3343.8	14405.0	3343.8	14405.0	-1.8%	6.2%	9.7%	NaN
3	1983	3634.0	15513.7	3634.0	15513.7	4.6%	3.2%	9.6%	NaN
4	1984	4037.7	17086.4	4037.7	17086.4	7.2%	4.4%	7.5%	NaN
5	1985	4339.0	18199.3	4339.0	18199.3	4.2%	3.5%	7.2%	NaN
6	1986	4579.6	19034.8	4579.6	19034.8	3.5%	1.9%	7.0%	NaN
7	1987	4855.3	20001.0	4855.3	20001.0	3.5%	3.6%	6.2%	NaN
8	1988	5236.4	21376.0	5236.4	21376.0	4.2%	4.1%	5.5%	NaN

	Year	GDP (in Bil. US$PPP)	GDP per capita (in US$ PPP)	GDP (in Bil. US$nominal)	GDP per capita (in US$ nominal)	GDP growth (real)	Inflation rate (in Percent)	Unemployment (in Percent)	Government debt (in % of GDP)
9	1989	5641.6	22814.1	5641.6	22814.1	3.7%	4.8%	5.3%	NaN
10	1990	5963.1	23848.0	5963.1	23848.0	1.9%	5.4%	5.6%	NaN
11	1991	6158.1	24302.8	6158.1	24302.8	-0.1%	4.2%	6.9%	NaN
12	1992	6520.3	25392.9	6520.3	25392.9	3.5%	3.0%	7.5%	NaN
13	1993	6858.6	26364.2	6858.6	26364.2	2.8%	3.0%	6.9%	NaN
14	1994	7287.3	27674.0	7287.3	27674.0	4.0%	2.6%	6.1%	NaN
15	1995	7639.8	28671.5	7639.8	28671.5	2.7%	2.8%	5.6%	NaN
16	1996	8073.1	29947.0	8073.1	29947.0	3.8%	2.9%	5.4%	NaN
17	1997	8577.6	31440.1	8577.6	31440.1	4.4%	2.3%	4.9%	NaN
18	1998	9062.8	32833.7	9062.8	32833.7	4.5%	1.5%	4.5%	NaN
19	1999	9631.2	34496.2	9631.2	34496.2	4.8%	2.2%	4.2%	NaN
20	2000	10251.0	36312.8	10251.0	36312.8	4.1%	3.4%	4.0%	NaN
21	2001	10581.9	37101.5	10581.9	37101.5	1.0%	2.8%	4.7%	53.1%
22	2002	10929.1	37945.8	10929.1	37945.8	1.7%	1.6%	5.8%	55.5%
23	2003	11456.5	39405.4	11456.5	39405.4	2.8%	2.3%	6.0%	58.6%
24	2004	12217.2	41641.6	12217.2	41641.6	3.9%	2.7%	5.5%	66.1%
25	2005	13039.2	44034.3	13039.2	44034.3	3.5%	3.4%	5.1%	65.5%
26	2006	13815.6	46216.9	13815.6	46216.9	2.8%	3.2%	4.6%	64.2%
27	2007	14474.3	47943.4	14474.3	47943.4	2.0%	2.9%	4.6%	64.6%
28	2008	14769.9	48470.6	14769.9	48470.6	0.1%	3.8%	5.8%	73.4%
29	2009	14478.1	47102.4	14478.1	47102.4	-2.6%	-0.3%	9.3%	86.6%
30	2010	15049.0	48586.3	15049.0	48586.3	2.7%	1.6%	9.6%	95.1%
31	2011	15599.7	50008.1	15599.7	50008.1	1.6%	3.1%	8.9%	99.5%
32	2012	16254.0	51736.7	16254.0	51736.7	2.3%	2.1%	8.1%	103.0%
33	2013	16843.2	53245.5	16843.2	53245.5	1.8%	1.5%	7.4%	104.5%
34	2014	17550.7	55083.5	17550.7	55083.5	2.3%	1.6%	6.2%	104.5%
35	2015	18206.0	56729.7	18206.0	56729.7	2.7%	0.1%	5.3%	104.9%
36	2016	18695.1	57840.0	18695.1	57840.0	1.7%	1.3%	4.9%	106.9%
37	2017	19479.6	59885.7	19479.6	59885.7	2.3%	2.1%	4.4%	106.0%
38	2018	20527.2	62769.7	20527.2	62769.7	2.9%	2.4%	3.9%	107.1%
39	2019	21372.6	65051.9	21372.6	65051.9	2.3%	1.8%	3.7%	108.5%
40	2020	20893.8	63358.5	20893.8	63358.5	-3.4%	1.2%	8.1%	133.9%
41	2021	22939.6	69375.4	22939.6	69375.4	6.0%	4.3%	5.4%	133.3%
42	2022	24796.1	74725.0	24796.1	74725.0	5.2%	3.5%	3.5%	130.7%
43	2023	25938.2	77881.3	25938.2	77881.3	2.2%	2.7%	3.0%	131.1%
44	2024	26980.4	80714.8	26980.4	80714.8	1.7%	2.6%	3.0%	131.7%

In [24]:
```
df_GDP.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47 entries, 0 to 46
Data columns (total 9 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Year                            47 non-null     int64
 1   GDP (in Bil. US$PPP)            47 non-null     float64
 2   GDP per capita (in US$ PPP)     47 non-null     float64
 3   GDP (in Bil. US$nominal)        47 non-null     float64
 4   GDP per capita (in US$ nominal) 47 non-null     float64
 5   GDP growth (real)               47 non-null     object
 6   Inflation rate (in Percent)     47 non-null     object
 7   Unemployment (in Percent)       47 non-null     object
 8   Government debt (in % of GDP)   26 non-null     object
dtypes: float64(4), int64(1), object(4)
```

In [25]:
```
df_GDP['GDP growth (real)'].replace({'%': ''}, regex=True).astype('float')
```

Out[25]:

```
0    -0.3
1     2.5
2    -1.8
```

In [26]:
```
#strip removes extra spaces
from unicodedata import normalize
def clean_normalize_whitespace(x):
    if isinstance(x, str):
        return normalize('NFKC', x).strip()
    else:
        return x
```

In [27]:
```
df_GDP = df_GDP.applymap(clean_normalize_whitespace)
```

In [28]:
```
df_GDP
```

Out[28]:

	Year	GDP (in Bil. US$PPP)	GDP per capita (in US$ PPP)	GDP (in Bil. US$nominal)	GDP per capita (in US$ nominal)	GDP growth (real)	Inflation rate (in Percent)	Unemployment (in Percent)	Government debt (in % of GDP)
0	1980	2857.3	12552.9	2857.3	12552.9	-0.3%	13.5%	7.2%	NaN
1	1981	3207.0	13948.7	3207.0	13948.7	2.5%	10.4%	7.6%	NaN
2	1982	3343.8	14405.0	3343.8	14405.0	-1.8%	6.2%	9.7%	NaN
3	1983	3634.0	15513.7	3634.0	15513.7	4.6%	3.2%	9.6%	NaN
4	1984	4037.7	17086.4	4037.7	17086.4	7.2%	4.4%	7.5%	NaN
5	1985	4339.0	18199.3	4339.0	18199.3	4.2%	3.5%	7.2%	NaN
6	1986	4579.6	19034.8	4579.6	19034.8	3.5%	1.9%	7.0%	NaN
7	1987	4855.3	20001.0	4855.3	20001.0	3.5%	3.6%	6.2%	NaN
8	1988	5236.4	21376.0	5236.4	21376.0	4.2%	4.1%	5.5%	NaN
9	1989	5641.6	22814.1	5641.6	22814.1	3.7%	4.8%	5.3%	NaN
10	1990	5963.1	23848.0	5963.1	23848.0	1.9%	5.4%	5.6%	NaN
11	1991	6158.1	24302.8	6158.1	24302.8	-0.1%	4.2%	6.9%	NaN
12	1992	6520.3	25392.9	6520.3	25392.9	3.5%	3.0%	7.5%	NaN
13	1993	6858.6	26364.2	6858.6	26364.2	2.8%	3.0%	6.9%	NaN
14	1994	7287.3	27674.0	7287.3	27674.0	4.0%	2.6%	6.1%	NaN
15	1995	7639.8	28671.5	7639.8	28671.5	2.7%	2.8%	5.6%	NaN
16	1996	8073.1	29947.0	8073.1	29947.0	3.8%	2.9%	5.4%	NaN
17	1997	8577.6	31440.1	8577.6	31440.1	4.4%	2.3%	4.9%	NaN
18	1998	9062.8	32833.7	9062.8	32833.7	4.5%	1.5%	4.5%	NaN
19	1999	9631.2	34496.2	9631.2	34496.2	4.8%	2.2%	4.2%	NaN
20	2000	10251.0	36312.8	10251.0	36312.8	4.1%	3.4%	4.0%	NaN
21	2001	10581.9	37101.5	10581.9	37101.5	1.0%	2.8%	4.7%	53.1%
22	2002	10929.1	37945.8	10929.1	37945.8	1.7%	1.6%	5.8%	55.5%
23	2003	11456.5	39405.4	11456.5	39405.4	2.8%	2.3%	6.0%	58.6%
24	2004	12217.2	41641.6	12217.2	41641.6	3.9%	2.7%	5.5%	66.1%
25	2005	13039.2	44034.3	13039.2	44034.3	3.5%	3.4%	5.1%	65.5%
26	2006	13815.6	46216.9	13815.6	46216.9	2.8%	3.2%	4.6%	64.2%

	Year	GDP (in Bil. US$PPP)	GDP per capita (in US$ PPP)	GDP (in Bil. US$nominal)	GDP per capita (in US$ nominal)	GDP growth (real)	Inflation rate (in Percent)	Unemployment (in Percent)	Government debt (in % of GDP)
27	2007	14474.3	47943.4	14474.3	47943.4	2.0%	2.9%	4.6%	64.6%
28	2008	14769.9	48470.6	14769.9	48470.6	0.1%	3.8%	5.8%	73.4%
29	2009	14478.1	47102.4	14478.1	47102.4	-2.6%	-0.3%	9.3%	86.6%
30	2010	15049.0	48586.3	15049.0	48586.3	2.7%	1.6%	9.6%	95.1%
31	2011	15599.7	50008.1	15599.7	50008.1	1.6%	3.1%	8.9%	99.5%
32	2012	16254.0	51736.7	16254.0	51736.7	2.3%	2.1%	8.1%	103.0%
33	2013	16843.2	53245.5	16843.2	53245.5	1.8%	1.5%	7.4%	104.5%
34	2014	17550.7	55083.5	17550.7	55083.5	2.3%	1.6%	6.2%	104.5%
35	2015	18206.0	56729.7	18206.0	56729.7	2.7%	0.1%	5.3%	104.9%
36	2016	18695.1	57840.0	18695.1	57840.0	1.7%	1.3%	4.9%	106.9%
37	2017	19479.6	59885.7	19479.6	59885.7	2.3%	2.1%	4.4%	106.0%
38	2018	20527.2	62769.7	20527.2	62769.7	2.9%	2.4%	3.9%	107.1%
39	2019	21372.6	65051.9	21372.6	65051.9	2.3%	1.8%	3.7%	108.5%
40	2020	20893.8	63358.5	20893.8	63358.5	-3.4%	1.2%	8.1%	133.9%
41	2021	22939.6	69375.4	22939.6	69375.4	6.0%	4.3%	5.4%	133.3%
42	2022	24796.1	74725.0	24796.1	74725.0	5.2%	3.5%	3.5%	130.7%
43	2023	25938.2	77881.3	25938.2	77881.3	2.2%	2.7%	3.0%	131.1%

In [29]:
```
df_GDP.columns[8]
```
Out[29]: 'Government debt (in\xa0% of GDP)'

In [30]:
```
df_GDP.columns = df_GDP.columns.to_series().apply(clean_normalize_whitespace)
df_GDP.columns[8]
```
Out[30]: 'Government debt (in % of GDP)'

In [31]:

Out[31]: 'Government debt (in % of GDP)'

In [32]:
```
df_GDP['GDP growth (real)'].replace({'%': ''}, regex=True).astype('float')
```
Out[32]:

```
0     -0.3
1      2.5
2     -1.8
3      4.6
4      7.2
5      4.2
6      3.5
7      3.5
8      4.2
9      3.7
10     1.9
11    -0.1
12     3.5
13     2.8
14     4.0
15     2.7
16     3.8
17     4.4
18     4.5
19     4.8
20     4.1
21     1.0
22     1.7
23     2.8
24     3.9
25     3.5
26     2.8
27     2.0
28     0.1
29    -2.6
30     2.7
31     1.6
```

In [33]:
```
df_GDP['GDP growth (real)']=df_GDP['GDP growth (real)'].replace({'%': ''}, regex=Tr
```

In [34]:

Out[34]:

```
0     -0.3
1      2.5
2     -1.8
3      4.6
4      7.2
5      4.2
6      3.5
7      3.5
8      4.2
9      3.7
10     1.9
11    -0.1
12     3.5
13     2.8
14     4.0
15     2.7
16     3.8
17     4.4
18     4.5
19     4.8
20     4.1
21     1.0
22     1.7
```

In [35]:
```
df_GDP['GDP growth (real)'].replace({'%':'','-':'-'}, regex=True).astype('float')
```

Out[35]:

```
0     -0.3
1      2.5
2     -1.8
3      4.6
4      7.2
5      4.2
6      3.5
7      3.5
8      4.2
9      3.7
10     1.9
```

In [36]:
```
#if your data have any character
df['Year'].replace({'%': '', '-': '-', '\(est\)': ''}, regex=True).astype('int')
```

Out[36]:
```
0      2020
1      2020
2      2018
3      2018
4      2018
5      2016
6      2014
7      2014
8      2012
9      2012
10     2010
11     2008
12     2008
13     2006
14     2006
15     2004
16     2002
17     2002
18     2000
19     2000
20     1998
21     1996
22     1996
23     1994
24     1994
25     1992
Name: Year, dtype: int32
```

In [37]:
```
#change the columns according to the recent one(dict.fromkeys(df_GDP.columns, 'floa
col_type = {
'Year': 'int',
 'GDP (in Bil. US$PPP)': 'float',
 'GDP per capita (in US$ PPP)': 'float',
 'GDP (in Bil. US$nominal)': 'float',
 'GDP per capita (in US$ nominal)': 'float',
 'GDP growth (real)': 'float',
 'Inflation rate (in Percent)': 'float',
 'Unemployment (in Percent)': 'float',
 'Government debt (in % of GDP)': 'float'
}
```

In [38]:
```
dict.fromkeys(df_GDP.columns, 'float')
```

Out[38]:

```
{'Year': 'float',
 'GDP (in Bil. US$PPP)': 'float',
 'GDP per capita (in US$ PPP)': 'float',
 'GDP (in Bil. US$nominal)': 'float',
 'GDP per capita (in US$ nominal)': 'float'
```

In [39]:
```
clean_dict = {'%': '', '-': '-', '\(est\)': ''}
```

In [40]:
```

df_GDP.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47 entries, 0 to 46
Data columns (total 9 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Year                             47 non-null     int64
 1   GDP (in Bil. US$PPP)             47 non-null     float64
 2   GDP per capita (in US$ PPP)      47 non-null     float64
 3   GDP (in Bil. US$nominal)         47 non-null     float64
 4   GDP per capita (in US$ nominal)  47 non-null     float64
 5   GDP growth (real)                47 non-null     object
 6   Inflation rate (in Percent)      47 non-null     object
 7   Unemployment (in Percent)        47 non-null     object
 8   Government debt (in % of GDP)    26 non-null     object
dtypes: float64(4), int64(1), object(4)
memory usage: 3.4+ KB
```

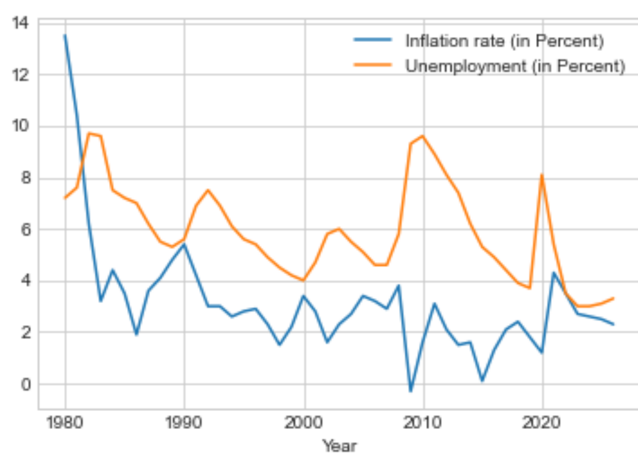
In [41]:
```
df_GDP = df_GDP.replace(clean_dict, regex=True).replace({ '-n/a ': np.nan}).astype(
```

In [42]:
```
df_GDP.columns
```

Out[42]:
```
Index(['Year', 'GDP (in Bil. US$PPP)', 'GDP per capita (in US$ PPP)',
       'GDP (in Bil. US$nominal)', 'GDP per capita (in US$ nominal)',
       'GDP growth (real)', 'Inflation rate (in Percent)',
       'Unemployment (in Percent)', 'Government debt (in % of GDP)'],
      dtype='object')
```

In [43]:
```
plt.style.use('seaborn-whitegrid')
df_GDP.plot.line(x='Year', y=['Inflation rate (in Percent)', 'Unemployment (in Perc
```

Out[43]:
```
<AxesSubplot:xlabel='Year'>
```



In [44]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [45]:
```
mtcars = pd.read_csv('mtcars_analysis.csv')
```

In [46]:
```
mtcars
```

Out[46]:

	Unnamed: 0	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
5	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
6	Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
7	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
8	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
9	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
10	Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
11	Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
12	Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
13	Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
14	Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
15	Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
16	Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
17	Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
18	Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
19	Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
20	Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
21	Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
22	AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
23	Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
24	Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
25	Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
26	Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
27	Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
28	Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
29	Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
30	Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	8
31	Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2

In [47]:
```
mtcars=mtcars.rename(columns={'Unnamed: 0':'model'})
```

In [48]:
```
mtcars
```

Out[48]:

	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
5	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
6	Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
7	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
8	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
9	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
10	Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
11	Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
12	Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
13	Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
14	Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
15	Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
16	Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
17	Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
18	Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
19	Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
20	Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
21	Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
22	AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
23	Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
24	Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
25	Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
26	Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
27	Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
28	Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
29	Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
30	Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	8
31	Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2

In [49]:
```
mtcars.index=mtcars.model
```

In [50]:
```
mtcars
```

Out[50]:

	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
model												
Mazda RX4	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4

	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
model												
Datsun 710	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Dodge Challenger	Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
AMC Javelin	AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
Camaro Z28	Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
Pontiac Firebird	Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
Fiat X1-9	Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Lotus Europa	Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
Ford Pantera L	Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
Ferrari Dino	Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6

In [51]:
```
del mtcars['model']
```

In [52]:
```
mtcars
```

Out[52]:

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
model											
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
model											
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6

In [53]:
```
mtcars.mean()
```

Out[53]:
```
mpg      20.090625
cyl       6.187500
disp    230.721875
hp      146.687500
drat      3.596563
wt        3.217250
qsec     17.848750
vs        0.437500
am        0.406250
gear      3.687500
carb      2.812500
dtype: float64
```

In [54]:
```
mtcars.mean(axis=0)
```

Out[54]:

```
mpg      20.090625
cyl       6.187500
disp    230.721875
hp      146.687500
drat      3.596563
wt        3.217250
qsec     17.848750
vs        0.437500
```

In [55]:
```
mtcars.mean(axis=1)
```

Out[55]:
```
model
Mazda RX4              29.907273
Mazda RX4 Wag          29.981364
Datsun 710             23.598182
Hornet 4 Drive         38.739545
Hornet Sportabout      53.664545
Valiant                35.049091
Duster 360             59.720000
Merc 240D              24.634545
Merc 230               27.233636
Merc 280               31.860000
Merc 280C              31.787273
Merc 450SE             46.430909
Merc 450SL             46.500000
Merc 450SLC            46.350000
Cadillac Fleetwood     66.232727
Lincoln Continental    66.058545
Chrysler Imperial      65.972273
Fiat 128               19.440909
Honda Civic            17.742273
Toyota Corolla         18.814091
Toyota Corona          24.888636
Dodge Challenger       47.240909
AMC Javelin            46.007727
Camaro Z28             58.752727
Pontiac Firebird       57.379545
Fiat X1-9              18.928636
Porsche 914-2          24.779091
Lotus Europa           24.880273
Ford Pantera L         60.971818
Ferrari Dino           34.508182
Maserati Bora          63.155455
Volvo 142E             26.262727
dtype: float64
```

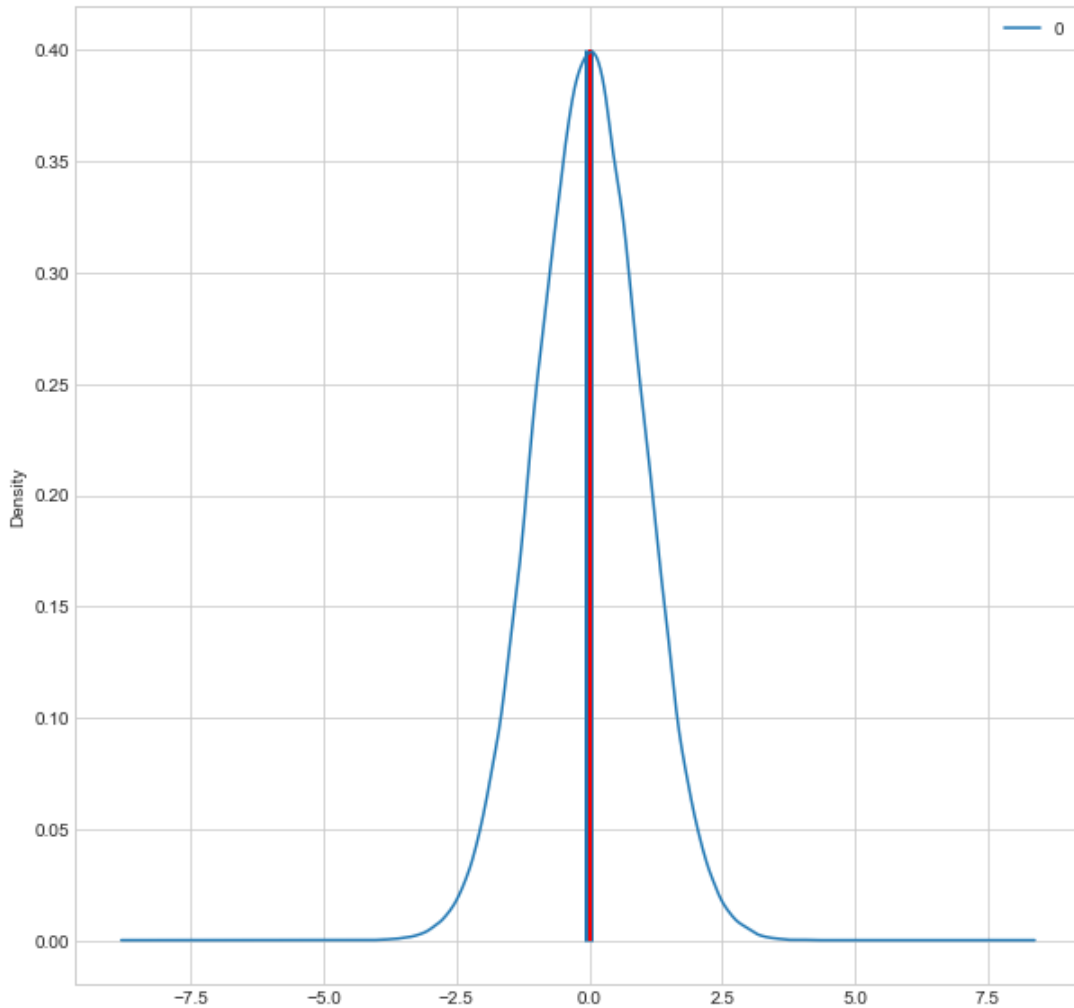
In [56]:
```
mtcars.median()
```

Out[56]:
```
mpg      19.200
cyl       6.000
disp    196.300
hp      123.000
drat      3.695
wt        3.325
qsec     17.710
vs        0.000
am        0.000
gear      4.000
carb      2.000
dtype: float64
```

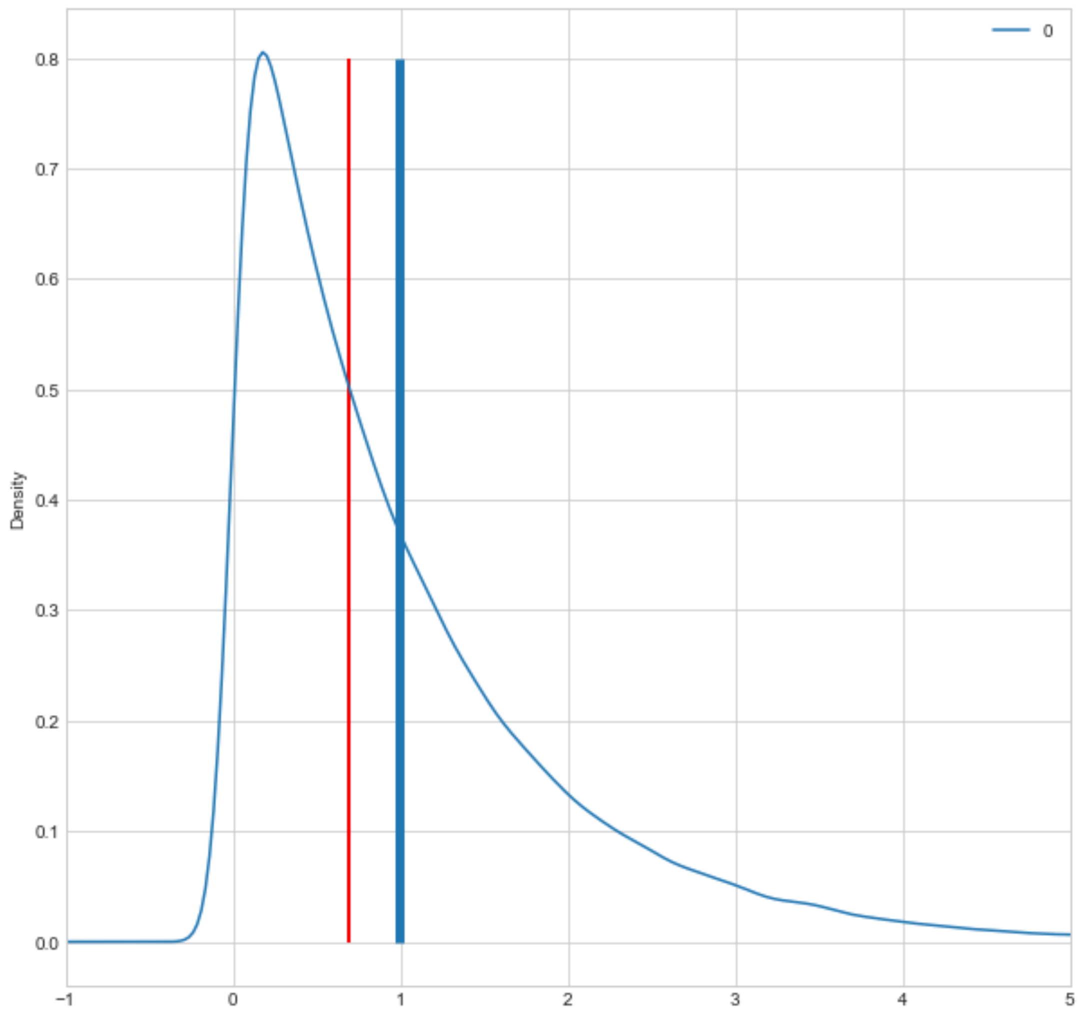
In [57]:
```
mtcars.median(axis=1)
```

Out[57]:
```
model
Mazda RX4              4.000
Mazda RX4 Wag          4.000
Datsun 710             4.000
Hornet 4 Drive         3.215
Hornet Sportabout      3.440
Valiant                3.460
Duster 360             4.000
Merc 240D              4.000
Merc 230               4.000
Merc 280               4.000
Merc 280C              4.000
Merc 450SE             4.070
Merc 450SL             3.730
Merc 450SLC            3.780
Cadillac Fleetwood     5.250
Lincoln Continental    5.424
Chrysler Imperial      5.345
Fiat 128               4.000
Honda Civic            4.000
Toyota Corolla         4.000
Toyota Corona          3.700
Dodge Challenger       3.520
AMC Javelin            3.435
Camaro Z28             4.000
Pontiac Firebird       3.845
Fiat X1-9              4.000
Porsche 914-2          4.430
Lotus Europa           4.000
Ford Pantera L         5.000
Ferrari Dino           6.000
Maserati Bora          8.000
Volvo 142E             4.000
dtype: float64
```

In [58]:
```
norm_data = pd.DataFrame(np.random.normal(size=100000))
norm_data.plot(kind="density",
 figsize=(10,10));
plt.vlines(norm_data.mean(), # Plot black line at mean
 ymin=0,
 ymax=0.4,
 linewidth=5.0);
plt.vlines(norm_data.median(), # Plot red line at median
 ymin=0,
 ymax=0.4,
 linewidth=2.0,
 color="red");
```

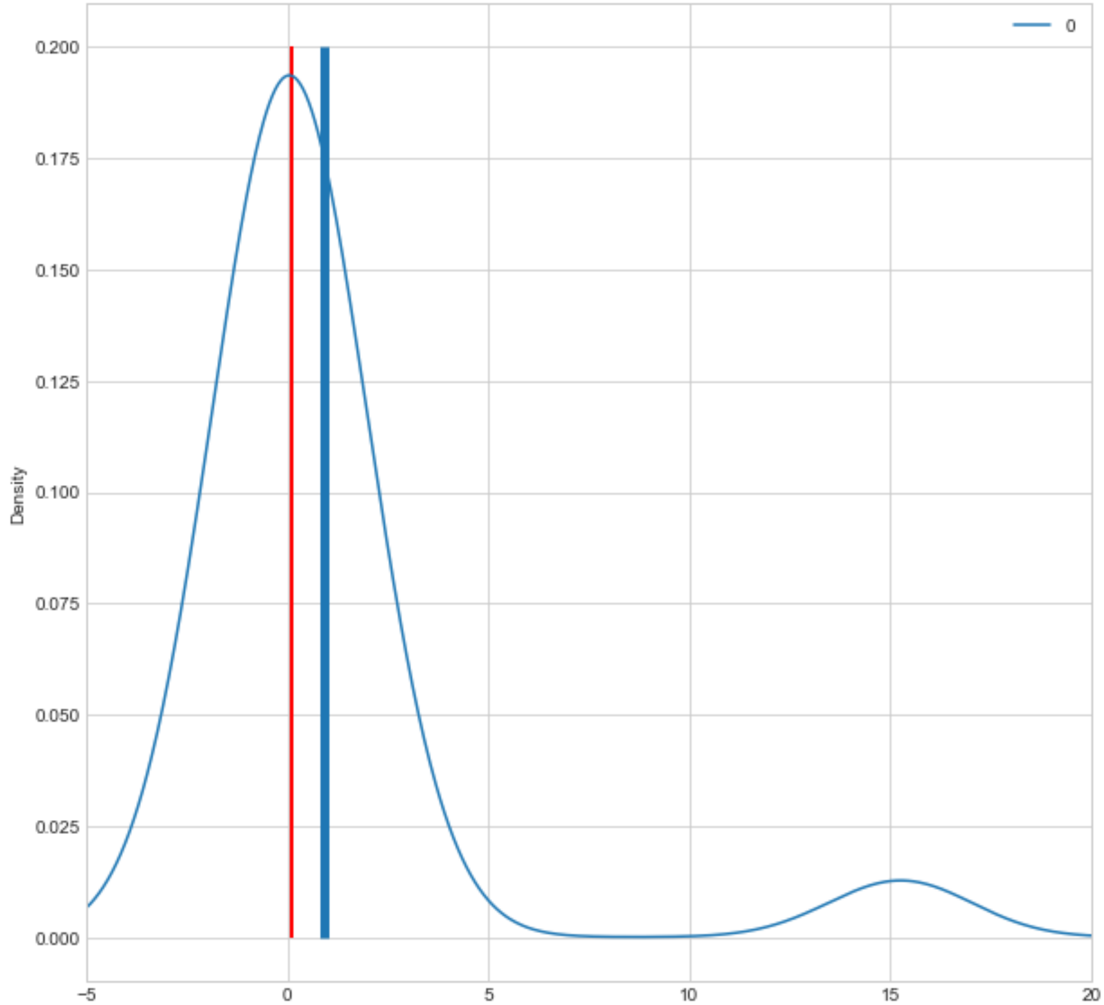


In [59]:
```
skewed_data = pd.DataFrame(np.random.exponential(size=100000))
skewed_data.plot(kind="density",
 figsize=(10,10),
 xlim=(-1,5));
plt.vlines(skewed_data.mean(), # Plot black line at mean
 ymin=0,
 ymax=0.8,
 linewidth=5.0);
plt.vlines(skewed_data.median(), # Plot red line at median
 ymin=0,
 ymax=0.8,
 linewidth=2.0,
 color="red");
```



In [60]:
```
norm_data = np.random.normal(size=50)
outliers = np.random.normal(15, size=3)
combined_data = pd.DataFrame(np.concatenate((norm_data, outliers), axis=0))
combined_data.plot(kind="density",
 figsize=(10,10),
 xlim=(-5,20));
plt.vlines(combined_data.mean(), # Plot black line at mean
 ymin=0,
 ymax=0.2,
 linewidth=5.0);
plt.vlines(combined_data.median(), # Plot red line at median
 ymin=0,
 ymax=0.2,
```

```
linewidth=2.0,
color="red");
```



In [61]:
```
mtcars.mode()
```

Out[61]:

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	10.4	8.0	275.8	110.0	3.07	3.44	17.02	0.0	0.0	3.0	2.0
1	15.2	NaN	NaN	175.0	3.92	NaN	18.90	NaN	NaN	NaN	4.0
2	19.2	NaN	NaN	180.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	21.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	21.4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	22.8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	30.4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [62]:
```
max(mtcars["mpg"])
```

Out[62]: 33.9

In [63]:
```
min(mtcars["mpg"])
```

Out[63]: 10.4

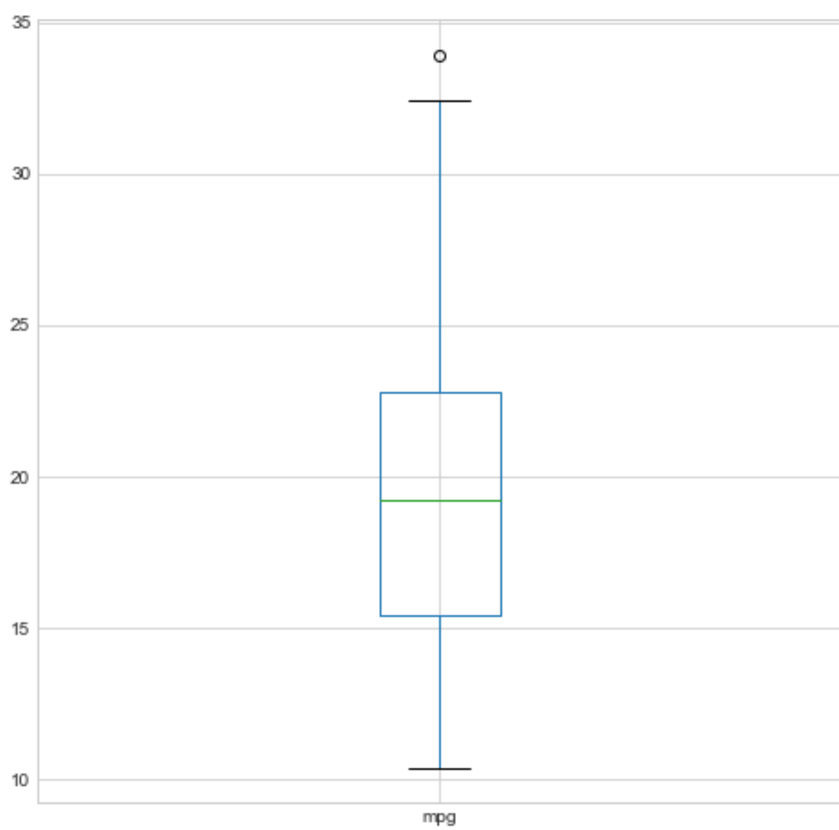
In [64]:
```
max(mtcars["mpg"])-min(mtcars["mpg"])
```
Out[64]:
```
23.5
```

In [65]:
```
five_num = [mtcars["mpg"].quantile(0),
 mtcars["mpg"].quantile(0.25),
 mtcars["mpg"].quantile(0.50),
 mtcars["mpg"].quantile(0.75),
 mtcars["mpg"].quantile(1)]
five_num
```
Out[65]:
```
[10.4, 15.425, 19.2, 22.8, 33.9]
```

In [66]:
```
mtcars["mpg"].describe()
```
Out[66]:
```
count    32.000000
mean     20.090625
std       6.026948
min      10.400000
25%      15.425000
50%      19.200000
75%      22.800000
max      33.900000
Name: mpg, dtype: float64
```

In [67]:
```
mtcars["mpg"].quantile(0.75) - mtcars["mpg"].quantile(0.25)
```
Out[67]:
```
7.375
```

In [68]:
```
mtcars.boxplot(column="mpg",
 return_type='axes',
 figsize=(8,8))
```
Out[68]:
```
<AxesSubplot:>
```



In [69]:
```
mtcars["mpg"].quantile(0.75)-mtcars["mpg"].quantile(0.25)
```

Out[69]: 7.375

In [70]:
```
mtcars["mpg"].var()
```

Out[70]: 36.32410282258065

In [71]:
```
mtcars["mpg"].std()
```

Out[71]: 6.026948052089105

In [72]:
```
abs_median_devs = abs(mtcars["mpg"] - mtcars["mpg"].median())
abs_median_devs.median() * 1.4826
```

Out[72]: 5.411490000000001

In [73]:
```
mtcars["mpg"].skew() # Check skewness
```

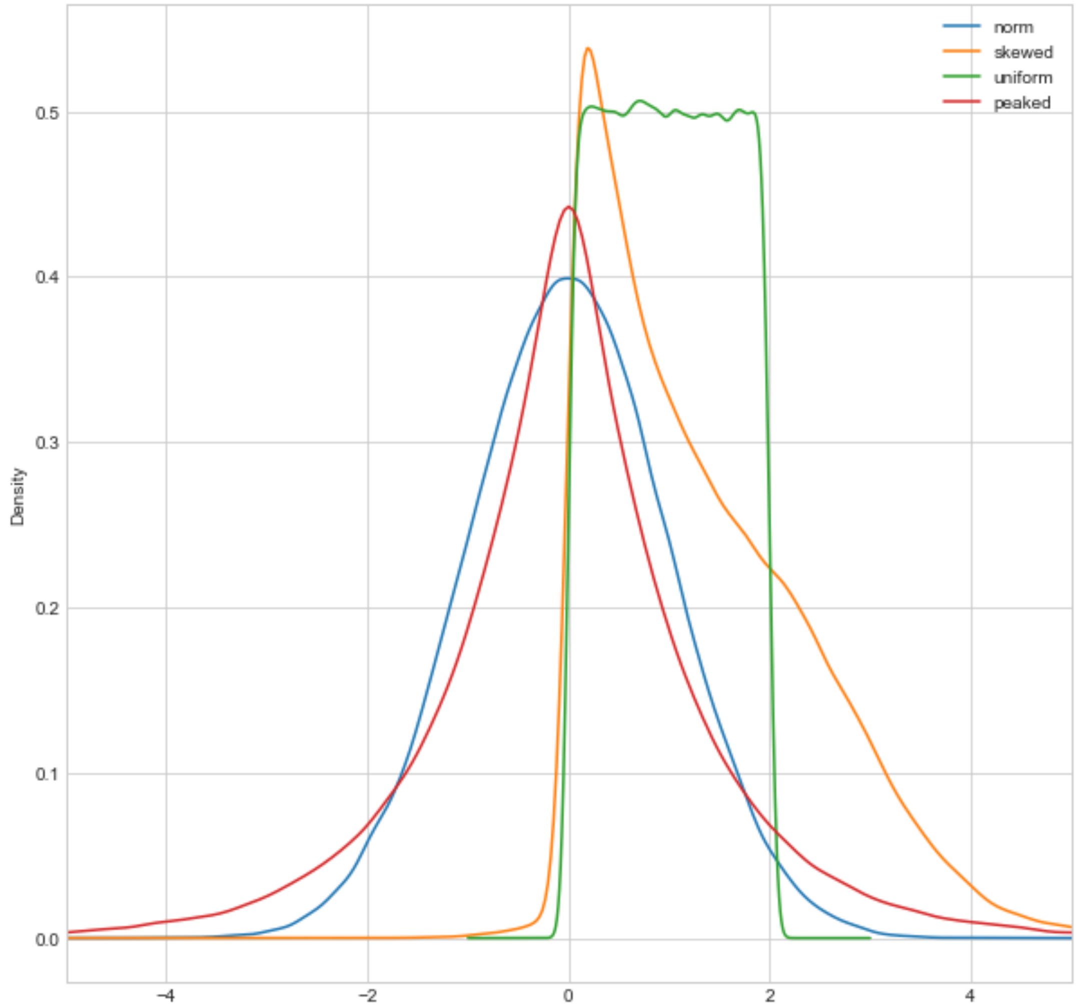
Out[73]: 0.6723771376290805

In [74]:
```
mtcars["mpg"].kurt() # Check kurtosis
```

Out[74]: -0.0220062914240855

In [75]:
```
norm_data = np.random.normal(size=100000)
skewed_data = np.concatenate((np.random.normal(size=35000)+2,
 np.random.exponential(size=65000)),
axis=0)
uniform_data = np.random.uniform(0,2, size=100000)
peaked_data = np.concatenate((np.random.exponential(size=50000),
 np.random.exponential(size=50000)*(-1)),
axis=0)
data_df = pd.DataFrame({"norm":norm_data,
 "skewed":skewed_data,
"uniform":uniform_data,
"peaked":peaked_data})
```

In [76]:
```
data_df.plot(kind="density",
 figsize=(10,10),
 xlim=(-5,5));
```



In [77]:
```
data_df.skew()
```

Out[77]:
```
norm       0.000344
skewed     1.029248
uniform    0.004356
peaked     0.010979
dtype: float64
```

In [78]:
```
data_df.kurt()
```

Out[78]:
```
norm      -0.017676
skewed     1.440956
uniform   -1.200284
peaked     2.927368
dtype: float64
```

In [79]:
```
#Question 1. Find any other HTML data table online that potentially can be useful f
#your assignments project. Read it using read_html() function and apply appropriate
#cleaning.

table_bd = table_MN = pd.read_html('https://en.wikipedia.org/wiki/Economy_of_Bangla
table_bd
```

Out[79]:

```
[                                                 0  \
 0          Dhaka, the financial centre of Bangladesh
 1                                          Currency
 2                                       Fiscal year
 3                              Trade organizations
 4                                     Country group
 5                                        Statistics
 6                                        Population
 7                                               GDP
 8                                          GDP rank
 9                                        GDP growth
 10                                   GDP per capita
 11                              GDP per capita rank
 12                                    GDP by sector
 13                                  Inflation (CPI)
 14                     Population below poverty line
 15                                Gini coefficient
 16                         Human Development Index
```

In [80]:

```python
#Question 1. Find any other HTML data table online that potentially can be useful f
#your assignments project. Read it using read_html() function and apply appropriate
#cleaning.
print(f'Total tables: {len(table_bd)}')
```

```
Total tables: 21
```

In [81]:

```python
#Question 1. Find any other HTML data table online that potentially can be useful f
#your assignments project. Read it using read_html() function and apply appropriate
#cleaning.
table_bd[2]
```

Out[81]:

	Year	GDP(in bn. US$ PPP)	GDP per capita(in US$ PPP)	GDP growth(real)	Inflation rate(in Percent)	Unemployment Rate (in Percent)	Government debt(in % of GDP)	Total Investment (in % of GDP)
0	1980	41.2	500	3.1 %	15.4 %	NaN	NaN	14.44 %
1	1981	47.4	560	5.6 %	14.5 %	NaN	NaN	17.16 %
2	1982	52.0	597	3.2 %	12.9 %	NaN	NaN	17.36%
3	1983	56.5	633	4.6 %	9.5 %	NaN	NaN	16.56 %
4	1984	61.0	664	4.2 %	10.4 %	NaN	NaN	16.48 %
5	1985	65.3	693	3.7 %	10.5 %	NaN	NaN	15.83 %
6	1986	69.3	715	4.0 %	10.2 %	NaN	NaN	16.18 %
7	1987	73.1	735	2.9 %	10.8 %	NaN	NaN	15.47 %
8	1988	77.5	759	2.4 %	9.7 %	NaN	NaN	15.74 %
9	1989	84.0	801	4.3 %	8.7 %	NaN	NaN	16.12 %
10	1990	91.1	848	4.6 %	10.5 %	NaN	NaN	16.46 %
11	1991	98.1	892	4.2 %	8.3 %	2.20 %	NaN	16.90 %
12	1992	105.1	935	4.8 %	3.6 %	2.25 %	NaN	17.31 %
13	1993	112.3	977	4.3 %	3.0 %	2.37 %	NaN	17.95 %
14	1994	119.9	1021	4.5 %	6.2 %	2.44 %	NaN	18.40 %
15	1995	128.2	1069	4.8 %	10.1 %	2.48 %	NaN	19.12 %
16	1996	137.1	1120	5.0 %	2.5 %	2.51 %	NaN	20.73 %
17	1997	146.8	1175	5.3 %	5.0 %	2.69 %	NaN	21.82 %

	Year	GDP(in bn. US$ PPP)	GDP per capita(in US$ PPP)	GDP growth(real)	Inflation rate(in Percent)	Unemployment Rate (in Percent)	Government debt(in % of GDP)	Total Investment (in % of GDP)
18	1998	155.9	1223	5.0 %	8.6 %	2.83 %	NaN	22.12 %
19	1999	166.9	1284	5.4 %	6.2 %	3.10 %	NaN	22.72 %
20	2000	180.2	1361	5.6 %	2.5 %	3.27 %	NaN	23.81 %
21	2001	193.2	1434	4.8 %	1.9 %	3.55 %	NaN	24.17 %
22	2002	205.7	1501	4.8 %	3.7 %	3.96 %	NaN	24.34 %
23	2003	221.9	1594	5.8 %	5.4 %	4.32 %	44.3 %	24.68 %
24	2004	241.9	1713	6.1 %	6.1 %	4.30 %	43.5 %	24.99 %
25	2005	265.5	1855	6.3 %	7.0 %	4.25 %	42.3 %	25.83 %
26	2006	292.4	2018	6.9 %	6.8 %	3.59 %	42.3 %	26.14 %
27	2007	319.7	2183	6.5 %	9.1 %	3.77 %	41.9 %	26.18 %
28	2008	344.0	2325	5.5 %	8.9 %	4.07 %	40.6 %	26.20 %
29	2009	365.0	2441	5.3 %	4.9 %	5.00 %	39.5 %	26.21 %
30	2010	391.7	2592	6.0 %	9.4 %	3.37 %	35.5 %	26.25 %
31	2011	425.8	2785	6.5 %	11.5 %	3.71 %	36.6 %	27.42 %
32	2012	460.8	2979	6.3 %	6.2 %	4.04 %	36.2 %	28.26 %
33	2013	496.5	3171	6.0 %	7.5 %	4.43 %	35.8 %	28.39 %
34	2014	537.3	3396	6.3 %	7.0 %	4.41 %	35.3 %	28.58 %
35	2015	581.6	3638	6.8 %	6.2 %	4.42 %	33.6 %	28.89 %
36	2016	629.9	3900	7.2 %	5.7 %	4.35 %	33.3 %	29.65 %
37	2017	710.5	4331	7.6 %	5.6 %	4.37 %	32.6 %	30.51 %

In [82]:

```
#Question 1. Find any other HTML data table online that potentially can be useful f
#your assignments project. Read it using read_html() function and apply appropriate
#cleaning.
len(table_bd)
```

Out[82]: 21

In [83]:

```
#Question 1. Find any other HTML data table online that potentially can be useful f
#your assignments project. Read it using read_html() function and apply appropriate
#cleaning.
df = table_bd[2]
df.head(10)
```

Out[83]:

	Year	GDP(in bn. US$ PPP)	GDP per capita(in US$ PPP)	GDP growth(real)	Inflation rate(in Percent)	Unemployment Rate (in Percent)	Government debt(in % of GDP)	Total Investment (in % of GDP)
0	1980	41.2	500	3.1 %	15.4 %	NaN	NaN	14.44 %
1	1981	47.4	560	5.6 %	14.5 %	NaN	NaN	17.16 %
2	1982	52.0	597	3.2 %	12.9 %	NaN	NaN	17.36%
3	1983	56.5	633	4.6 %	9.5 %	NaN	NaN	16.56 %
4	1984	61.0	664	4.2 %	10.4 %	NaN	NaN	16.48 %
5	1985	65.3	693	3.7 %	10.5 %	NaN	NaN	15.83 %
6	1986	69.3	715	4.0 %	10.2 %	NaN	NaN	16.18 %

Year	GDP(in bn. US$ PPP)	GDP per capita(in US$ PPP)	GDP growth(real)	Inflation rate(in Percent)	Unemployment Rate (in Percent)	Government debt(in % of GDP)	Total Investment (in % of GDP)

In [84]:
```
#Question 1. Find any other HTML data table online that potentially can be useful f
#your assignments project. Read it using read_html() function and apply appropriate
#cleaning.
df.columns
```

Out[84]:
```
Index(['Year', 'GDP(in bn. US$ PPP)', 'GDP per capita(in US$ PPP)',
       'GDP growth(real)', 'Inflation rate(in Percent)',
       'Unemployment Rate (in Percent)', 'Government debt(in % of GDP)',
       'Total Investment (in % of GDP)'],
      dtype='object')
```

In [85]:
```
#Question 1. Find any other HTML data table online that potentially can be useful f
#your assignments project. Read it using read_html() function and apply appropriate
#cleaning.
#changing value from integer to float, first replace the integer with nothing
#df['GDP per capita(in US$ PPP)'].replace({'%':''}, regex=True).astype('float')
df=df.replace({'%':''},regex=True)
```

In [86]:
```
#Question 1. Find any other HTML data table online that potentially can be useful f
#your assignments project. Read it using read_html() function and apply appropriate
#cleaning.
df.head(10)
```

Out[86]:

	Year	GDP(in bn. US$ PPP)	GDP per capita(in US$ PPP)	GDP growth(real)	Inflation rate(in Percent)	Unemployment Rate (in Percent)	Government debt(in % of GDP)	Total Investment (in % of GDP)
0	1980	41.2	500	3.1	15.4	NaN	NaN	14.44
1	1981	47.4	560	5.6	14.5	NaN	NaN	17.16
2	1982	52.0	597	3.2	12.9	NaN	NaN	17.36
3	1983	56.5	633	4.6	9.5	NaN	NaN	16.56
4	1984	61.0	664	4.2	10.4	NaN	NaN	16.48
5	1985	65.3	693	3.7	10.5	NaN	NaN	15.83
6	1986	69.3	715	4.0	10.2	NaN	NaN	16.18
7	1987	73.1	735	2.9	10.8	NaN	NaN	15.47
8	1988	77.5	759	2.4	9.7	NaN	NaN	15.74
9	1989	84.0	801	4.3	8.7	NaN	NaN	16.12

In [87]:
```
#Question 1. Find any other HTML data table online that potentially can be useful f
#your assignments project. Read it using read_html() function and apply appropriate
#cleaning.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40 entries, 0 to 39
Data columns (total 8 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
```

In [88]:
```
#Question 1. Find any other HTML data table online that potentially can be useful f
#your assignments project. Read it using read_html() function and apply appropriate
#cleaning.
#strip removes extra spaces
from unicodedata import normalize
def clean_normalize_whitespace(x):
    if isinstance(x, str):
        return normalize('NFKC', x).strip()
    else:
        return x
```

In [89]:
```
#Question 1. Find any other HTML data table online that potentially can be useful f
#your assignments project. Read it using read_html() function and apply appropriate
#cleaning.
df = df.applymap(clean_normalize_whitespace)
```

In [90]:
```
#Question 1. Find any other HTML data table online that potentially can be useful f
#your assignments project. Read it using read_html() function and apply appropriate
#cleaning.
df
```

Out[90]:

	Year	GDP(in bn. US$ PPP)	GDP per capita(in US$ PPP)	GDP growth(real)	Inflation rate(in Percent)	Unemployment Rate (in Percent)	Government debt(in % of GDP)	Total Investment (in % of GDP)
0	1980	41.2	500	3.1	15.4	NaN	NaN	14.44
1	1981	47.4	560	5.6	14.5	NaN	NaN	17.16
2	1982	52.0	597	3.2	12.9	NaN	NaN	17.36
3	1983	56.5	633	4.6	9.5	NaN	NaN	16.56
4	1984	61.0	664	4.2	10.4	NaN	NaN	16.48
5	1985	65.3	693	3.7	10.5	NaN	NaN	15.83
6	1986	69.3	715	4.0	10.2	NaN	NaN	16.18
7	1987	73.1	735	2.9	10.8	NaN	NaN	15.47
8	1988	77.5	759	2.4	9.7	NaN	NaN	15.74
9	1989	84.0	801	4.3	8.7	NaN	NaN	16.12
10	1990	91.1	848	4.6	10.5	NaN	NaN	16.46
11	1991	98.1	892	4.2	8.3	2.20	NaN	16.90
12	1992	105.1	935	4.8	3.6	2.25	NaN	17.31
13	1993	112.3	977	4.3	3.0	2.37	NaN	17.95
14	1994	119.9	1021	4.5	6.2	2.44	NaN	18.40
15	1995	128.2	1069	4.8	10.1	2.48	NaN	19.12
16	1996	137.1	1120	5.0	2.5	2.51	NaN	20.73
17	1997	146.8	1175	5.3	5.0	2.69	NaN	21.82
18	1998	155.9	1223	5.0	8.6	2.83	NaN	22.12
19	1999	166.9	1284	5.4	6.2	3.10	NaN	22.72
20	2000	180.2	1361	5.6	2.5	3.27	NaN	23.81

	Year	GDP(in bn. US$ PPP)	GDP per capita(in US$ PPP)	GDP growth(real)	Inflation rate(in Percent)	Unemployment Rate (in Percent)	Government debt(in % of GDP)	Total Investment (in % of GDP)
21	2001	193.2	1434	4.8	1.9	3.55	NaN	24.17
22	2002	205.7	1501	4.8	3.7	3.96	NaN	24.34
23	2003	221.9	1594	5.8	5.4	4.32	44.3	24.68
24	2004	241.9	1713	6.1	6.1	4.30	43.5	24.99
25	2005	265.5	1855	6.3	7.0	4.25	42.3	25.83
26	2006	292.4	2018	6.9	6.8	3.59	42.3	26.14
27	2007	319.7	2183	6.5	9.1	3.77	41.9	26.18
28	2008	344.0	2325	5.5	8.9	4.07	40.6	26.20
29	2009	365.0	2441	5.3	4.9	5.00	39.5	26.21
30	2010	391.7	2592	6.0	9.4	3.37	35.5	26.25
31	2011	425.8	2785	6.5	11.5	3.71	36.6	27.42
32	2012	460.8	2979	6.3	6.2	4.04	36.2	28.26
33	2013	496.5	3171	6.0	7.5	4.43	35.8	28.39
34	2014	537.3	3396	6.3	7.0	4.41	35.3	28.58
35	2015	581.6	3638	6.8	6.2	4.42	33.6	28.89
36	2016	629.9	3900	7.2	5.7	4.35	33.3	29.65
37	2017	710.5	4331	7.6	5.6	4.37	32.6	30.51

In [91]:
```
#Question 1. Find any other HTML data table online that potentially can be useful f
#your assignments project. Read it using read_html() function and apply appropriate
#cleaning.
df.columns = df.columns.to_series().apply(clean_normalize_whitespace)
df.columns
```

Out[91]:
```
Index(['Year', 'GDP(in bn. US$ PPP)', 'GDP per capita(in US$ PPP)',
       'GDP growth(real)', 'Inflation rate(in Percent)',
       'Unemployment Rate (in Percent)', 'Government debt(in % of GDP)',
       'Total Investment (in % of GDP)'],
      dtype='object')
```

In [92]:
```
#Question 1. Find any other HTML data table online that potentially can be useful f
#your assignments project. Read it using read_html() function and apply appropriate
#cleaning.
#if your data have any character
df['Year'].replace({'%': '', '-': '-', '\(est\)': ''}, regex=True).astype('int')
```

Out[92]:

```
0     1980
1     1981
2     1982
3     1983
4     1984
5     1985
6     1986
7     1987
8     1988
9     1989
10    1990
11    1991
12    1992
13    1993
14    1994
15    1995
16    1996
17    1997
18    1998
19    1999
20    2000
21    2001
22    2002
23    2003
24    2004
25    2005
```

In [93]:
```
#Question 1. Find any other HTML data table online that potentially can be useful f
#your assignments project. Read it using read_html() function and apply appropriate
#cleaning.
clean_dict = {'%': '', '-': '-', '\(est\)': ''}
```

In [94]:
```
df1 = df.dropna(axis=1)
```

```
    Year  GDP(in bn. US$ PPP)  GDP per capita(in US$ PPP) GDP growth(real)  \
0   1980                 41.2                        500              3.1
1   1981                 47.4                        560              5.6
2   1982                 52.0                        597              3.2
3   1983                 56.5                        633              4.6
4   1984                 61.0                        664              4.2
5   1985                 65.3                        693              3.7
6   1986                 69.3                        715              4.0
7   1987                 73.1                        735              2.9
8   1988                 77.5                        759              2.4
9   1989                 84.0                        801              4.3
10  1990                 91.1                        848              4.6
11  1991                 98.1                        892              4.2
12  1992                105.1                        935              4.8
13  1993                112.3                        977              4.3
14  1994                119.9                       1021              4.5
15  1995                128.2                       1069              4.8
16  1996                137.1                       1120              5.0
17  1997                146.8                       1175              5.3
18  1998                155.9                       1223              5.0
19  1999                166.9                       1284              5.4
20  2000                180.2                       1361              5.6
21  2001                193.2                       1434              4.8
22  2002                205.7                       1501              4.8
23  2003                221.9                       1594              5.8
24  2004                241.9                       1713              6.1
25  2005                265.5                       1855              6.3
26  2006                292.4                       2018              6.9
27  2007                319.7                       2183              6.5
28  2008                344.0                       2325              5.5
29  2009                365.0                       2441              5.3
30  2010                391.7                       2592              6.0
31  2011                425.8                       2785              6.5
32  2012                460.8                       2979              6.3
33  2013                496.5                       3171              6.0
34  2014                537.3                       3396              6.3
35  2015                581.6                       3638              6.8
36  2016                629.9                       3900              7.2
37  2017                710.5                       4331              7.6
38  2018                785.9                       4730              7.9
39  2019                869.4                       5228              8.1

   Inflation rate(in Percent) Total Investment (in % of GDP)
0                        15.4                          14.44
1                        14.5                          17.16
2                        12.9                          17.36
3                         9.5                          16.56
4                        10.4                          16.48
5                        10.5                          15.83
6                        10.2                          16.18
7                        10.8                          15.47
8                         9.7                          15.74
9                         8.7                          16.12
10                       10.5                          16.46
```

In [95]:
```
#Question 1. Find any other HTML data table online that potentially can be useful f
#your assignments project. Read it using read_html() function and apply appropriate
#cleaning.
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40 entries, 0 to 39
Data columns (total 6 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   Year                             40 non-null     int64  
 1   GDP(in bn. US$ PPP)              40 non-null     float64
 2   GDP per capita(in US$ PPP)       40 non-null     int64  
 3   GDP growth(real)                 40 non-null     object 
 4   Inflation rate(in Percent)       40 non-null     object 
 5   Total Investment (in % of GDP)   40 non-null     object 
```

In [96]:
```python
#Question 2. Implement an example of data retrieval using one of the packages liste
#in the table.
import tweepy
from tweepy import Stream
from tweepy import OAuthHandler
import pandas as pd
import numpy as np
import re
import csv
```

In [97]:
```python
#Question 2. Implement an example of data retrieval using one of the packages liste
#in the table.
bearer_tocken = 'AAAAAAAAAAAAAAAAAAAAAIQnVgEAAAAAG3oZJnERuECSPzZXlklkx8FRXNc%3DxxEI
client = tweepy.Client(bearer_tocken)
path = 'C:/DIT45602/lectures/'
screen_name = 'iamsrk'
```

In [98]:
```python
#Question 2. Implement an example of data retrieval using one of the packages liste
#in the table.
tweets = client.get_users_tweets(client.get_user(username=screen_name).data['id'])
outtweets = [[tweet.id, tweet.text] for tweet in tweets.data]

with open(path + '%s_tweets.csv' % screen_name, 'w', encoding='utf-8') as f:
        writer = csv.writer(f)
        writer.writerow(["id","text"])
        writer.writerows(outtweets)
        print('saved - ' + path   )
pass
```

```
saved - C:/DIT45602/lectures/
```

In [99]:
```python
#Question 2. Implement an example of data retrieval using one of the packages liste
#in the table.
# Printing tweets of Prime Minister of Bangladesh Sheikh Hasina form her Twitter pr
def remove_pattern(text,pattern):

    # re.findall() finds the pattern i.e @user and puts it in a list for further ta
    r = re.findall(pattern,text)

    # re.sub() removes @user from the sentences in the dataset
    for i in r:
        text = re.sub(i,"",text)

    return text
```

In [100]:
```python
#Question 2. Implement an example of data retrieval using one of the packages liste
#in the table.
## Suppose we have a text with many email addresses
str = 'purple alice@google.com, blah monkey bob@abc.com blah dishwasher'
```

```
## Here re.findall() returns a list of all the found email strings
emails = re.findall(r'[\w\.-]+@[\w\.-]+', str) ## ['alice@google.com', 'bob@abc.com
for email in emails:
    # do something with each found email string
    print(email)
```

```
alice@google.com
bob@abc.com
```

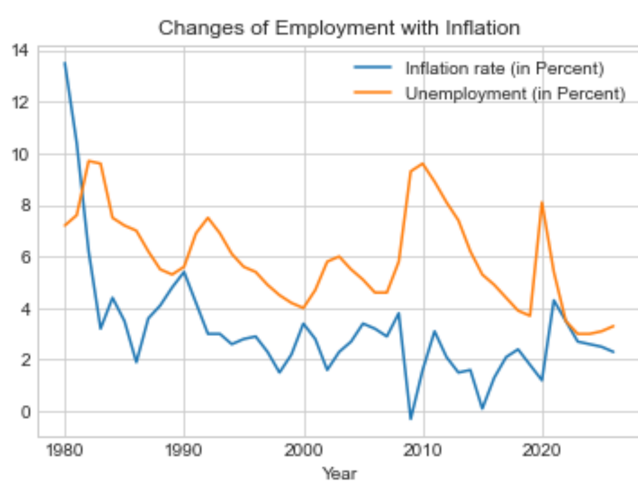
In [101]:
```
#Question 2. Implement an example of data retrieval using one of the packages liste
#in the table.
# Printing tweets of Prime Minister of Bangladesh Sheikh Hasina form her Twitter pr
raw_tweets = pd.read_csv('C:/DIT45602/lectures/'+screen_name+'_tweets.csv')
raw_tweets
```

Out[101]:

	id	text
0	1503604476934242308	Kuch kuch hone wala hai, OTT ki duniya mein. h...
1	1499700448005337089	Extremely happy to see #LoveHostel receiving ...
2	1499011302806605826	Ok boys and girls time to get back to work. Ha...
3	1499010940058345475	If I knew you were coming home would have told...
4	1499010607403524104	Aaj kal toh Thums Up hi pi raha hoon….maybe it...
5	1499010039322853376	Toh aadha reply….kar….ok #Pathaan https://t.co...
6	1499009660505911302	Dimaag try kar shaayad work karega…Mann pyaar ...
7	1499009443773616128	Thoda tum adjust kar lena thoda main kar dunga...
8	1499009111400144896	Arre yaar Aamir kehta hai pehle Pathaan dikha!...
9	1499008466035167234	Ok next time I will be 'Khabardaar' #Pathaan h...

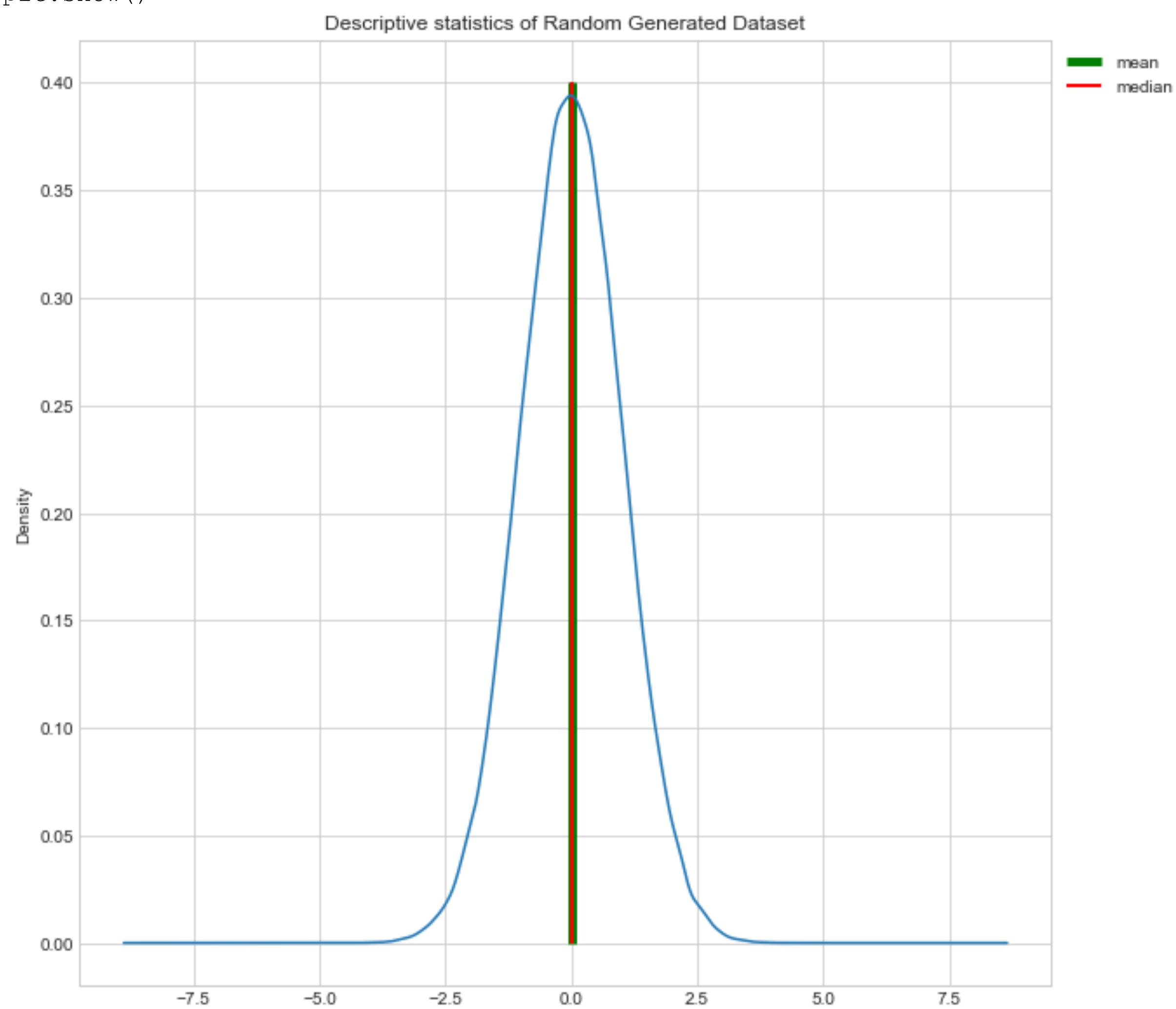
In [102]:
```
#Question 3: Add legend to the plot with caption for all lines on the plot.
#Graph-1
#Comment-In this plot which colors represent which data comes automatically.
#I addeed title in the graph.
plt.style.use('seaborn-whitegrid')
df_GDP.plot.line(x='Year', y=['Inflation rate (in Percent)', 'Unemployment (in Perc
plt.title('Changes of Employment with Inflation')
```

Out[102]:
```
Text(0.5, 1.0, 'Changes of Employment with Inflation')
```



In [103]:
```
#Question 3: Add legend to the plot with caption for all lines on the plot.
#Graph-2
```

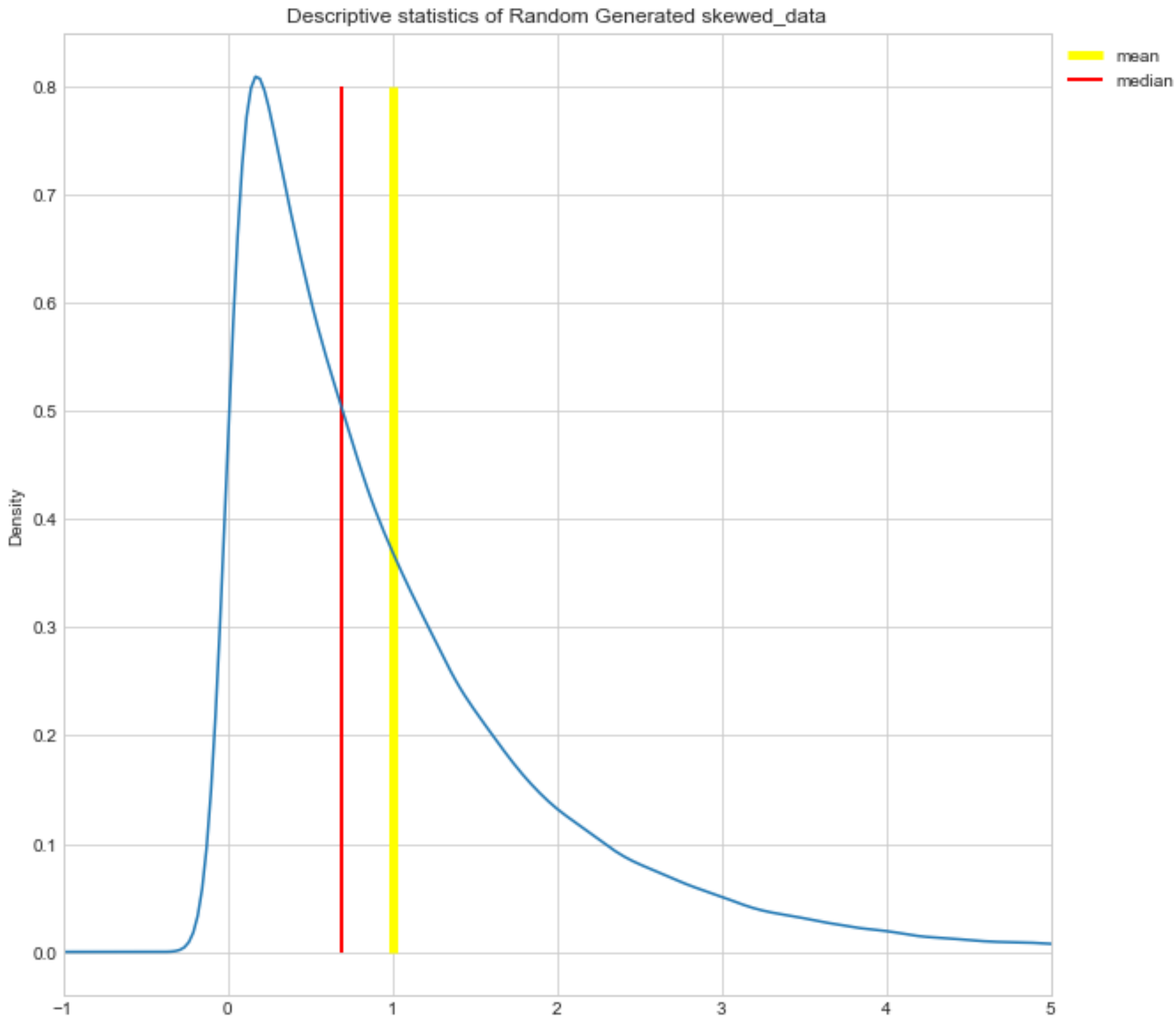
```
norm_data = pd.DataFrame(np.random.normal(size=100000))
norm_data.plot(kind="density",
 figsize=(10,10));
arr1 = plt.vlines(norm_data.mean(), # Plot green line at mean
 ymin=0,
 ymax=0.4,
 linewidth=5.0,
 color="green");
arr2 = plt.vlines(norm_data.median(), # Plot red line at median
 ymin=0,
 ymax=0.4,
 linewidth=2.0,
 color="red");
plt.legend([arr1, arr2], ['mean','median'],bbox_to_anchor=(1.0, 1),loc='upper left'
plt.title('Descriptive statistics of Random Generated Dataset')
plt.show()
```



In [104]:
```
#Question 3: Add legend to the plot with caption for all lines on the plot.
#Graph-3

skewed_data = pd.DataFrame(np.random.exponential(size=100000))
skewed_data.plot(kind="density",
 figsize=(10,10),
 xlim=(-1,5));
arr1 = plt.vlines(skewed_data.mean(), # Plot yellow line at mean
 ymin=0,
 ymax=0.8,
 linewidth=5.0,
```

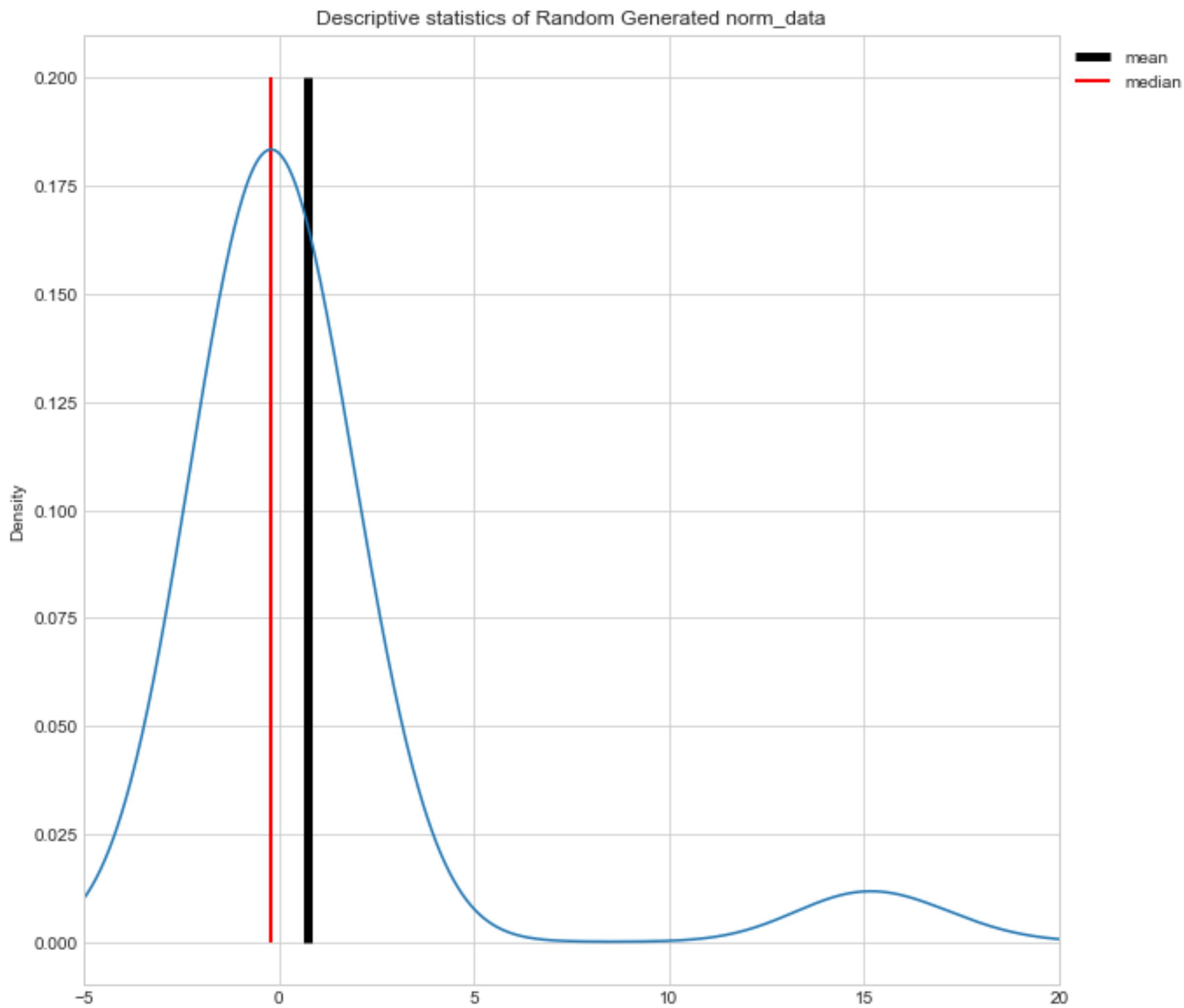
```
 color="yellow");
arr2 = plt.vlines(skewed_data.median(), # Plot red line at median
 ymin=0,
 ymax=0.8,
 linewidth=2.0,
 color="red");
plt.legend([arr1, arr2], ['mean','median'],bbox_to_anchor=(1.0, 1),loc='upper left'
plt.title('Descriptive statistics of Random Generated skewed_data')
plt.show()
```



In [105]:

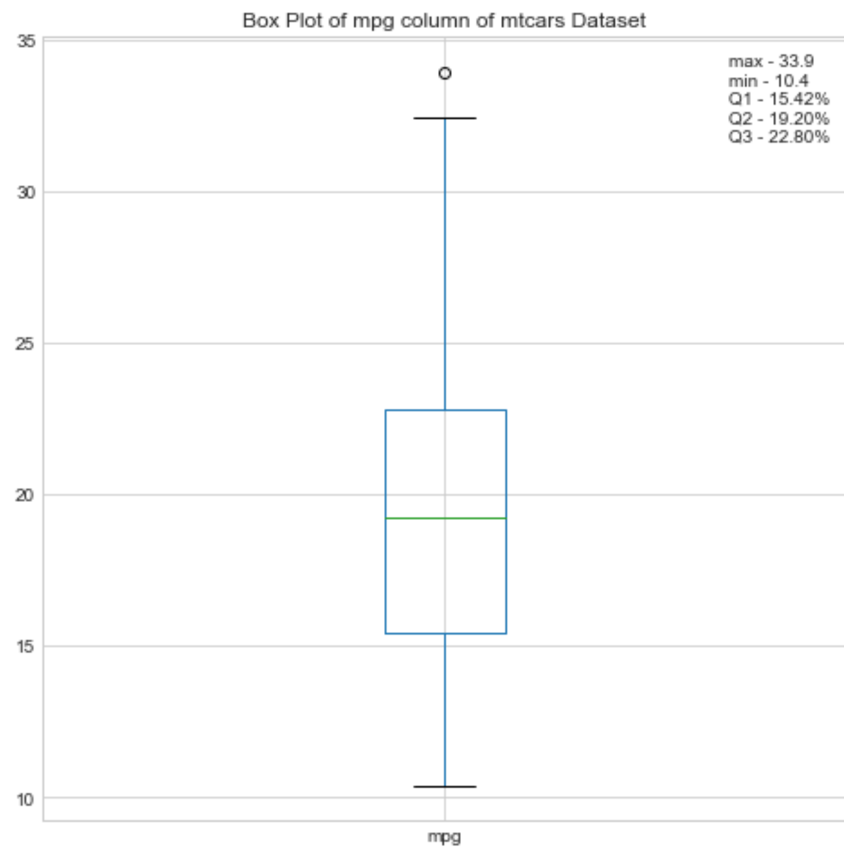
```
#Question 3: Add legend to the plot with caption for all lines on the plot.
#Graph-4
norm_data = np.random.normal(size=50)
outliers = np.random.normal(15, size=3)
combined_data = pd.DataFrame(np.concatenate((norm_data, outliers), axis=0))
combined_data.plot(kind="density",
 figsize=(10,10),
 xlim=(-5,20));
arr1 = plt.vlines(combined_data.mean(), # Plot black line at mean
 ymin=0,
 ymax=0.2,
 linewidth=5.0,
 color="black");
arr2 = plt.vlines(combined_data.median(), # Plot red line at median
 ymin=0,
 ymax=0.2,
 linewidth=2.0,
 color="red");
```

```
plt.legend([arr1, arr2], ['mean','median'],bbox_to_anchor=(1.0, 1),loc='upper left'
plt.title('Descriptive statistics of Random Generated norm_data')
plt.show()
```



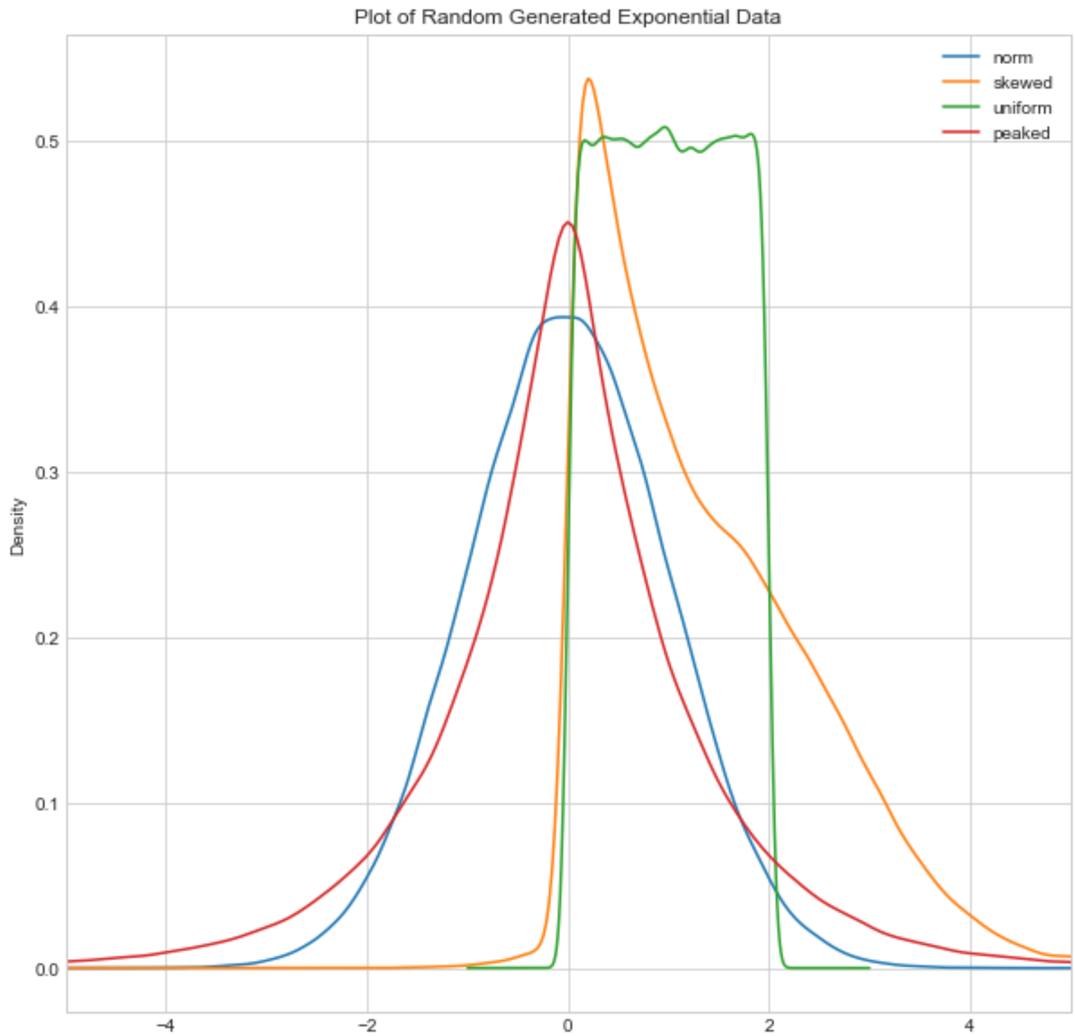
In [106]:
```
#Question 3: Add legend to the plot with caption for all lines on the plot.
#Graph-4 (Box plot)
mtcars.boxplot(column="mpg",
 return_type='axes',
 figsize=(8,8)
)
plt.title('Box Plot of mpg column of mtcars Dataset')
plt.legend(bbox_to_anchor=(1.0, 1),loc='upper right', title='max - 33.9\nmin - 10.4
plt.show()
```

No handles with labels found to put in legend.



In [107]:
```
#Question 3: Add legend to the plot with caption for all lines on the plot.
#Graph-5
#Comment-In this plot which colors represent which data comes automatically.
#I addeed title in the graph.
norm_data = np.random.normal(size=100000)
skewed_data = np.concatenate((np.random.normal(size=35000)+2,
 np.random.exponential(size=65000)),
axis=0)
uniform_data = np.random.uniform(0,2, size=100000)
peaked_data = np.concatenate((np.random.exponential(size=50000),
 np.random.exponential(size=50000)*(-1)),
axis=0)
data_df = pd.DataFrame({"norm":norm_data,
 "skewed":skewed_data,
"uniform":uniform_data,
"peaked":peaked_data})
data_df.plot(kind="density",
 figsize=(10,10),
 xlim=(-5,5));
plt.title('Plot of Random Generated Exponential Data')
```

```
plt.show()
```



In [108]:
```
#Question 4: Plot boxplots for all numerical columns in the dataset and provide you
#comments on what you observe.
mtcars
```

Out[108]:

model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
model											
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6

In [109]:
```
#Question 4: Plot boxplots for all numerical columns in the dataset and provide you
#comments on what you observe.
mtcars.describe()
```

Out[109]:

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am
count	32.000000	32.000000	32.000000	32.000000	32.000000	32.000000	32.000000	32.000000	32.000000
mean	20.090625	6.187500	230.721875	146.687500	3.596563	3.217250	17.848750	0.437500	0.406250
std	6.026948	1.785922	123.938694	68.562868	0.534679	0.978457	1.786943	0.504016	0.498991
min	10.400000	4.000000	71.100000	52.000000	2.760000	1.513000	14.500000	0.000000	0.000000
25%	15.425000	4.000000	120.825000	96.500000	3.080000	2.581250	16.892500	0.000000	0.000000
50%	19.200000	6.000000	196.300000	123.000000	3.695000	3.325000	17.710000	0.000000	0.000000
75%	22.800000	8.000000	326.000000	180.000000	3.920000	3.610000	18.900000	1.000000	1.000000
max	33.900000	8.000000	472.000000	335.000000	4.930000	5.424000	22.900000	1.000000	1.000000

In [110]:
```
#Question 4: Plot boxplots for all numerical columns in the dataset and provide you
#comments on what you observe.
mtcars.median()
```

Out[110]:

```
mpg      19.200
cyl       6.000
disp    196.300
hp      123.000
```

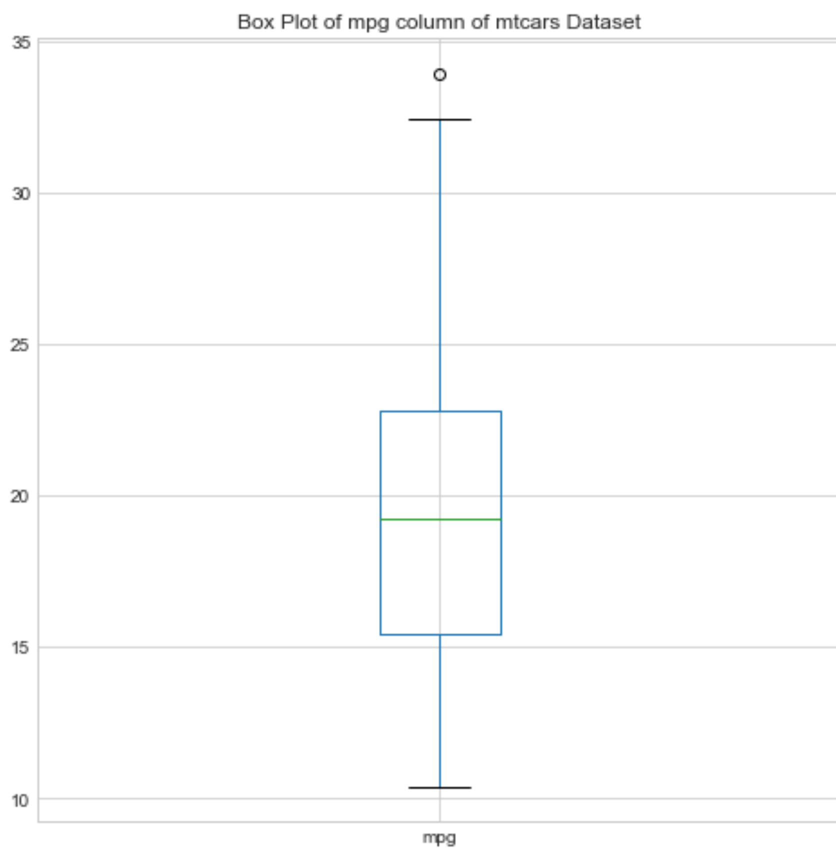
In [111]:

```
#Question 4: Plot boxplots for all numerical columns in the dataset and provide you
#comments on what you observe.
mtcars.boxplot(column="mpg",
 return_type='axes',
 figsize=(8,8)
)
plt.title('Box Plot of mpg column of mtcars Dataset')

#Comments on Box-plot
#A boxplot describes summary of five numbers of a dataset.
#These are minimum, first quartile, median, third quartile, and maximum numbers.
#Min number is 10.40 (the black line below the box).
#Median is 19.2 (the middle grean line in the box).
#First quarantile number is 15.42 which is below median.
#Third quarantile number is 22.8 which is greater than median.
#Max number is 33.9 (the black line above the box).
#There is one outlier in the dataset(circle above the max line).
```

Out[111]:

```
Text(0.5, 1.0, 'Box Plot of mpg column of mtcars Dataset')
```



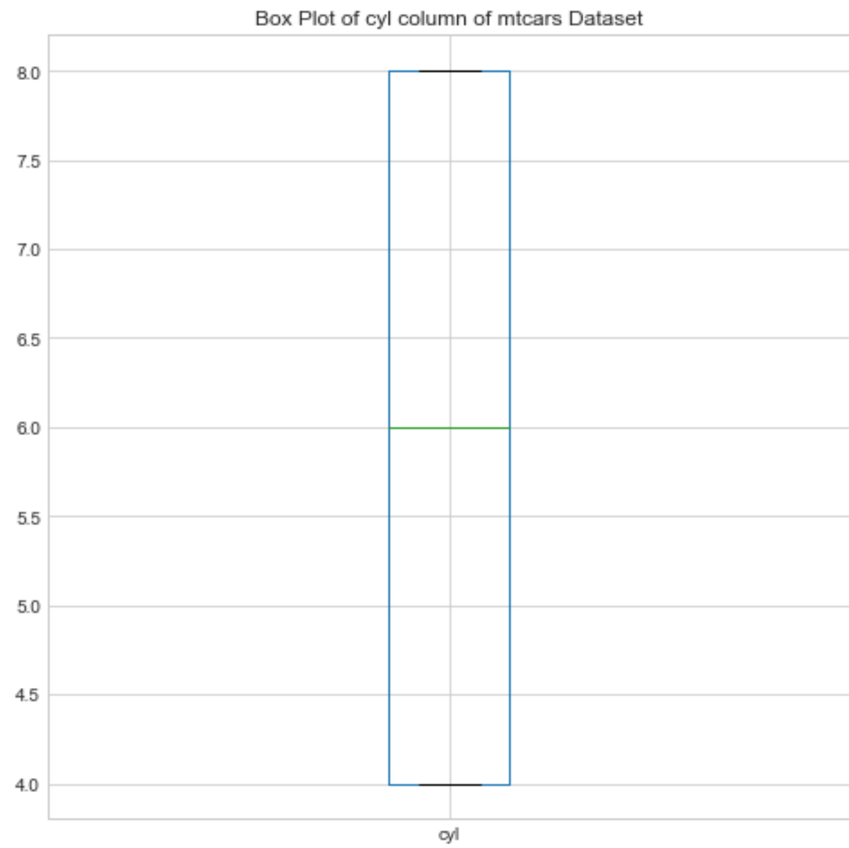
In [112]:

```
#Question 4: Plot boxplots for all numerical columns in the dataset and provide you
#comments on what you observe.
mtcars.boxplot(column="cyl",
 return_type='axes',
 figsize=(8,8)
)
plt.title('Box Plot of cyl column of mtcars Dataset')

#Comments on Box-plot
```

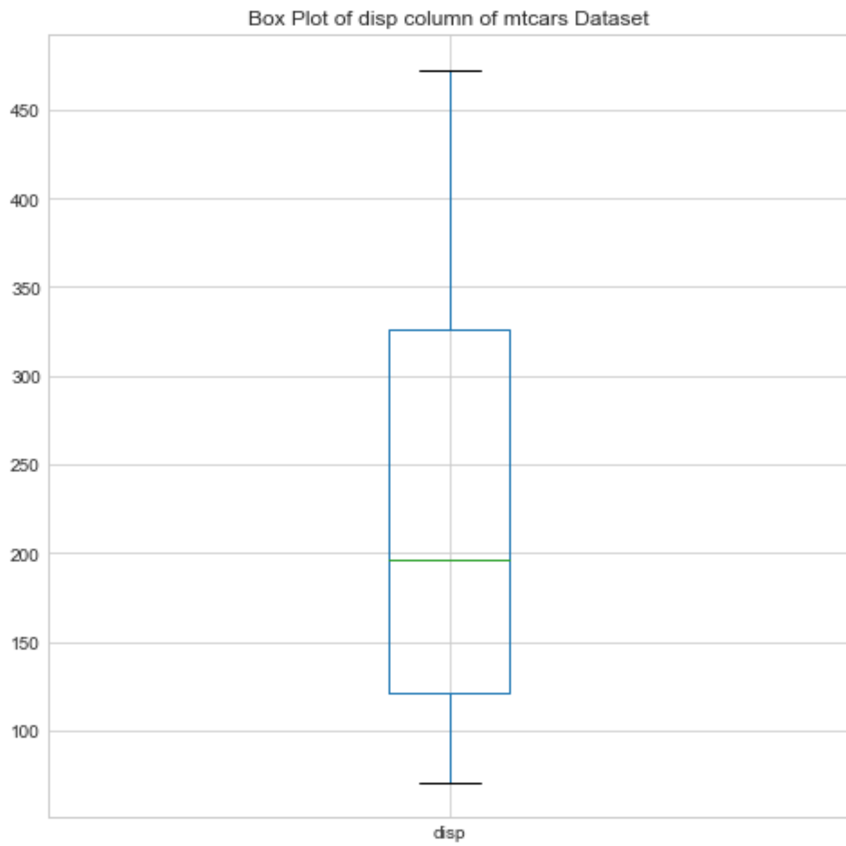
```
#A boxplot describes summary of five numbers of a dataset.
#These are minimum, first quartile, median, third quartile, and maximum numbers.
#Min number is 4 (the black line below the box).
#Median is 6 (the middle grean line in the box).
#First quarantile number is 4 which is equal to median.
#Third quarantile number is 8 which is greater than median.
#Max number is 8 (the black line above the box).
#There is no outliers in the dataset.
```

Out[112]: Text(0.5, 1.0, 'Box Plot of cyl column of mtcars Dataset')



In [113]:
```
#Question 4: Plot boxplots for all numerical columns in the dataset and provide you
#comments on what you observe.
mtcars.boxplot(column="disp",
 return_type='axes',
 figsize=(8,8)
)
plt.title('Box Plot of disp column of mtcars Dataset')
#Comments on Box-plot
#A boxplot describes summary of five numbers of a dataset.
#These are minimum, first quartile, median, third quartile, and maximum numbers.
#Min number is 71.1 (the black line below the box).
#Median is 196.3 (the middle grean line in the box).
#First quarantile number is 120.82 which is below median.
#Third quarantile number is 326 which is greater than median.
#Max number is 472 (the black line above the box).
#There is no outliers in the dataset.
```

Out[113]: Text(0.5, 1.0, 'Box Plot of disp column of mtcars Dataset')

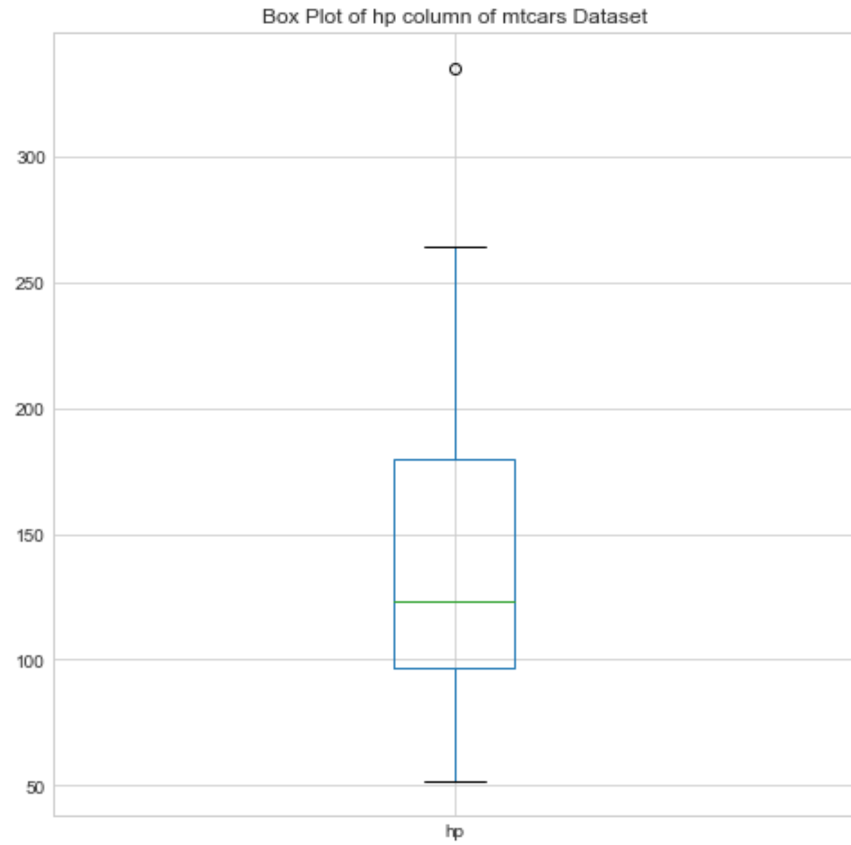


In [114]:
```
#Question 4: Plot boxplots for all numerical columns in the dataset and provide you
#comments on what you observe.
mtcars.boxplot(column="hp",
 return_type='axes',
 figsize=(8,8)
)
plt.title('Box Plot of hp column of mtcars Dataset')
#Comments on Box-plot
#A boxplot describes summary of five numbers of a dataset.
#These are minimum, first quartile, median, third quartile, and maximum numbers.
#Min number is 52 (the black line below the box).
#Median is 123 (the middle grean line in the box).
#First quarantile number is 96.5 which is below median.
#Third quarantile number is 180 which is greater than median.
```

```
#Max number is 335 (the black line above the box).
#There is one outlier in the dataset(circle above the max line).
```

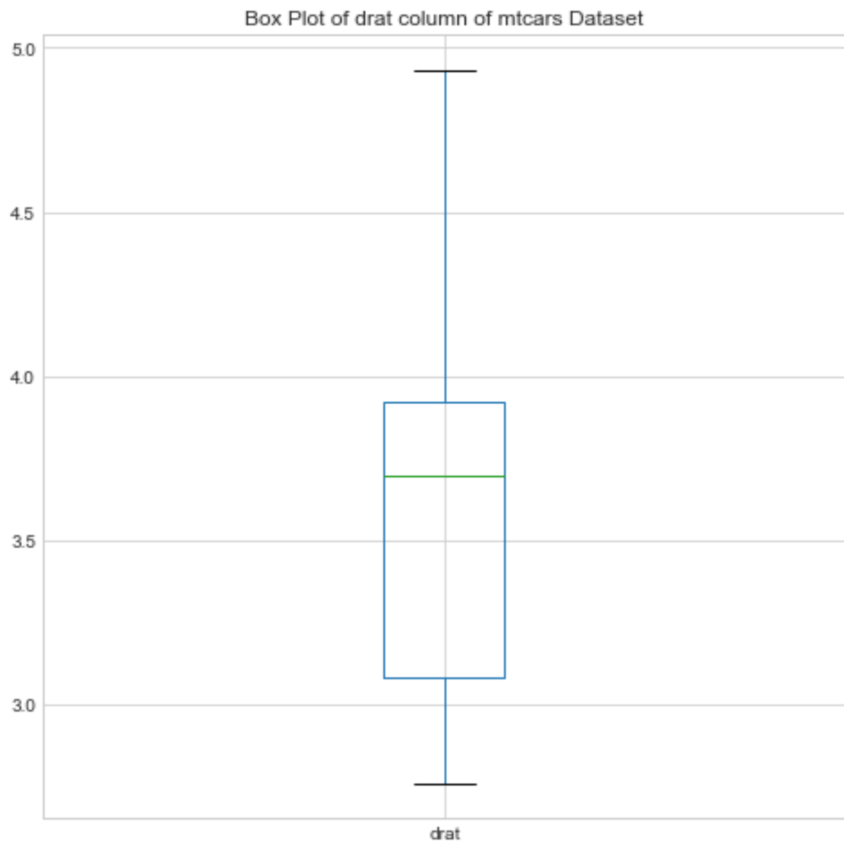
Out[114]:

```
Text(0.5, 1.0, 'Box Plot of hp column of mtcars Dataset')
```



In [115]:
```
#Question 4: Plot boxplots for all numerical columns in the dataset and provide you
#comments on what you observe.
mtcars.boxplot(column="drat",
 return_type='axes',
 figsize=(8,8)
)
plt.title('Box Plot of drat column of mtcars Dataset')
#Comments on Box-plot
#A boxplot describes summary of five numbers of a dataset.
#These are minimum, first quartile, median, third quartile, and maximum numbers.
#Min number is 2.76 (the black line below the box).
#Median is 3.695 (the middle grean line in the box).
#First quarantile number is 3.08 which is below median.
#Third quarantile number is 3.92 which is greater than median.
#Max number is 4.93 (the black line above the box).
#There is no outliers.
```

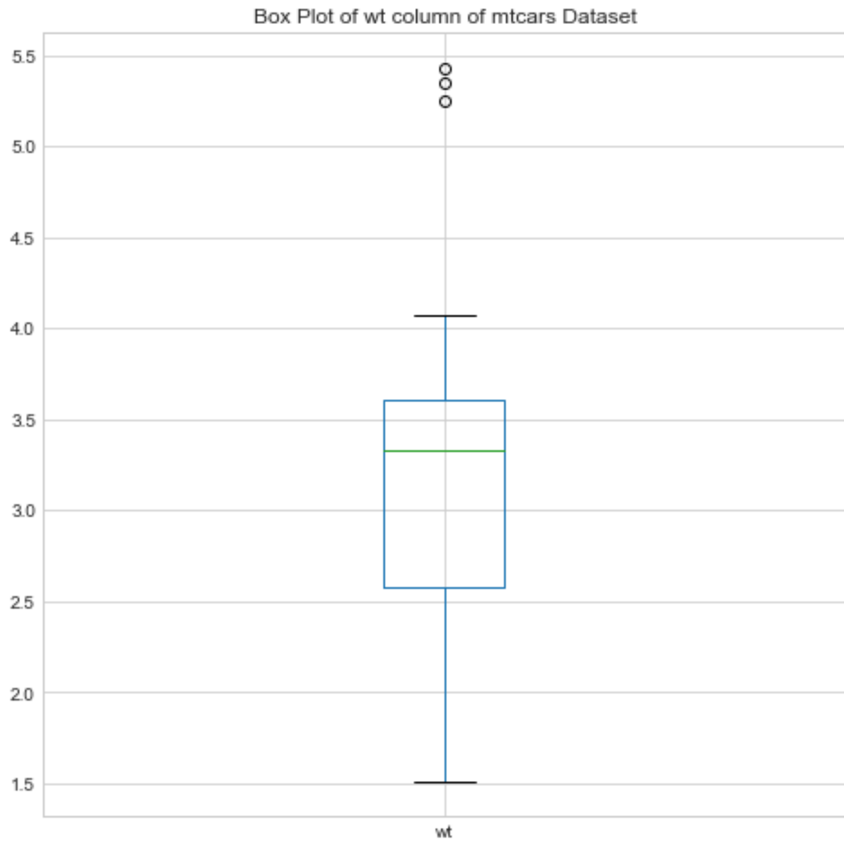
Out[115]: Text(0.5, 1.0, 'Box Plot of drat column of mtcars Dataset')



In [116]:
```
#Question 4: Plot boxplots for all numerical columns in the dataset and provide you
#comments on what you observe.
mtcars.boxplot(column="wt",
 return_type='axes',
 figsize=(8,8)
)
plt.title('Box Plot of wt column of mtcars Dataset')
#Comments on Box-plot
#A boxplot describes summary of five numbers of a dataset.
#These are minimum, first quartile, median, third quartile, and maximum numbers.
#Min number is 1.51 (the black line below the box).
#Median is 3.325 (the middle grean line in the box).
#First quarantile number is 2.58 which is below median.
#Third quarantile number is 3.61 which is greater than median.
```

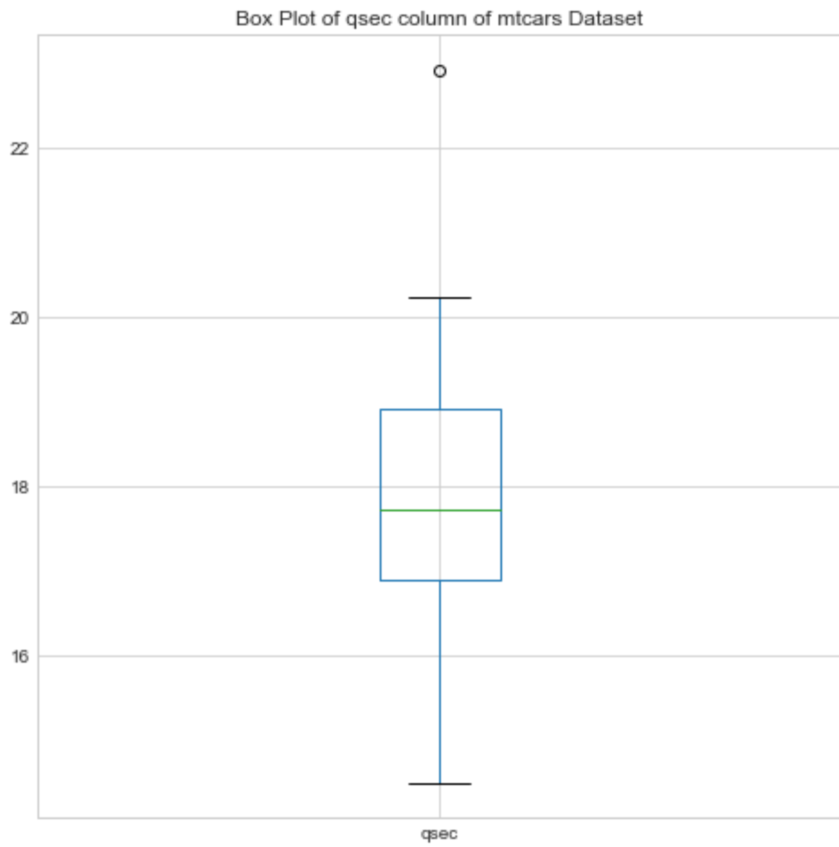
```
#Max number is 5.42 (the black line above the box).
#There are three outliers in the dataset(circle above the max line).
```

Out[116]: Text(0.5, 1.0, 'Box Plot of wt column of mtcars Dataset')



In [117]:
```
#Question 4: Plot boxplots for all numerical columns in the dataset and provide you
#comments on what you observe.
mtcars.boxplot(column="qsec",
 return_type='axes',
 figsize=(8,8)
)
plt.title('Box Plot of qsec column of mtcars Dataset')
#Comments on Box-plot
#A boxplot describes summary of five numbers of a dataset.
#These are minimum, first quartile, median, third quartile, and maximum numbers.
#Min number is 14.5 (the black line below the box).
#Median is 17.7 (the middle grean line in the box).
#First quarantile number is 16.89 which is below median.
#Third quarantile number is 18.9 which is greater than median.
#Max number is 22.9 (the black line above the box).
#There is one outlier in the dataset(circle above the max line).
```

Out[117]: Text(0.5, 1.0, 'Box Plot of qsec column of mtcars Dataset')

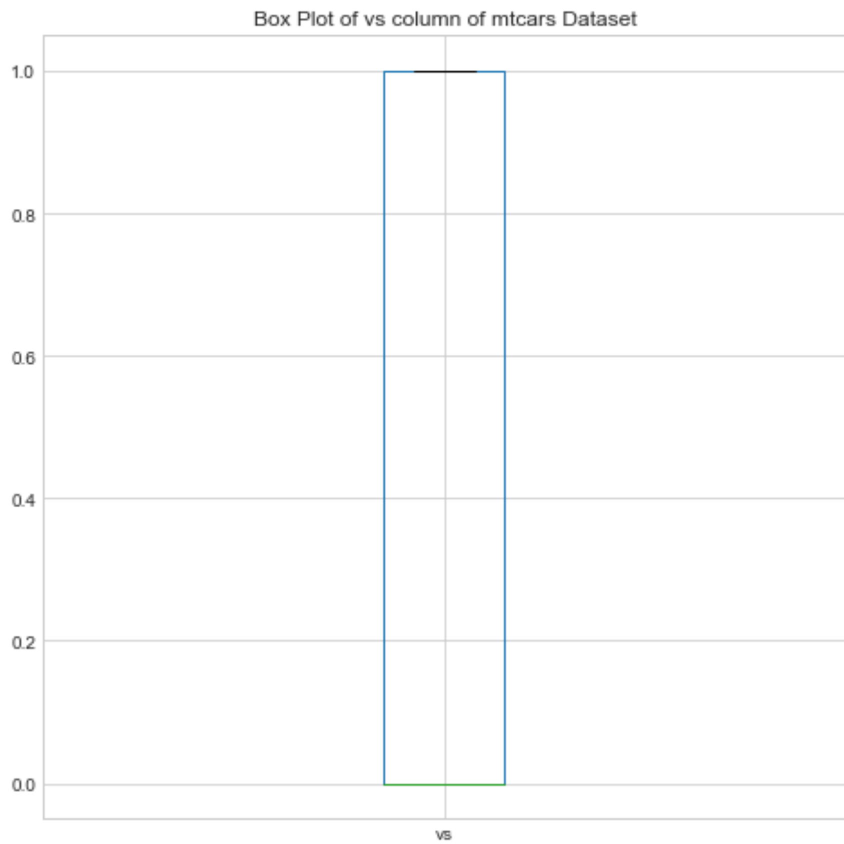


In [118]:
```
#Question 4: Plot boxplots for all numerical columns in the dataset and provide you
#comments on what you observe.
mtcars.boxplot(column="vs",
 return_type='axes',
 figsize=(8,8)
)
plt.title('Box Plot of vs column of mtcars Dataset')
#Comments on Box-plot
#A boxplot describes summary of five numbers of a dataset.
#These are minimum, first quartile, median, third quartile, and maximum numbers.
#Min number is 0.
#Median is 0 (grean line in the box).
#First quarantile number is 0.
#Third quarantile number is 1 which is greater than median.
```

```
#Max number is 1(the black line above the box).
```

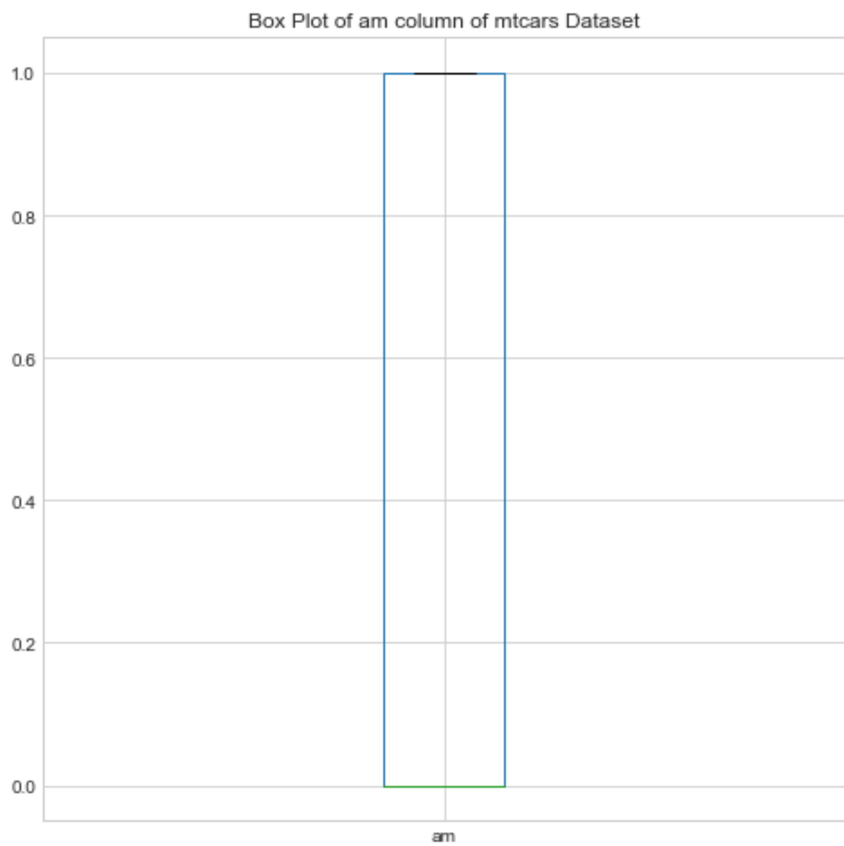
Out[118]:

```
Text(0.5, 1.0, 'Box Plot of vs column of mtcars Dataset')
```



In [119]:
```
#Question 4: Plot boxplots for all numerical columns in the dataset and provide you
#comments on what you observe.
mtcars.boxplot(column="am",
 return_type='axes',
 figsize=(8,8)
)
plt.title('Box Plot of am column of mtcars Dataset')
#Comments on Box-plot
#A boxplot describes summary of five numbers of a dataset.
#These are minimum, first quartile, median, third quartile, and maximum numbers.
#Min number is 0.
#Median is 0 (grean line in the box).
#First quarantile number is 0.
#Third quarantile number is 1 which is greater than median.
#Max number is 1(the black line above the box).
```

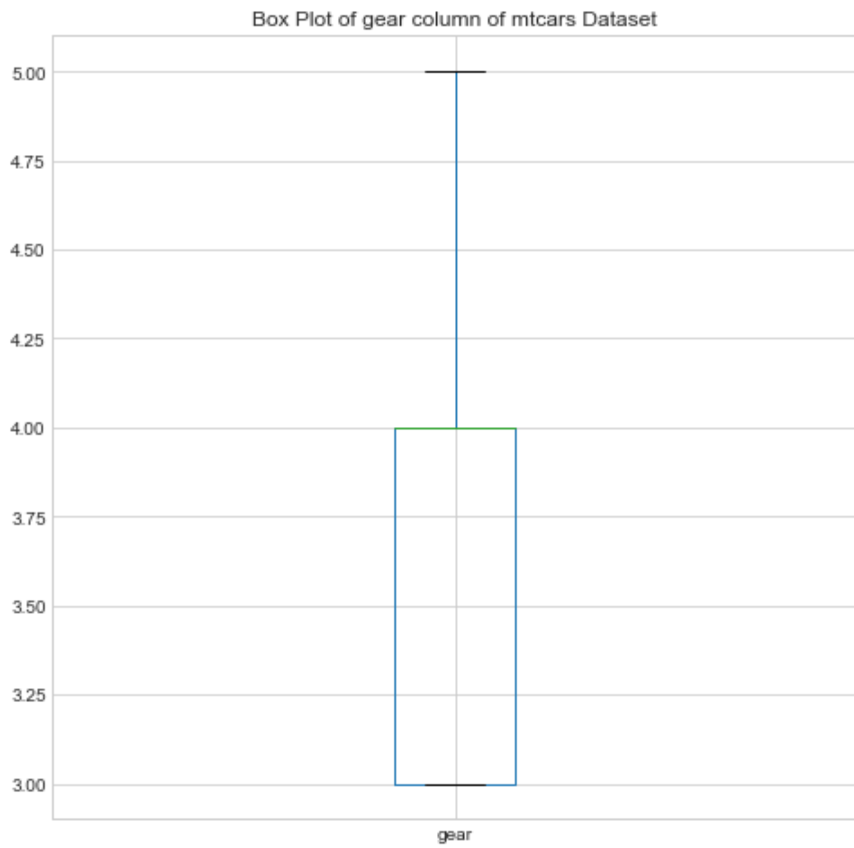
Out[119]:
```
Text(0.5, 1.0, 'Box Plot of am column of mtcars Dataset')
```



In [120]:
```
#Question 4: Plot boxplots for all numerical columns in the dataset and provide you
#comments on what you observe.
mtcars.boxplot(column="gear",
 return_type='axes',
 figsize=(8,8)
)
plt.title('Box Plot of gear column of mtcars Dataset')
#Comments on Box-plot
#A boxplot describes summary of five numbers of a dataset.
#These are minimum, first quartile, median, third quartile, and maximum numbers.
#Min number is 3 (the black line below the box).
#Median is 4 (grean up in the box).
#First quarantile number is 3 which is below median.
#Third quarantile number is 4 which is greater than median.
#Max number is 5(the black line above the box).
```

```
#No outliers
```

Out[120]: Text(0.5, 1.0, 'Box Plot of gear column of mtcars Dataset')



In [121]:
```
#Question 4: Plot boxplots for all numerical columns in the dataset and provide you
#comments on what you observe.
mtcars.boxplot(column="carb",
 return_type='axes',
 figsize=(8,8)
)
plt.title('Box Plot of carb column of mtcars Dataset')
#Comments on Box-plot
#A boxplot describes summary of five numbers of a dataset.
#These are minimum, first quartile, median, third quartile, and maximum numbers.
#Min number is 1(the black line below the box).
#Median is 2(grean line below in the box).
#First quarantile number is 2 which is below median.
#Third quarantile number is 4 which is greater than median.
#Max number is 8(the black line above the box).
#There is one outlier in the dataset(circle above the max line).
```

Out[121]:
```
Text(0.5, 1.0, 'Box Plot of carb column of mtcars Dataset')
```