Description:

Honour killings in Pakistan are known locally as karo-kari (Urdu: کاروکاری .(Pakistan has the highest number of documented and estimated honour killings per capita of any country in the world; about one-fifth of the world's honour killings are performed in Pakistan

Question 1

The dataset contains reported cases from 2011 to 2019. Your task is to Visualize the data in best possible manner. For example showing a plot of districts with most number of cases. Note that this is one example, you need to be creative.

Answer

> **First we converted the file honour-killing-reported-cases-2011-2019(1).xslx into .csv file. We can work with csv files using pandas library.**

Step 1 : Getting the Data

In [13]:
```
import pandas as pd

df=pd.read_csv("honour-killing-reported-cases-2011-2019(1).csv")

# Reading the csv File => Making sure we got the data
df.head()
```

Out[13]:

	Districts	2011	2012	2013	2014	2015	2016	2017	2018	2019
0	Lahore	10	15	11	24	13	7	9	10	12
1	Sheikhupura	14	13	5	17	11	11	9	10	5
2	Nankana Sahib	13	10	10	15	14	0	2	1	7
3	Kasur	10	13	10	10	13	13	8	7	6
4	Region Total	37	36	25	42	38	24	19	18	18

Step 2: Exploring the data

In [14]:
```
# Checking the types of data
print(df.dtypes)
df.info()
df.describe()
```

```
Districts    object
2011          int64
2012          int64
2013          int64
2014          int64
2015          int64
2016          int64
2017          int64
2018          int64
2019          int64
dtype: object
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46 entries, 0 to 45
Data columns (total 10 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   Districts  46 non-null     object
 1   2011       46 non-null     int64 
 2   2012       46 non-null     int64 
 3   2013       46 non-null     int64 
 4   2014       46 non-null     int64 
 5   2015       46 non-null     int64 
 6   2016       46 non-null     int64 
 7   2017       46 non-null     int64 
 8   2018       46 non-null     int64 
 9   2019       46 non-null     int64 
dtypes: int64(9), object(1)
memory usage: 3.7+ KB
```

Out[14]:

	2011	2012	2013	2014	2015	2016	2017	2018	2019
count	46.000000	46.000000	46.000000	46.000000	46.000000	46.000000	46.000000	46.000000	46.000000
mean	23.521739	23.543478	25.065217	25.826087	21.108696	16.021739	11.608696	15.695652	12.586957
std	53.626999	53.757359	57.448490	59.438971	48.782615	36.895341	26.767707	35.767067	29.178817
min	1.000000	0.000000	1.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000
25%	5.000000	7.000000	6.250000	6.250000	4.000000	3.250000	2.000000	4.000000	3.000000
50%	11.000000	11.000000	10.500000	12.000000	9.000000	7.500000	6.500000	8.000000	6.000000
75%	20.750000	19.000000	25.750000	24.000000	18.750000	13.750000	11.750000	14.000000	10.750000
max	364.000000	366.000000	388.000000	404.000000	328.000000	248.000000	181.000000	244.000000	197.000000

Hence, our dataset has int values..

Step 3: Visualizing our data

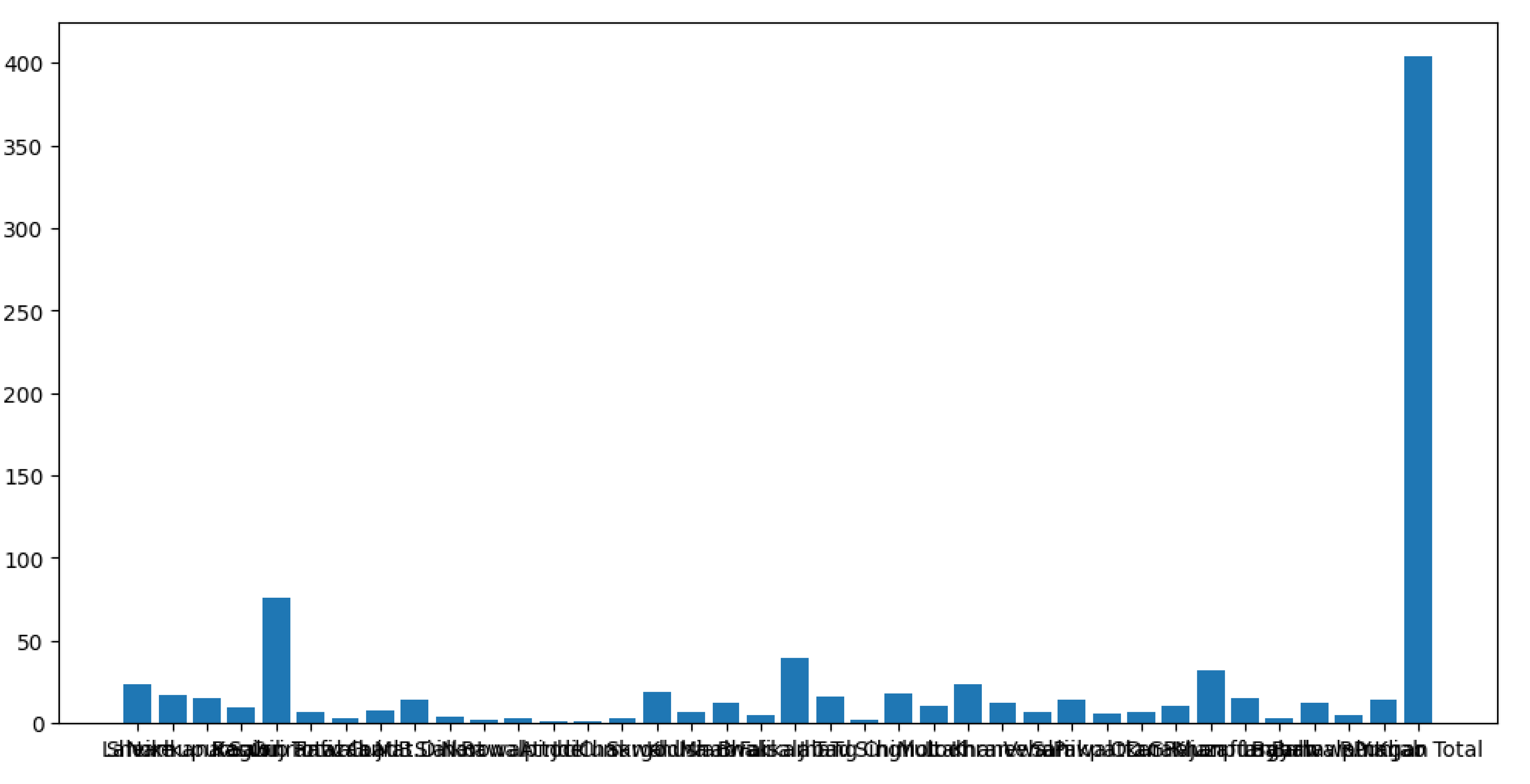
1. Matplot library is used to make plots from dataset. so, first we will import the required library.
2. Seaborn provides a high-level interface for drawing attractive and informative statistical graphics. So, we will also import the required library.

In [15]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

Bar chart

In [16]:
```
fig=plt.figure(figsize=(12,6), dpi= 100, facecolor='w', edgecolor='k')
plt.bar(df["Districts"],height=df['2014'],align="center")
```

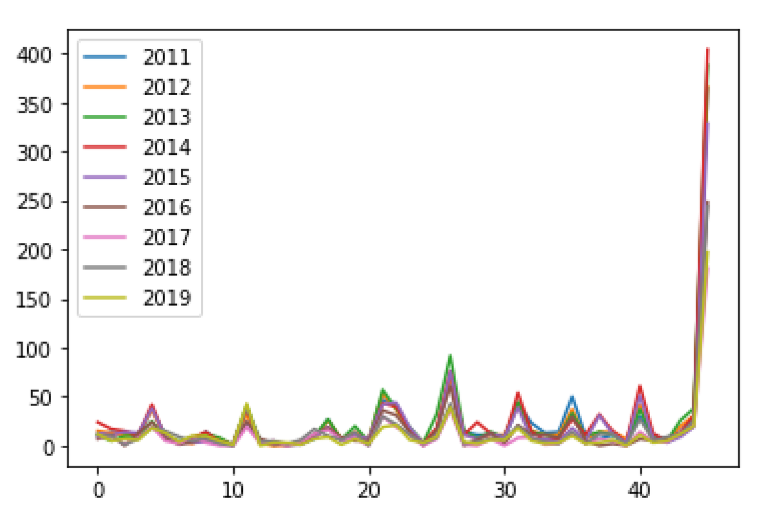
Out[16]: <BarContainer object of 46 artists>



Trending Plot

In [18]:
```
df.plot()
```

Out[18]: <AxesSubplot:>

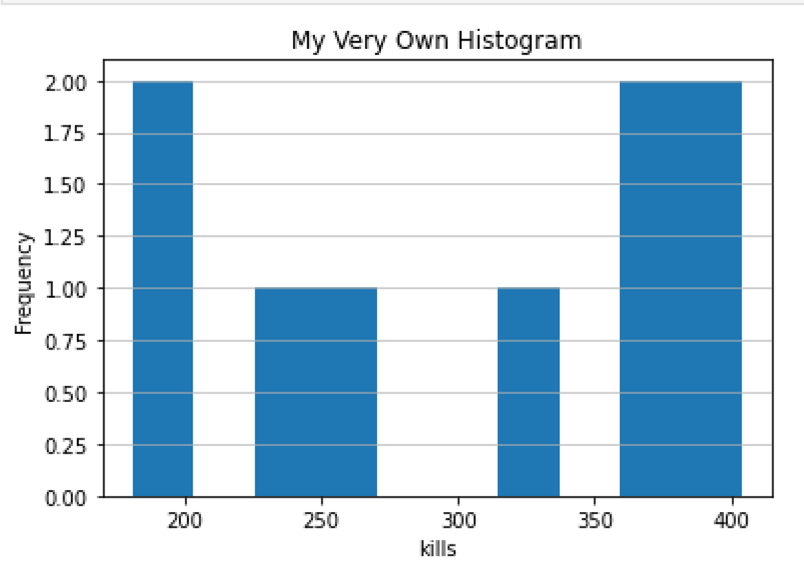


Frequency graph

In [41]:
```
import matplotlib.pyplot as plt
need_values=total_values[1:]

# An "interface" to matplotlib.axes.Axes.hist() method

n, bins, patches = plt.hist(x=need_values)
plt.grid(axis='y', alpha=0.75)
plt.xlabel('kills')
plt.ylabel('Frequency')
plt.title('My Very Own Histogram')
maxfreq = n.max()
```



More Exploration of the data

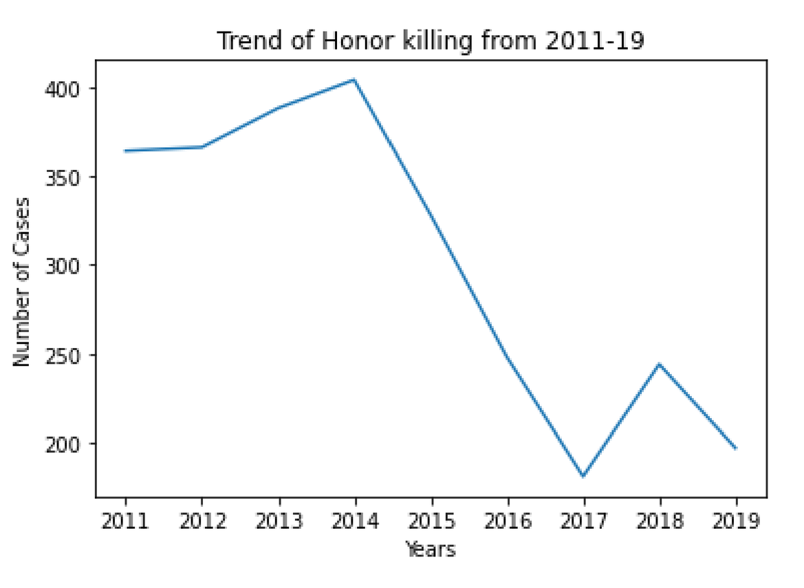
In [19]:
```
total_values = df.max()
total_values
```

Out[19]:
```
Districts    Vehari
2011            364
2012            366
2013            388
2014            404
2015            328
2016            248
2017            181
2018            244
2019            197
dtype: object
```

Plots to see the increase in honour killings over 9 years

In [20]:
```
need_values=total_values[1:]
plt.plot(need_values)
plt.xlabel("Years")
plt.ylabel("Number of Cases")
plt.title("Trend of Honor killing from 2011-19")
```

Out[20]: Text(0.5, 1.0, 'Trend of Honor killing from 2011-19')



In [21]:
```
# Maximum kills during 2011-19 and the corresponding Year
dict_of_values=dict(need_values)
years=list(dict_of_values.keys())
# print(years)
values=list(dict_of_values.values())
# print(values)
# print(list_of_values)
print(f"Maximum Honor killings = {need_values.max()}, during the year {years[values.index(404)]}")
```

```
Maximum Honor killings = 404, during the year 2014
```

Overal Data Grapgh

Creating different Models and thier Visualization

In [22]:
```
# import sys
# !{sys.executable} -m pip install statsmodel

# imports
import pandas as pd
import seaborn as sns
# import statsmodels.formula.api as smf
from sklearn.linear_model import LinearRegression
from sklearn import metrics
import numpy as np

# allow plots to appear directly in the notebook
%matplotlib inline
```

In [23]:
```
data=pd.read_csv("CleanData.csv",index_col=0)
data.head()
```

Out[23]:

	2012	2013	2014	2015	2016	2017	2018	2019	2020	Total kills
2011										
10	15	11	24	13	7	9	10	12	3	114
14	13	5	17	11	11	9	10	5	8	103
13	10	10	15	14	0	2	1	7	8	80
10	13	10	10	13	13	8	7	6	5	95
11	9	7	7	9	8	5	15	13	13	97

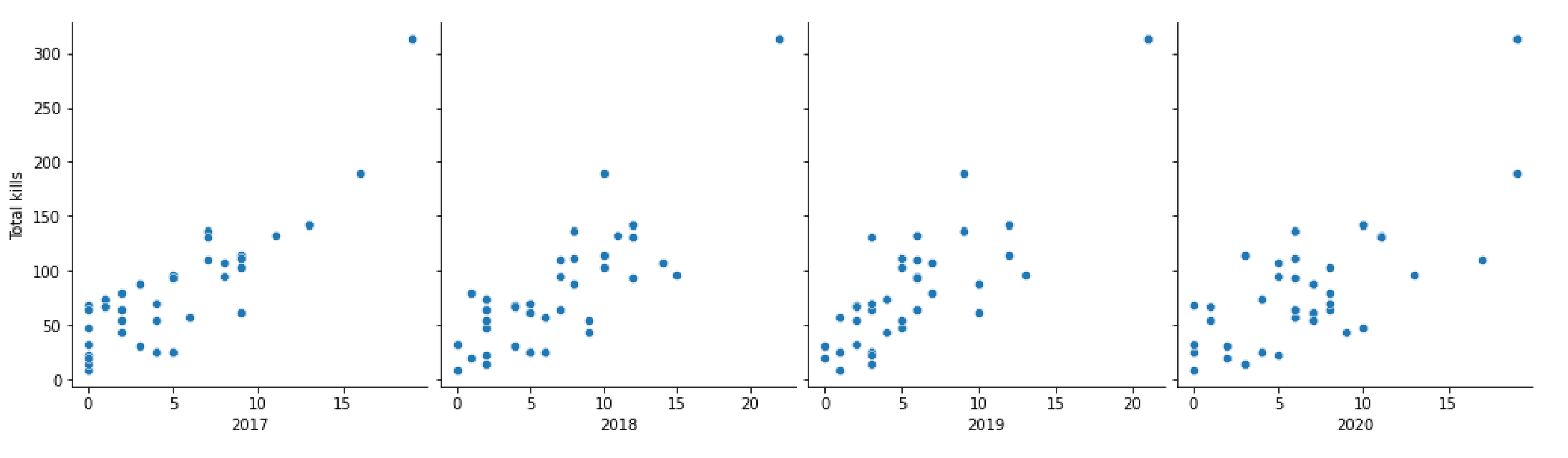
In [24]:
```
data.shape
```

Out[24]: (36, 10)

In [25]:
```
sns.pairplot(data, x_vars=["2017","2018","2019","2020"], y_vars='Total kills', size=4, aspect=0.9)
```

```
/home/jawad/.local/lib/python3.8/site-packages/seaborn/axisgrid.py:1969: UserWarning: The `size` parameter has been renamed to `height`; please updat
e your code.
  warnings.warn(msg, UserWarning)
```

Out[25]: <seaborn.axisgrid.PairGrid at 0x7f1fb3660400>



Question 2

> Get reported cases from 2019 to 2021 (from any source, this can be only 10 rows), handle it as a regression problem and use the model built on previous data. The task should be to predict the district wise total number of cases. Compare your machine learning models performance on unseen data.

Step 1: Obtaining Dataset

We use data from Pujab police data from 2011-2020.
There was no data for this Year

Step 2: Data Processing for Multiple Regression Algorithm

We pre-processed the data and formed the file CleanData.csv which we will use for trainig our Multiregression algorithm and test its performance in the following section.

> **We are using MultiRegression Algorithm because the data is for several years i.e 2011-2020.**

In [26]:
```
#import the libraries
import numpy as py
import matplotlib.pyplot as plt
import pandas as pd

#Import the data set from Desktop
dataset = pd.read_csv('CleanData.csv')
dataset.head()

X=dataset.iloc[:,:-1].values
y=dataset.iloc[:,10].values

#Training and Testing Data (divide the data into two part)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =train_test_split(X,y,test_size=0.30, random_state=40)
# We are using 30 % for testing and 70% for training our Algo.

#regression
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X_train,y_train)

#for predict the test values
y_prdict=reg.predict(X_test)
print(y_prdict);
print('\nIntercept: ',reg.intercept_)
```

```
[ 80. 112.  26. 107.  97.  43.  74.  20.  23. 131.  94.]

 Intercept:  -1.4210854715202004e-14
```

Step 3: Algorithm Evluation

There are many ways to evaluate model performance but in classical statistics, the performance of linear regression models is evaluated with R^2 — which gives a value between 0 and 1, and the higher the R^2 the better the model.

In [27]:
```
from sklearn.metrics import r2_score
from sklearn.metrics import *
import numpy as np
score=r2_score(y_test,y_prdict)
print(score)

# Our Algorithm gives 100% accurate results
# Evaluating the model using MAE Evaluation Metric
print(f"mean_squared_error is {mean_absolute_error(y_test, y_prdict)}")
# Evaluating the model using RMSE Evaluation Metric
print(f"root_mean_squared_error is {np.sqrt(mean_absolute_error(y_test, y_prdict))}")
```

```
1.0
mean_squared_error is 2.5514943693203597e-14
root_mean_squared_error is 1.5973397789200517e-07
```

The End

In [ ]: