Data Analysis and Visualisation

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
```

In [2]:
```
data=pd.read_csv("C:/Users/Akruti/Downloads/ESE_2.csv")
```

In [3]:
```
#(Basic description of data)
```

In [4]:
```
data.head()
```

Out[4]:

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

In [5]:
```
data.tail()
```

Out[5]:

	R&D Spend	Administration	Marketing Spend	State	Profit
995	54135.00	118451.999	173232.6695	California	95279.96251
996	134970.00	130390.080	329204.0228	California	164336.60550
997	100275.47	241926.310	227142.8200	California	413956.48000
998	128456.23	321652.140	281692.3200	California	333962.19000
999	161181.72	270939.860	295442.1700	New York	476485.43000

In [6]:
```
data.describe()
```

Out[6]:

	R&D Spend	Administration	Marketing Spend	Profit
count	1000.000000	1000.000000	1000.000000	1000.000000
mean	81668.927200	122963.897612	226205.058419	119546.164656
std	46537.567891	12613.927535	91578.393542	42888.633848
min	0.000000	51283.140000	0.000000	14681.400000
25%	43084.500000	116640.684850	150969.584600	85943.198543
50%	79936.000000	122421.612150	224517.887350	117641.466300
75%	124565.500000	129139.118000	308189.808525	155577.107425
max	165349.200000	321652.140000	471784.100000	476485.430000

In [10]:
```
data.nunique()
```

Out[10]:
```
R&D Spend          997
Administration     998
Marketing Spend    996
State                3
Profit             998
dtype: int64
```

In [17]:
```
data.shape
```

Out[17]:
```
(1000, 5)
```

In [18]:
```
data.describe()
```

Out[18]:

	R&D Spend	Administration	Marketing Spend	Profit
count	1000.000000	1000.000000	1000.000000	1000.000000
mean	81668.927200	122963.897612	226205.058419	119546.164656
std	46537.567891	12613.927535	91578.393542	42888.633848
min	0.000000	51283.140000	0.000000	14681.400000
25%	43084.500000	116640.684850	150969.584600	85943.198543
50%	79936.000000	122421.612150	224517.887350	117641.466300
75%	124565.500000	129139.118000	308189.808525	155577.107425
max	165349.200000	321652.140000	471784.100000	476485.430000

In [19]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   R&D Spend        1000 non-null   float64
 1   Administration   1000 non-null   float64
 2   Marketing Spend  1000 non-null   float64
 3   State            1000 non-null   object 
 4   Profit           1000 non-null   float64
dtypes: float64(4), object(1)
memory usage: 39.2+ KB
```

In [20]:
```
data.count()
```

Out[20]:
```
R&D Spend          1000
Administration     1000
Marketing Spend    1000
State              1000
Profit             1000
dtype: int64
```

In [21]:
```
data.columns
```

Out[21]:
```
Index(['R&D Spend', 'Administration', 'Marketing Spend', 'State', 'Profi
t'], dtype='object')
```

In [22]:
```
data.nunique()
```

Out[22]:
```
R&D Spend          997
Administration     998
Marketing Spend    996
State                3
Profit             998
dtype: int64
```

In [23]:
```
data.dtypes
```

Out[23]:
```
R&D Spend          float64
Administration     float64
Marketing Spend    float64
State               object
Profit             float64
dtype: object
```

In [24]:
```
data = data.drop_duplicates()
data.head(5)
```

Out[24]:

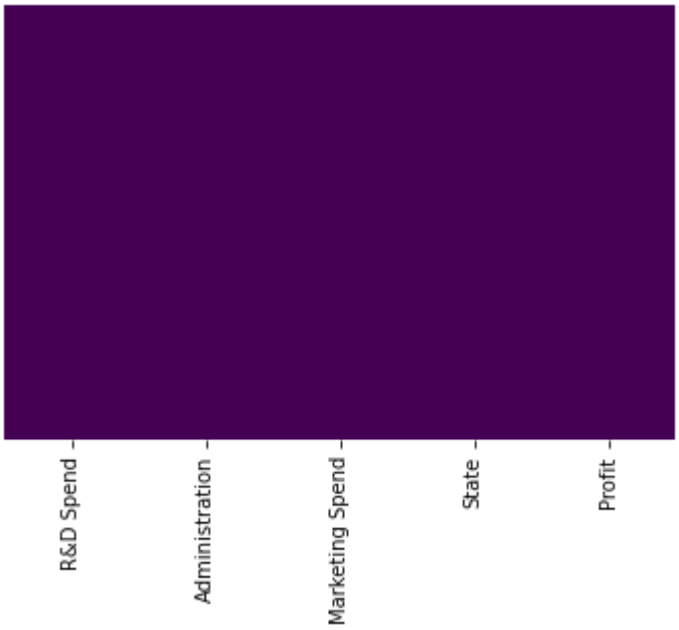
	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

In [25]:
```
print(data.isnull().sum())
```

```
R&D Spend          0
Administration     0
Marketing Spend    0
State              0
Profit             0
dtype: int64
```

In [27]:
```
import seaborn as sns
sns.heatmap(data.isnull(), cbar=False, yticklabels=False, cmap='viridis')
```

Out[27]: <AxesSubplot:>



HANDLING MISSING VALUES

In [28]:
```
data = data.dropna()
data.count()
```

Out[28]:
```
R&D Spend          999
Administration     999
Marketing Spend    999
State              999
Profit             999
dtype: int64
```

In [29]:
```
missing = data.isnull().sum(axis=0).reset_index()
missing.columns = ['column_name', 'missing_count']
missing
```

Out[29]:

	column_name	missing_count
0	R&D Spend	0
1	Administration	0
2	Marketing Spend	0
3	State	0
4	Profit	0

In [30]:
```
Q1 = data.quantile(0.25)
Q3 = data.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
R&D Spend          81512.000000
Administration     12502.014500
Marketing Spend    157757.108050
Profit             69725.937345
dtype: float64
```

In [31]:
```
data_out = data[~((data < (Q1 - 1.5 * IQR)) |(data > (Q3 + 1.5 * IQR))).any(a
print(data_out.shape)
```

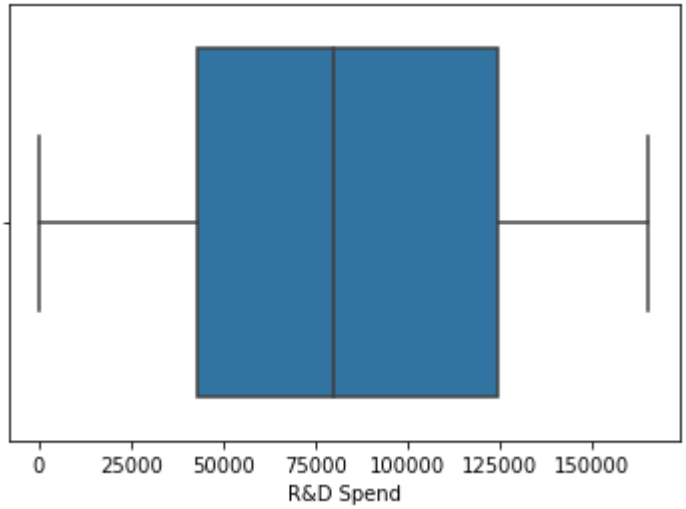
```
(977, 5)
```

In [32]:
```
lower_limit = Q1-1.5*IQR
upper_limit = Q3+1.5*IQR
lower_limit,upper_limit
```

Out[32]:
```
(R&D Spend          -79187.000000
 Administration      97886.518550
 Marketing Spend    -85907.747675
 Profit             -18679.238413
 dtype: float64,
 R&D Spend           246861.000000
 Administration      147894.576550
 Marketing Spend     545120.684525
 Profit              260224.510968
 dtype: float64)
```

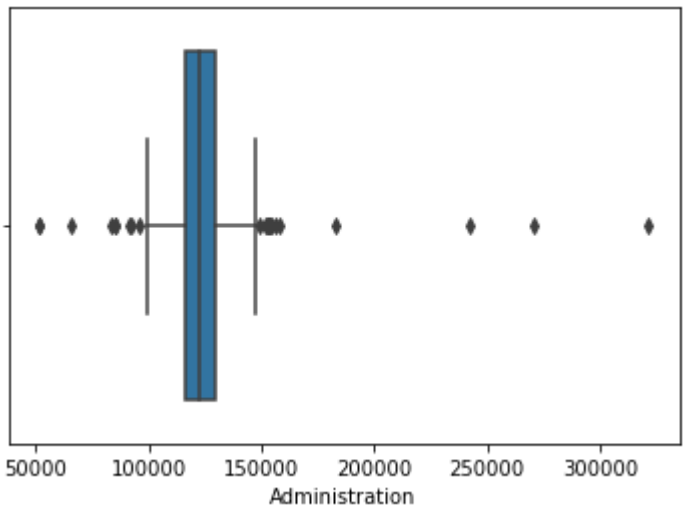
In [33]:
```
sns.boxplot(x=data['R&D Spend'])
```

Out[33]:
```
<AxesSubplot:xlabel='R&D Spend'>
```

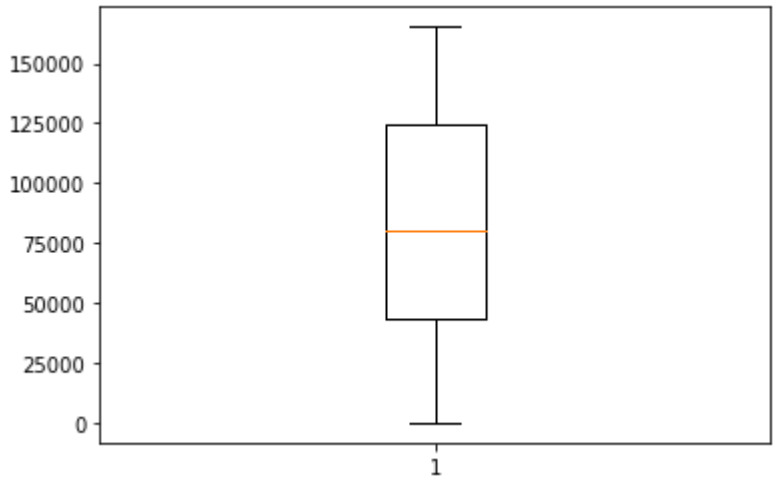


In [34]:
```
sns.boxplot(x=data['Administration'])
```

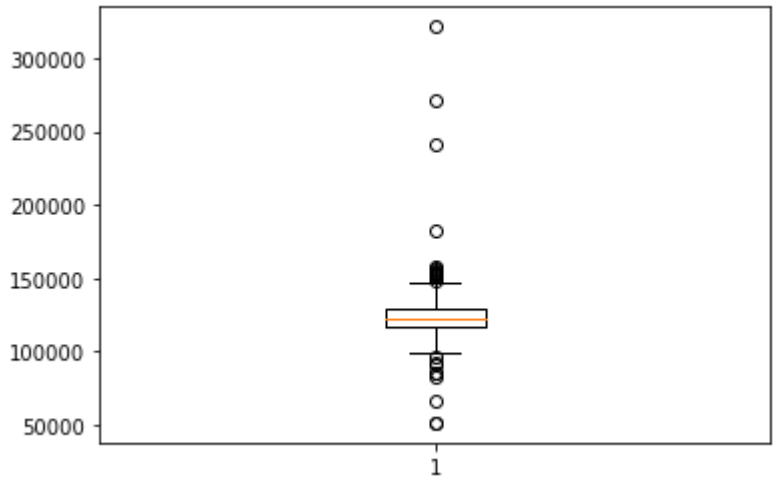
Out[34]:
```
<AxesSubplot:xlabel='Administration'>
```



In [35]:
```
import matplotlib.pyplot as plt
plt.boxplot(data["R&D Spend"])
plt.show()
```



In [36]:
```
plt.boxplot(data["Administration"])
plt.show()
```



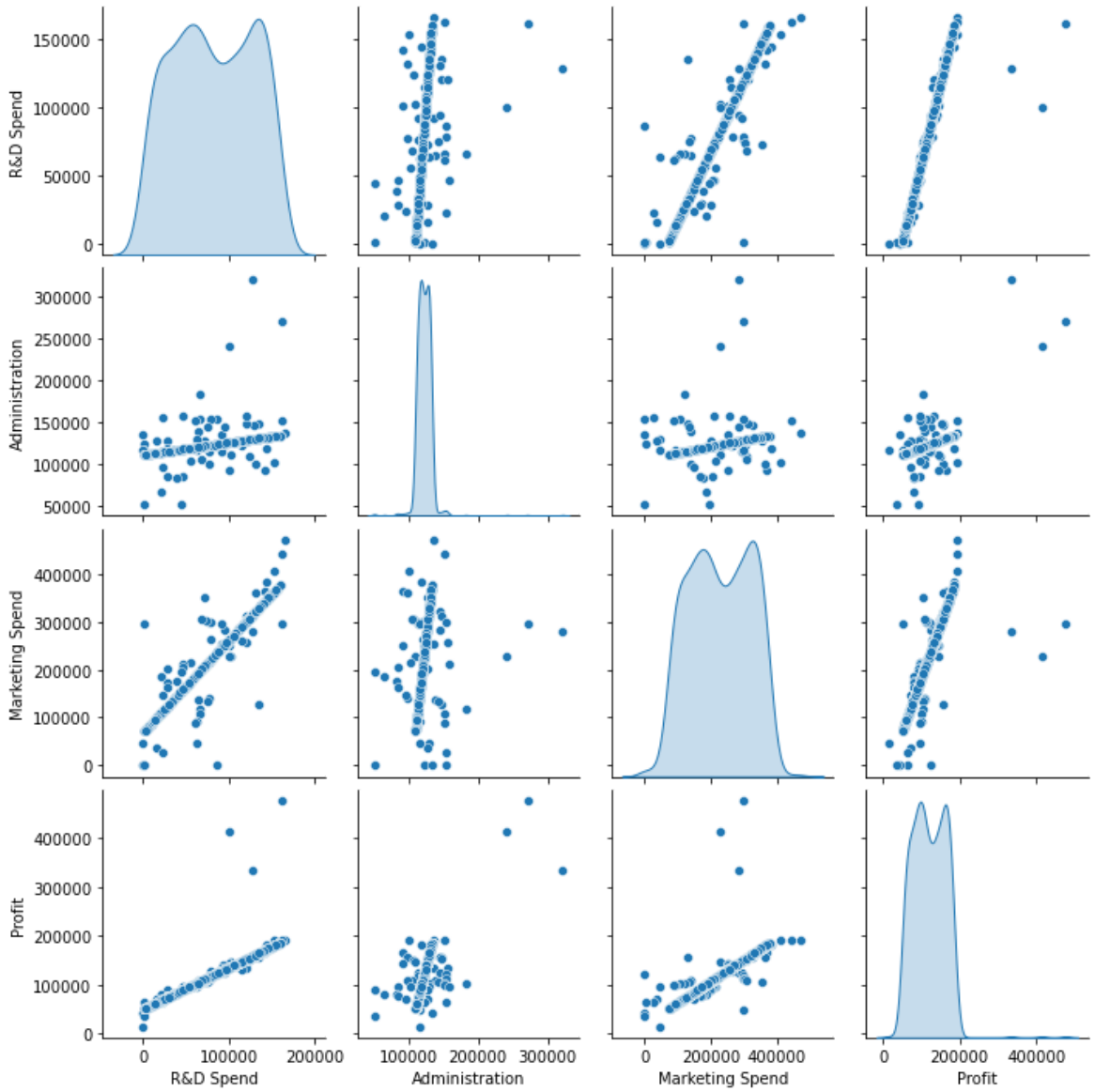
In [37]:
```
corr = data.corr()
corr.style.background_gradient(cmap='coolwarm').set_precision(2)
```

Out[37]:

	R&D Spend	Administration	Marketing Spend	Profit
R&D Spend	1.00	0.58	0.98	0.95
Administration	0.58	1.00	0.52	0.74
Marketing Spend	0.98	0.52	1.00	0.92
Profit	0.95	0.74	0.92	1.00

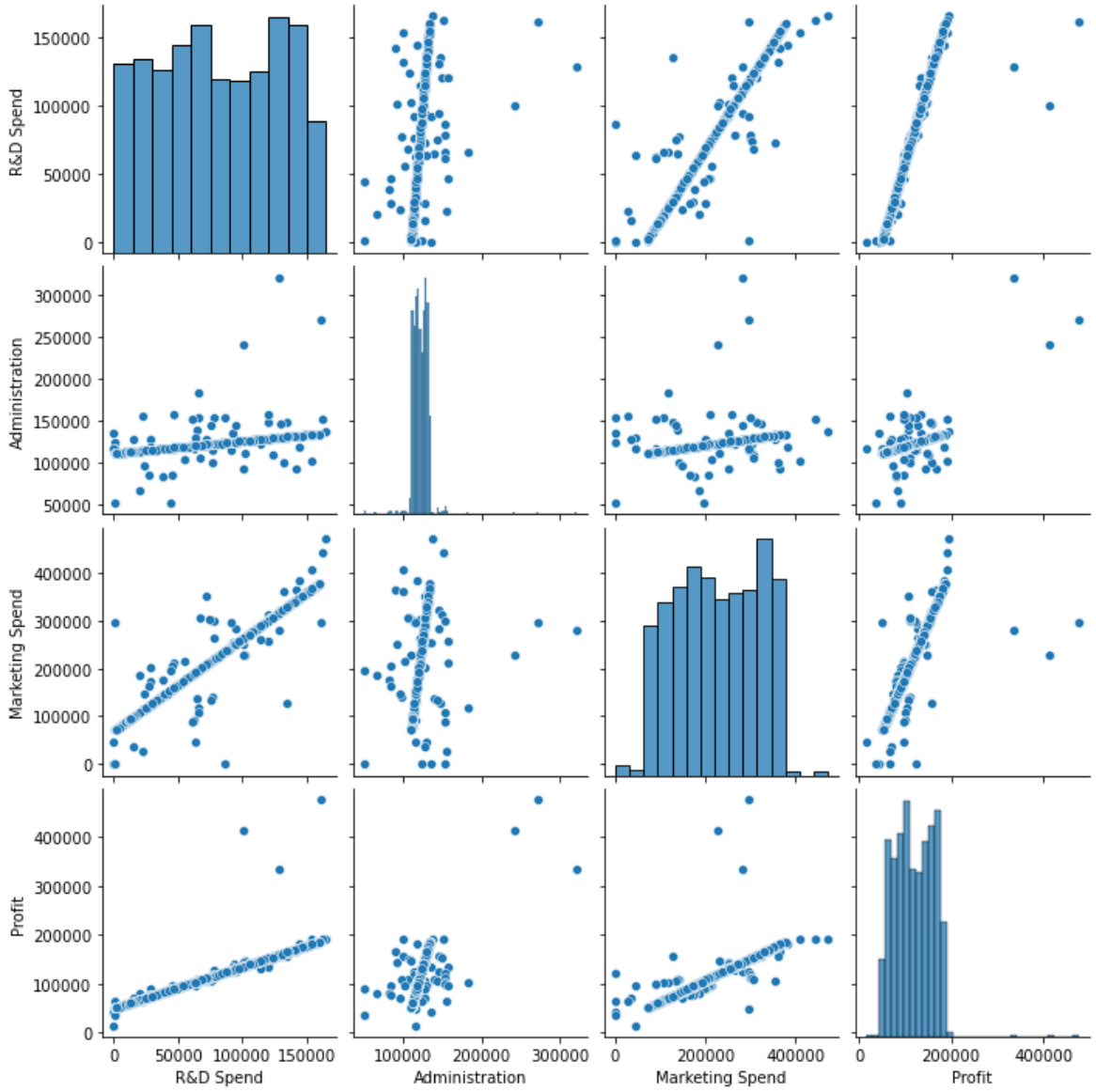
In [38]:
```
sns.pairplot(data,diag_kind='kde')
```

Out[38]: <seaborn.axisgrid.PairGrid at 0x19a244dedc0>



In [42]:
```
# Overall scatterplot for all the features
sns.pairplot(data)
```

Out[42]: <seaborn.axisgrid.PairGrid at 0x19a24f66610>



In [43]:
```
data.corr()
```

Out[43]:

	R&D Spend	Administration	Marketing Spend	Profit
R&D Spend	1.000000	0.582435	0.978406	0.945245
Administration	0.582435	1.000000	0.520466	0.741561
Marketing Spend	0.978406	0.520466	1.000000	0.917270
Profit	0.945245	0.741561	0.917270	1.000000

In [44]:
```
# Converting to categorical to numeric data for model building
data_state = pd.get_dummies(data['State'],drop_first=True)
data_state
```

Out[44]:

	Florida	New York
0	0	1
1	0	0
2	1	0
3	0	1
4	1	0
...	...	...
995	0	0
996	0	0
997	0	0
998	0	0
999	0	1

999 rows × 2 columns

In [45]:
```
data_final = pd.concat([data,data_state],axis=1)
data_final
```

Out[45]:

	R&D Spend	Administration	Marketing Spend	State	Profit	Florida	New York
0	165349.20	136897.800	471784.1000	New York	192261.83000	0	1
1	162597.70	151377.590	443898.5300	California	191792.06000	0	0
2	153441.51	101145.550	407934.5400	Florida	191050.39000	1	0
3	144372.41	118671.850	383199.6200	New York	182901.99000	0	1
4	142107.34	91391.770	366168.4200	Florida	166187.94000	1	0
...	...	...	...	...	...	...	...
995	54135.00	118451.999	173232.6695	California	95279.96251	0	0
996	134970.00	130390.080	329204.0228	California	164336.60550	0	0
997	100275.47	241926.310	227142.8200	California	413956.48000	0	0
998	128456.23	321652.140	281692.3200	California	333962.19000	0	0
999	161181.72	270939.860	295442.1700	New York	476485.43000	0	1

999 rows × 7 columns

In [46]:
```
train_test_split
x_train, x_test, y_train, y_test = train_test_split(data_final[['R&D Spend','
```

In [47]:
```
from sklearn.linear_model import LinearRegression
```

In [48]:
```
R = LinearRegression().fit(x_train, y_train)
```

In [49]:
```
R.score(x_train, y_train), R.score(x_test,y_test)
```

Out[49]: (0.9612784814607421, 0.9299668422955786)

In [50]:
```
# Predicted values
R.predict(x_test)
```

Out[50]:
```
array([169982.20286498,  80641.24700154, 129511.04153582,  92269.1960125 ,
        68933.23946026,  60186.07292562,  65520.47987953,  52601.60263338,
        72494.82613067, 101558.29835384, 173456.33234257, 162500.19521233,
       181831.46802509, 184096.34017613,  67236.26835125, 165372.62536587,
        97492.64835164,  98205.16094642, 183129.81935742, 150402.39602984,
        85053.64724659,  91137.83482228,  86314.40098733, 152544.17488462,
        97967.63059905, 106731.94064372,  19544.52153036, 111485.81782539,
       124897.63923479, 182465.22933334, 366974.98148135, 171149.51210129,
        58441.20337148,  65226.24706666, 115346.80642883,  70008.3870321 ,
       166071.15352307, 183792.9736927 , 120034.82335081,  65443.53739889,
        59512.07432934, 121654.85197919,  90279.70256862, 123581.28915474,
       106269.17108537, 120416.35074067,  51716.3748946 , 136233.30397672,
       139820.27544705, 128693.58576346,  64956.81861372, 150341.70478117,
       148856.81794255,  52230.66519788, 178992.59679799, 160060.18182775,
       140601.2863126 , 175846.69964206, 123563.05222548, 167937.75468568,
       178055.15705711, 158259.71604803, 138748.58619645,  73191.10031155,
       161058.93021204, 141181.95793953, 131534.54102802, 174109.56502902,
        92878.42776309, 160284.27787481,  87017.4438322 , 156548.20429421,
        75316.55394839, 138969.22391573, 101045.71873968, 135074.91455429,
        85273.28508548, 152552.96618758, 170145.97402836, 118730.48482257,
        79630.86638127, 174585.982843  , 130155.92200016, 163703.8801055 ,
       168997.7563952 , 120276.9267834 ,  95171.56269792,  98491.67174344,
        88887.46626521, 170599.26101873, 128087.1581199 , 184556.54370636,
       101003.19031861, 141828.86115854, 169757.61806942, 120427.45593158,
        69266.23715226, 118463.00512525, 182791.10936786,  99875.01244212,
       175000.81701259, 138205.45277136,  78429.94854431, 105513.91633906,
       123196.39140133,  89345.68420076, 128898.00919899,  69099.21017762,
        86895.40689583, 146422.59085458, 175746.66317689,  88773.43265355,
       168088.29247167, 140936.75383008, 113146.90225089, 124441.74926153,
       157444.31323044,  95303.86343896, 146604.73880486, 105601.43480494,
       138254.20646072, 118155.70418237, 169976.61909143,  97388.57334355,
       121997.83858404,  92982.53971286,  98308.90651411, 150574.35405227,
        98725.45137348, 136958.62229376,  84848.36847272, 109938.52916401,
        98512.1995907 , 180166.96898808,  59243.50121737,  59768.43465029,
       109385.40683262, 174476.50085414,  54639.03146602, 198145.50180153,
       120519.79440191, 166263.02200455,  55150.0170348 , 104763.21322864,
       185217.95052588,  18396.08554792, 185457.20438109, 185146.72023364,
       131578.36984951, 169742.49719639, 181370.4462008 , 126062.32117646,
       118955.6741712 ,  64187.91550918, 103435.74377418,  57398.55825507,
        99520.66849965,  52216.70499371,  75044.83457746, 178161.49277655,
       125987.08927645,  72340.04867107, 136350.24588805,  51246.76278956,
        85132.57542452, 142353.79459604, 156739.55970234, 171027.20016394,
       143303.20895359, 101628.43533796, 133924.49814731,  74190.1235942 ,
        66914.93995261,  63378.73806904, 117663.2732088 , 169977.47443632,
       164138.38677746, 139957.25628599,  73720.78650255, 158539.17024633,
       128247.72175388, 159808.47733334, 131699.20916945, 144581.06927118,
       133005.02031509,  91882.55062764, 109396.28802549, 109338.94406126,
        86694.1666714 ,  67923.1338475 , 139444.82332649, 107880.64647663,
        74891.7307568 ,  63930.42483386,  72282.12447861,  82697.21797761,
        69230.31341793, 150818.08554432, 105318.93842722, 111633.88942729,
        57334.40868517, 156663.39858062, 130115.1413197 ,  94684.88084789,
       127573.61822056, 126139.30070138, 145894.23603752, 139272.90234836,
       160075.33975161, 143115.89198576, 139552.59472819, 101293.14661881,
```

```
111750.38933219, 133016.80946502,  89446.61284335, 113510.37976132,
154551.09277658, 155094.76355309, 176047.7016915 , 121237.45176839,
143028.61159226,  59891.84000752, 136283.76834932,  92513.13854305,
182248.25095683,  55827.90070033, 114900.56310995,  52453.90589481,
127157.99659555,  84131.84145209, 144672.5893383 ,  95872.93178815,
172102.34773578, 162747.38502561, 102705.05542458,  68771.0021808 ,
 99405.4372781 , 136785.19011824, 167508.10485723, 141613.3183486 ,
147740.65162441,  71466.14133019, 179246.90427529, 116342.37139024,
 81196.08002271,  61835.76296398, 124189.18942223, 178880.58572268,
106226.40470107,  61748.51951346, 185333.41981327, 161405.37513812,
 91920.22209604,  60531.62545663, 174189.92888267,  80885.59587526,
 80830.85488083, 140336.03836838,  74034.45384803, 117424.36161858,
 53022.69912144, 161571.30880465, 145458.87402067,  54238.70089236,
 77115.58416155, 181722.80444598,  58771.36001868,  66111.54868683,
 95631.49121409, 126928.76869747, 184821.65849971, 180107.98177029,
 55767.17240149, 168966.71450021, 120969.11672619,  54294.29723038,
 56941.53803856,  57537.97677254,  86683.28547199,  71689.99930487,
 52077.2864769 ,  72886.56642488, 170114.56498169,  80258.67732612,
159231.13076109,  80070.46807173, 174440.57701719,  70606.29838675,
 72326.94580032,  57690.22524766,  65091.96051263, 131124.81050018,
102159.35607399,  66262.08647282,  87751.31537982, 112552.41216664,
159552.69723864, 106458.19863382, 102610.11408707, 103432.04748972,
 56785.86829239,  50417.949792  ,  93238.28287242, 115139.19968612,
127023.9851582 , 166008.13419642, 107014.19894907, 138889.67834959,
180403.88833126,  94992.79906983,  87695.13881882, 120366.11035303,
 78129.76536805, 330556.89838025, 150508.73174928, 107969.5635674 ,
172227.84298685, 153447.40148486, 171614.5356425 , 128421.31637096,
177656.57411295,  82540.0756075 , 116598.38955732,  99264.0701109 ,
143985.14350467,  55619.20054298, 134379.56971081, 162161.72329515,
114047.52587479, 132999.92529659])
```

In [ ]: