In [1]:
```python
# Importing libraries

import numpy as np
import pandas as pd
import statistics
import math
from matplotlib import pyplot as plt
import statsmodels.formula.api as smf
```

In [2]:
```python
# Reading and defining a dataframe of the csv file

df = pd.read_csv("ColoradoRiverData.csv")
df.head()
```

Out[2]:

	Date	PPT	Tmin	Tav	Tmax	delt	SMI	ET	Flow_cfs	Flow?
0	1988-03	16.09	2.8	12.3	21.8	19.0	197.907	31.429845	2.68	1
1	1988-04	17.63	7.9	17.0	26.1	18.2	299.710	61.427057	2.33	1
2	1988-05	63.44	14.1	21.4	28.7	14.6	1357.616	104.931753	4.56	1
3	1988-06	58.01	18.6	25.8	33.0	14.4	1496.658	149.374517	18.20	1
4	1988-07	91.98	20.5	26.6	32.7	12.2	2446.668	161.751506	76.10	1

In [3]:
```python
# Using describe function to obtain important statistical information and basic information

df.describe()
```

Out[3]:

	PPT	Tmin	Tav	Tmax	delt	SMI	ET	Flow_cfs	Flow?
count	373.000000	373.000000	373.000000	373.000000	373.000000	373.000000	373.000000	373.000000	373.000000
mean	42.483217	10.456568	17.771046	25.087668	14.631099	862.258668	80.490330	13.153506	0.678284
std	42.894824	8.182669	7.896588	7.715736	1.889885	1014.255443	63.165859	51.729368	0.467762
min	0.000000	-4.600000	3.300000	8.800000	9.900000	0.000000	2.151330	0.000000	0.000000
25%	9.910000	2.500000	10.500000	18.200000	13.400000	126.100000	19.647938	0.000000	0.000000
50%	30.700000	10.400000	18.100000	25.900000	14.500000	428.280000	65.127493	0.550000	1.000000
75%	62.850000	18.400000	25.300000	31.800000	16.000000	1389.024000	138.843235	7.600000	1.000000
max	251.240000	24.300000	31.500000	38.800000	19.400000	5672.665000	218.125353	709.900000	1.000000

In [4]:
```python
# Finding the amount of columns and rows in dataframe

df.shape
```

Out[4]:
```
(373, 10)
```

In [5]:
```python
# using the correlation function to get the correlation

corr = df.corr(method ='pearson')
corr
```

Out[5]:

	PPT	Tmin	Tav	Tmax	delt	SMI	ET	Flow_cfs	Flow?
PPT	1.000000	0.377617	0.317593	0.249061	-0.618147	0.938977	0.289487	0.614588	0.452067
Tmin	0.377617	1.000000	0.993709	0.973441	-0.355501	0.572704	0.974184	0.151265	0.234497
Tav	0.317593	0.993709	1.000000	0.992938	-0.248663	0.526083	0.975157	0.123433	0.197389
Tmax	0.249061	0.973441	0.992938	1.000000	-0.132076	0.468836	0.962877	0.092345	0.155013
delt	-0.618147	-0.355501	-0.248663	-0.132076	1.000000	-0.565551	-0.286853	-0.277924	-0.382443
SMI	0.938977	0.572704	0.526083	0.468836	-0.565551	1.000000	0.502235	0.575503	0.423881
ET	0.289487	0.974184	0.975157	0.962877	-0.286853	0.502235	1.000000	0.101266	0.184756
Flow_cfs	0.614588	0.151265	0.123433	0.092345	-0.277924	0.575503	0.101266	1.000000	0.175355
Flow?	0.452067	0.234497	0.197389	0.155013	-0.382443	0.423881	0.184756	0.175355	1.000000

In [6]:
```python
# Determining how many rows of data have a flow value of 0 and move them into its own dataframe

NoFlowperiods = df[df["Flow?"] < 0.5]
NoFlowperiods
```

Out[6]:

	Date	PPT	Tmin	Tav	Tmax	delt	SMI	ET	Flow_cfs	Flow?
19	1989-10	6.50	11.0	19.4	27.7	16.7	126.100	71.323279	0.0	0
20	1989-11	0.83	4.0	12.6	21.3	17.3	10.458	28.028436	0.0	0
21	1989-12	6.00	-4.6	3.3	11.2	15.8	19.800	2.151330	0.0	0
27	1990-06	11.21	21.9	29.3	36.7	14.8	328.453	190.533113	0.0	0
33	1990-12	18.24	-2.0	5.8	13.6	15.6	105.792	6.327737	0.0	0
...	...	...	...	...	...	...	...	...	...	...
359	2018-02	8.06	-0.1	8.5	17.0	17.1	68.510	12.878851	0.0	0
360	2018-03	13.15	6.5	14.8	23.1	16.6	194.620	44.778675	0.0	0
361	2018-04	0.69	6.9	16.4	26.0	19.1	11.316	57.346363	0.0	0
370	2019-01	6.00	-0.1	6.8	13.8	13.9	40.800	8.740291	0.0	0
371	2019-02	3.71	1.6	9.4	17.2	15.6	34.874	15.613347	0.0	0

120 rows × 10 columns

In [7]:
```python
# Using the describe function to view basic and statistical information of the new dataframe of rows only with non-flowperiods

NoFlowperiods.describe()
```

Out[7]:

	PPT	Tmin	Tav	Tmax	delt	SMI	ET	Flow_cfs	Flow?
count	120.000000	120.000000	120.000000	120.000000	120.000000	120.000000	120.000000	120.0	120.0
mean	14.364583	7.674167	15.510833	23.353333	15.679167	238.843025	63.567709	0.0	0.0
std	17.226562	8.205547	8.041675	7.948986	1.549947	344.112492	64.449411	0.0	0.0
min	0.000000	-4.600000	3.300000	10.700000	12.200000	0.000000	2.151330	0.0	0.0
25%	2.380000	0.575000	8.575000	16.650000	14.475000	22.949500	13.431416	0.0	0.0
50%	8.690000	5.350000	13.350000	21.900000	15.650000	103.686000	31.725971	0.0	0.0
75%	20.670000	13.225000	21.025000	28.875000	16.700000	335.676750	85.466834	0.0	0.0
max	104.370000	24.300000	31.500000	38.800000	19.400000	2035.215000	218.125353	0.0	0.0

In [8]:
```python
# Determining how many rows of data have a flow value of 1 and move them into its own dataframe

Flowingperiods = df[df["Flow?"] > 0.5]
Flowingperiods
```

Out[8]:

	Date	PPT	Tmin	Tav	Tmax	delt	SMI	ET	Flow_cfs	Flow?
0	1988-03	16.09	2.8	12.3	21.8	19.0	197.907	31.429845	2.68	1
1	1988-04	17.63	7.9	17.0	26.1	18.2	299.710	61.427057	2.33	1
2	1988-05	63.44	14.1	21.4	28.7	14.6	1357.616	104.931753	4.56	1
3	1988-06	58.01	18.6	25.8	33.0	14.4	1496.658	149.374517	18.20	1
4	1988-07	91.98	20.5	26.6	32.7	12.2	2446.668	161.751506	76.10	1
...	...	...	...	...	...	...	...	...	...	...
366	2018-09	77.57	17.7	23.2	28.7	11.0	1799.624	106.116280	22.20	1
367	2018-10	169.51	11.5	16.9	22.4	10.9	2864.719	54.778328	39.20	1
368	2018-11	8.10	1.8	9.1	16.3	14.5	73.710	15.039182	0.10	1
369	2018-12	30.35	0.1	6.6	13.0	12.9	200.310	8.102200	0.01	1
372	2019-03	19.36	4.3	10.9	17.5	13.2	211.024	24.942857	0.15	1

253 rows × 10 columns

In [9]:
```python
# Using the describe function to view basic and statistical information of the new dataframe of rows only with non-flowperiods

Flowingperiods.describe()
```

Out[9]:

	PPT	Tmin	Tav	Tmax	delt	SMI	ET	Flow_cfs	Flow?
count	253.000000	253.000000	253.000000	253.000000	253.000000	253.000000	253.000000	253.000000	253.0
mean	55.820119	11.776285	18.843083	25.910277	14.133992	1157.949881	88.516869	19.392323	1.0
std	44.954686	7.849178	7.610853	7.478751	1.834899	1090.784231	61.052095	61.876599	0.0
min	0.000000	-2.400000	3.500000	8.800000	9.900000	0.000000	2.453093	0.010000	1.0
25%	22.020000	4.500000	12.000000	19.600000	12.800000	270.648000	28.885725	0.480000	1.0
50%	45.180000	12.900000	20.200000	27.500000	14.000000	838.101000	90.359437	2.760000	1.0
75%	75.250000	19.400000	25.700000	32.200000	15.400000	1678.095000	147.841050	13.900000	1.0
max	251.240000	22.900000	30.100000	37.200000	19.000000	5672.665000	204.904749	709.900000	1.0

In [10]:
```python
# Highlighting rows that were recorded only in the 1990s and moving it into its own dataframe

The1990s = df.iloc[22:142]
The1990s
```

Out[10]:

	Date	PPT	Tmin	Tav	Tmax	delt	SMI	ET	Flow_cfs	Flow?
22	1990-01	16.49	0.4	8.3	16.2	15.8	136.867	12.797940	3.94	1
23	1990-02	35.33	2.4	10.6	18.9	16.5	374.498	19.647938	0.54	1
24	1990-03	47.16	5.9	12.3	18.8	12.9	580.068	31.429845	10.00	1
25	1990-04	97.88	10.1	17.4	24.7	14.6	1703.112	64.221802	51.50	1
26	1990-05	12.53	14.4	21.6	28.9	14.5	270.648	106.815862	2.32	1
...	...	...	...	...	...	...	...	...	...	...
137	1999-08	45.77	22.2	29.6	37.0	14.8	1354.792	188.740241	1.53	1
138	1999-09	42.66	16.9	23.9	30.9	14.0	1019.574	112.325867	1.20	1
139	1999-10	9.16	9.9	18.0	26.2	16.3	164.880	61.801659	0.00	0
140	1999-11	0.00	6.0	14.7	23.4	17.4	0.000	37.642092	0.00	0
141	1999-12	0.00	-0.3	7.9	16.0	16.3	0.000	11.428327	0.00	0

120 rows × 10 columns

In [11]:
```python
# Using the describe function to obtain basic and statistical information of the 1990s dataframe

The1990s.describe()
```

Out[11]:

	PPT	Tmin	Tav	Tmax	delt	SMI	ET	Flow_cfs	Flow?
count	120.000000	120.000000	120.000000	120.000000	120.000000	120.000000	120.000000	120.000000	120.000000
mean	40.184250	10.226667	17.613333	25.006667	14.780000	803.839033	78.862285	11.191917	0.616667
std	40.396408	8.051439	7.767265	7.581653	1.778282	955.286509	62.475109	32.796915	0.488237
min	0.000000	-2.300000	4.600000	10.700000	11.000000	0.000000	4.137878	0.000000	0.000000
25%	11.162500	2.375000	10.375000	18.350000	13.475000	106.964250	19.031779	0.000000	0.000000
50%	27.700000	9.950000	17.800000	25.950000	14.700000	479.093000	64.575780	0.445000	1.000000
75%	53.125000	18.000000	25.025000	31.725000	16.000000	1095.277750	137.966614	7.330000	1.000000
max	196.190000	23.600000	30.900000	38.200000	18.500000	3943.419000	215.449682	263.200000	1.000000

In [12]:
```python
# Highlighting rows that were recorded only in the 2000s and moving it into its own dataframe

The2000s = df.iloc[142:262]
The2000s
```

Out[12]:

	Date	PPT	Tmin	Tav	Tmax	delt	SMI	ET	Flow_cfs	Flow?
142	2000-01	9.72	0.3	8.5	16.7	16.4	82.620	13.394391	0.00	0
143	2000-02	5.82	3.5	12.4	21.4	17.9	72.168	26.523347	0.00	0
144	2000-03	105.42	6.6	14.7	22.8	16.2	1549.674	44.201641	51.20	1
145	2000-04	8.34	10.6	18.9	27.3	16.7	157.626	75.229168	0.37	1
146	2000-05	17.53	17.7	25.6	33.4	15.7	448.768	147.841050	2.51	1
...	...	...	...	...	...	...	...	...	...	...
257	2009-08	30.93	20.9	28.2	35.6	14.7	872.226	172.031597	0.27	1
258	2009-09	57.72	16.0	22.5	29.0	13.0	1298.700	100.075445	12.70	1
259	2009-10	25.80	9.5	16.6	23.6	14.1	428.280	52.933136	0.00	0
260	2009-11	0.04	4.6	13.3	22.0	17.4	0.532	31.082804	0.00	0
261	2009-12	52.53	-2.4	4.7	11.8	14.2	246.891	4.231809	0.04	1

120 rows × 10 columns

In [13]:
```python
# Using the describe function to obtain basic and statistical information of the 2000s dataframe

The2000s.describe()
```

Out[13]:

	PPT	Tmin	Tav	Tmax	delt	SMI	ET	Flow_cfs	Flow?
count	120.000000	120.000000	120.000000	120.000000	120.000000	120.000000	120.000000	120.000000	120.000000
mean	46.928000	10.650000	17.785833	24.919167	14.269167	939.826842	80.251048	13.266583	0.808333
std	44.229143	8.052308	7.805291	7.664699	1.903460	1049.003852	62.231058	30.141890	0.395263
min	0.000000	-2.400000	3.500000	8.800000	10.300000	0.000000	2.453093	0.000000	0.000000
25%	10.935000	3.100000	10.500000	18.475000	12.875000	132.383500	21.389656	0.085000	1.000000
50%	36.055000	10.900000	18.200000	25.400000	14.200000	442.567000	66.463042	1.095000	1.000000
75%	66.600000	18.325000	25.225000	31.725000	15.700000	1498.260000	136.909704	10.077500	1.000000
max	205.320000	23.400000	30.500000	37.600000	18.200000	5317.788000	210.145647	179.300000	1.000000

In [14]:
```python
# Highlighting rows to only present data in the 2010s and moving it into its own dataframe

The2010s = df.iloc[262:]
The2010s
```

Out[14]:

	Date	PPT	Tmin	Tav	Tmax	delt	SMI	ET	Flow_cfs	Flow?
262	2010-01	39.64	-1.8	5.6	12.9	14.7	221.984	6.028550	0.98	1
263	2010-02	75.24	-0.5	5.3	11.1	11.6	398.772	5.217003	7.47	1
264	2010-03	48.90	4.1	12.0	19.9	15.8	586.800	29.979651	3.15	1
265	2010-04	115.83	10.2	17.4	24.5	14.3	2015.442	64.221802	72.40	1
266	2010-05	25.92	14.8	21.8	28.9	14.1	565.056	108.715967	2.77	1
...	...	...	...	...	...	...	...	...	...	...
368	2018-11	8.10	1.8	9.1	16.3	14.5	73.710	15.039182	0.10	1
369	2018-12	30.35	0.1	6.6	13.0	12.9	200.310	8.102200	0.01	1
370	2019-01	6.00	-0.1	6.8	13.8	13.9	40.800	8.740291	0.00	0
371	2019-02	3.71	1.6	9.4	17.2	15.6	34.874	15.613347	0.00	0
372	2019-03	19.36	4.3	10.9	17.5	13.2	211.024	24.942857	0.15	1

111 rows × 10 columns

In [15]:
```python
# Using the describe function to obtain basic and statistical information of the 2010s dataframe

The2010s.describe()
```

Out[15]:

	PPT	Tmin	Tav	Tmax	delt	SMI	ET	Flow_cfs	Flow?
count	111.000000	111.000000	111.000000	111.000000	111.000000	111.000000	111.000000	111.000000	111.000000
mean	42.730721	10.512613	17.852252	25.192793	14.680180	885.072532	82.097774	15.909350	0.567568
std	45.675354	8.562074	8.261013	8.068499	1.942436	1074.395558	66.411398	82.574002	0.497660
min	0.000000	-3.800000	4.600000	10.400000	9.900000	0.000000	4.137878	0.000000	0.000000
25%	11.980000	2.100000	9.350000	17.150000	13.600000	144.622500	15.645602	0.000000	0.000000
50%	31.100000	10.900000	18.100000	25.900000	14.400000	405.870000	67.798591	0.100000	1.000000
75%	62.315000	18.850000	25.700000	32.050000	15.800000	1431.070000	144.236413	3.000000	1.000000
max	251.240000	24.300000	31.500000	38.800000	19.400000	5672.665000	218.125353	709.900000	1.000000

In [16]:
```python
# Highlighting rows to only present data in the 2010s WITH NO OUTLIERS and moving it into its own dataframe
# Using the describe function to obtain basic and statistical information of the 2010s (WITH NO OUTLIERS) dataframe

The2010sNO_OUTLIERS = The2010s[The2010s["Flow_cfs"] < 500]
The2010sNO_OUTLIERS.describe()
```

Out[16]:

	PPT	Tmin	Tav	Tmax	delt	SMI	ET	Flow_cfs	Flow?
count	109.000000	109.000000	109.000000	109.000000	109.000000	109.000000	109.000000	109.000000	109.000000
mean	38.995229	10.402752	17.773394	25.144037	14.741284	801.326550	81.694683	5.057228	0.559633
std	36.644383	8.597033	8.314207	8.134203	1.896010	884.268337	66.952524	12.381805	0.498724
min	0.000000	-3.800000	4.600000	10.400000	10.200000	0.000000	4.137878	0.000000	0.000000
25%	11.750000	2.100000	9.300000	17.100000	13.600000	143.715000	15.613347	0.000000	0.000000
50%	30.350000	10.500000	18.000000	25.800000	14.500000	405.573000	65.127493	0.100000	1.000000
75%	58.950000	19.100000	25.700000	32.200000	15.800000	1369.377000	148.268856	2.770000	1.000000
max	169.510000	24.300000	31.500000	38.800000	19.400000	3753.208000	218.125353	72.400000	1.000000

In [17]:
```python
# Establishing a dataframe of the 1990s that is all non-flow states

NoFlowperiods90s = The1990s[The1990s["Flow?"] < 0.5]
# NoFlowperiods90s
NoFlowperiods90s.shape
```

Out[17]:
```
(46, 10)
```

In [18]:
```python
# Establishing a dataframe of the 2000s that is all non-flow states

NoFlowperiods00s = The2000s[The2000s["Flow?"] < 0.5]
# NoFlowperiods00s
NoFlowperiods00s.shape
```

Out[18]:
```
(23, 10)
```

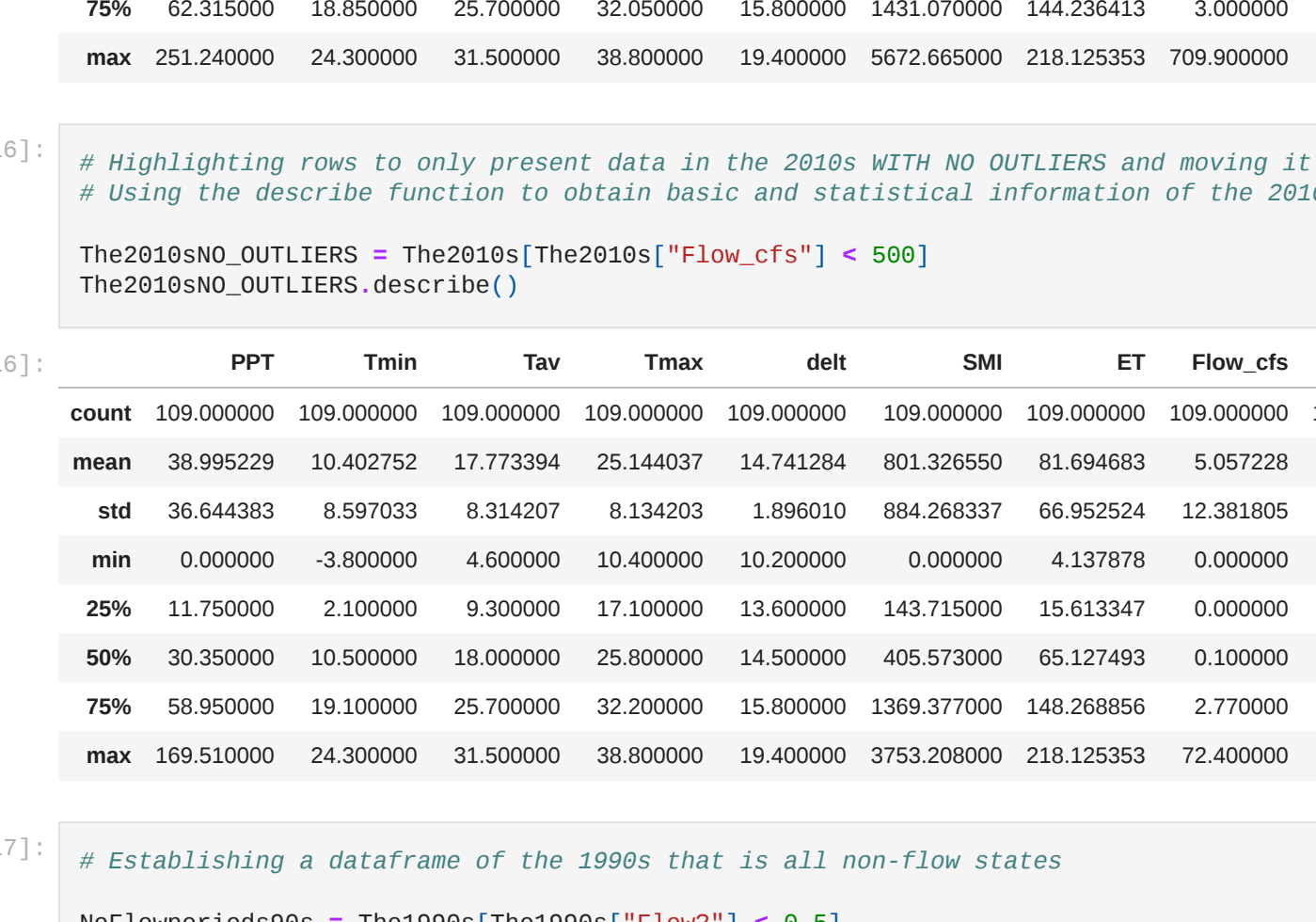
In [19]:
```python
# Establishing a dataframe of the 2010s that is all non-flow states

NoFlowperiods10s = The2010s[The2010s["Flow?"] < 0.5]
# NoFlowperiods10s
NoFlowperiods10s.shape
```

Out[19]:
```
(48, 10)
```

In [20]:
```python
# Creating the linear model using preciptation values to predict the average streamflow and fitting it
# Establishing key values of intercept, slope, R^2, and RMSE

import statsmodels.formula.api as smf

# Initialise and fit linear regression model using `statsmodels`
model = smf.ols('Flow_cfs ~ PPT', data=df) # model object constructor syntax
model = model.fit()

intercept = model.params[0]
slope = model.params[1]
Rsquare = model.rsquared
RMSE = math.sqrt(model.mse_total)



model.params
```

Out[20]:
```
Intercept   -18.333692
PPT           0.741168
dtype: float64
```

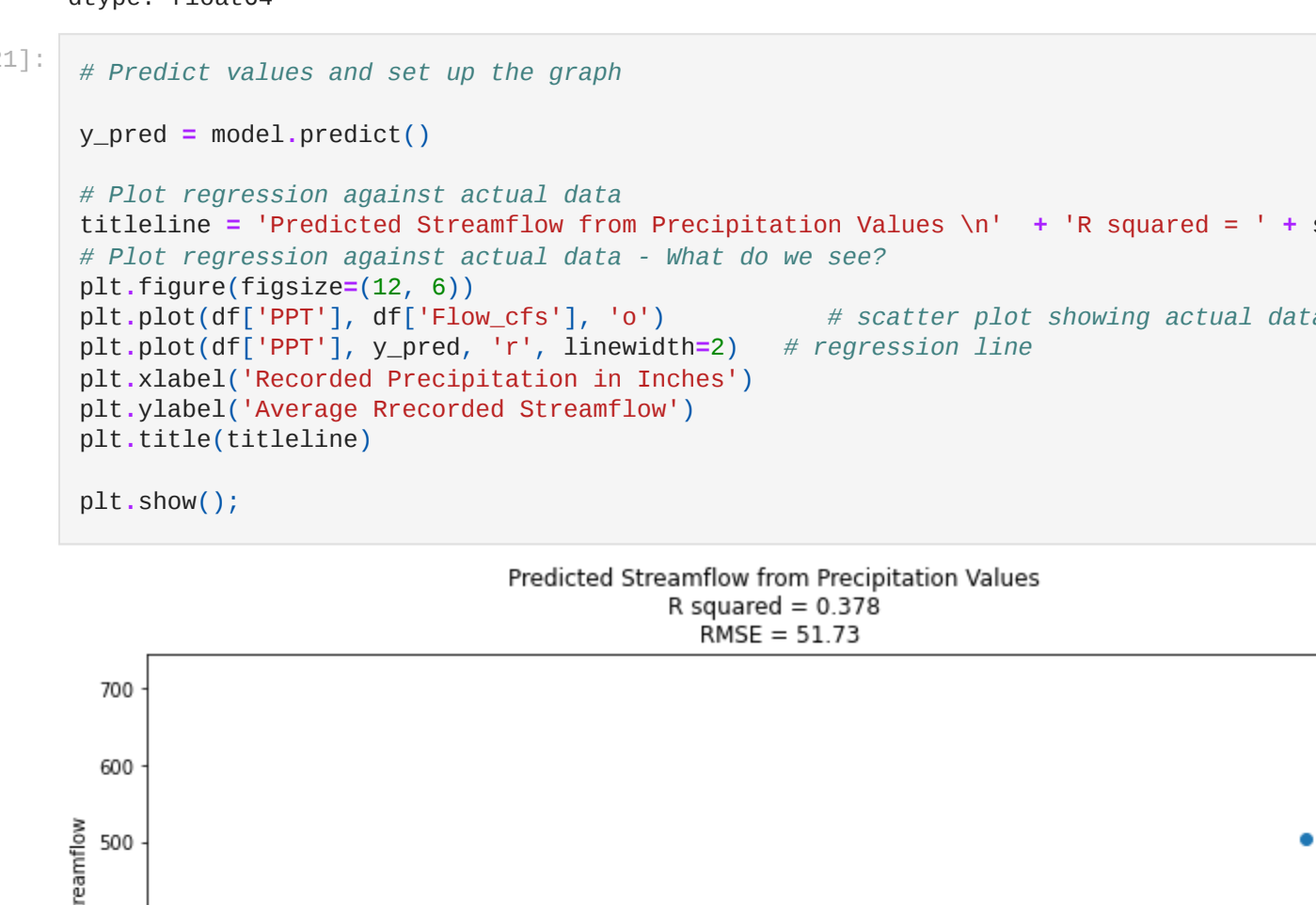
In [21]:
```python
# Predict values and set up the graph

y_pred = model.predict()

# Plot regression against actual data
titleline = 'Predicted Streamflow from Precipitation Values \n'  + 'R squared = ' + str(round(Rsquare,3)) + ' \n RMSE = ' + str(round(RMSE,2))
# Plot regression against actual data - What do we see?
plt.figure(figsize=(12, 6))
plt.plot(df['PPT'], df['Flow_cfs'], 'o')           # scatter plot showing actual data
plt.plot(df['PPT'], y_pred, 'r', linewidth=2)   # regression line
plt.xlabel('Recorded Precipitation in Inches')
plt.ylabel('Average Rrecorded Streamflow')
plt.title(titleline)

plt.show();
```



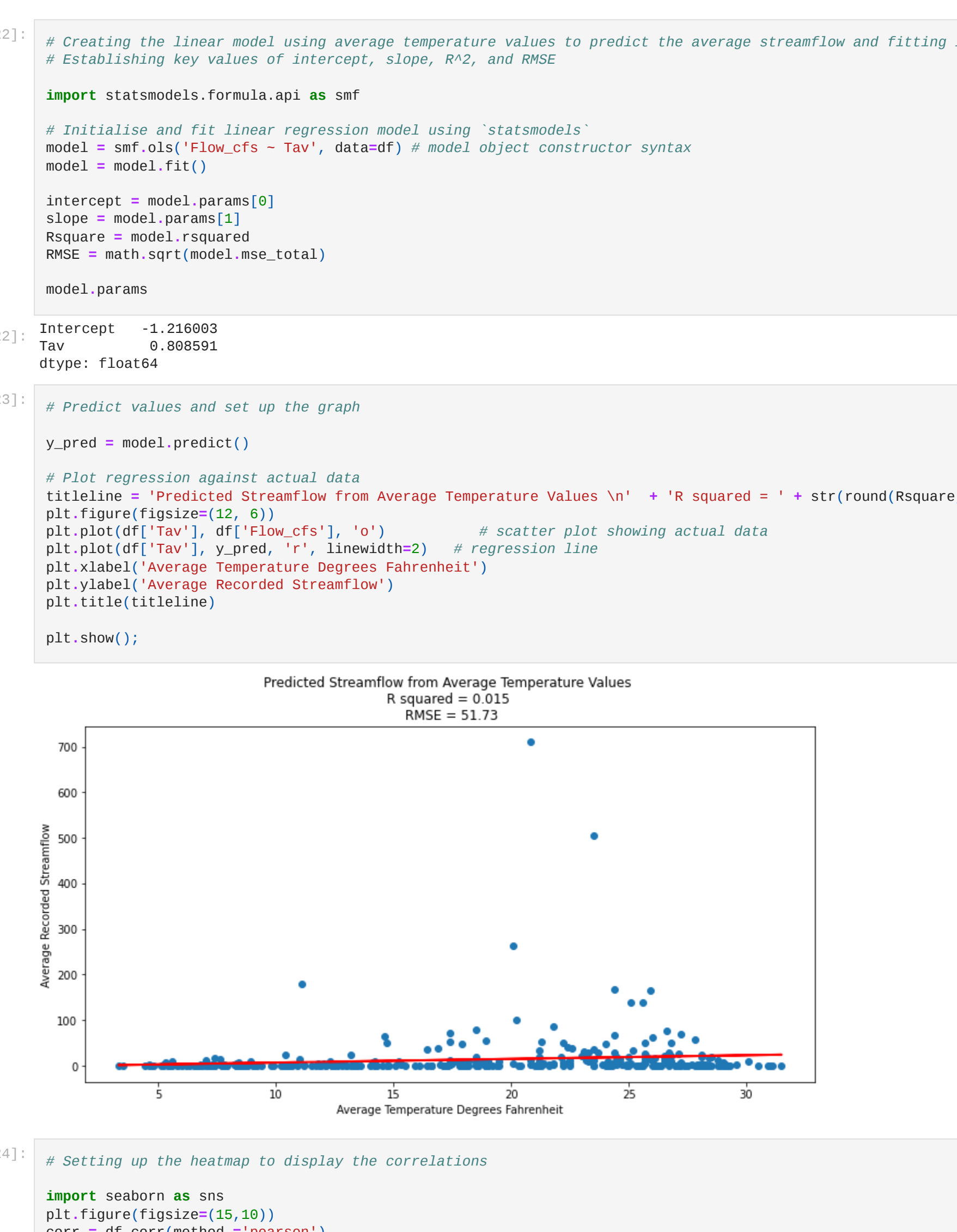
In [22]:
```python
# Creating the linear model using average temperature values to predict the average streamflow and fitting it
# Establishing key values of intercept, slope, R^2, and RMSE

import statsmodels.formula.api as smf

# Initialise and fit linear regression model using `statsmodels`
model = smf.ols('Flow_cfs ~ Tav', data=df) # model object constructor syntax
model = model.fit()

intercept = model.params[0]
slope = model.params[1]
Rsquare = model.rsquared
RMSE = math.sqrt(model.mse_total)

model.params
```

Out[22]:
```
Intercept   -1.216003
Tav          0.808591
dtype: float64
```

In [23]:
```python
# Predict values and set up the graph

y_pred = model.predict()

# Plot regression against actual data
titleline = 'Predicted Streamflow from Average Temperature Values \n'  + 'R squared = ' + str(round(Rsquare,3)) + ' \n RMSE = ' + str(round(RMSE,2)
plt.figure(figsize=(12, 6))
plt.plot(df['Tav'], df['Flow_cfs'], 'o')           # scatter plot showing actual data
plt.plot(df['Tav'], y_pred, 'r', linewidth=2)   # regression line
plt.xlabel('Average Temperature Degrees Fahrenheit')
plt.ylabel('Average Recorded Streamflow')
plt.title(titleline)

plt.show();
```

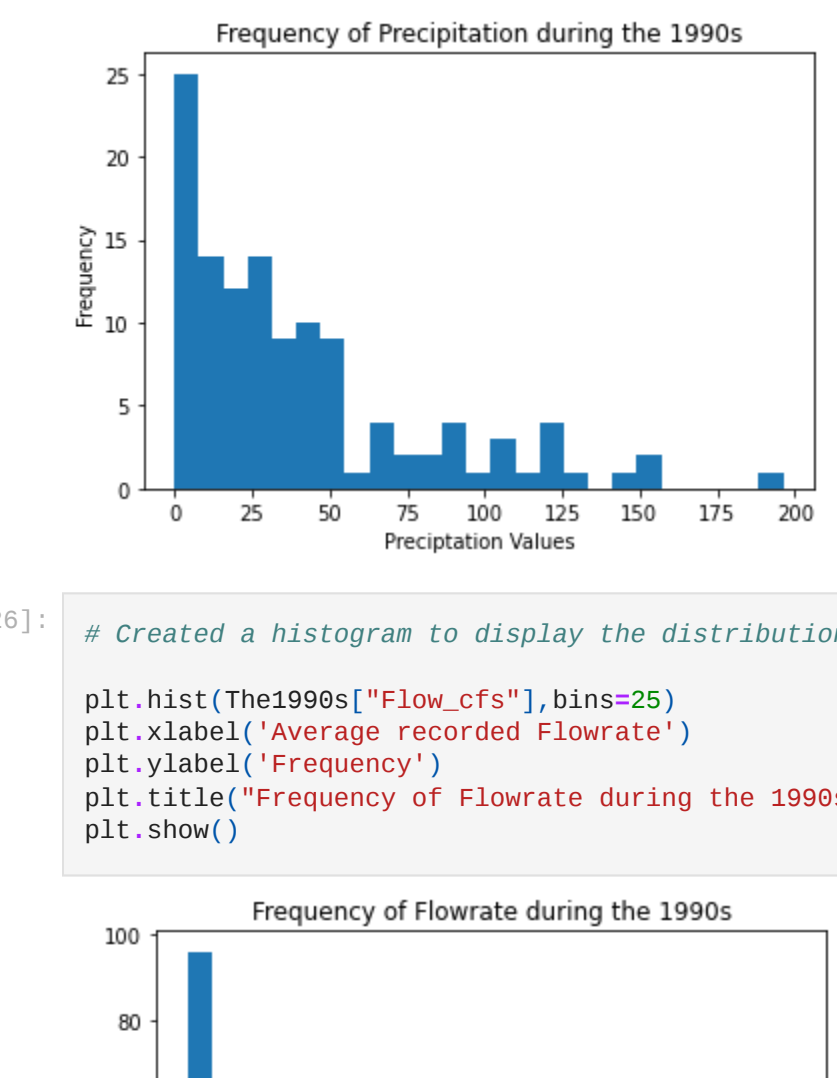


In [24]:
```python
# Setting up the heatmap to display the correlations

import seaborn as sns
plt.figure(figsize=(15,10))
corr = df.corr(method ='pearson')
sns.heatmap(corr, annot=True)
plt.show()
```

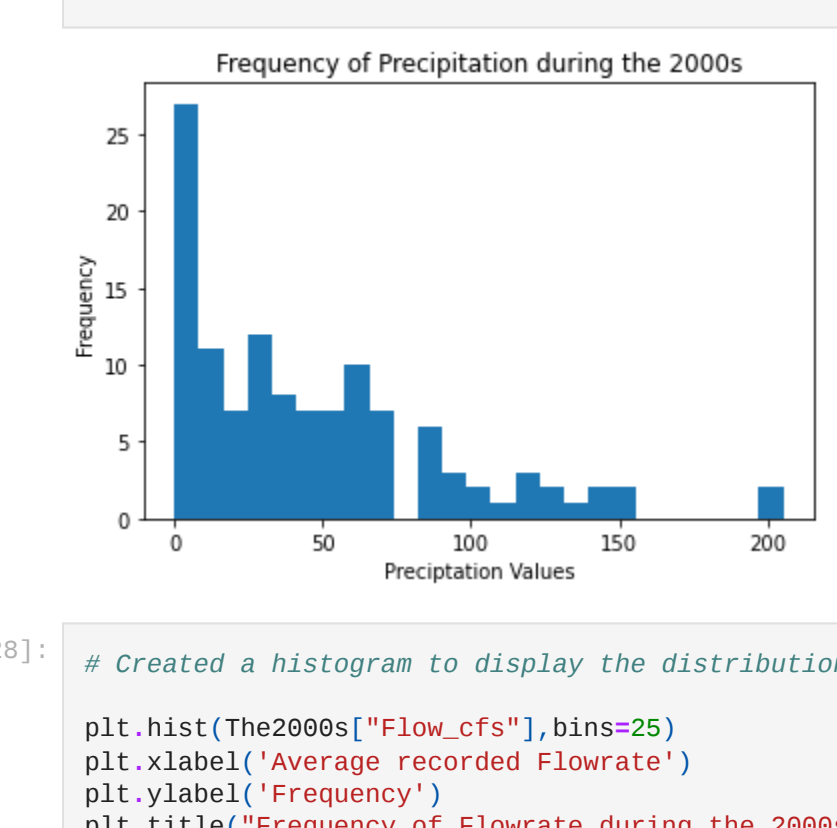


In [25]:
```python
# Created a histogram to display the distribution of the preciptation of the 1990s

plt.hist(The1990s["PPT"],bins=25)
plt.xlabel('Preciptation Values')
plt.ylabel('Frequency')
plt.title("Frequency of Precipitation during the 1990s")
plt.show()
```

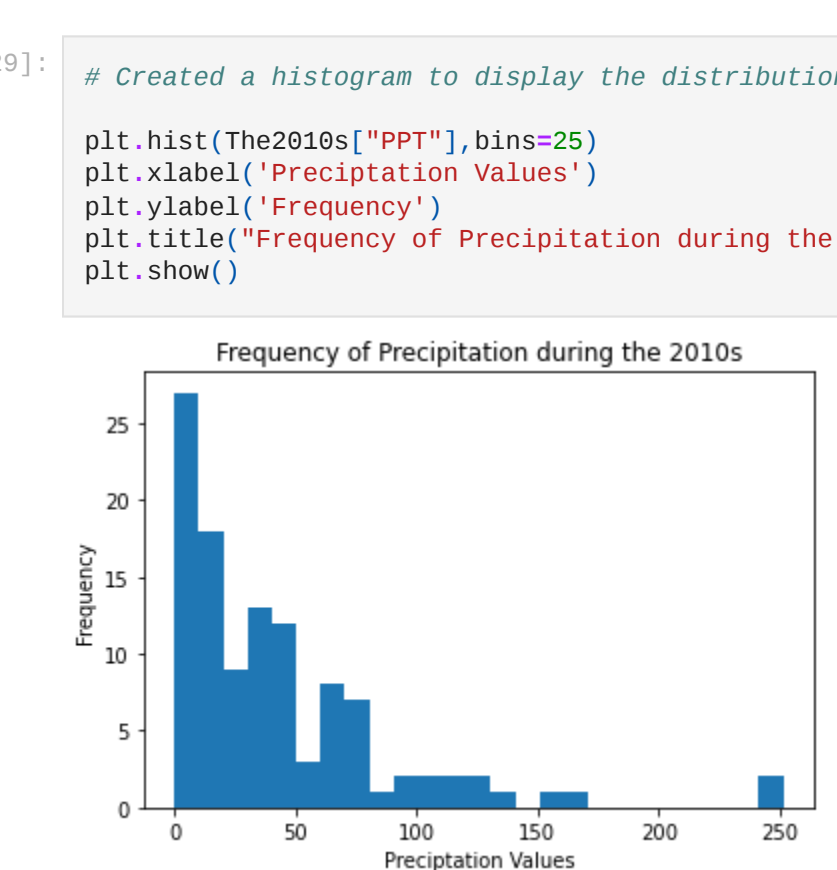
In [26]:
```python
# Created a histogram to display the distribution of the average flowrate of the 1990s

plt.hist(The1990s["Flow_cfs"],bins=25)
plt.xlabel('Average recorded Flowrate')
plt.ylabel('Frequency')
plt.title("Frequency of Flowrate during the 1990s")
plt.show()
```



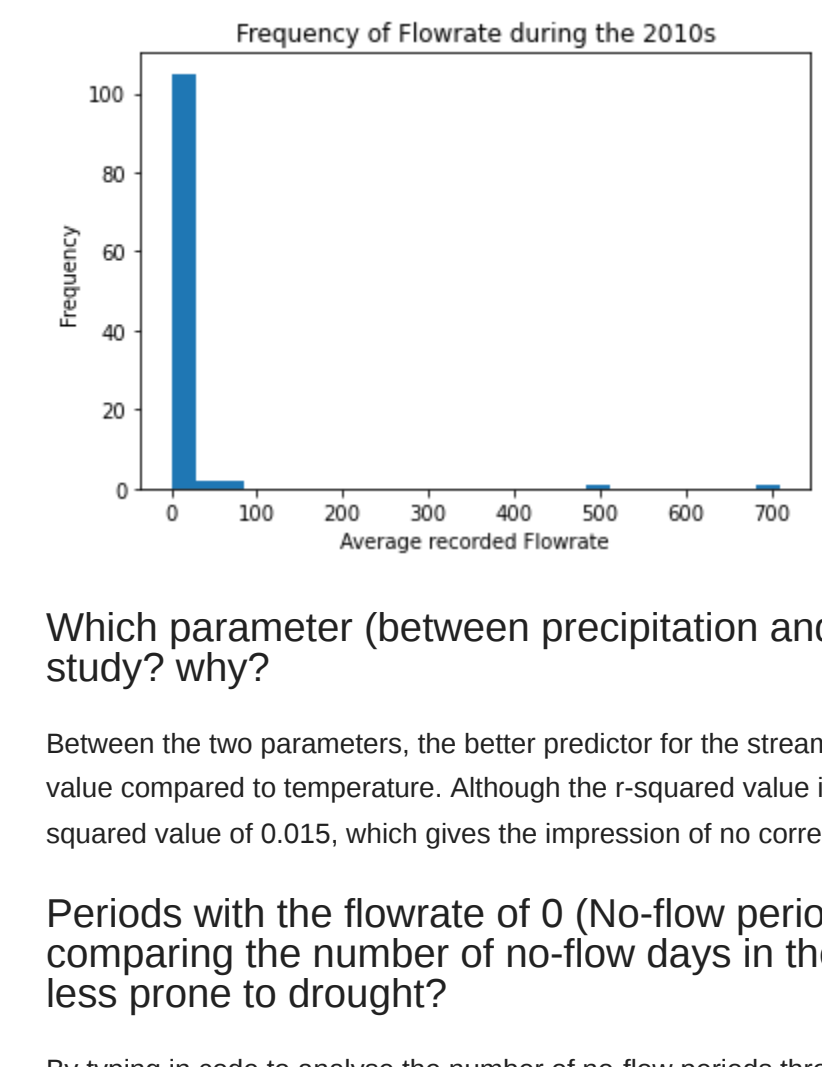
In [27]:
```python
# Created a histogram to display the distribution of the preciptation of the 2000s

plt.hist(The2000s["PPT"],bins=25)
plt.xlabel('Preciptation Values')
plt.ylabel('Frequency')
plt.title("Frequency of Precipitation during the 2000s")
plt.show()
```



In [28]:
```python
# Created a histogram to display the distribution of the average flowrate of the 2000s

plt.hist(The2000s["Flow_cfs"],bins=25)
plt.xlabel('Average recorded Flowrate')
plt.ylabel('Frequency')
plt.title("Frequency of Flowrate during the 2000s")
plt.show()
```



In [29]:
```python
# Created a histogram to display the distribution of the preciptation of the 2010s

plt.hist(The2010s["PPT"],bins=25)
plt.xlabel('Preciptation Values')
plt.ylabel('Frequency')
plt.title("Frequency of Precipitation during the 2010s")
plt.show()
```



In [30]:
```python
# Created a histogram to display the distribution of the average flowrate of the 2010s

plt.hist(The2010s["Flow_cfs"],bins=25)
plt.xlabel('Average recorded Flowrate')
plt.ylabel('Frequency')
plt.title("Frequency of Flowrate during the 2010s")
plt.show()
```


Which parameter (between precipitation and temperature values) can be a better predictor for streamflow at the station of study? why?

Between the two parameters, the better predictor for the streamflow is precipitation, as when fitting a linear data model reveals that the precipitation has a vastly superior r-squared value compared to temperature. Although the r-squared value is 0.378 out of 1, which is considered a weak or low effect size; however, it is considerably better than the temperature's r-squared value of 0.015, which gives the impression of no correlation regarding streamflow output.

Periods with the flowrate of 0 (No-flow periods) can be viewed as indicators of drought. What can you understand from comparing the number of no-flow days in the 90s, 2000s, and 2010s? Is the upstream of the Colorado River becoming more or less prone to drought?

By typing in code to analyse the number of no-flow periods throughout the three decades, it is revealed that the 90s had 46 no-flow periods, the 2000's had 23 no-flow periods, and the 2010s had 48 no-flow periods. In analysis, I was quite surprised to see that the amount of no-flow periods in the 2000's differed greater compared to the other 2 decades. As a result, I created histograms in order to visualize the data set and realized that the flow rate of the 2010s had two outliers that greatly influenced the average flow rate, this had made it appear to be on edge with the two other decades at 15.9; however, once removing the outlier, the average flow rate greatly decreased to 5.06 granting me an assumption that the upstream of the Colorado river is becoming more prone to drought since the correlations of the "Flow?" column has a high, positive correlation with average flow rate.

What can you infer from comparing the maximum recorded floods in the 90s, 2000s, and 2010s? Based on the records available in the dataset, would you expect to see more or less extreme floods in the future?

By ultizing histograms to view the frequencies of the average flowrate for the past three decades, one can understand that amount of floodings has significantly declined; however, it can be noted that the amount of extreme floods would be expected to increased as before the 2010s, there was no average flowrate above 300, and the 2010s manages to have two extreme floodings at the 500 and 700 mark.