In [1]:

```
import pandas as pd
global_temp = pd.read_csv("GlobalTemperatures.csv")
print(global_temp.shape)
print(global_temp.columns)
print(global_temp.info())
print(global_temp.isnull().sum())
```

```
(3192, 9)
Index(['dt', 'LandAverageTemperature', 'LandAverageTemperatureUncertainty',
       'LandMaxTemperature', 'LandMaxTemperatureUncertainty',
       'LandMinTemperature', 'LandMinTemperatureUncertainty',
       'LandAndOceanAverageTemperature',
       'LandAndOceanAverageTemperatureUncertainty'],
      dtype='object')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3192 entries, 0 to 3191
Data columns (total 9 columns):
 #   Column                                     Non-Null Count  Dtype
---  ------                                     --------------  -----
 0   dt                                         3192 non-null   object
 1   LandAverageTemperature                     3180 non-null   float64
 2   LandAverageTemperatureUncertainty          3180 non-null   float64
 3   LandMaxTemperature                         1992 non-null   float64
 4   LandMaxTemperatureUncertainty              1992 non-null   float64
 5   LandMinTemperature                         1992 non-null   float64
 6   LandMinTemperatureUncertainty              1992 non-null   float64
 7   LandAndOceanAverageTemperature             1992 non-null   float64
 8   LandAndOceanAverageTemperatureUncertainty  1992 non-null   float64
dtypes: float64(8), object(1)
memory usage: 224.6+ KB
None
dt                                              0
LandAverageTemperature                         12
LandAverageTemperatureUncertainty              12
LandMaxTemperature                           1200
LandMaxTemperatureUncertainty                1200
LandMinTemperature                           1200
LandMinTemperatureUncertainty                1200
LandAndOceanAverageTemperature               1200
LandAndOceanAverageTemperatureUncertainty    1200
dtype: int64
```

In [9]:

```
def wrangle(df):
    df.copy()
    df.drop(columns=["LandAverageTemperatureUncertainty","LandMaxTemperatureUncertainty"
,"LandMinTemperatureUncertainty","LandAndOceanAverageTemperatureUncertainty"])
    def converttemp(x):
        x=(x*1.8)+32
        return float(x)
    df["LandAverageTemperature"] = df["LandAverageTemperature"].apply(converttemp)
    df["LandMaxTemperature"] = df["LandMaxTemperature"].apply(converttemp)
    df["LandMinTemperature"] = df["LandMinTemperature"].apply(converttemp)
    df["LandAndOceanAverageTemperature"] = df["LandAndOceanAverageTemperature"].apply(co
nverttemp)
    df["dt"] = pd.to_datetime(df["dt"])
    df["Month"] = df["dt"].dt.month
    df["Year"] = df["dt"].dt.year
    df = df.drop("dt", axis=1)
    df = df.drop("Month", axis=1)
    df = df[df.Year >= 1850]
    df = df.set_index(["Year"])
    df = df.dropna()
    return df
```

In [10]:

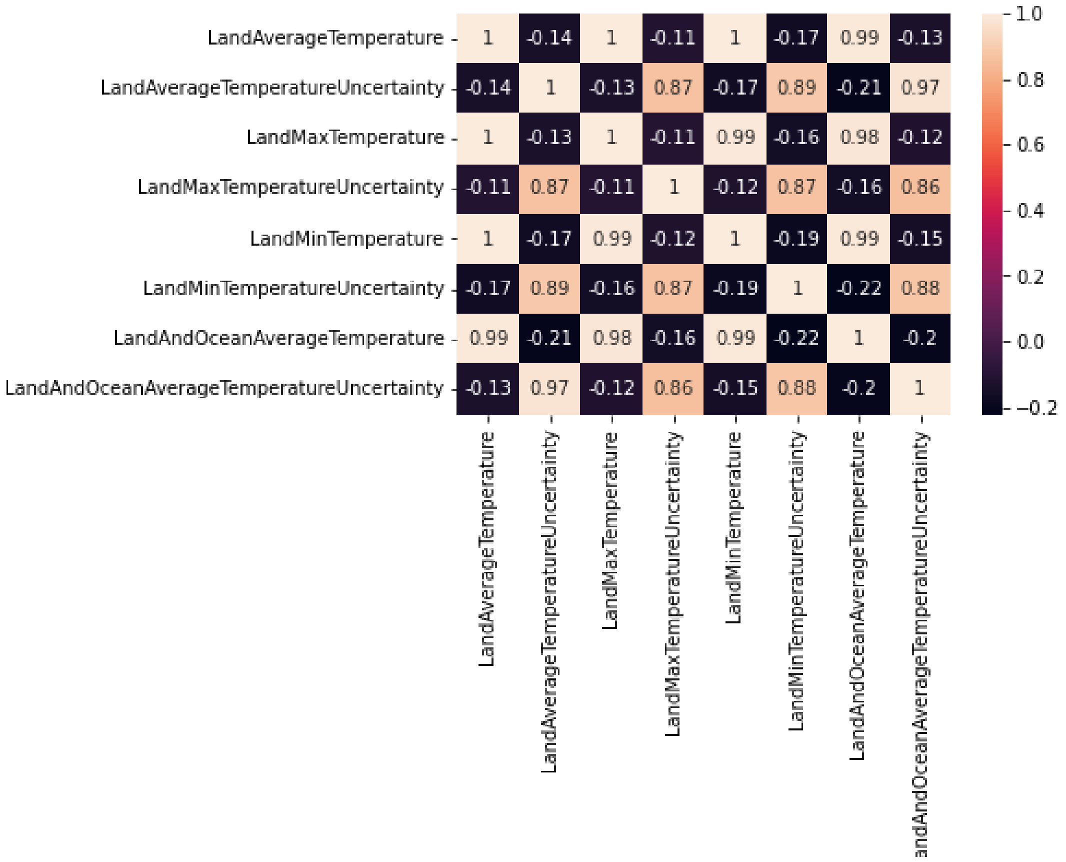
In [10]:

```
global_temp = wrangle(global_temp)
print(global_temp.head())
```

```
      LandAverageTemperature  LandAverageTemperatureUncertainty  \
Year                                                              
1850                 33.3482                              1.105   
1850                 37.5278                              1.275   
1850                 40.9172                              0.955   
1850                 44.9906                              0.665   
1850                 50.0072                              0.617   

      LandMaxTemperature  LandMaxTemperatureUncertainty  LandMinTemperature  \
Year                                                                          
1850             46.8356                          1.738             26.2292   
1850             49.9460                          3.007             27.8762   
1850             50.6246                          2.401             28.5710   
1850             55.2812                          1.004             33.8324   
1850             60.1790                          2.406             38.8598   

      LandMinTemperatureUncertainty  LandAndOceanAverageTemperature  \
Year                                                                  
1850                          2.822                         55.0994   
1850                          1.623                         56.4584   
1850                          1.410                         57.2774   
1850                          1.329                         58.4006   
1850                          1.347                         59.9126   

      LandAndOceanAverageTemperatureUncertainty  
Year                                             
1850                                      0.367  
1850                                      0.414  
1850                                      0.341  
1850                                      0.267  
1850                                      0.249  
```

In [12]:

```
import seaborn as sns
import matplotlib.pyplot as plt
corrMatrix = global_temp.corr()
sns.heatmap(corrMatrix, annot=True)
plt.show()
```



In [13]:

```
target = "LandAndOceanAverageTemperature"
y = global_temp[target]
x = global_temp[["LandAverageTemperature", "LandMaxTemperature", "LandMinTemperature"]]
```

In [14]:

```
from sklearn.model_selection import train_test_split
xtrain, xval, ytrain, yval = train_test_split(x, y, test_size=0.25, random_state=42)
print(xtrain.shape)
print(xval.shape)
print(ytrain.shape)
print(yval.shape)
```

```
(1494, 3)
(498, 3)
(1494,)
(498,)
```

In [15]:

```
from sklearn.metrics import mean_squared_error
ypred = [ytrain.mean()] * len(ytrain)
print("Baseline MAE: ", round(mean_squared_error(ytrain, ypred), 5))
```

```
Baseline MAE:  5.29374
```

In [16]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.feature_selection import SelectKBest
from sklearn.ensemble import RandomForestRegressor
forest = make_pipeline(SelectKBest(k="all"),StandardScaler(),
RandomForestRegressor(
n_estimators=100,
max_depth=50,
random_state=77,
n_jobs=-1
)
)
forest.fit(xtrain, ytrain)
```

Out[16]:

```
Pipeline(steps=[('selectkbest', SelectKBest(k='all')),
                ('standardscaler', StandardScaler()),
                ('randomforestregressor',
                 RandomForestRegressor(max_depth=50, n_jobs=-1,
                                       random_state=77))])
```

In [18]:

```
print("Training Accuracy :", forest.score(xtrain, ytrain))
print("Testing Accuracy :", forest.score(xval, yval))
```

```
Training Accuracy : 0.997300169812254
Testing Accuracy : 0.9802030421320909
```