```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_csv('winequality-red.csv')
```

```
df.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	p
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.2
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.2
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.1
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.5

Next steps: Generate code with `df` View recommended plots

```
df.shape
```

```
(1599, 12)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

```
df.describe()
```

	fixed acidity	volatile acidity	citric acid	re
count	1599.000000	1599.000000	1599.000000	1599.
mean	8.319637	0.527821	0.270976	2.
std	1.741096	0.179060	0.194801	1.
min	4.600000	0.120000	0.000000	0.
25%	7.100000	0.390000	0.090000	1.
50%	7.900000	0.520000	0.260000	2.
75%	9.200000	0.640000	0.420000	2.
max	15.900000	1.580000	1.000000	15.

```
df.isnull().sum()
```

```
fixed acidity          0
volatile acidity       0
citric acid            0
residual sugar         0
chlorides              0
```

NameError NameError AttributeError •••

You

Please explain the error:

```
KeyError: "None of [Index(['total sulphur dioxide'
```

Colab AI

The error message indicates that the `df` DataFrame does not have a column named "total sulphur dioxide". To fix the issue, ensure that the column name is correct or change the column name in the DataFrame.

Here's how to investigate the issue and fix the code:

1. **Check the column names of the DataFrame:**

```
print(df.columns)
```

Use code with caution

This code will print the names of all the columns in the `df` DataFrame. Verify if "total sulphur dioxide" is present in the list.

2. **If the column name is incorrect, update the code:**

```
population_fit.fit(df[["Total Sulphur Dioxide"]]
```

Use code with caution

This code assumes that the correct column name is "Total Sulphur Dioxide". Adjust the code accordingly if the actual column name is different.

3. **If the column name is correct, investigate further:**

```
print(df["total sulphur dioxide"].head())
```

Use code with caution

This code will print the first few rows of the "total sulphur dioxide" column. Check if the data in the column is valid.

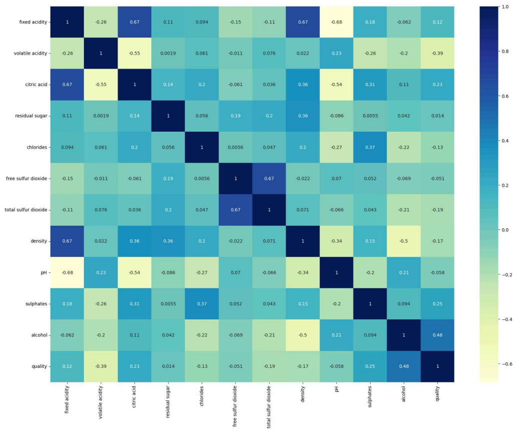
4. **Fix the issue based on the investigation:**

- If the column name is incorrect, update the code as shown in step 2.
- If the data in the column is invalid, clean or pre-process the data before fitting the model.
- If the issue persists, provide the complete code and error traceback for further assistance.

Rate this answer

```
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

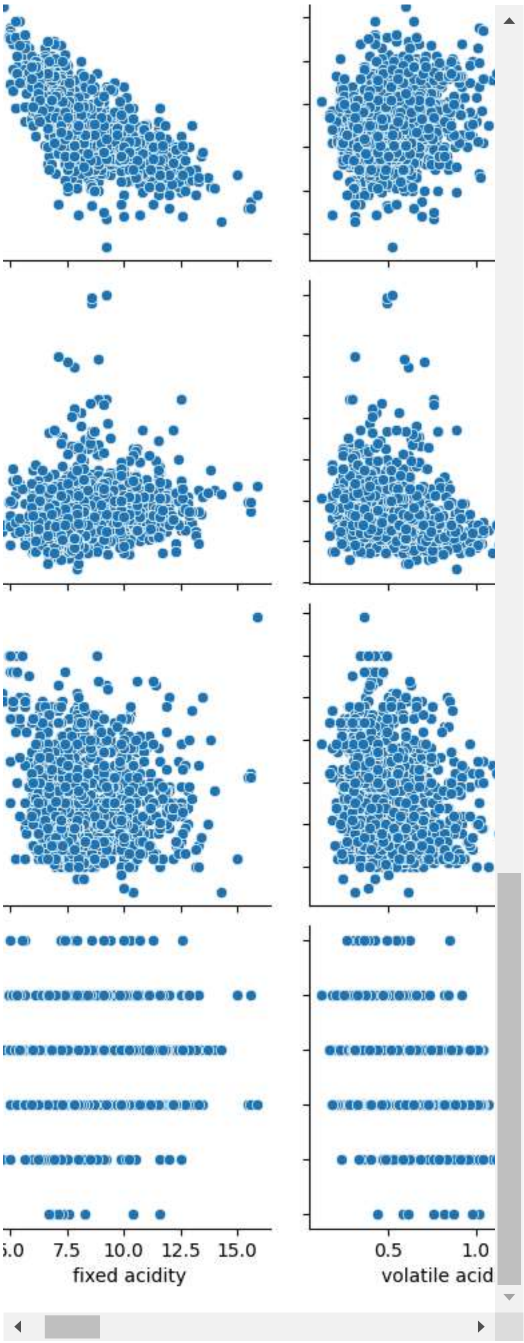
```
plt.figure(figsize=(20,15))
sns.heatmap(df.corr(), annot = True, cmap = 'YlGnBu')
plt.show()
```



	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1	-0.26	0.67	0.11	0.094	-0.15	-0.11	0.67	-0.68	0.18	-0.062	0.12
volatile acidity	-0.26	1	-0.55	0.0019	0.061	-0.011	0.076	0.022	0.23	-0.26	-0.2	-0.39
citric acid	0.67	-0.55	1	0.14	0.2	-0.061	0.036	0.36	-0.54	0.31	0.11	0.23
residual sugar	0.11	0.0019	0.14	1	0.056	0.19	0.2	0.36	-0.086	0.0055	0.042	0.014
chlorides	0.094	0.061	0.2	0.056	1	0.0056	0.047	0.2	-0.27	0.37	-0.22	-0.13
free sulfur dioxide	-0.15	-0.011	-0.061	0.19	0.0056	1	0.67	-0.022	0.07	0.052	-0.069	-0.051
total sulfur dioxide	-0.11	0.076	0.036	0.2	0.047	0.67	1	0.071	-0.066	0.043	-0.21	-0.19
density	0.67	0.022	0.36	0.36	0.2	-0.022	0.071	1	-0.34	0.15	-0.5	-0.17
pH	-0.68	0.23	-0.54	-0.086	-0.27	0.07	-0.066	-0.34	1	-0.2	0.21	-0.058
sulphates	0.18	-0.26	0.31	0.0055	0.37	0.052	0.043	0.15	-0.2	1	0.094	0.25
alcohol	-0.062	-0.2	0.11	0.042	-0.22	-0.069	-0.21	-0.5	0.21	0.094	1	0.48
quality	0.12	-0.39	0.23	0.014	-0.13	-0.051	-0.19	-0.17	-0.058	0.25	0.48	1

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from joblib import dump, load
%matplotlib inline
```

```
sns.pairplot(df)
```


.0
7.5
10.0
12.5
15.0
fixed acidity
0.5
1.0
volatile acid

```
y = df['quality']
X = df['chlorides']
```

```
polynomial_converter = PolynomialFeatures(degree = 2, include_bias = False)
X = X.values.reshape(-1,1)
poly_features = polynomial_converter.fit_transform(X)
print(poly_features.shape)
```

```
(1599, 2)
```

```
X.shape
```

```
(1599, 1)
```

```
X_train, X_test, y_train, y_test = train_test_split(poly_features,y,train_size = 0.
```

```
polymodel = LinearRegression()
```

```
polymodel.fit(X_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
y_pred = polymodel.predict(X_test)
```

```
pd.DataFrame({'Y_test':y_test,'y_pred':y_pred, 'residuals':(y_test-y_pred)}).head()
```

	Y_test	y_pred	residuals
1254	5	5.659075	-0.659075
1087	6	5.766731	0.233269
822	5	5.673221	-0.673221
1514	6	5.682746	0.317254
902	7	5.654397	1.345603

```
mae_poly = metrics.mean_absolute_error(y_test,y_pred)
mse_poly = metrics.mean_squared_error(y_test,y_pred)
rmse_poly = np.sqrt(mse_poly)
r_score = metrics.r2_score(y_test,y_pred)
pd.DataFrame([mae_poly,mse_poly,rmse_poly, r_score], index = ['mae','mse','rmse','r
```

	metrics
mae	0.655147
mse	0.612797
rmse	0.782813
rscore	0.030039

```
coeff = polymodel.coef_
intercept = polymodel.intercept_
print('coeff:',coeff)
print('intercept:', intercept)
```

```
coeff: [-6.16723919  9.36615106]
intercept: 6.087832574693612
```

```
#exponential regression
def exponential_func(x,a,b):
  return a*np.exp(b*x)
from scipy.optimize import curve_fit
popt, pcov = curve_fit(exponential_func,df['total sulfur dioxide'],df['quality'])
```

```
a_fit, b_fit = popt
time_fit = np.linspace(0,len(df))
population_fit = exponential_func(time_fit,a_fit,b_fit)
```

```
<ipython-input-26-f9aa9179710c>:3: RuntimeWarning: overflow encountered in exp
  return a*np.exp(b*x)
```