In [3]:
```python
import matplotlib.pyplot as plt
```

In [4]:
```python
import pandas as pd
```

In [5]:
```python
import numpy as np
```

In [6]:
```python
import seaborn as sns
```

In [7]:
```python
data = pd.read_csv("D:/water_potability.csv")
```

In [8]:
```python
data.head()
```

Out[8]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783	86.990970	2.963135	0
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	15.180013	56.329076	4.500656	0
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	16.868637	66.420093	3.055934	0
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524	100.341674	4.628771	0
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279	31.997993	4.075075	0

In [9]:
```python
data.isnull().sum()
```

Out[9]:
```
ph                 491
Hardness             0
Solids               0
Chloramines          0
Sulfate            781
Conductivity         0
Organic_carbon       0
Trihalomethanes    162
Turbidity            0
Potability           0
dtype: int64
```

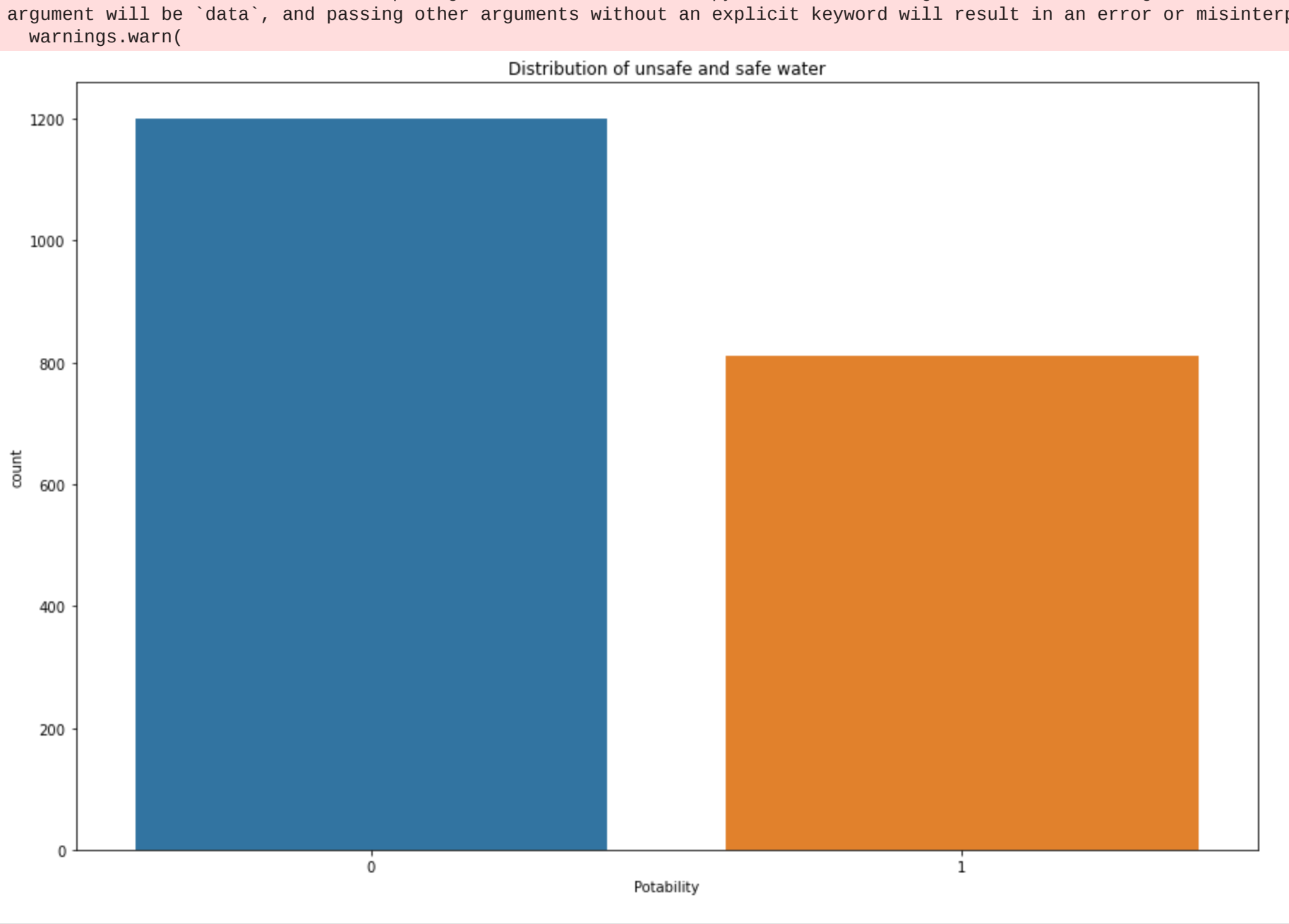
In [10]:
```python
data=data.dropna()
```

In [11]:
```python
data.isnull().sum()
```

Out[11]:
```
ph                 0
Hardness           0
Solids             0
Chloramines        0
Sulfate            0
Conductivity       0
Organic_carbon     0
Trihalomethanes    0
Turbidity          0
Potability         0
dtype: int64
```

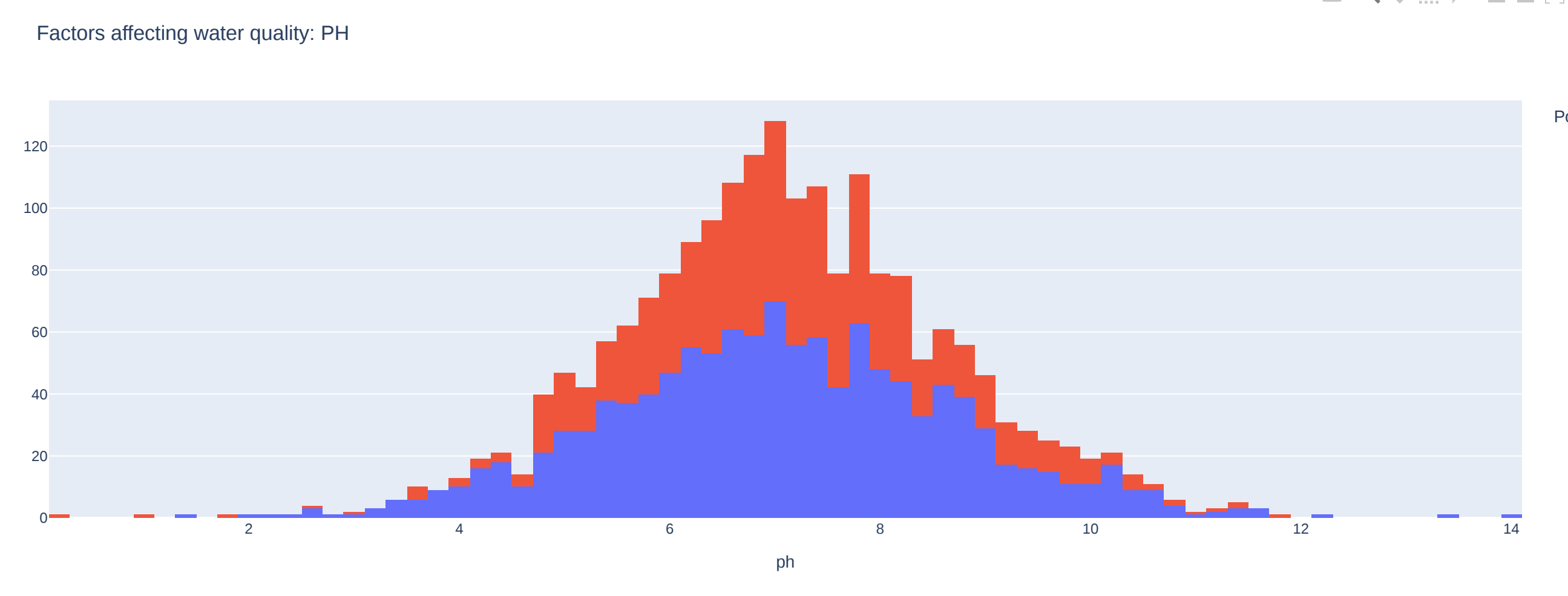
In [12]:
```python
plt.figure(figsize=(15,10))
sns.countplot(data.Potability)
plt.title("Distribution of unsafe and safe water")
plt.show()
```

```
C:\Users\lenovo\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

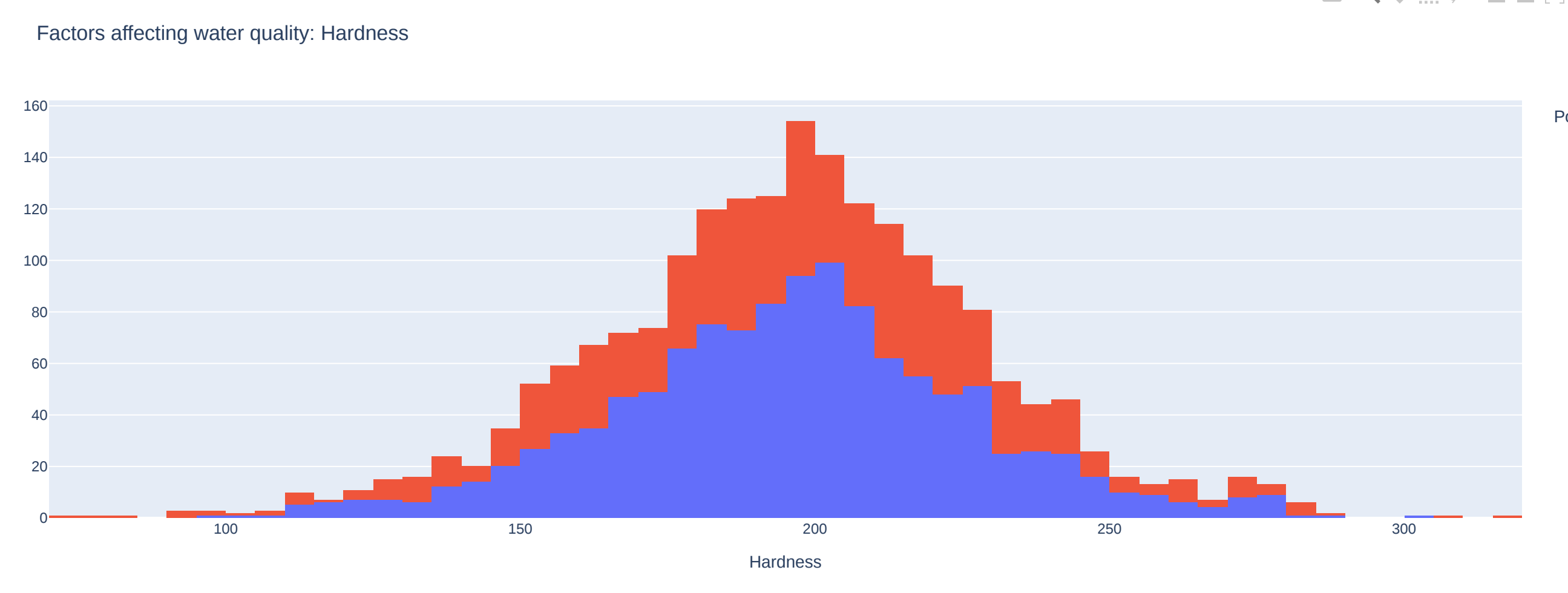


In [13]:
```python
import plotly.express as px
```

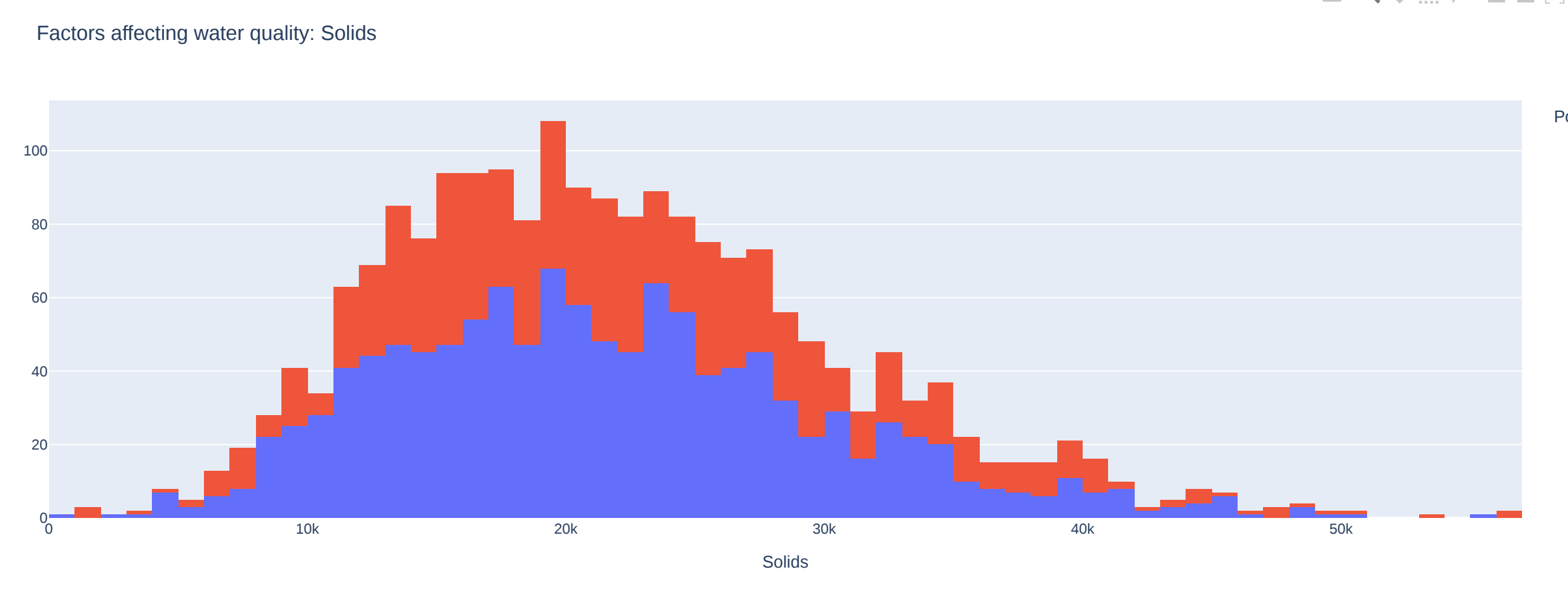
In [14]:
```python
data = data
figure = px.histogram(data, x ="ph",
                      color = "Potability",
                      title = "Factors affecting water quality: PH")
figure.show()
```



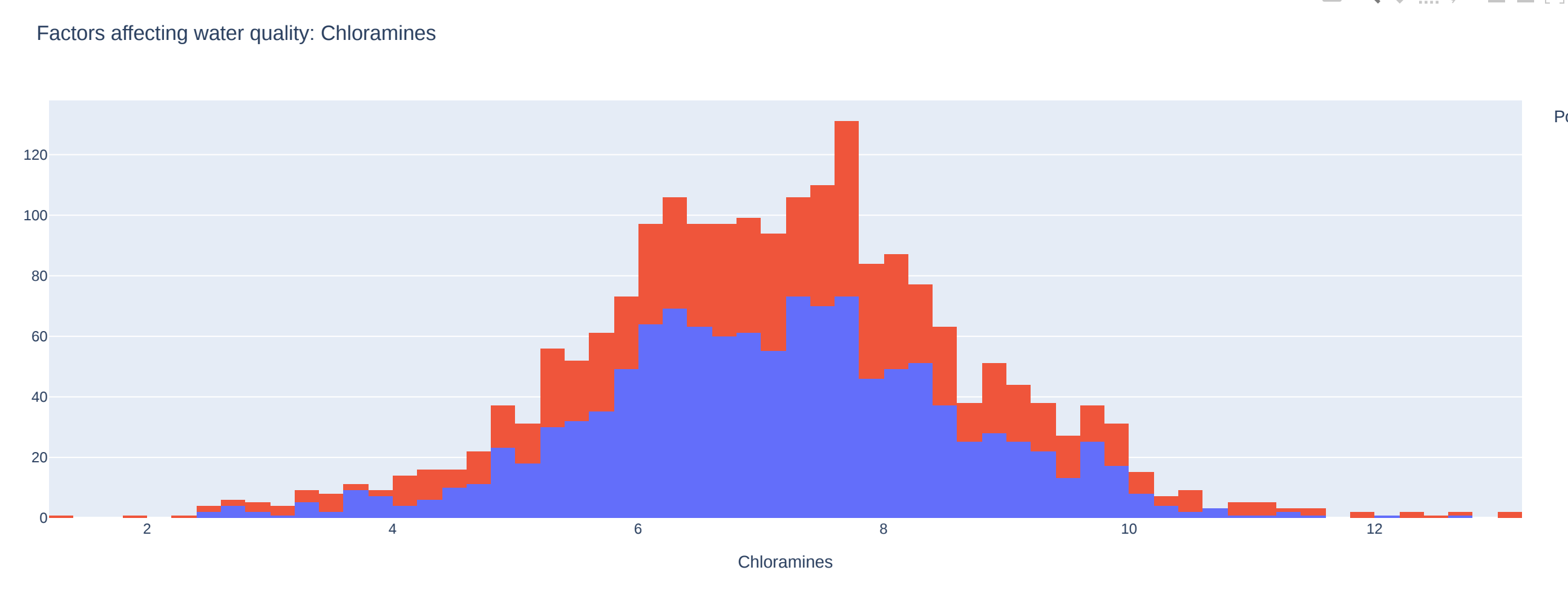
In [15]:
```python
data = data
figure = px.histogram(data, x ="Hardness",
                      color = "Potability",
                      title = "Factors affecting water quality: Hardness")
figure.show()
```



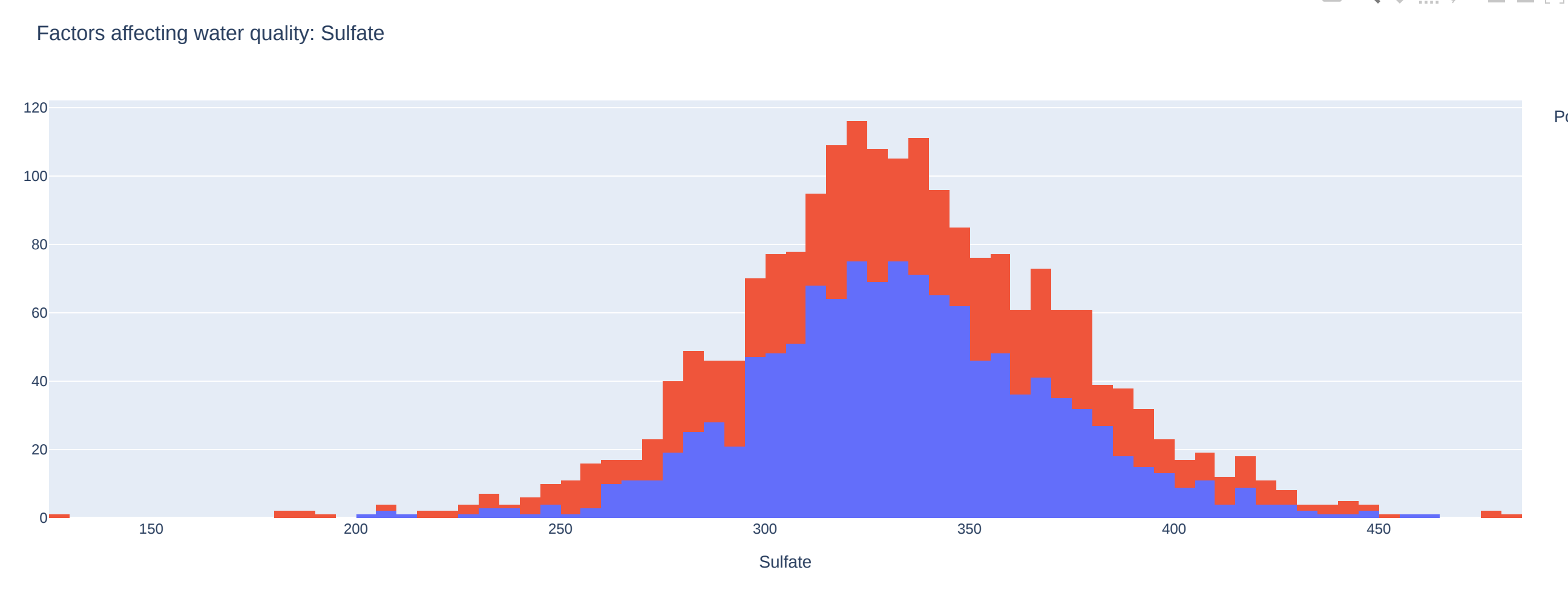
In [16]:
```python
data = data
figure = px.histogram(data, x ="Solids",
                      color = "Potability",
                      title = "Factors affecting water quality: Solids")
figure.show()
```



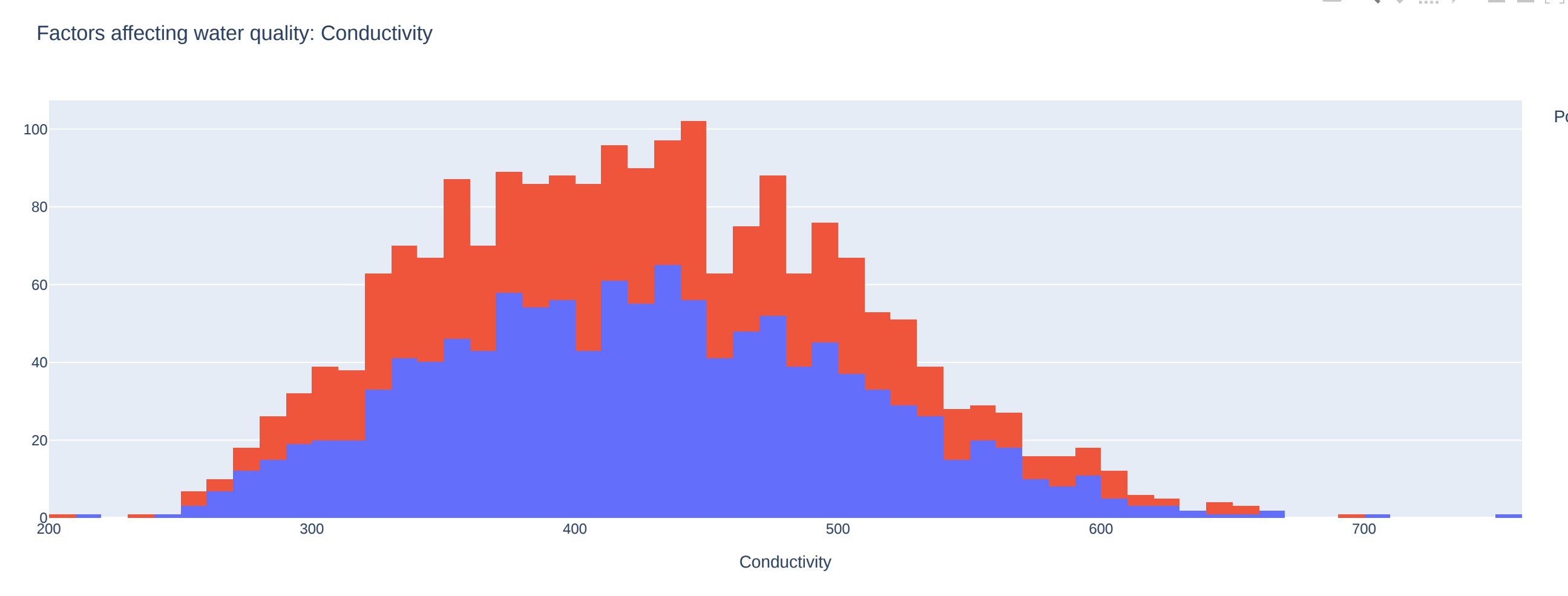
In [17]:
```python
data = data
figure = px.histogram(data, x ="Chloramines",
                      color = "Potability",
                      title = "Factors affecting water quality: Chloramines")
figure.show()
```



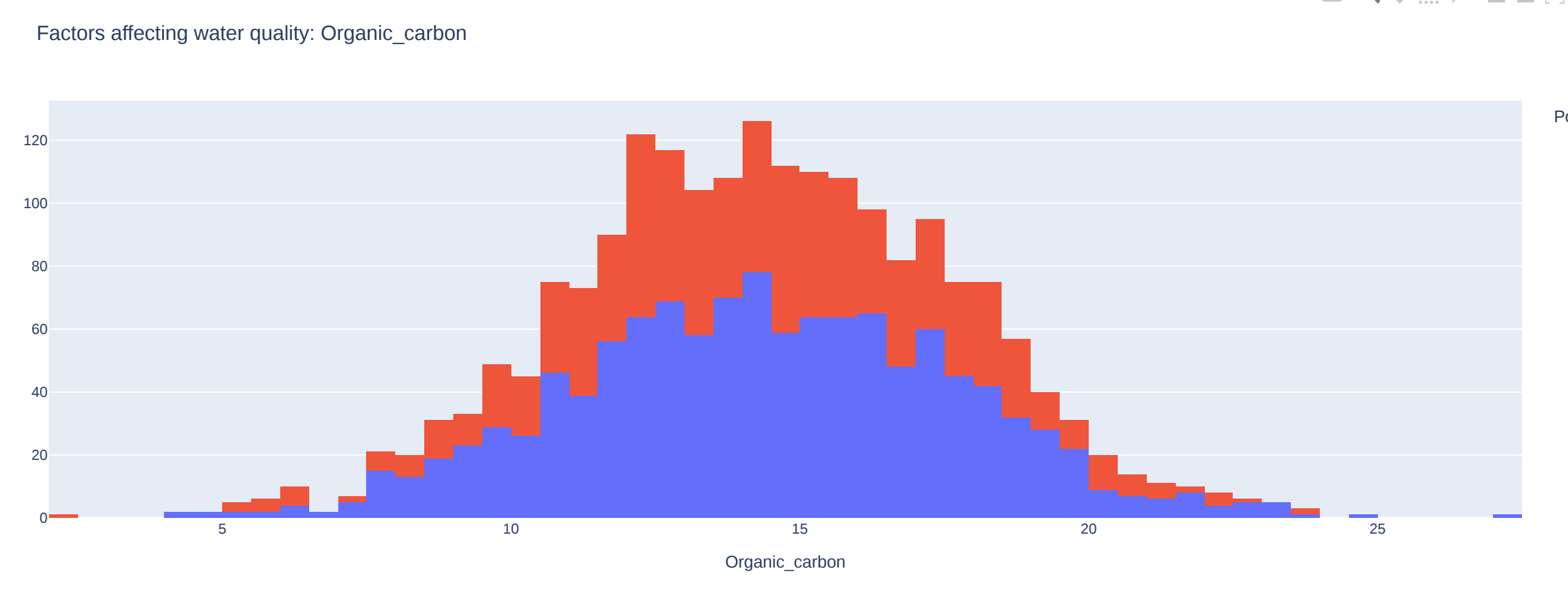
In [18]:
```python
data = data
figure = px.histogram(data, x ="Sulfate",
                      color = "Potability",
                      title = "Factors affecting water quality: Sulfate")
figure.show()
```



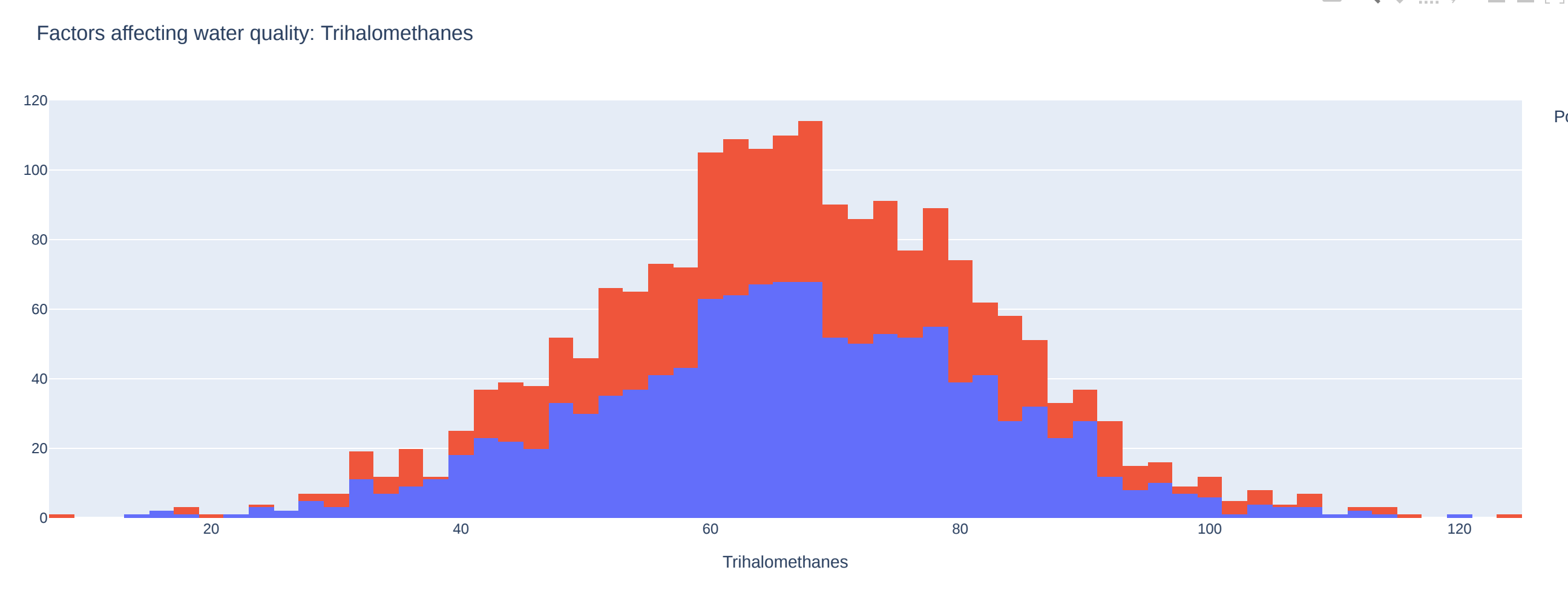
In [19]:
```python
data = data
figure = px.histogram(data, x ="Conductivity",
                      color = "Potability",
                      title = "Factors affecting water quality: Conductivity")
figure.show()
```



In [20]:
```python
data = data
figure = px.histogram(data, x ="Organic_carbon",
                      color = "Potability",
                      title = "Factors affecting water quality: Organic_carbon")
figure.show()
```



In [21]:
```python
data = data
figure = px.histogram(data, x ="Trihalomethanes",
                      color = "Potability",
                      title = "Factors affecting water quality: Trihalomethanes")
figure.show()
```



In [22]:
```python
correlation = data.corr()
```

In [23]:
```python
correlation.style.background_gradient('coolwarm')
```

Out[23]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
ph	1.000000	0.108948	-0.087615	-0.024768	0.010524	0.014128	0.028375	0.018278	-0.035849	0.014530
Hardness	0.108948	1.000000	-0.053269	-0.022685	-0.108521	0.011731	0.013224	-0.015400	-0.034831	-0.001505
Solids	-0.087615	-0.053269	1.000000	-0.051789	-0.162769	-0.005198	-0.005484	-0.015668	0.019409	0.040674
Chloramines	-0.024768	-0.022685	-0.051789	1.000000	0.006254	-0.028277	-0.023808	0.014990	0.013137	0.020784
Sulfate	0.010524	-0.108521	-0.162769	0.006254	1.000000	-0.016192	0.026776	-0.023347	-0.009934	-0.015303
Conductivity	0.014128	0.011731	-0.005198	-0.028277	-0.016192	1.000000	0.015647	0.004888	0.012495	-0.015496
Organic_carbon	0.028375	0.013224	-0.005484	-0.023808	0.026776	0.015647	1.000000	-0.005667	-0.015428	-0.015567
Trihalomethanes	0.018278	-0.015400	-0.015668	0.014990	-0.023347	0.004888	-0.005667	1.000000	-0.020497	0.009244
Turbidity	-0.035849	-0.034831	0.019409	0.013137	-0.009934	0.012495	-0.015428	-0.020497	1.000000	0.022682
Potability	0.014530	-0.001505	0.040674	0.020784	-0.015303	-0.015496	-0.015567	0.009244	0.022682	1.000000

In [24]:
```python
y = data["Potability"]
X = data.drop(["Potability"], axis= 1)
```

In [25]:
```python
from sklearn.preprocessing import StandardScaler
```

In [26]:
```python
scaler = StandardScaler()
scaler.fit(X)
```

Out[26]:
```
StandardScaler()
```

In [27]:
```python
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report, precision_score
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from warnings import filterwarnings
filterwarnings('ignore')
```

In [28]:
```python
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2,
                                                    random_state=42)
```

In [29]:
```python
models =[("LogisticReg", LogisticRegression(max_iter=1000)),("SVC", SVC()),
         ("DecisionTree", DecisionTreeClassifier()),("NaiveBayes", GaussianNB()),
         ('RandomForest',RandomForestClassifier()),('GradientBoost',GradientBoostingClassifier()),
         ('NeuralNetwork', MLPClassifier())]
```

In [30]:
```python
results = []
names = []
accuracies = []
finalResults = []

for name,model in models:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    score = precision_score(y_test, y_pred,average='macro')
    accuracyScore = accuracy_score(y_test, y_pred)
    accuracies.append(accuracyScore)
    results.append(score)
    names.append(name)
    finalResults.append((name,score, accuracyScore))
```

In [38]:
```python
finalResults_df = pd.DataFrame({'Model-Name': names, 'Accuracy-Score': accuracies, 'Precision-Score': results})
finalResults_df.sort_values(by='Accuracy-Score', ascending=False)
```

Out[38]:

	Model-Name	Accuracy-Score	Precision-Score
4	RandomForest	0.650124	0.647632
5	GradientBoost	0.645161	0.648925
2	DecisionTree	0.627792	0.616706
3	NaiveBayes	0.622829	0.636385
0	LogisticReg	0.573201	0.286600
1	SVC	0.573201	0.286600
6	NeuralNetwork	0.468983	0.514313

In [41]:
```python
rf_params = {"max_depth": [12, 14, 15, 17 ,19],
             "max_features": [3, 5, 7, 10, 11],
             "n_estimators": [100, 150, 180, 200, 300, 500],
             "min_samples_split": [11, 13, 15, 16, 18]}

rf = RandomForestClassifier()

rf_cv_model = GridSearchCV(rf, rf_params, cv=10, n_jobs=-1, verbose=2)

rf_cv_model.fit(X_train, y_train)
```

```
Fitting 10 folds for each of 750 candidates, totalling 7500 fits
```

Out[41]:
```
GridSearchCV(cv=10, estimator=RandomForestClassifier(), n_jobs=-1,
             param_grid={'max_depth': [12, 14, 15, 17, 19],
                         'max_features': [3, 5, 7, 10, 11],
                         'min_samples_split': [11, 13, 15, 16, 18],
                         'n_estimators': [100, 150, 180, 200, 300, 500]},
             verbose=2)
```

In [42]:
```python
rf_cv_model.best_params_
```

Out[42]:
```
{'max_depth': 12,
 'max_features': 7,
 'min_samples_split': 15,
 'n_estimators': 180}
```

In [39]:
```python
rf_tuned = RandomForestClassifier(max_depth = 12, max_features = 7, min_samples_split = 10, n_estimators = 100)
rf_tuned.fit(X_train, y_train)
```

Out[39]:
```
RandomForestClassifier(max_depth=10, max_features=8, min_samples_split=10)
```

In [40]:
```python
y_pred = rf_tuned.predict(X_test)
accuracy_score(y_test, y_pred)
```

Out[40]:
```
0.674937965260546
```

In [ ]: