一：Aim: to master the functions and methods used in the pandas process.

## Experiment content

In this experiment, we will work with pandas and we will do the following:

1. Working with large data using pandas where we Create, Read, Update and read (CRUD) the data
2. Wuxi weather condition of a week where we create the data and work with pandas to find and calculate the maximum, minimum, the average of the temperature and save the new data.

## Working with large data using pandas

In this section, we Create, Read, Update and read (CRUD) the data using pandas and our local csv file.

I**mporting pandas library**

import pandas as pd

# Create

df = pd.read\_csv('/Users/boom/Desktop/Sixth Experiment/elco\_churn.csvt')

## 1.2 Create from a Dictionary

tempdict = {'col1':[1,2,3], 'col2':[4,5,6], 'col3':[7,8,9]}

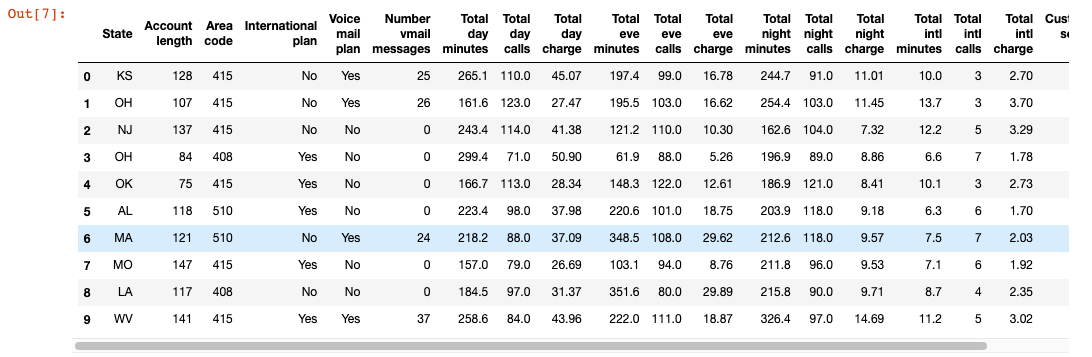
dictdf = pd.DataFrame.from\_dict(tempdict)

# 2. Read

## 2.1 Show Top 5 and Bottom 5 Rows

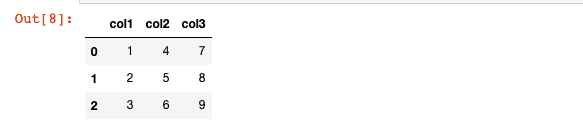
df.head(10)

**Output**



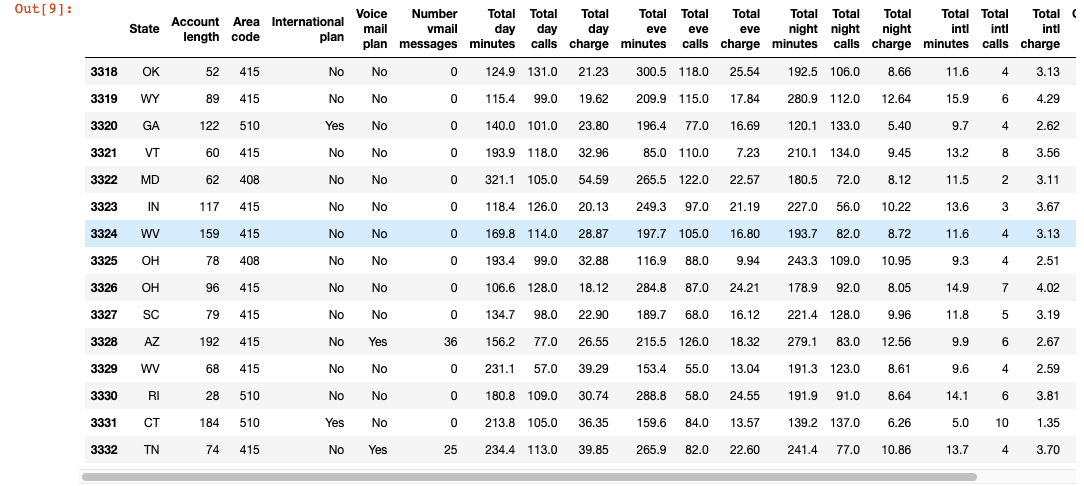
dictdf.head()

**Output**



df.tail(15)

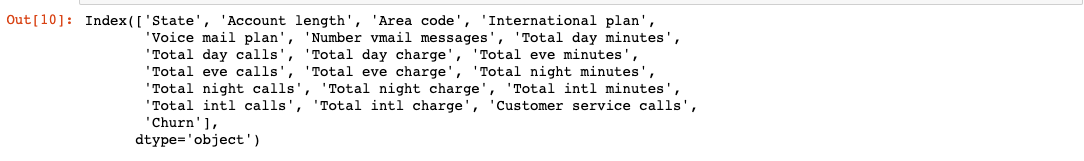
**Output**



## 2.2 Show Columns and Data Type

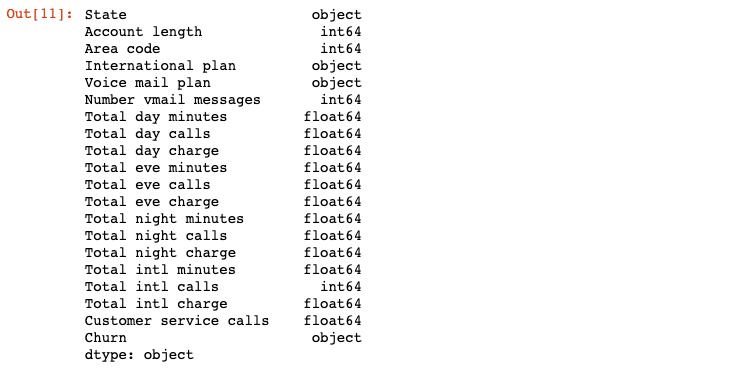
df.columns

**Output**



df.dtypes

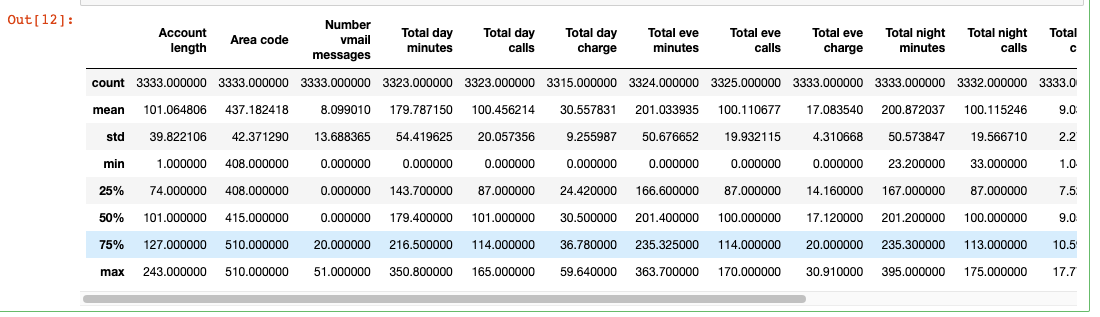
**Output**



## 2.3 Summary Statistics

df.describe()

**Output**



df.describe(*include*='object')

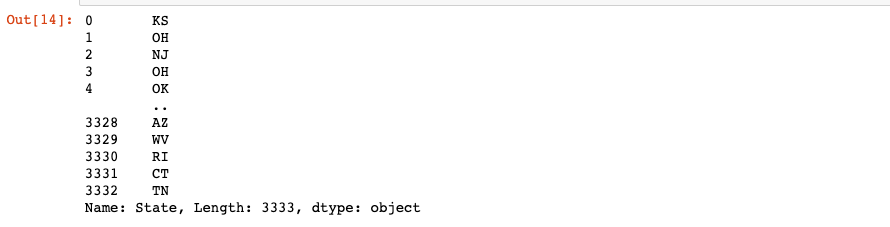
**Outpu**t



## 2.4 Filtering Columns

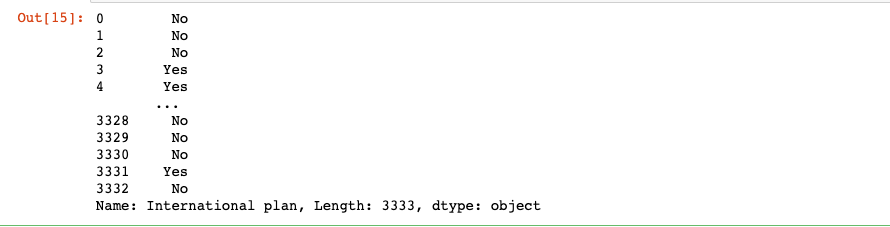
df.State

**Output**



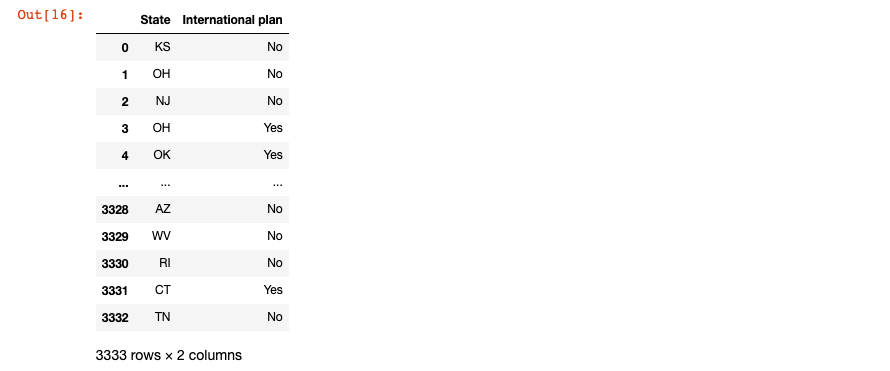
df['International plan']

**Output**



df[['State', 'International plan']]

**Outpu**t:



df.Churn.unique()

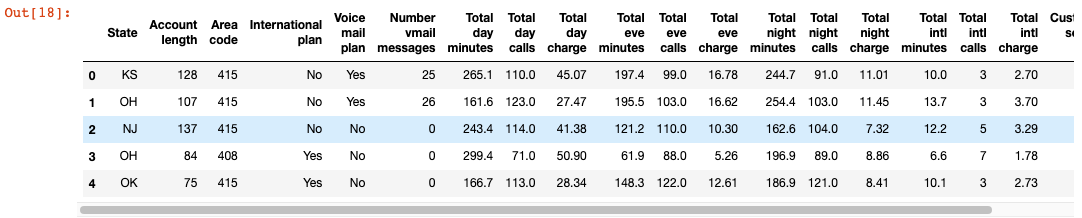
**Output**



## 2.5 Filtering on Rows

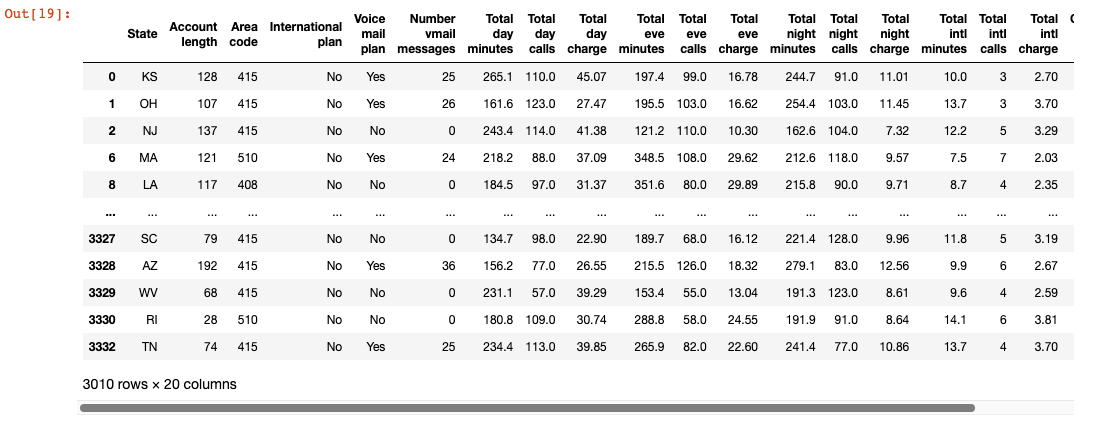
df.head()

**Output:**



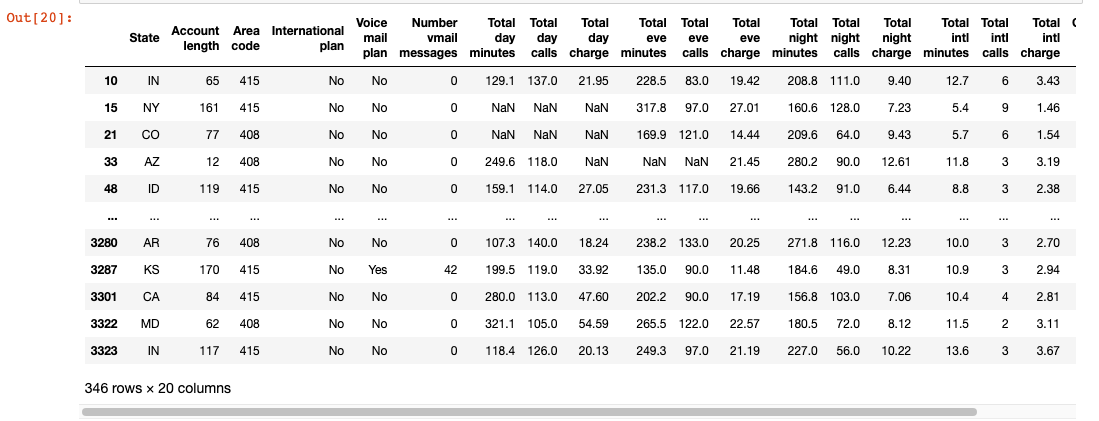
df[df['International plan']=='No']

**Output:**



df[(df['International plan']=='No') & (df['Churn']==True)]

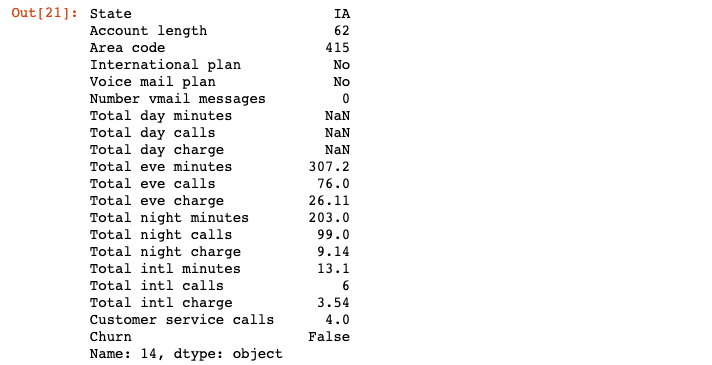
**Output:**



## 2.6 Indexing with iloc

df.iloc[14]

**Output:**



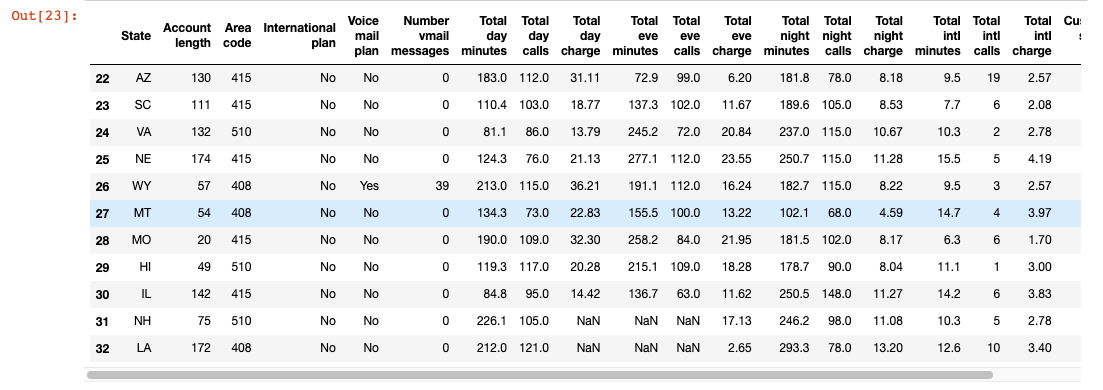
df.iloc[14,-1]

**Output:**



df.iloc[22:33]

**Output:**



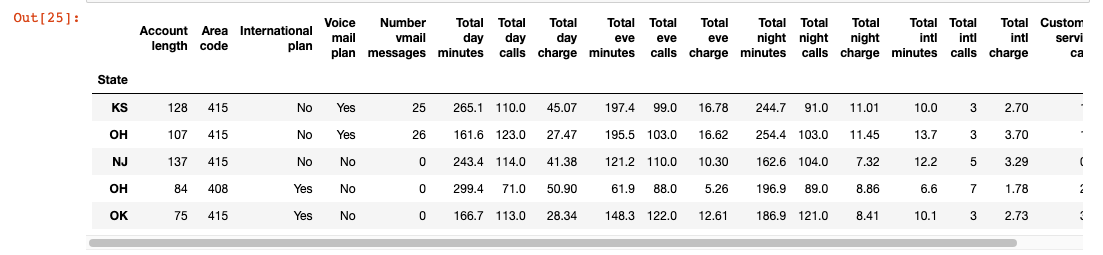
## 2.7 Indexing with loc

state = df.copy()

state.set\_index('State', *inplace*=True)

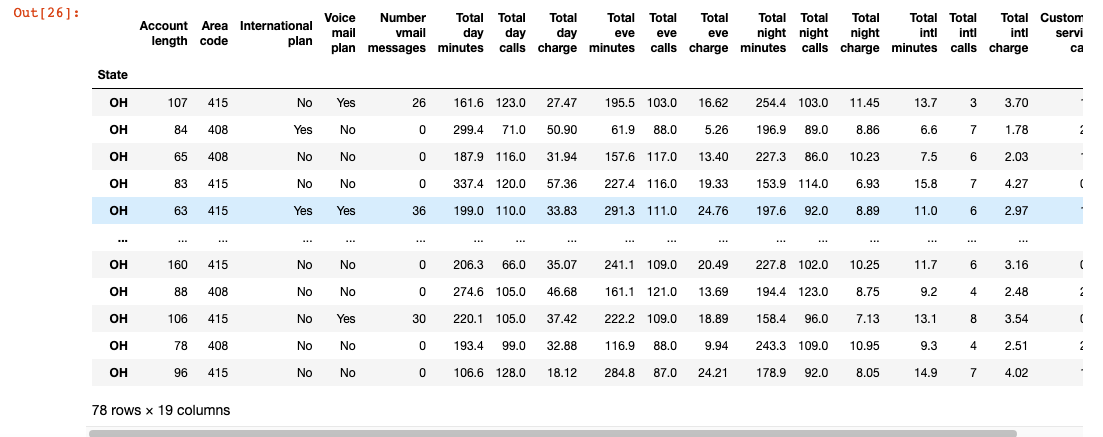
state.head()

**Output:**



state.loc['OH']

**Output:**

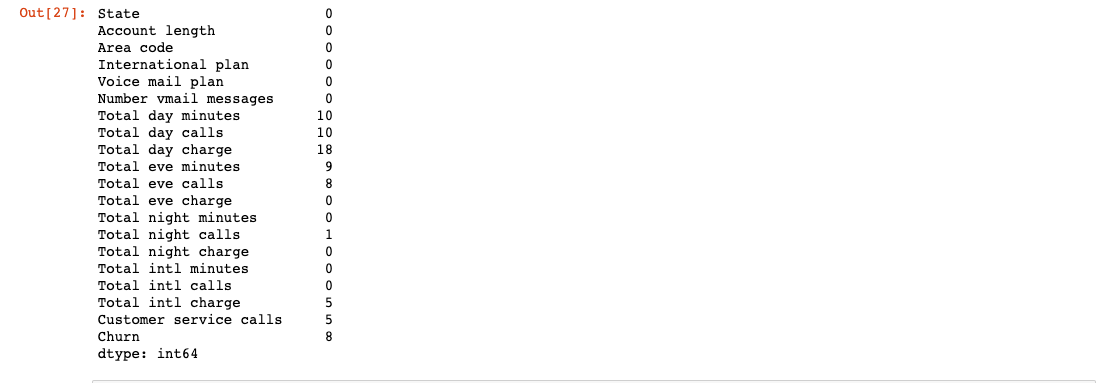


# Update

## 3.1 Dropping Rows

df.isnull().sum()

**Output:**



df.dropna(*inplace*=True)

df.isnull().sum()

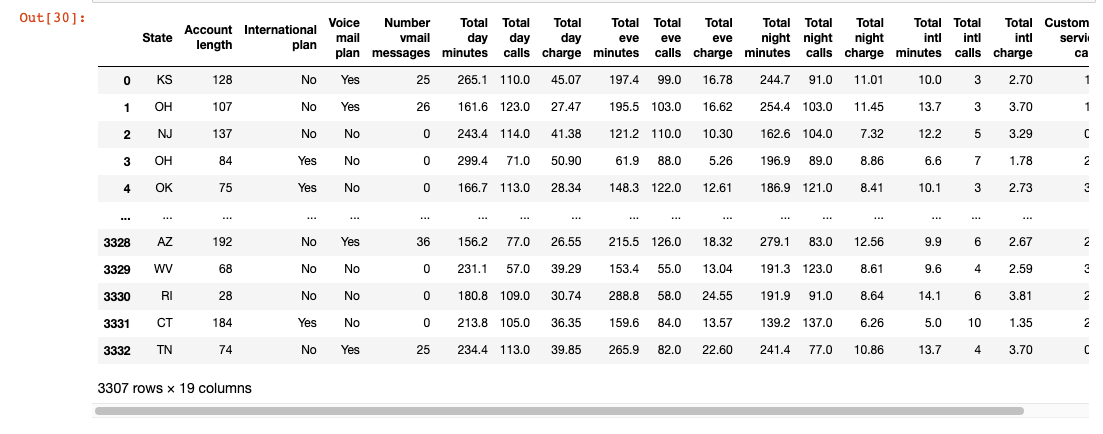
**Output:**



## 3.2 Dropping Columns

df.drop('Area code', *axis*=1)

**Output:**

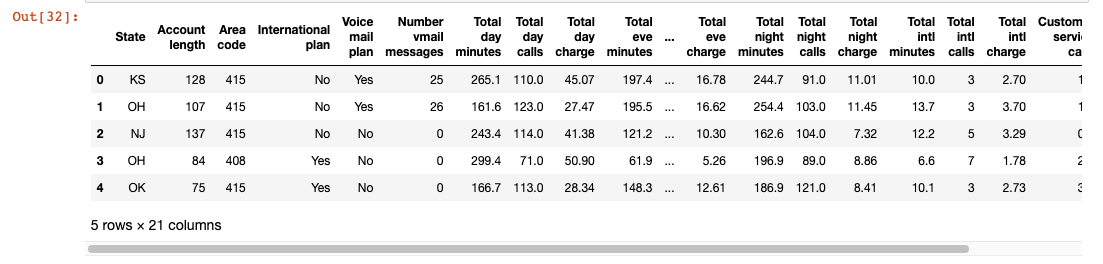


## 3.3 Creating Calculated Columns

df['New Column'] = df['Total night minutes'] + df['Total intl minutes']

df.head()

**Output:**

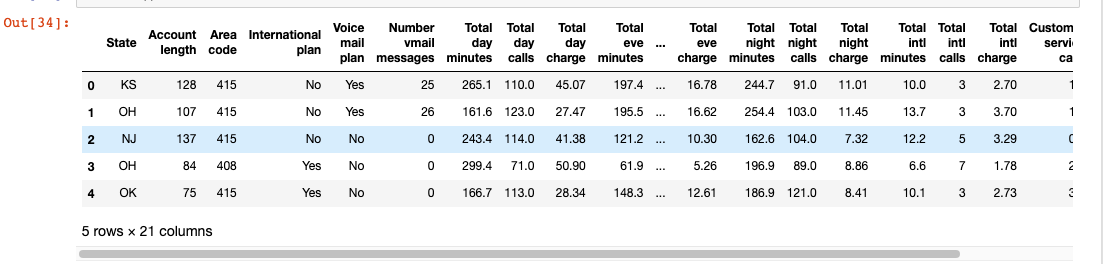


## 3.4 Updating an Entire Column

df['New Column'] = 100

df.head()

**Output:**

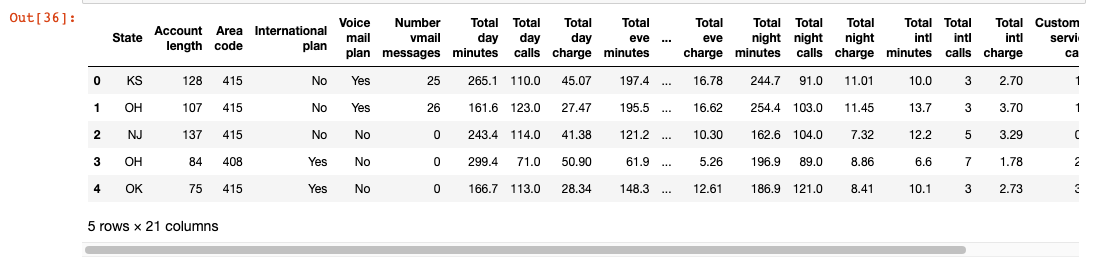


## 3.5 Updating a Single Value

df.iloc[0,-1] = 10

df.head()

**Output:**

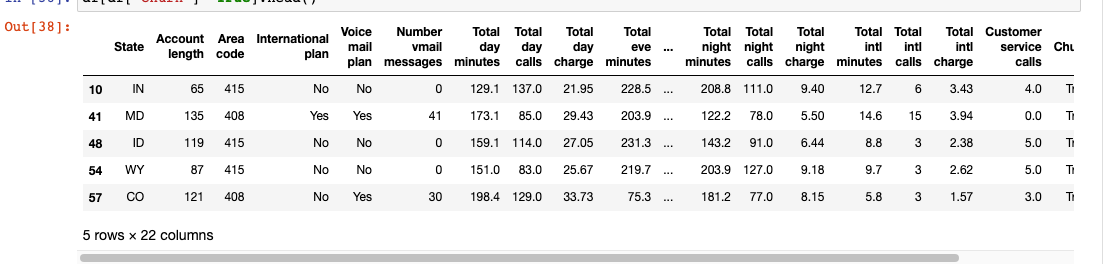


## 3.6 Condition based Updating using Apply

df['Churn Binary'] = df['Churn'].apply(*lambda* *x*: 1 if *x*==True else 0)

df[df['Churn']==True].head()

**Output:**



# 4. Delete/Output

## 4.1 Output to CSV

df.to\_csv('/Users/boom/Desktop/Sixth Experiment/output.csv')

**Output:**

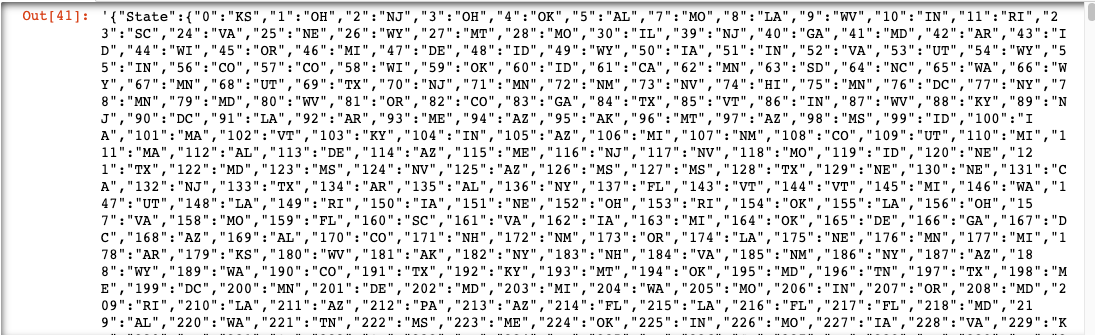
File created in the folder sixth Experiment



## 4.2 Output to JSON

df.to\_json()

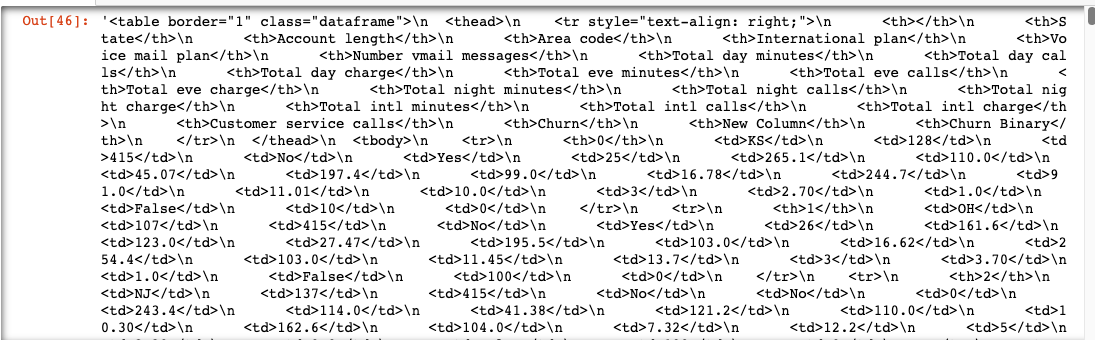
**Output:**



## 4.3 Output to HTML

df.to\_html()

**Output:**



## 4.4 Delete a DataFrame

del df

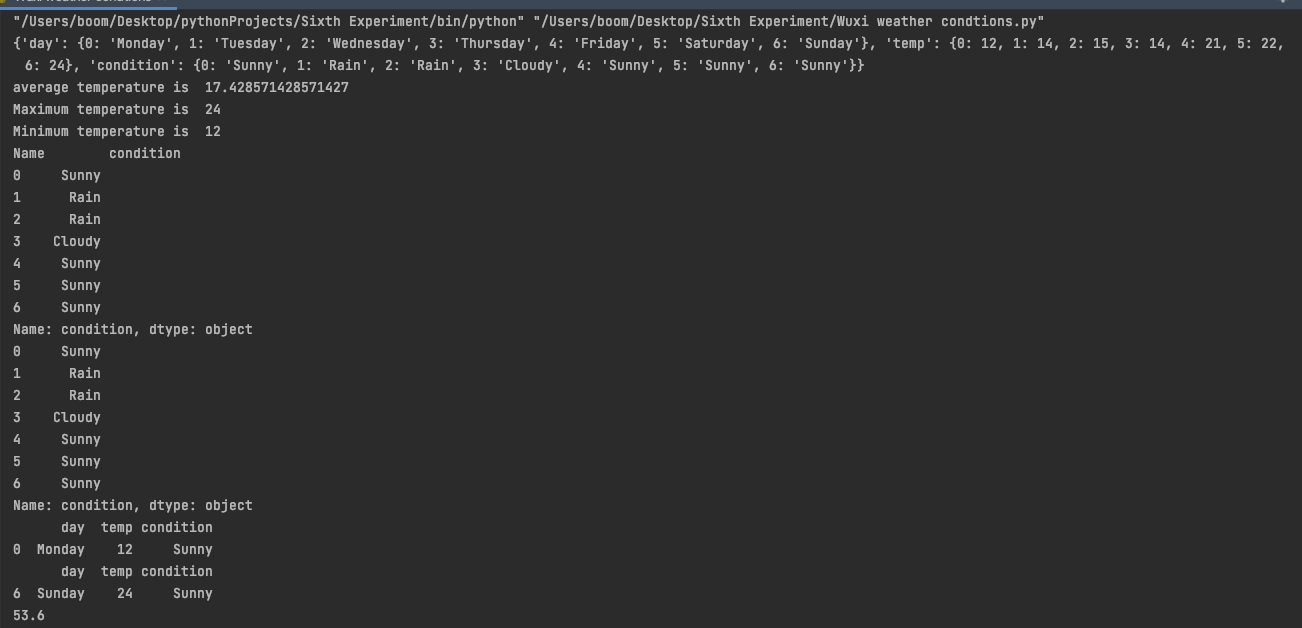
1. **Wuxi weather condition of a week**

In this section, we create the data and work with pandas using our local csv file to find and calculate the maximum, minimum, the average of the temperature and save the new file.

### **The program**

import pandas  
data = pandas.read\_csv("Wuxi\_weather\_data.csv")  
data\_dict = data.to\_dict()  
print(data\_dict)  
temp\_list = data["temp"].to\_list()  
print('average temperature is ', data["temp"].mean())  
print('Maximum temperature is ',data["temp"].max())  
print('Minimum temperature is ',data["temp"].min())  
#Get Data in Columns  
print('Name\t\tcondition')  
print(data["condition"])  
print(data.condition)  
# Get Data in Row  
print(data[data.day == "Monday"])  
print(data[data.temp == data.temp.max()])  
  
# Get Row data value  
monday = data[data.day == "Monday"]  
monday\_temp = int(monday.temp)  
monday\_temp\_F = monday\_temp \* 9/5 + 32  
print(monday\_temp\_F)  
# Create a dataframe from scratch  
data = pandas.DataFrame(data\_dict)  
data.to\_csv("new\_data.csv")

**Output**



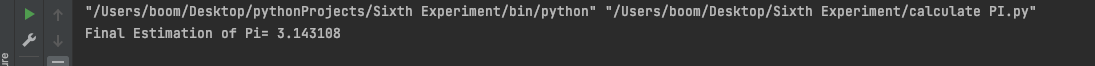
二：A small experiment based on monte carlo algorithm is implemented in Python

1. Calculate the value of PI

### **The program**

import random  
  
INTERVAL = 1000  
circle\_points = 0  
square\_points = 0  
# Total Random numbers generated= possible x  
# values\* possible y values  
for i in range(INTERVAL\*\*2):  
# Randomly generated x and y values from a  
# uniform distribution  
# Range of x and y values is -1 to 1  
 rand\_x = random.uniform(-1, 1)  
 rand\_y = random.uniform(-1, 1)  
# Distance between (x, y) from the origin  
 origin\_dist = rand\_x\*\*2 + rand\_y\*\*2  
# Checking if (x, y) lies inside the circle  
 if origin\_dist <= 1:  
 circle\_points += 1  
 square\_points += 1  
# Estimating value of pi,  
# pi= 4\*(no. of points generated inside the  
# circle)/ (no. of points generated inside the square)  
 pi = 4 \* circle\_points / square\_points  
  
## print(rand\_x, rand\_y, circle\_points, square\_points, "-", pi)  
# print("\n")  
print("Final Estimation of Pi=", pi)

**Output**



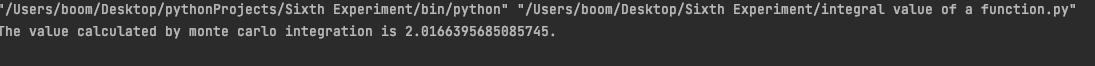
Final Estimation of Pi= 3.143108

1. Calculate the integral value of the function

**The program**

# importing the modules  
from scipy import random  
import numpy as np  
  
# limits of integration  
a = 0  
b = np.pi # gets the value of pi  
N = 1000  
  
# array of zeros of length N  
ar = np.zeros(N)  
  
# iterating over each Value of ar and filling  
# it with a random value between the limits a  
# and b  
for i in range (len(ar)):  
 ar[i] = random.uniform(a,b)  
  
# variable to store sum of the functions of  
# different values of x  
integral = 0.0  
  
# function to calculate the sin of a particular  
# value of x  
def f(x):  
 return np.sin(x)  
  
# iterates and sums up values of different functions  
# of x  
for i in ar:  
 integral += f(i)  
  
# we get the answer by the formula derived adobe  
ans = (b-a)/float(N)\*integral  
  
# prints the solution  
print ("The value calculated by monte carlo integration is {}.".format(ans))

**Output**



The value calculated by monte carlo integration is 2.0166395685085745.