dv14zvnhv

September 12, 2024

[3]:
```
import pandas as pd
import numpy as np
```

[6]:
```
Employee_data = pd.DataFrame({
    'EmployeeID': [101, 102, 103, 104, 105],
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
    'Department': ['HR', 'IT', 'Finance', 'IT', 'HR'],
    'Salary': [50000, 60000, 55000, 65000, 58000]
})
```

[9]:
```
print(Employee_data[['Name','Salary']])
```

```
      Name  Salary
0    Alice   50000
1      Bob   60000
2  Charlie   55000
3    David   65000
4      Eva   58000
```

[15]:
```
average_salary = Employee_data[Employee_data['Department']=='IT']['Salary'].
 ↪mean()
print(average_salary)
```

```
62500.0
```

[16]:
```
Employee_data['Bonus'] = Employee_data['Salary']*0.1
print(Employee_data)
```

```
   EmployeeID     Name Department  Salary   Bonus
0         101    Alice         HR   50000  5000.0
1         102      Bob         IT   60000  6000.0
2         103  Charlie    Finance   55000  5500.0
3         104    David         IT   65000  6500.0
4         105      Eva         HR   58000  5800.0
```

[19]:
```
Employee_data.rename(columns={'EmployeeID': 'ID'}, inplace=True)
print(Employee_data)
```

```
    ID     Name Department  Salary   Bonus
0  101    Alice         HR   50000  5000.0
1  102      Bob         IT   60000  6000.0
2  103  Charlie    Finance   55000  5500.0
3  104    David         IT   65000  6500.0
4  105      Eva         HR   58000  5800.0
```

[24]:
```
print(Employee_data)
```

```
    ID     Name  Salary   Bonus
0  101    Alice   50000  5000.0
1  102      Bob   60000  6000.0
2  103  Charlie   55000  5500.0
3  104    David   65000  6500.0
4  105      Eva   58000  5800.0
```

[25]:
```
sales_data = pd.DataFrame({
    'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May'],
    'Product_A': [150, 200, 250, 300, 350],
    'Product_B': [300, 250, 200, 150, 100]
})
```

[26]:
```
total_sales_A = sales_data['Product_A'].sum()
total_sales_B = sales_data['Product_B'].sum()
print(total_sales_A)
print(total_sales_B)
```

```
1250
1000
```

[28]:
```
highest_sales_A_mon
th = sales_data.loc[sales_data['Product_A'].idxmax(), 'Month']
print(highest_sales_A_month)
```

```
May
```

[30]:
```
sales_data['Total_sales'] = sales_data['Product_A']+sales_data['Product_B']
print(sales_data)
```

```
  Month  Product_A  Product_B  Total_sales
0   Jan        150        300          450
1   Feb        200        250          450
2   Mar        250        200          450
3   Apr        300        150          450
4   May        350        100          450
```

[31]:
```
sales_data['Normalized_Product_A'] = sales_data['Product_A']/␣
↪sales_data['Product_A'].max()
```

```
print(sales_data)
```

```
  Month  Product_A  Product_B  Total_sales  Normalized_Product_A
0   Jan        150        300          450              0.428571
1   Feb        200        250          450              0.571429
2   Mar        250        200          450              0.714286
3   Apr        300        150          450              0.857143
4   May        350        100          450              1.000000
```

[33]:
```
highest_sales = sales_data[sales_data['Total_sales'] > 400]
print(highest_sales)
```

```
  Month  Product_A  Product_B  Total_sales  Normalized_Product_A
0   Jan        150        300          450              0.428571
1   Feb        200        250          450              0.571429
2   Mar        250        200          450              0.714286
3   Apr        300        150          450              0.857143
4   May        350        100          450              1.000000
```

[34]:
```
customer_data = pd.DataFrame({
    'CustomerID': [1, 2, 3, 4, 5],
    'Name': ['John', 'Jane', None, 'Alice', 'Bob'],
    'Purchase': [200, 300, 150, None, 500]
})
```

[35]:
```
missing_values = customer_data.isnull().sum()
print(missing_values)
```

```
CustomerID    0
Name          1
Purchase      1
dtype: int64
```

[36]:
```
customer_data['Name'].fillna('Unknown', inplace = True)
print(customer_data)
```

```
   CustomerID     Name  Purchase
0           1     John     200.0
1           2     Jane     300.0
2           3  Unknown     150.0
3           4    Alice       NaN
4           5      Bob     500.0
```

```
C:\Users\HDC0422079\AppData\Local\Temp\ipykernel_12472\1221914968.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
```

```
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  customer_data['Name'].fillna('Unknown', inplace = True)
```

[38]:
```
customer_data['Purchase'].fillna(customer_data['Purchase'].median(), inplace =␣
↪True)
print(customer_data)
```

```
   CustomerID     Name  Purchase
0           1     John     200.0
1           2     Jane     300.0
2           3  Unknown     150.0
3           4    Alice     250.0
4           5      Bob     500.0
C:\Users\HDC0422079\AppData\Local\Temp\ipykernel_12472\587324257.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  customer_data['Purchase'].fillna(customer_data['Purchase'].median(), inplace =
True)
```

[39]:
```
customer_data.dropna(inplace = True)
```

[40]:
```
print(customer_data)
```

```
   CustomerID     Name  Purchase
0           1     John     200.0
1           2     Jane     300.0
2           3  Unknown     150.0
3           4    Alice     250.0
4           5      Bob     500.0
```

[41]:
```
median_purchase = customer_data['Purchase'].median()
```

```
high_purchase_customers = customer_data[customer_data['Purchase'] >␣
  ↪median_purchase]
print(high_purchase_customers)
```

```
   CustomerID  Name  Purchase
1           2  Jane     300.0
4           5   Bob     500.0
```

[42]:
```
temperature = pd.Series([23, 21, 20, 25, 27, 30, 28, 22, 24, 26])
```

[44]:
```
mean_temp = temperature.mean()
median_temp = temperature.median()
variance = temperature.var()
print(mean_temp)
print(median_temp)
print(variance)
```

```
24.6
24.5
10.266666666666666
```

[45]:
```
above_mean = temperature > mean_temp
print(above_mean)
```

```
0    False
1    False
2    False
3     True
4     True
5     True
6     True
7    False
8    False
9     True
dtype: bool
```

[46]:
```
temperature_Kelvin = temperature + 273.15
print(temperature_Kelvin)
```

```
0    296.15
1    294.15
2    293.15
3    298.15
4    300.15
5    303.15
6    301.15
7    295.15
8    297.15
```

```
9    299.15
dtype: float64
```

[48]:
```
rolling_mean = temperature.rolling(window = 3).mean()
print(rolling_mean)
```

```
0          NaN
1          NaN
2    21.333333
3    22.000000
4    24.000000
5    27.333333
6    28.333333
7    26.666667
8    24.666667
9    24.000000
dtype: float64
```

[49]:
```
std_dev = temperature.std()
filtered_temperatures = temperature[(temperature < mean_temp - std_dev) |␣
 ↪(temperature > mean_temp + std_dev)]
print(filtered_temperatures)
```

```
1    21
2    20
5    30
6    28
dtype: int64
```

[50]:
```
orders = pd.DataFrame({
    'OrderID': [1, 2, 3, 4, 5],
    'CustomerID': [101, 102, 103, 104, 101],
    'Product': ['A', 'B', 'A', 'C', 'B'],
    'Quantity': [2, 1, 4, 2, 3]
})
```

[51]:
```
customers = pd.DataFrame({
    'CustomerID': [101, 102, 103, 104],
    'Name': ['Alice', 'Bob', 'Charlie', 'David'],
    'Location': ['New York', 'Los Angeles', 'Chicago', 'Houston']
})
```

[53]:
```
merged_df = pd.merge(orders, customers, on='CustomerID')
print(merged_df)
```

```
   OrderID  CustomerID Product  Quantity     Name     Location
0        1         101       A         2    Alice     New York
1        2         102       B         1      Bob  Los Angeles
```

```
2         3         103       A         4  Charlie      Chicago
3         4         104       C         2    David      Houston
4         5         101       B         3    Alice     New York
```

[56]:
```
quantity_by_location = merged_df.groupby('Product')['Quantity'].sum().
↪reset_index()
print(quantity_by_location)
```

```
  Product  Quantity
0       A         6
1       B         4
2       C         2
```

[63]:
```
highest_order_customer = merged_df['CustomerID'].value_counts().idxmax()
highest_order_customer_name = customers[customers['CustomerID'] ==␣
↪highest_order_customer]['Name'].values[0]
print(highest_order_customer_name)
```

```
Alice
```

[66]:
```
merged_df['Total_Quantity'] = merged_df.groupby('Product')['Quantity'].cumsum()
print(merged_df)
```

```
   OrderID  CustomerID Product  Quantity     Name     Location  Total_Quantity
0        1         101       A         2    Alice     New York               2
1        2         102       B         1      Bob  Los Angeles               1
2        3         103       A         4  Charlie      Chicago               6
3        4         104       C         2    David      Houston               2
4        5         101       B         3    Alice     New York               4
```

[67]:
```
filtered_orders = merged_df[merged_df['Product'].isin(['A','B'])]
print(filtered_orders)
```

```
   OrderID  CustomerID Product  Quantity     Name     Location  Total_Quantity
0        1         101       A         2    Alice     New York               2
1        2         102       B         1      Bob  Los Angeles               1
2        3         103       A         4  Charlie      Chicago               6
4        5         101       B         3    Alice     New York               4
```

[ ]:
```

```