Count of Categorical Values

DataFrame:

Out[12]:

	Name	Age	Gender	Salary	City
0	John	25	Male	50000	New York
1	Alice	30	Female	60000	Los Angeles
2	Bob	35	Male	70000	Chicago
3	Emily	28	Female	55000	San Francisco
4	David	40	Male	75000	Boston

Correlation Heatmap
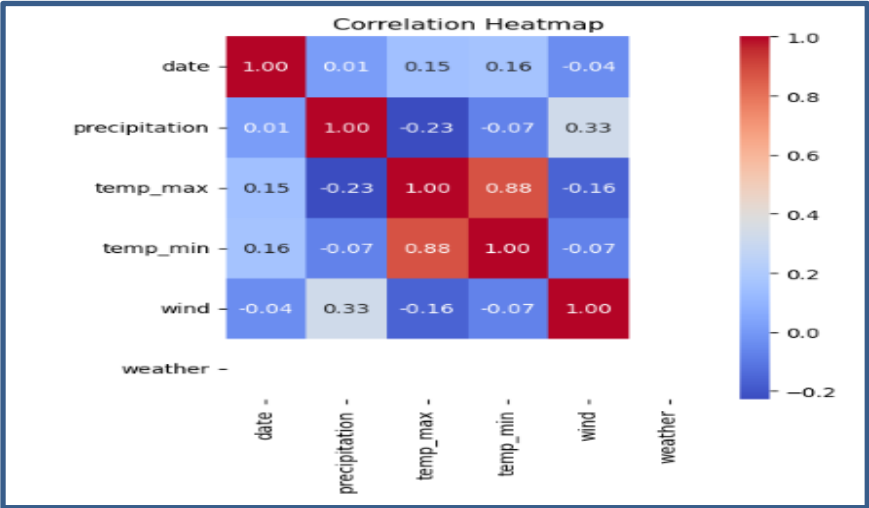
```
In [2]:  # from functools import reduce

         # Dictionary of students' grades
         student_grades = {
             "Alice": [85, 90, 92],
             "Bob": [70, 65, 80],
             "Charlie": [55, 60, 58]
         }
         print("Data in dictionary :", student_grades)

         # Higher-order function - Map: Add 5 bonus marks to each student's grades
         updated_grades = {name: list(map(lambda x: x + 5, grades)) for name, grades in student_grades.items()}
         print("Updated grades:", updated_grades)

         # Higher-order function - Filter: Find students who passed (average grade >= 60)
         passed_students = list(filter(lambda x: sum(x[1]) / len(x[1]) >= 60, student_grades.items()))
         print("Students who passed:", passed_students)

         # Higher-order function - Reduce: Calculate the total number of students
         total_students = reduce(lambda x, _: x + 1, student_grades, 0)
         print("Total number of students:", total_students)


         Data in dictionary : {'Alice': [85, 90, 92], 'Bob': [70, 65, 80], 'Charlie': [55, 60, 58]}
         Updated grades: {'Alice': [90, 95, 97], 'Bob': [75, 70, 85], 'Charlie': [60, 65, 63]}
         Students who passed: [('Alice', [85, 90, 92]), ('Bob', [70, 65, 80])]
         Total number of students: 3
```

```
In [1]:  from functools import reduce

         # List of students' ages
         ages = [18, 21, 19, 22, 20, 23]
         print("List of student ages :", ages)

         # Higher-order function - Map: Calculate age after 5 years
         ages_after_5_years = list(map(lambda x: x + 5, ages))
         print("Ages after 5 years:", ages_after_5_years)

         # Higher-order function - Filter: Find students above 20 years old
         above_20 = list(filter(lambda x: x > 20, ages))
         print("Students above 20 years old:", above_20)

         # Higher-order function - Reduce: Calculate average age
         average_age = reduce(lambda x, y: x + y, ages) / len(ages)
         print("Average age of students:", average_age)


         List of student ages : [18, 21, 19, 22, 20, 23]
         Ages after 5 years: [23, 26, 24, 27, 25, 28]
         Students above 20 years old: [21, 22, 23]
         Average age of students: 20.5
```
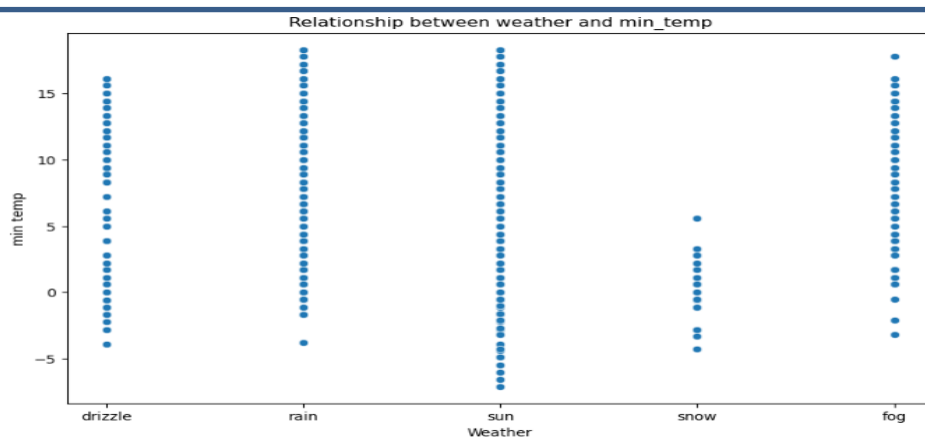

Relationship between weather and min_temp

```
Basic EDA:
1. Summary Statistics:
            Age        Salary
count    5.00000      5.000000
mean    31.60000  62000.000000
std      5.94138  10368.220677
min     25.00000  50000.000000
25%     28.00000  55000.000000
50%     30.00000  60000.000000
75%     35.00000  70000.000000
max     40.00000  75000.000000
```