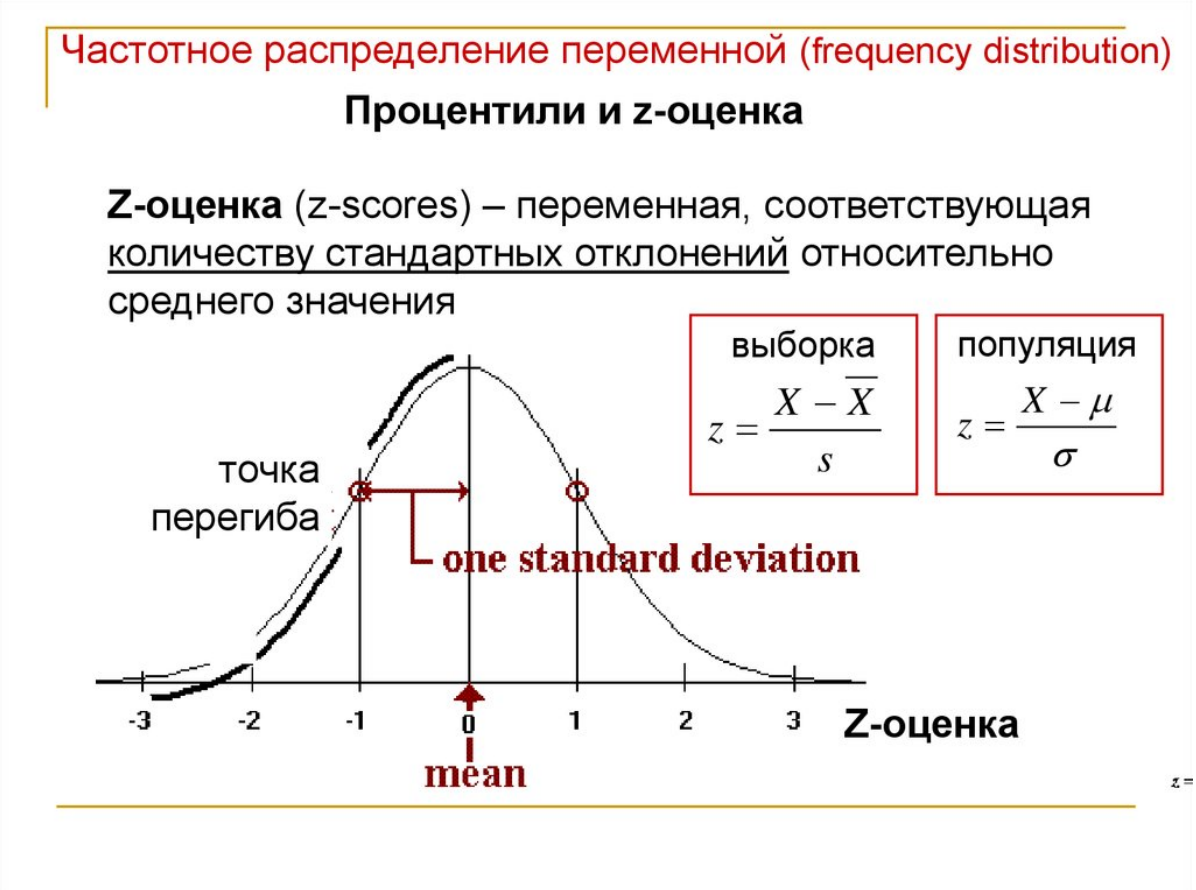
Министерство цифрового развития, связи и массовых коммуникаций
Российской Федерации
Ордена Трудового Красного Знамени федеральное государственное
бюджетное образовательное учреждение высшего образования
«Московский технический университет связи и информатики»
Кафедра «МКиИТ»

Лабораторная работа №2

по дисциплине «Data mining»

Москва 2023

Стандартизированная оценка (z-оценка) — это относительная мера, которая показывает, на сколько среднеквадратичных отклонений наблюдаемое значение отличается от среднего значения распределения. Знак z-оценки показывает, находится ли значение левее среднего (–) или правее среднего (+).



Выброс — это аномальное значение в данных, которое значительно отличается от значения, выраженного мерой центральной тенденции.

Z-оценка

In [7]:
```
import pandas as pd
import numpy as np
import scipy.stats
%matplotlib inline
import matplotlib.pyplot as plt
import scipy.stats as stats
```

Загрузите датасет eng_test.csv. Обратите внимание на сепаратор (sep=';'). Выведите первые 5 элементов

In [8]:
```
data =  pd.read_csv('./eng_test.csv',  sep=';')
data.head(5)
```

Out[8]:

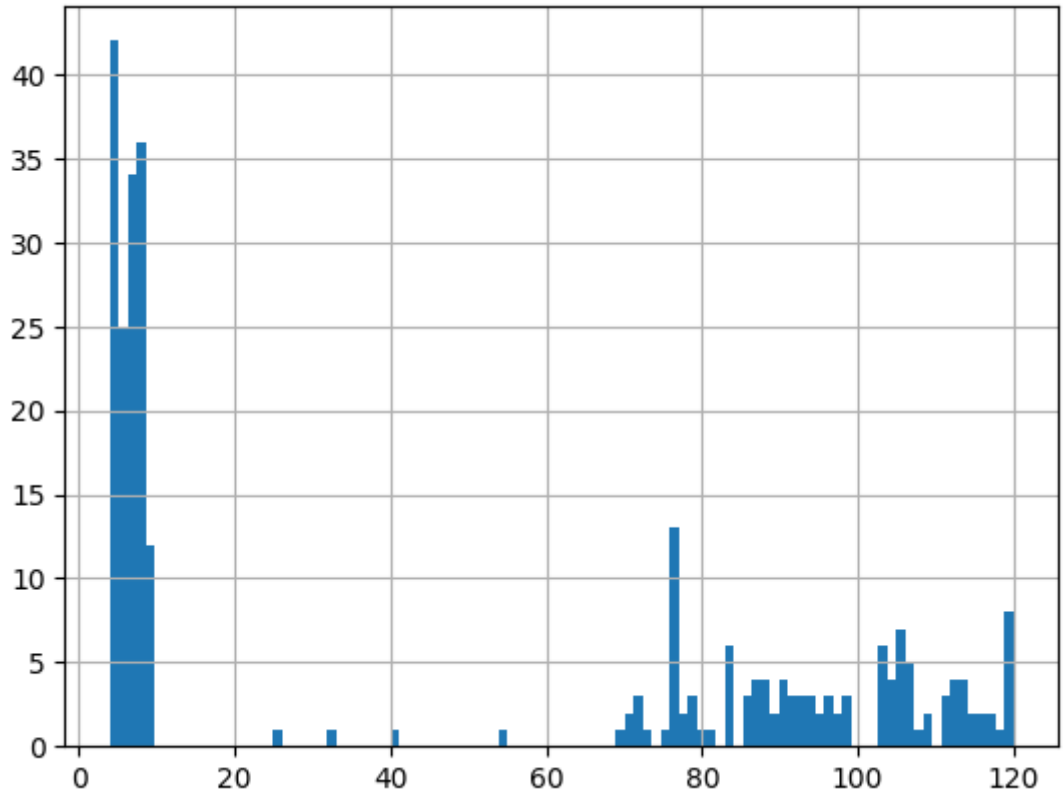
	Id	Exam	Score	Advanced
0	ID1	TOEFL	77.0	NO
1	ID10	TOEFL	105.0	NO
2	ID100	TOEFL	107.0	YES

	Id	Exam	Score	Advanced
3	ID101	TOEFL	72.0	NO
4	ID102	TOEFL	120.0	YES

Постойте гистограмму по оценкам студентов

In [9]:
```
data.Score.hist(bins = 100)
```

Out[9]: <Axes: >



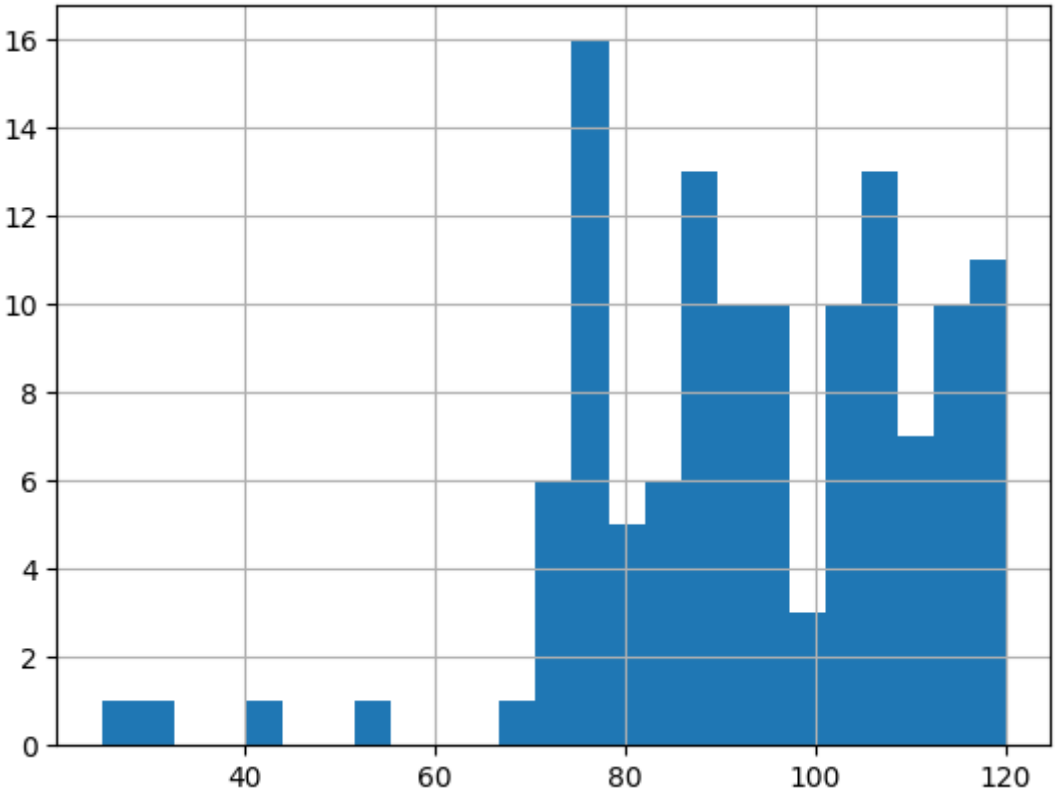
Создайте переменную, содержащую информацию только об оценках TOEFL и выведите гисограмму и основные статистики

In [10]:
```
toefl = data[data['Exam'] == 'TOEFL']
toefl.Score.hist(bins = 25)
print(len(toefl))
toefl.head(5)
```

125

Out[10]:

	Id	Exam	Score	Advanced
0	ID1	TOEFL	77.0	NO
1	ID10	TOEFL	105.0	NO
2	ID100	TOEFL	107.0	YES
3	ID101	TOEFL	72.0	NO
4	ID102	TOEFL	120.0	YES



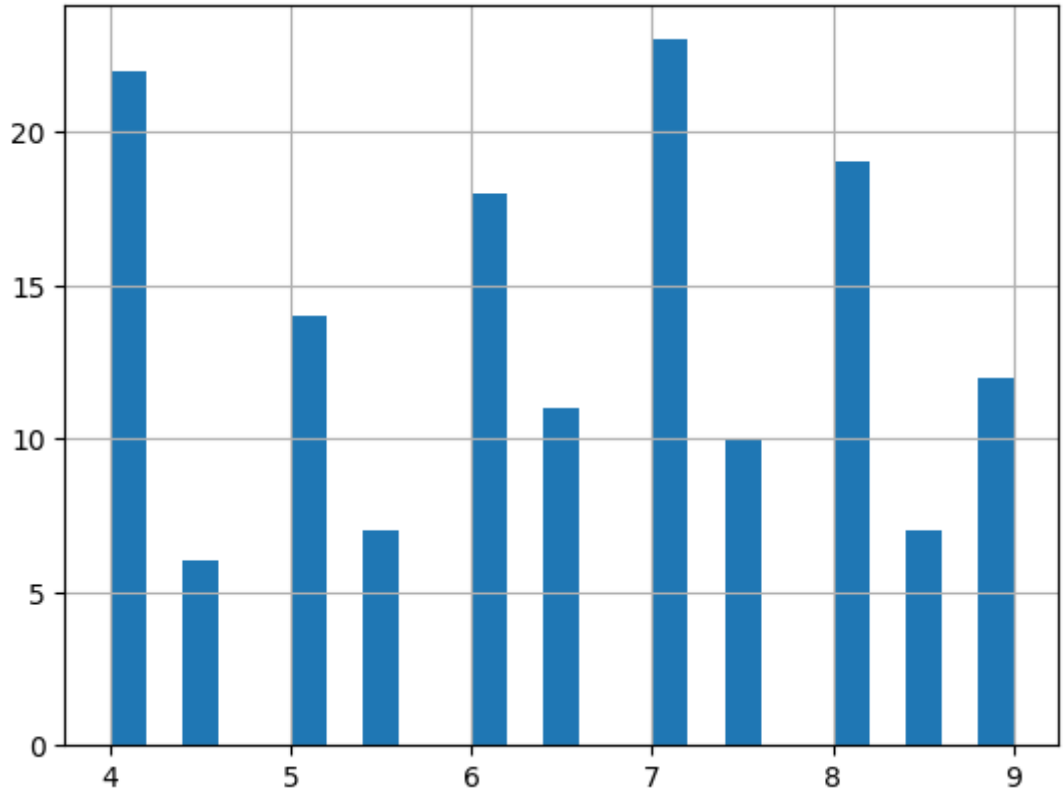
Создайте переменную, содержащую информацию только об оценках IELTS и выведите гисограмму и основные статистики

In [11]:
```
ielts = data[data['Exam'] == 'IELTS']
ielts.Score.hist(bins = 25)
print(len(ielts))
ielts.head(5)
```

149

Out[11]:

	Id	Exam	Score	Advanced
30	ID126	IELTS	9.0	YES
31	ID127	IELTS	5.0	NO
32	ID128	IELTS	7.5	YES
33	ID129	IELTS	5.0	NO
35	ID130	IELTS	4.0	NO



Загрузите датасет eng_test.csv. Обратите внимание на сепаратор (sep=';')

In [12]:
```
data =  pd.read_csv('./eng_test.csv',  sep=';')
data.head(5)
```

Out[12]:

	Id	Exam	Score	Advanced
0	ID1	TOEFL	77.0	NO
1	ID10	TOEFL	105.0	NO
2	ID100	TOEFL	107.0	YES
3	ID101	TOEFL	72.0	NO
4	ID102	TOEFL	120.0	YES

Посчитайте z-score для первого студента в списке toefl. Также выведите стандартное отклонение, среднее и само кол-во баллов.

In [13]:
```
print("z-score для первого студента в списке toefl", stats.zscore(toefl.Score)[0])
print("Стандартное отклонение",np.std(toefl.Score))
print("Среднее", np.average(toefl.Score))
print("\"Само кол-во баллов\"", toefl.Score[0])
```

```
z-score для первого студента в списке toefl -0.9407782522297515
Стандартное отклонение 17.832044863110905
Среднее 93.776
"Само кол-во баллов" 77.0
```

In [15]:
```
stats.zscore(toefl['Score'])   # Z-score через библиотеку scipy.stats
```

Out[15]:
```
0     -0.940778
1      0.629429
2      0.741586
```

```
3     -1.221172
4      1.470611
          ...
269   -0.323911
270   -0.211754
271   -0.211754
272   -2.230591
273    0.685507
Name: Score, Length: 125, dtype: float64
```

сохраните в переменные Z-score для ielts и toefl

In [18]:
```
ielts["z-score"] = stats.zscore(ielts.Score)
toefl["z-score"] = stats.zscore(toefl.Score)
```

In [19]:
```
eng = pd.concat([toefl, ielts]) #собираем результаты обратно в 1 датафрейм
```

In [20]:
```
eng[eng['z-score'] < -3] # кто написал экзамен хуже, чем 3 стандартных отклонения
```

Out[20]:

	Id	Exam	Score	Advanced	z-score
25	ID121	TOEFL	32.0	NO	-3.464325
27	ID123	TOEFL	25.0	YES	-3.856877

In [21]:
```
eng.groupby('Advanced')['z-score'].mean() #кто сдал экзамен лучше? Кто брал продвину
```

Out[21]:
```
Advanced
NO    -0.397672
YES    0.440499
Name: z-score, dtype: float64
```

Выбросы

Разберем, как выбросы влияют на меры центральной тенденции

In [22]:
```
import pandas as pd
import numpy as np

bikes = pd.read_pickle('BikesDataVars.pkl')
bikes.head()
```

Out[22]:

	Date	Hour	Temperature	Humidity	Wind speed	Rainfall	Snowfall	Seasons	Holiday	Functioning Day
0	2017-12-01	0	-5.2	37	2.2	0.0	0.0	Winter	0	True
1	2017-12-01	1	-5.5	38	0.8	0.0	0.0	Winter	0	True
2	2017-12-01	2	-6.0	39	1.0	0.0	0.0	Winter	0	True
3	2017-12-01	3	-6.2	40	0.9	0.0	0.0	Winter	0	True

	Date	Hour	Temperature	Humidity	Wind speed	Rainfall	Snowfall	Seasons	Holiday	Functioning Day
4	2017-12-01	4	-6.0	36	2.3	0.0	0.0	Winter	0	True

In [23]:
```
bikes['Rental Count'].describe()
```

Out[23]:
```
count    8760.000000
mean      696.582078
std       749.812613
min         0.000000
25%       157.000000
50%       425.500000
75%      1009.000000
max      6012.000000
Name: Rental Count, dtype: float64
```

Найдите интерквартильный размах по атрибуту 'Rental Count'

In [24]:
```
iqr = np.percentile(bikes['Rental Count'], 75) - np.percentile(bikes['Rental Count']
iqr
```

Out[24]:
```
852.0
```

Найдите интерквартильный размах по атрибуту 'Rental Count', а также выведите значения q1 - 1.5 *iqr, q1 + 1.5* iqr

In [25]:
```
print("Интерквартильный размах по атрибуту 'Rental Count'", iqr)
print("q1-1.5*iqr =", np.percentile(bikes['Rental Count'], 25) - 1.5 * iqr)
print("q3+1.5*iqr =", np.percentile(bikes['Rental Count'], 75) + 1.5 * iqr)
iqr_outlier_threshold_up = np.percentile(bikes['Rental Count'], 75) + 1.5 * iqr
```

```
Интерквартильный размах по атрибуту 'Rental Count' 852.0
q1-1.5*iqr = -1121.0
q3+1.5*iqr = 2287.0
```

print(iqr_outlier_threshold_up) print(iqr_outlier_threshold_bottom)

In [28]:
```
vibros = bikes[bikes['Rental Count'] > iqr_outlier_threshold_up] # Определили элемен
len(vibros)
```

Out[28]:
```
413
```

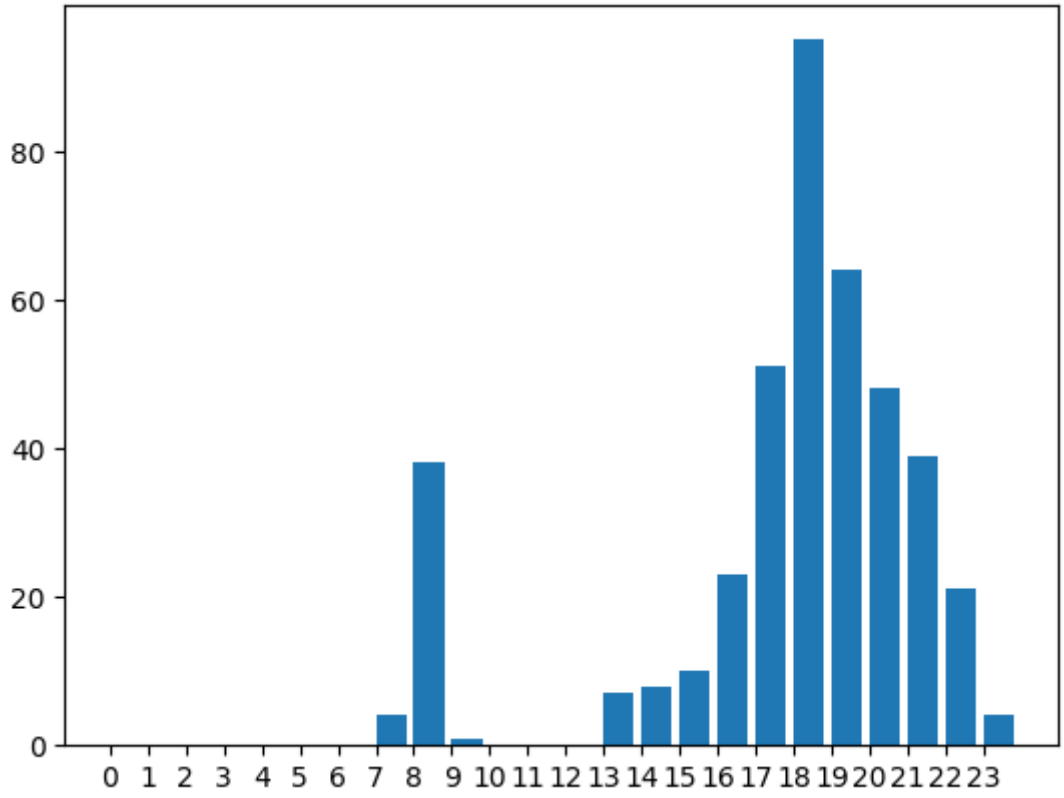
Определите, в какие часы какое количество выбросов было зафиксировано. (value_counts)

In [93]:
```
value_counts = [len(vibros[vibros["Hour"] == i]) for i in range(0, 24)]
print(*value_counts)
x = range(len(value_counts))
ax = plt.gca()
ax.bar(x, value_counts, align='edge') # align='edge' - выравнивание по границе, а не
ax.set_xticks(x)
ax.set_xticklabels(range[24])
plt.hist(value_counts)
```

```
0 0 0 0 0 0 0 4 38 1 0 0 0 7 8 10 23 51 95 64 48 39 21 4
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
```

```
Cell In[93], line 7
      5 ax.bar(x, value_counts, align='edge') # align='edge' - выравнивание по грани
це, а не по центру
      6 ax.set_xticks(x)
----> 7 ax.set_xticklabels(range[24])
      8 plt.hist(value_counts)

TypeError: type 'range' is not subscriptable
```



Выведите количество выбросов по сезонам

In [30]:
```
print("Весна", len(vibros[vibros['Seasons'] == "Spring"]))
print("Зима", len(vibros[vibros['Seasons'] == "Winter"]))
print("Лето", len(vibros[vibros['Seasons'] == "Summer"]))
print("Осень", len(vibros[vibros['Seasons'] == "Autumn"]))
```

```
Весна 101
Зима 0
Лето 196
Осень 116
```

Выведите среднее, среднеквадратичное отклонение и пороги для атрибута Rental Count (+-2.5 стандартных отклонений)

In [31]:
```
#Ваш код

print("Среднее", np.mean(bikes["Rental Count"]))
print("Среднеквадратичное отклонение", np.std(bikes["Rental Count"]))
print("botton", np.mean(bikes["Rental Count"] - 2.5 * np.std(bikes["Rental Count"])))
print("top", np.mean(bikes["Rental Count"] + 2.5 * np.std(bikes["Rental Count"])))
```

```
Среднее 696.5820776255708
Среднеквадратичное отклонение 749.7698143709189
botton -1177.8424583017265
top 2571.006613552868
```

Определите количество выбросов по данной метрике (с shape)

In [32]:
```
vibrosi = bikes[bikes['Rental Count'] > np.mean(bikes["Rental Count"] + 2.5 * np.std
vibrosi[0]
```

Out[32]: 278

Выведите количество выбросов по сезонам

In [34]:
```
a = bikes[bikes['Rental Count'] > np.mean(bikes["Rental Count"] + 2.5 * np.std(bikes
print("Весна", len(a[a['Seasons'] == "Spring"]))
print("Зима", len(a[a['Seasons'] == "Winter"]))
print("Лето", len(a[a['Seasons'] == "Summer"]))
print("Осень", len(a[a['Seasons'] == "Autumn"]))
```

```
Весна 67
Зима 0
Лето 136
Осень 75
```

In [35]:
```
#Посмотрим "нормальные" значения после фильтрации каждой метрикой
std_outlier_threshold_up = np.mean(bikes["Rental Count"] + 2.5 * np.std(bikes["Renta
iqr_bikes_no_outliers = bikes[bikes['Rental Count'] <= iqr_outlier_threshold_up]
std_bikes_no_outliers = bikes[bikes['Rental Count'] <= std_outlier_threshold_up]
```

In [36]:
```
#посмотрим меры центральной тенденции
print(bikes['Rental Count'].mean())
print(iqr_bikes_no_outliers['Rental Count'].mean())
print(std_bikes_no_outliers['Rental Count'].mean())
```

```
696.5820776255708
584.146280100635
613.2393303466164
```

In [37]:
```
print(bikes['Rental Count'].median())
print(iqr_bikes_no_outliers['Rental Count'].median())
print(std_bikes_no_outliers['Rental Count'].median())
```

```
425.5
392.0
402.0
```

Сравните устойчивость мер центральной тенденции к применению фильтров

Пропущенные значения

In [38]:
```
import pandas as pd
import numpy as np

bikes = pd.read_pickle('BikesData.pkl')
bikes.head()
```

Out[38]:

	Date	Hour	Temperature	Humidity	Wind speed	Rainfall	Snowfall	Seasons	Holiday	Functioning Day	I
0	2017-12-01	0	-5.2	37	2.2	0.0	0.0	Winter	0	True	

	Date	Hour	Temperature	Humidity	Wind speed	Rainfall	Snowfall	Seasons	Holiday	Functioning Day
1	2017-12-01	1	-5.5	38	0.8	0.0	0.0	Winter	0	True
2	2017-12-01	2	-6.0	39	1.0	0.0	0.0	Winter	0	True
3	2017-12-01	3	-6.2	40	0.9	0.0	0.0	Winter	0	True
4	2017-12-01	4	-6.0	36	2.3	0.0	0.0	Winter	0	True

In [39]:
```
bikes.info() #обратите внимание на кол-во значений в Temperature. Есть пропущенные з
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8760 entries, 0 to 8759
Data columns (total 14 columns):
 #   Column               Non-Null Count  Dtype         
---  ------               --------------  -----         
 0   Date                 8760 non-null   datetime64[ns]
 1   Hour                 8760 non-null   int64         
 2   Temperature          8581 non-null   float64       
 3   Humidity             8760 non-null   int64         
 4   Wind speed           8760 non-null   float64       
 5   Rainfall             8760 non-null   float64       
 6   Snowfall             8760 non-null   float64       
 7   Seasons              8760 non-null   object        
 8   Holiday              8760 non-null   int64         
 9   Functioning Day      8760 non-null   bool          
 10  Rental Count         8760 non-null   int64         
 11  Normal Humidity      8760 non-null   int64         
 12  Temperature Category  8581 non-null   category      
 13  Good Weather         8760 non-null   int64         
dtypes: bool(1), category(1), datetime64[ns](1), float64(4), int64(6), object(1)
memory usage: 838.5+ KB
```

Посчитайте количество пустых ячеек (isna())

In [40]:
```
bikes["Temperature"].isna().sum()
```

Out[40]: 179

Посчитайте количество заполненных ячеек (notna())

In [41]:
```
bikes["Temperature"].notna().sum()
```

Out[41]: 8581

Создайте новый датафрэйм bikes1. Скопируйте туда старый. Заполните пустые ячейки числом 42.

In [42]:
```
bikes1 = bikes.copy()
bikes1["Temperature"] = bikes1["Temperature"].fillna(44)
bikes1["Temperature"].isna().sum()
```

Out[42]: 0

dropna не удаляет значения из датасета, если не добавить параметр inplace=True (но лучше так не надо)

In [43]:
```
bikes.dropna(subset=['Temperature']).shape
```

Out[43]: (8581, 14)

Заполните пустые ячейки в bikes медианой. (но созраняем в другую колонку)

In [44]:
```
bikes['Temperature_Median'] = bikes['Temperature'].fillna(bikes['Temperature'].media
bikes.iloc[38:42] # выводим строку с пропуском + ближайшие
```

Out[44]:

	Date	Hour	Temperature	Humidity	Wind speed	Rainfall	Snowfall	Seasons	Holiday	Functioning Day
38	2017-12-02	14	7.3	35	1.3	0.0	0.0	Winter	0	True
39	2017-12-02	15	NaN	41	2.3	0.0	0.0	Winter	0	True
40	2017-12-02	16	6.4	48	2.6	0.0	0.0	Winter	0	True
41	2017-12-02	17	6.0	51	2.5	0.0	0.0	Winter	0	True

In [45]:
```
np.random.choice(bikes['Temperature'].dropna()) # генератор случайных объектов из ма
```

Out[45]: 16.7

В функцию fillna можно передавать вектор значений. Он будет служить для заполнений пустых значений, главное чтобы количество элементов совпала

In [46]:
```
temps = np.random.choice(bikes['Temperature'].dropna(), 8760)
temps[:100]
```

Out[46]:
```
array([ 21.3,  18. ,  -0.2,  24. ,  26.4,  28.7,  20. ,  26.7,   2.6,
        -1.9,  24.1,  33.3,  -8.1,  -5.2,  20.4,  15.5,  17.8,  12.8,
        18.6,  22.6,  -4.2,   5.2,  -1.3,  20.4,  11.9,  20.1,  34.3,
        27.9,  -6. ,   7. ,  24.6,  28.5,  21.3,  16.5,  19.7,  11.7,
        20.9,  -9.6,   7.5,  27.4,   9.2,  26.8,  20.8,  18.9,  -7.2,
        12.9,   0.5,  26.2,  11.7,  11.7,  28.2,  -5.8,  20.3,  30.7,
         7.1,   9.1,  17.1,   6.7,   9.9,   9.2,  11.4,   5.8,  24.8,
         1.1,  -1.3,  -5.3,  24.7,   6.8,  28.3,  -5.6, -15.3,  11.4,
        23.2,   7.7,   2.7,  11.2,  -5.4,  20.2,   7.1,  29.2,  21.6,
        -5.4,  24.5,  15.9,  15.9,   0.4,   5.1,   2.6,  14.8,  10.5,
        22.6,   0.2,   3.7,  19.2,  20.5,   6.7,  10.4, -10. ,   2.9,
        -9.7])
```

Создать новую колонку 'Temperature_Random', заполнить пустые значения из Temperature, сохранив новые значчения в Temperature_Random. Значения взять из temps

In [47]:
```
bikes['Temperature_Random'] = bikes['Temperature'].fillna(pd.Series(temps)) # сделал
```

Теперь заполните пробелы в Temperature в датафрейме bikes_1.

In [48]:
```
bikes1["Temperature"].isna().sum()
```

Out[48]:
```
0
```

Добавьте комментарии к коду ниже. Что он делает?

In [49]:
```
bikes.groupby([bikes['Date'].dt.isocalendar().week, 'Hour'])['Temperature'].median()
```

Out[49]:
```
week  Hour
1     0      -4.3
      1      -4.8
      2      -5.3
      3      -5.5
      4      -5.1
             ...
52    19     -0.4
      20     -1.0
      21     -1.6
      22     -1.7
      23     -1.0
Name: Temperature, Length: 1248, dtype: float64
```

In [50]:
```
bikes.groupby([bikes['Date'].dt.isocalendar().week, 'Hour'])['Temperature'].median()
```

Out[50]:
```
week  Hour
1     0      -4.3
      1      -4.8
      2      -5.3
      3      -5.5
      4      -5.1
             ...
52    19     -0.4
      20     -1.0
      21     -1.6
      22     -1.7
      23     -1.0
Name: Temperature, Length: 1248, dtype: float64
```

In [52]:
```
bikes.groupby([bikes['Date'].dt.isocalendar().week, 'Hour'])['Temperature'].transfor
temp_medians
```

Out[52]:
```
0       2.75
1       2.50
2       1.35
3       2.15
4       2.15
        ...
8755    5.45
8756    4.70
8757    4.15
8758    3.50
8759    3.40
Name: Temperature, Length: 8760, dtype: float64
```

In [53]:
```
bikes['Temperature_Median_Group'] = bikes['Temperature'].fillna(pd.Series(temp_media
bikes[bikes['Temperature'].isna()].head()
```

Out[53]:

	Date	Hour	Temperature	Humidity	Wind speed	Rainfall	Snowfall	Seasons	Holiday	Functioning Day
39	2017-12-02	15	NaN	41	2.3	0.0	0.0	Winter	0	True
50	2017-12-03	2	NaN	79	1.4	0.0	0.0	Winter	0	True
64	2017-12-03	16	NaN	77	1.6	0.0	0.0	Winter	0	True
105	2017-12-05	9	NaN	31	1.3	0.0	0.0	Winter	0	True
151	2017-12-07	7	NaN	93	0.5	0.0	0.9	Winter	0	True

In [54]:

```
bikes = pd.read_pickle('BikesDataVars.pkl')
bikes['Temperature'] = bikes['Temperature'].fillna(pd.Series(temp_medians))
```

In [55]:

```
def get_temp_cat(temp):
    if temp < 0:
        return 'Freezing'
    elif temp < 15:
        return 'Chilly'
    elif temp < 26:
        return 'Nice'
    elif temp >= 26:
        return 'Hot'
    else:
        return temp

bikes['Temperature Category'] = pd.Categorical(bikes['Temperature'].apply(get_temp_c
```

In [56]:

```
bikes.head()
```

Out[56]:

	Date	Hour	Temperature	Humidity	Wind speed	Rainfall	Snowfall	Seasons	Holiday	Functioning Day	I
0	2017-12-01	0	-5.2	37	2.2	0.0	0.0	Winter	0	True	
1	2017-12-01	1	-5.5	38	0.8	0.0	0.0	Winter	0	True	
2	2017-12-01	2	-6.0	39	1.0	0.0	0.0	Winter	0	True	
3	2017-12-01	3	-6.2	40	0.9	0.0	0.0	Winter	0	True	
4	2017-12-01	4	-6.0	36	2.3	0.0	0.0	Winter	0	True	

Корреляция

Корреляция — это мера линейной взаимосвязи между двумя величинами. в ходе корреляционного анализа мы можем выявить только линейную взаимосвязь.

In [57]:
```
import pandas as pd
import warnings
import numpy as np
warnings.filterwarnings('ignore')
bikes = pd.read_pickle('BikesDataImputed.pkl')
bikes.head()
```

Out[57]:

	Date	Hour	Temperature	Humidity	Wind speed	Rainfall	Snowfall	Seasons	Holiday	Functioning Day
0	2017-12-01	0	-5.2	37	2.2	0.0	0.0	Winter	0	True
1	2017-12-01	1	-5.5	38	0.8	0.0	0.0	Winter	0	True
2	2017-12-01	2	-6.0	39	1.0	0.0	0.0	Winter	0	True
3	2017-12-01	3	-6.2	40	0.9	0.0	0.0	Winter	0	True
4	2017-12-01	4	-6.0	36	2.3	0.0	0.0	Winter	0	True

In [58]:
```
#подготовим данные, возьмём аггрегированные по неделям значения температуры и количе
temp_mean = bikes.groupby(bikes['Date'].dt.isocalendar().week)['Temperature'].mean()
bikes_sum = bikes.groupby(bikes['Date'].dt.isocalendar().week)['Rental Count'].sum()
```

In [59]:
```
#Сведём переменные в одну таблицу с помощью функции concat. axis=1 служит для раздел
bikes_week = pd.concat([temp_mean, bikes_sum], axis=1)
print(bikes_week)
```

```
      Temperature  Rental Count
week
1       -2.694940         39441
2       -5.079762         30871
3        2.662500         42193
4      -10.038690         23079
5       -5.650595         28415
6       -5.486310         33259
7       -1.225298         32139
8        1.631548         50136
9        4.004167         52958
10       5.269940         77316
11      10.133036         90547
12       6.162202         79109
13      13.890774        118031
14      10.283631         97000
15      11.713988         98468
16      14.669940        142918
17      14.061310        141147
18      16.365476        146127
19      16.220238        136607
20      19.307738        142346
21      18.988988        189749
22      22.223214        210326
23      22.628869        211869
```

```
24      21.751190         220392
25      24.036012         213553
26      23.757143         146455
27      24.758333         183652
28      25.858333         154848
29      29.297619         172636
30      30.740774         163447
31      32.193750         135086
32      30.518452         147911
33      29.238690         166679
34      26.901488         152282
35      24.691071         138529
36      22.962798         191800
37      22.782143         186208
38      20.743750         103925
39      18.413690         124820
40      17.019643          99622
41      12.672024         159527
42      13.542857         185695
43      11.788393         141509
44       8.882440         123557
45      11.590774          66682
46       7.953571         142787
47       4.422321         103454
48       3.505208         107727
49      -0.992560          48680
50      -6.645536          40147
51       0.210714          31938
52      -1.651786          34460
```

In [60]:

```
#Возьмём 5 первых значений датафрейма с помощью функции iloc
first_five = bikes_week.iloc[:5]
print(first_five)
```

```
      Temperature  Rental Count
week
1       -2.694940         39441
2       -5.079762         30871
3        2.662500         42193
4      -10.038690         23079
5       -5.650595         28415
```

In [61]:

```
#вычислим X-Mx и Y - My. Температура - X, аренды - Y
first_five['X - Mx'] = first_five['Temperature'].mean()
first_five['Y - My'] = first_five['Rental Count'].mean()
first_five
# должна получиться табличка вида Temperature   Rental Count    X - Mx  Y - My . И 5
```

Out[61]:

week	Temperature	Rental Count	X - Mx	Y - My
1	-2.694940	39441	-4.160298	32799.8
2	-5.079762	30871	-4.160298	32799.8
3	2.662500	42193	-4.160298	32799.8
4	-10.038690	23079	-4.160298	32799.8
5	-5.650595	28415	-4.160298	32799.8

In [62]:

```
# Найдём сумму квадратов для X и Y, а также сумму произведений
SSx = first_five['X - Mx'] ** 2 + first_five['Temperature'] ** 2
SSy = first_five['Y - My'] ** 2 + first_five['Rental Count'] ** 2
```

```
SP = SSx * SSy
print(SSx, SSy, SP)
```

```
week
1     24.570780
2     43.112057
3     24.396983
4    118.083383
5     49.237303
dtype: float64 week
1    2.631419e+09
2    2.028846e+09
3    2.856076e+09
4    1.608467e+09
5    1.883239e+09
dtype: float64 week
1    6.465603e+10
2    8.746770e+10
3    6.967964e+10
4    1.899332e+11
5    9.272561e+10
dtype: float64
```

In [63]:

```
#найдём коэффициент корреляции
def covariance(xs, ys):
    '''Вычисление ковариации (несмещенная, т.е. n-1)'''
    dx = xs - xs.mean()
    dy = ys - ys.mean()
    return (dx * dy).sum() / (dx.count() - 1)
def variance(xs):
    '''Вычисление корреляции,
       несмещенная дисперсия при n <= 30'''
    x_hat = xs.mean()
    n = xs.count()
    n = n - 1 if n in range( 1, 30 ) else n
    return sum((xs - x_hat) ** 2) / n

def standard_deviation(xs):
    '''Вычисление стандартного отклонения'''
    return np.sqrt(variance(xs))

def correlation(xs, ys):
    '''Вычисление корреляции'''
    return covariance(xs, ys) / (standard_deviation(xs) *
                                 standard_deviation(ys))
r = correlation(first_five['Temperature'], first_five['Rental Count'])
r
```

Out[63]: 0.9484760874616522

In [64]:

```
#проверим значение с помощью функции corr для first_five. Всё так же, корреляция Tem
first_five['Temperature'].corr(first_five['Rental Count'])
```

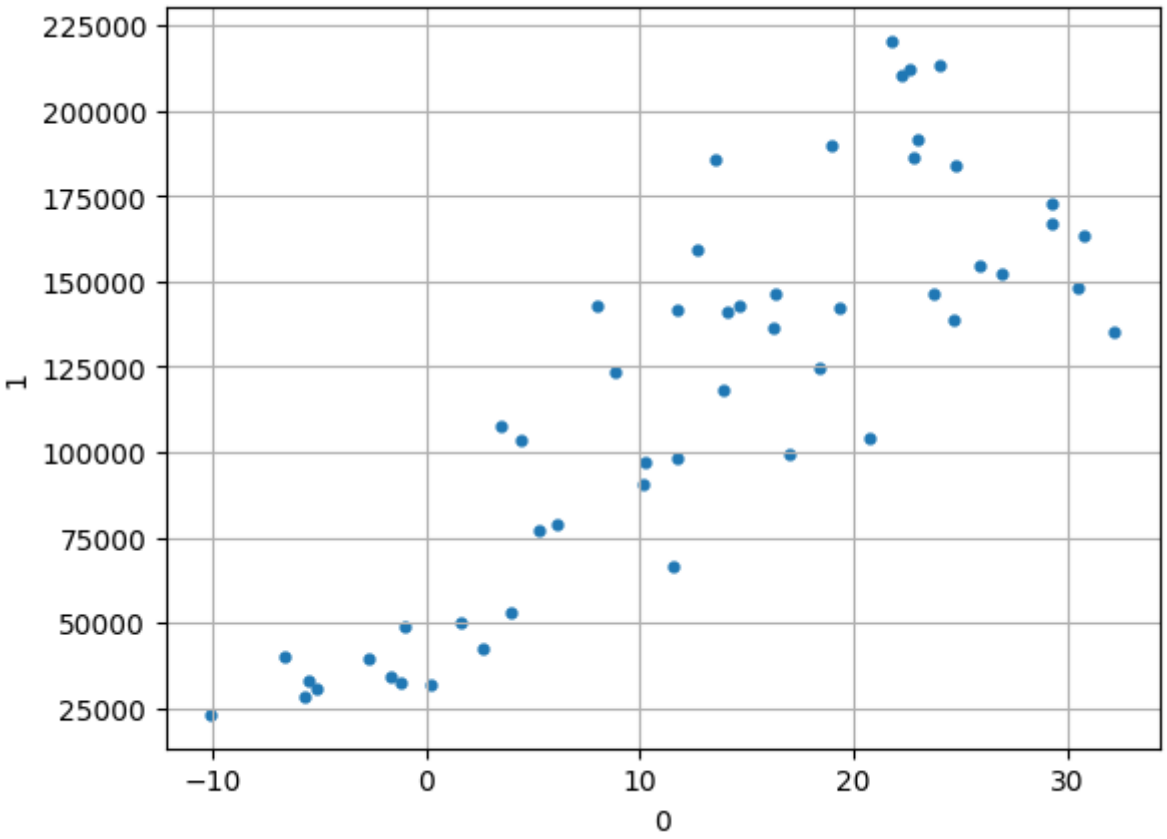
Out[64]: 0.9484760874616526

In [65]:

```
# Теперь найдём корреляцию не для 5 элементов, а ля всей генеральной совокупности (b
bikes_week['Temperature'].corr(bikes_week['Rental Count'])
```

Out[65]: 0.8458075200534891

In [66]:
```
# Построим график вида scatter (график рассеивания) по Temperature и Rental Count в
pd.DataFrame(np.array([bikes_week["Temperature"],bikes_week["Rental Count"]]).T).plo
plt.show()
```



Объясните код ниже

In [67]:
```
bikes.corr()
```

Out[67]:

	Hour	Temperature	Humidity	Wind speed	Rainfall	Snowfall	Holiday	F
Hour	1.000000e+00	0.123610	-0.241644	0.285197	0.008715	-0.021516	-1.391486e-16	
Temperature	1.236105e-01	1.000000	0.159793	-0.036418	0.050758	-0.217846	-5.570102e-02	
Humidity	-2.416438e-01	0.159793	1.000000	-0.336683	0.236397	0.108183	-5.027765e-02	
Wind speed	2.851967e-01	-0.036418	-0.336683	1.000000	-0.019674	-0.003554	2.301677e-02	
Rainfall	8.714642e-03	0.050758	0.236397	-0.019674	1.000000	0.008500	-1.426911e-02	
Snowfall	-2.151645e-02	-0.217846	0.108183	-0.003554	0.008500	1.000000	-1.259072e-02	
Holiday	-1.391486e-16	-0.055701	-0.050278	0.023017	-0.014269	-0.012591	1.000000e+00	
Functioning Day	5.439377e-03	-0.049849	-0.020800	0.005037	0.002055	0.032089	-2.762445e-02	

	Hour	Temperature	Humidity	Wind speed	Rainfall	Snowfall	Holiday	F
Rental Count	3.456218e-01	0.454749	-0.169085	0.097583	-0.103519	-0.120869	-6.882160e-02	
Normal Humidity	1.075026e-01	0.025467	-0.285947	0.074964	-0.095339	-0.067939	-2.015629e-02	
Good Weather	7.369784e-02	0.206979	-0.115874	0.032127	-0.042127	-0.054942	2.900771e-02	

In [68]:
```
humidity_mean = bikes.groupby(bikes['Date'].dt.isocalendar().week)['Humidity'].mean(
wind_mean = bikes.groupby(bikes['Date'].dt.isocalendar().week)['Wind speed'].mean()
```

In [69]:
```
bikes_week = pd.concat([bikes_week, humidity_mean, wind_mean], axis=1)
bikes_week
```

Out[69]:

week	Temperature	Rental Count	Humidity	Wind speed
1	-2.694940	39441	43.660714	1.524405
2	-5.079762	30871	53.958333	1.995833
3	2.662500	42193	55.178571	1.385119
4	-10.038690	23079	38.410714	2.575000
5	-5.650595	28415	47.815476	2.256548
6	-5.486310	33259	41.571429	2.275000
7	-1.225298	32139	38.422619	2.332738
8	1.631548	50136	50.041667	2.055357
9	4.004167	52958	55.166667	1.770238
10	5.269940	77316	60.988095	1.802381
11	10.133036	90547	58.119048	1.723810
12	6.162202	79109	59.136905	2.245833
13	13.890774	118031	62.321429	1.817262
14	10.283631	97000	67.178571	2.385714
15	11.713988	98468	51.166667	2.233333
16	14.669940	142918	41.059524	1.642262
17	14.061310	141147	56.172619	1.798810
18	16.365476	146127	66.392857	2.025595
19	16.220238	136607	66.142857	1.623214
20	19.307738	142346	69.363095	1.380952
21	18.988988	189749	48.666667	1.793452
22	22.223214	210326	53.220238	1.817262

week	Temperature	Rental Count	Humidity	Wind speed
23	22.628869	211869	59.916667	1.763095
24	21.751190	220392	64.244048	1.422024
25	24.036012	213553	57.553571	1.929167
26	23.757143	146455	80.690476	1.381548
27	24.758333	183652	67.958333	1.339881
28	25.858333	154848	78.303571	1.416667
29	29.297619	172636	61.255952	1.554167
30	30.740774	163447	63.547619	1.644048
31	32.193750	135086	56.791667	1.819048
32	30.518452	147911	64.642857	1.578571
33	29.238690	166679	53.708333	1.427976
34	26.901488	152282	66.357143	1.975000
35	24.691071	138529	76.297619	1.490476
36	22.962798	191800	63.065476	1.858333
37	22.782143	186208	62.607143	1.375000
38	20.743750	103925	66.869048	1.528571
39	18.413690	124820	51.065476	1.510714
40	17.019643	99622	64.946429	1.801786
41	12.672024	159527	53.934524	1.501190
42	13.542857	185695	56.696429	1.219643
43	11.788393	141509	63.410714	1.552976
44	8.882440	123557	54.797619	1.476786
45	11.590774	66682	74.005952	1.480357
46	7.953571	142787	54.488095	1.228571
47	4.422321	103454	51.904762	1.550000
48	3.505208	107727	54.859375	1.447917
49	-0.992560	48680	54.958333	1.697024
50	-6.645536	40147	42.898810	2.022619
51	0.210714	31938	70.178571	1.380952
52	-1.651786	34460	52.136905	1.900595

In [70]:
```
bikes_week.corr()
#обратите внимение на единицы по диагонали. Почему они появились? Оцените степень ко
```

Out[70]:

	Temperature	Rental Count	Humidity	Wind speed
Temperature	1.000000	0.845808	0.584642	-0.420474

	Temperature	Rental Count	Humidity	Wind speed
Rental Count	0.845808	1.000000	0.389963	-0.434142
Humidity	0.584642	0.389963	1.000000	-0.456225
Wind speed	-0.420474	-0.434142	-0.456225	1.000000

In [71]:
```
bikes_week.corr()['Rental Count']
```

Out[71]:
```
Temperature     0.845808
Rental Count    1.000000
Humidity        0.389963
Wind speed     -0.434142
Name: Rental Count, dtype: float64
```

In [72]:
```
#Составьте список убывания коэффициента корреляции между всеми столбцами по убыванию
cor = bikes_week.corr()
cor.abs().unstack().sort_values()
```

Out[72]:
```
Rental Count  Humidity        0.389963
Humidity      Rental Count    0.389963
Temperature   Wind speed      0.420474
Wind speed    Temperature     0.420474
Rental Count  Wind speed      0.434142
Wind speed    Rental Count    0.434142
Humidity      Wind speed      0.456225
Wind speed    Humidity        0.456225
Temperature   Humidity        0.584642
Humidity      Temperature     0.584642
Temperature   Rental Count    0.845808
Rental Count  Temperature     0.845808
Temperature   Temperature     1.000000
Rental Count  Rental Count    1.000000
Humidity      Humidity        1.000000
Wind speed    Wind speed      1.000000
dtype: float64
```

In [ ]:
```

```