cargar el conjunto de datos de entrenamiento

In [ ]:
```
import pandas as pd

bike_data = pd.read_csv('daily-bike-share.csv')
bike_data.head()
```

Out[ ]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	aten
0	1	1/1/2011	1	0	1	0	6	0	2	0.344167	0.3636
1	2	1/2/2011	1	0	1	0	0	0	2	0.363478	0.3537
2	3	1/3/2011	1	0	1	0	1	1	1	0.196364	0.1894
3	4	1/4/2011	1	0	1	0	2	1	1	0.200000	0.2121
4	5	1/5/2011	1	0	1	0	3	1	1	0.226957	0.2292

In [ ]:
```
bike_data['day'] = pd.DatetimeIndex(bike_data['dteday']).day
bike_data.head(32)
```

Out[ ]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	at
0	1	1/1/2011	1	0	1	0	6	0	2	0.344167	0.36
1	2	1/2/2011	1	0	1	0	0	0	2	0.363478	0.35
2	3	1/3/2011	1	0	1	0	1	1	1	0.196364	0.18
3	4	1/4/2011	1	0	1	0	2	1	1	0.200000	0.21
4	5	1/5/2011	1	0	1	0	3	1	1	0.226957	0.22
5	6	1/6/2011	1	0	1	0	4	1	1	0.204348	0.23
6	7	1/7/2011	1	0	1	0	5	1	2	0.196522	0.20
7	8	1/8/2011	1	0	1	0	6	0	2	0.165000	0.16
8	9	1/9/2011	1	0	1	0	0	0	1	0.138333	0.11
9	10	1/10/2011	1	0	1	0	1	1	1	0.150833	0.15
10	11	1/11/2011	1	0	1	0	2	1	2	0.169091	0.19
11	12	1/12/2011	1	0	1	0	3	1	1	0.172727	0.16
12	13	1/13/2011	1	0	1	0	4	1	1	0.165000	0.15
13	14	1/14/2011	1	0	1	0	5	1	1	0.160870	0.18
14	15	1/15/2011	1	0	1	0	6	0	2	0.233333	0.24
15	16	1/16/2011	1	0	1	0	0	0	1	0.231667	0.23
16	17	1/17/2011	1	0	1	1	1	0	2	0.175833	0.17
17	18	1/18/2011	1	0	1	0	2	1	2	0.216667	0.23
18	19	1/19/2011	1	0	1	0	3	1	2	0.292174	0.29
19	20	1/20/2011	1	0	1	0	4	1	2	0.261667	0.25
20	21	1/21/2011	1	0	1	0	5	1	1	0.177500	0.15
21	22	1/22/2011	1	0	1	0	6	0	1	0.059130	0.07
22	23	1/23/2011	1	0	1	0	0	0	1	0.096522	0.09
23	24	1/24/2011	1	0	1	0	1	1	1	0.097391	0.11
24	25	1/25/2011	1	0	1	0	2	1	2	0.223478	0.23
25	26	1/26/2011	1	0	1	0	3	1	3	0.217500	0.20
26	27	1/27/2011	1	0	1	0	4	1	1	0.195000	0.21
27	28	1/28/2011	1	0	1	0	5	1	2	0.203478	0.22
28	29	1/29/2011	1	0	1	0	6	0	1	0.196522	0.21
29	30	1/30/2011	1	0	1	0	0	0	1	0.216522	0.25
30	31	1/31/2011	1	0	1	0	1	1	2	0.180833	0.18
31	32	2/1/2011	1	0	2	0	2	1	2	0.192174	0.23

In [ ]:
```
numeric_features = ['temp', 'atemp', 'hum', 'windspeed']
bike_data[numeric_features + ['rentals']].describe()
```

Out[ ]:

	temp	atemp	hum	windspeed	rentals
count	731.000000	731.000000	731.000000	731.000000	731.000000
mean	0.495385	0.474354	0.627894	0.190486	848.176471
std	0.183051	0.162961	0.142429	0.077498	686.622488
min	0.059130	0.079070	0.000000	0.022392	2.000000
25%	0.337083	0.337842	0.520000	0.134950	315.500000
50%	0.498333	0.486733	0.626667	0.180975	713.000000
75%	0.655417	0.608602	0.730209	0.233214	1096.000000
max	0.861667	0.840896	0.972500	0.507463	3410.000000

Esto garantiza que los gráficos se muestren en línea en el bloc de notas de Jupyter

Obtener la columna de etiqueta

Crear una figura para 2 subgráficos (2 filas, 1 columna)

Trazar el histograma

Agregar líneas para la media, la mediana y el modo

Trazar el diagrama de caja

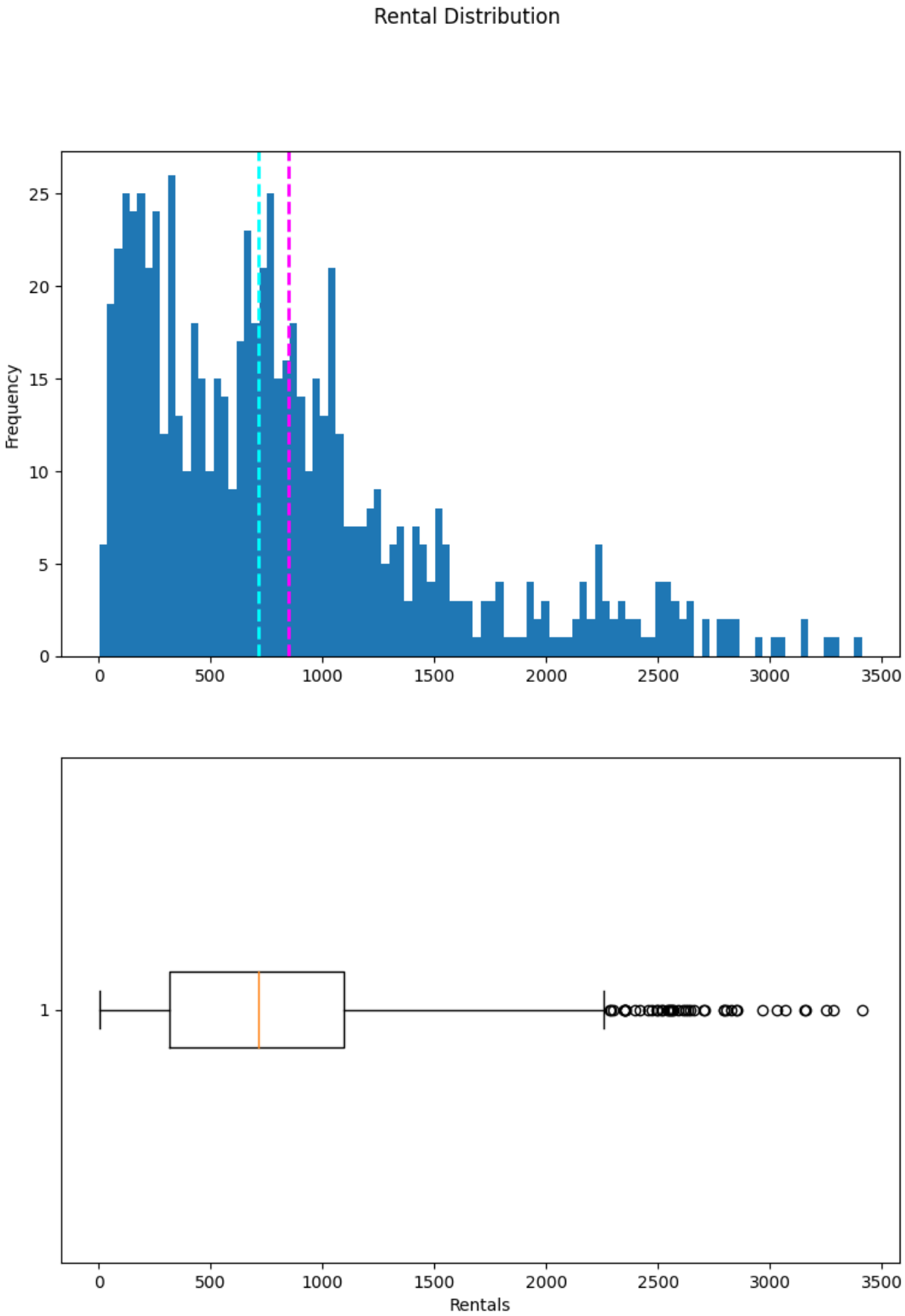
Agregar un título a la figura

Mostrar la figura

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt


%matplotlib inline

#
label = bike_data['rentals']


#
fig, ax = plt.subplots(2, 1, figsize = (9,12))

#
ax[0].hist(label, bins=100)
```

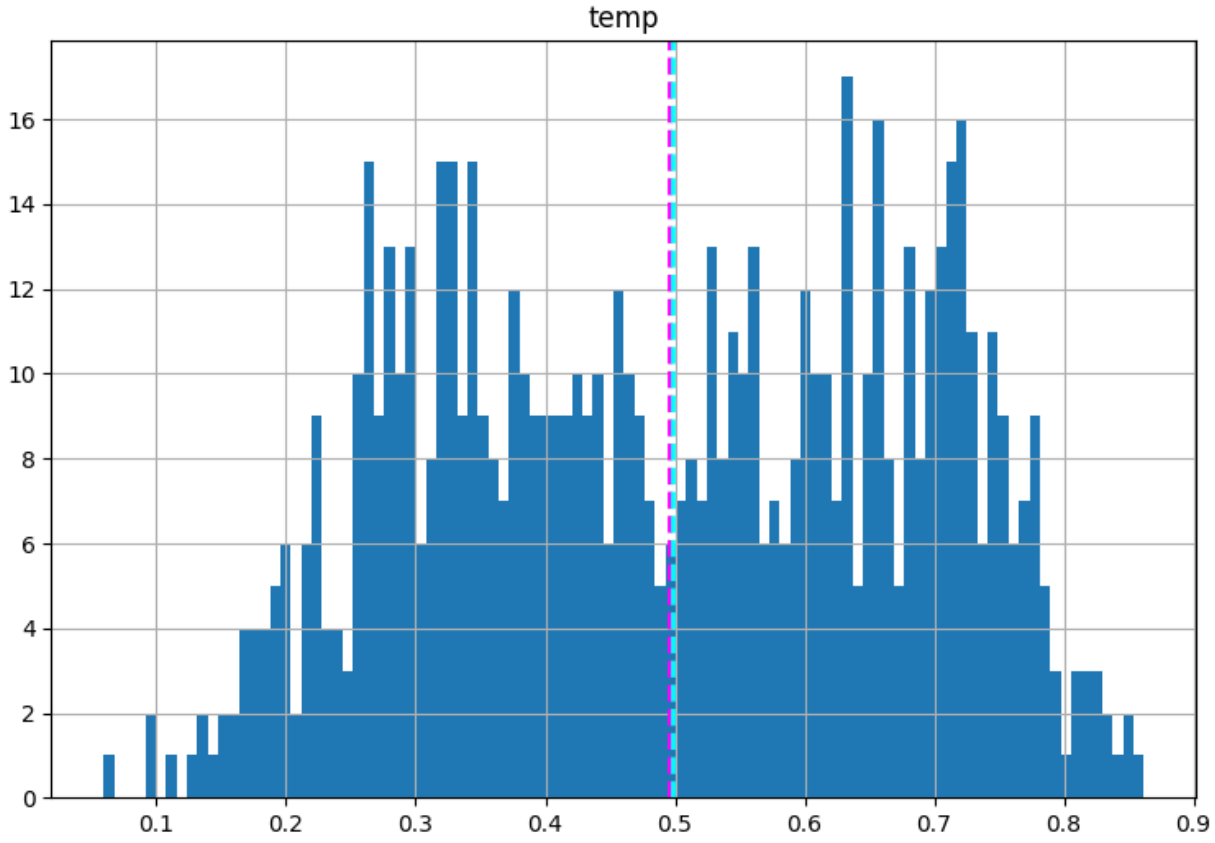
```
ax[0].set_ylabel('Frequency')

#
ax[0].axvline(label.mean(), color='magenta', linestyle='dashed', linewidth=2)
ax[0].axvline(label.median(), color='cyan', linestyle='dashed', linewidth=2)

#
ax[1].boxplot(label, vert=False)
ax[1].set_xlabel('Rentals')

#
fig.suptitle('Rental Distribution')

#
fig.show()
```

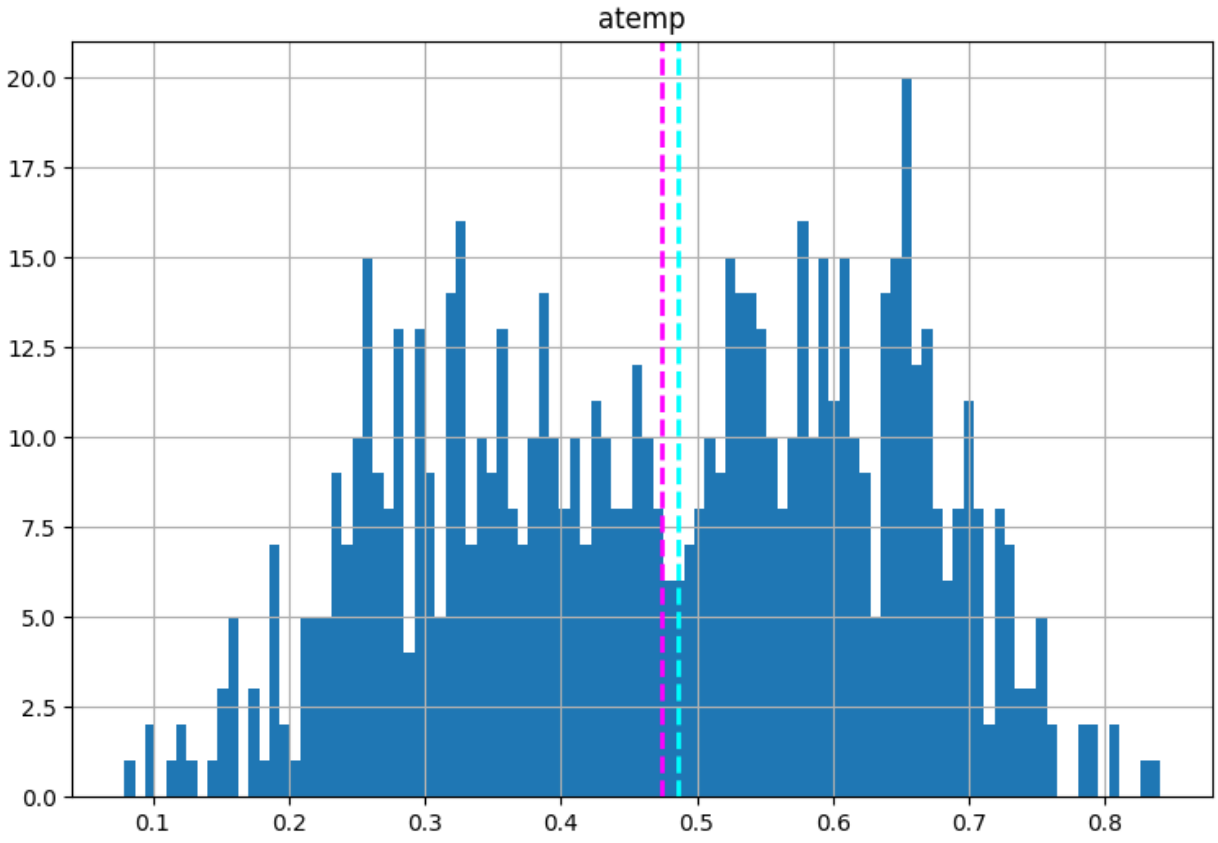
```
C:\Users\lenovo\AppData\Local\Temp\ipykernel_16684\2229708211.py:30: UserWarning: Mat
plotlib is currently using module://matplotlib_inline.backend_inline, which is a non-
GUI backend, so cannot show the figure.
  fig.show()
```

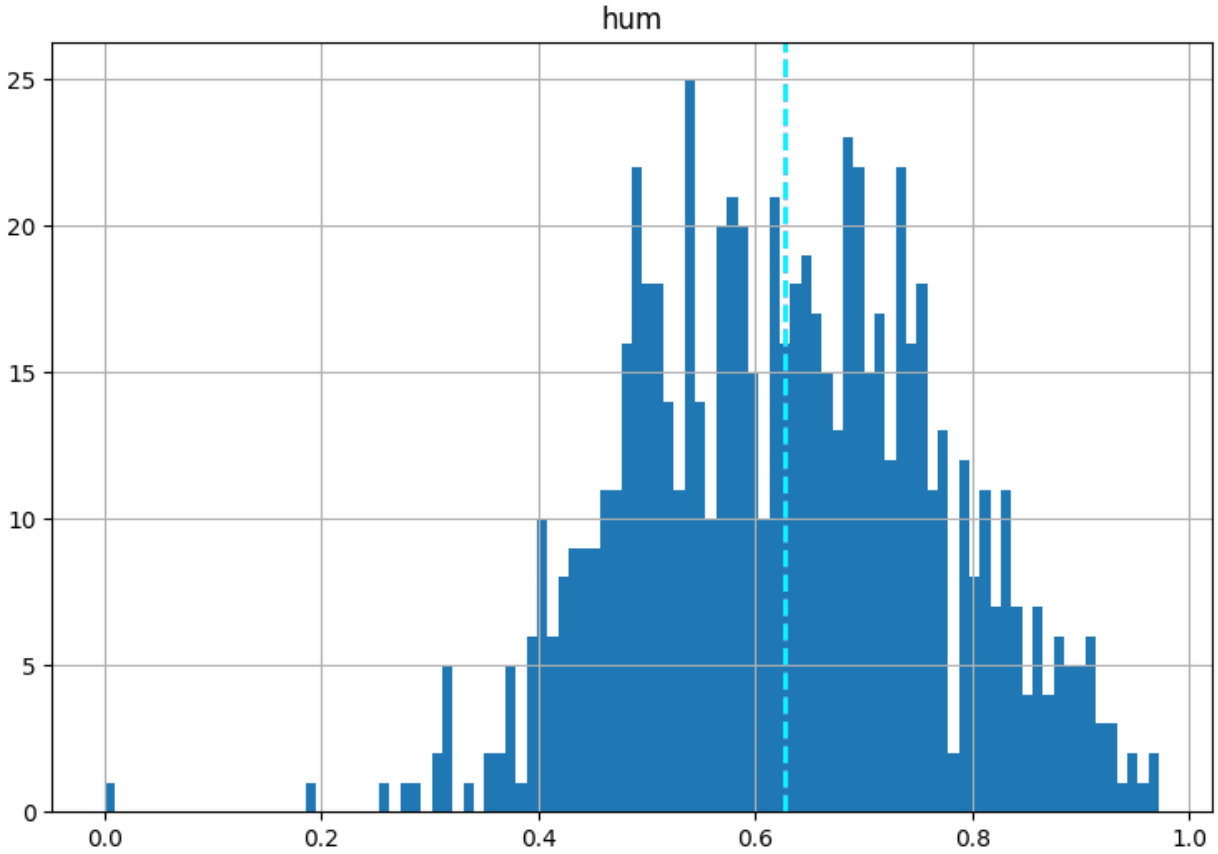


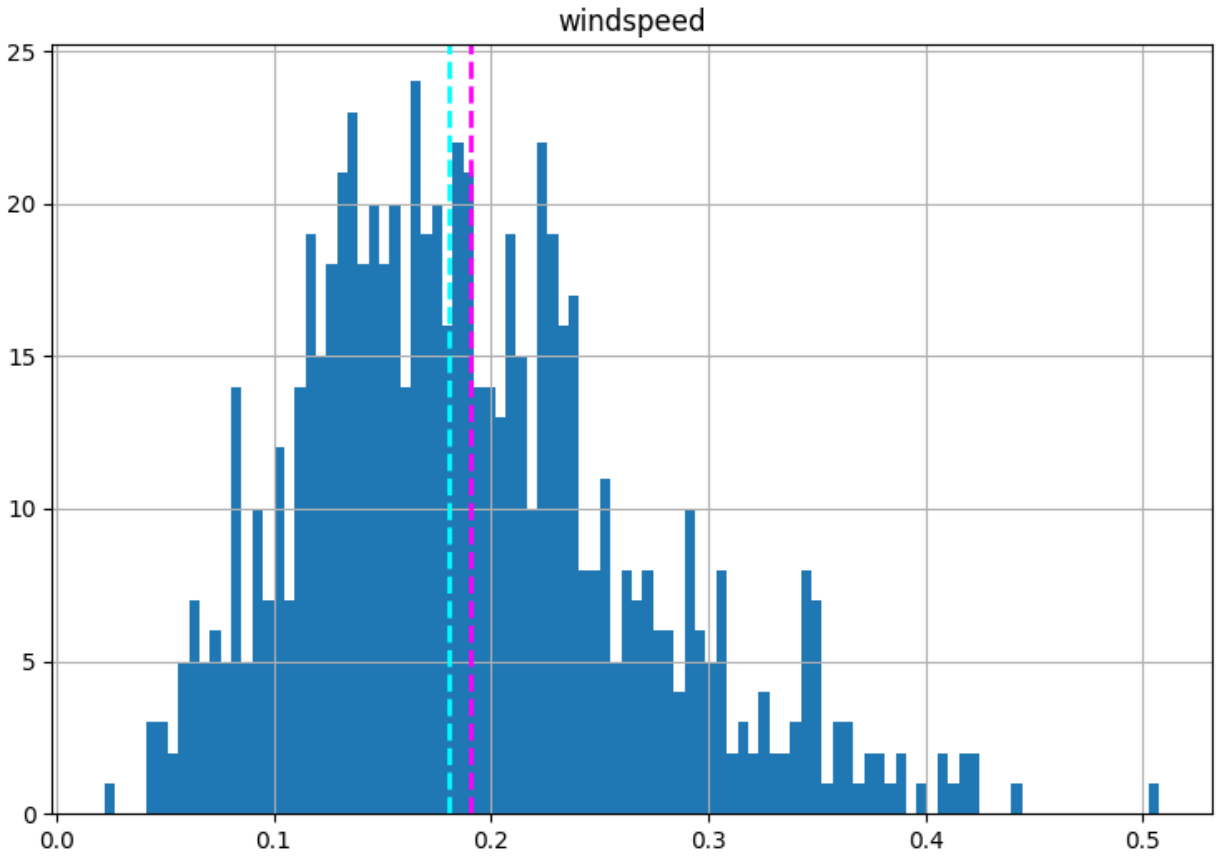
Trazar un histograma para cada entidad numérica

In [ ]:
```
#
for col in numeric_features:
    fig = plt.figure(figsize=(9, 6))
    ax = fig.gca()
    feature = bike_data[col]
    feature.hist(bins=100, ax = ax)
    ax.axvline(feature.mean(), color='magenta', linestyle='dashed', linewidth=2)
    ax.axvline(feature.median(), color='cyan', linestyle='dashed', linewidth=2)
    ax.set_title(col)
plt.show()
```



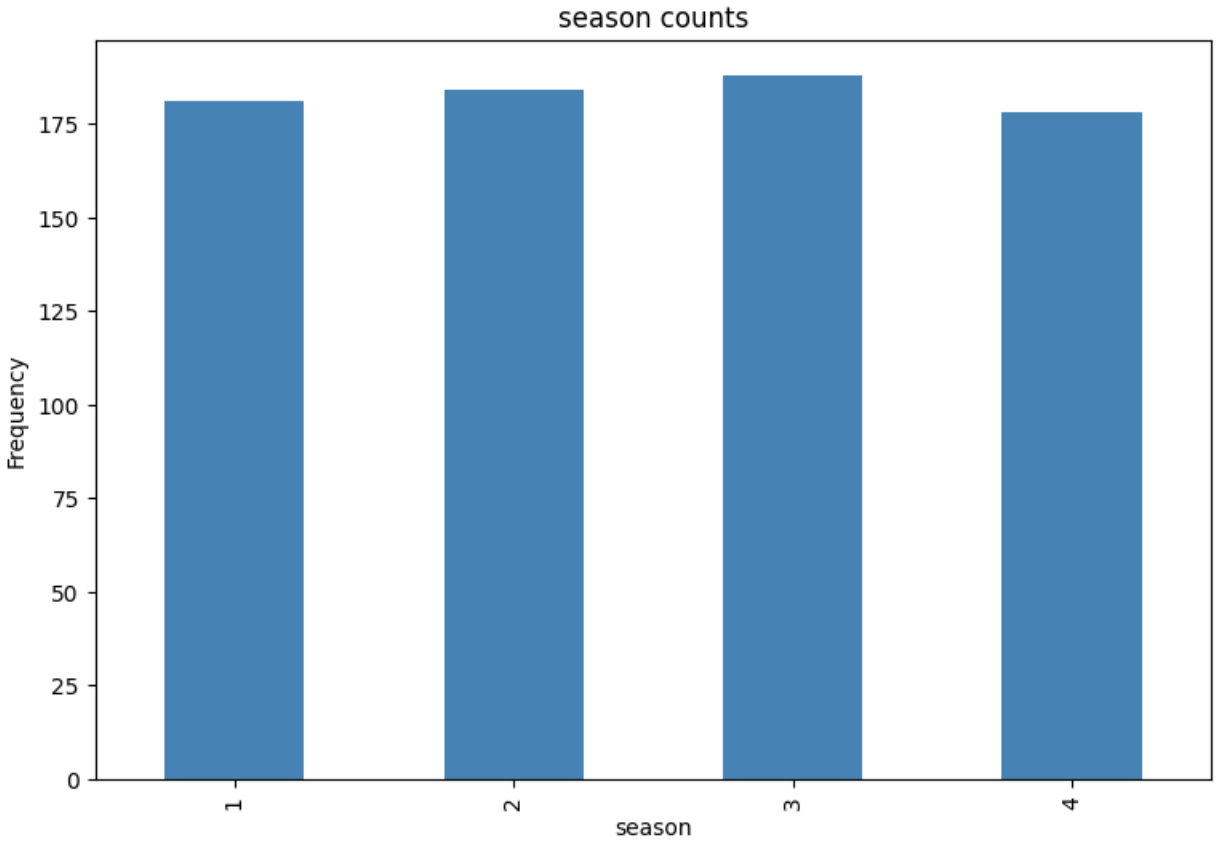

atemp
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

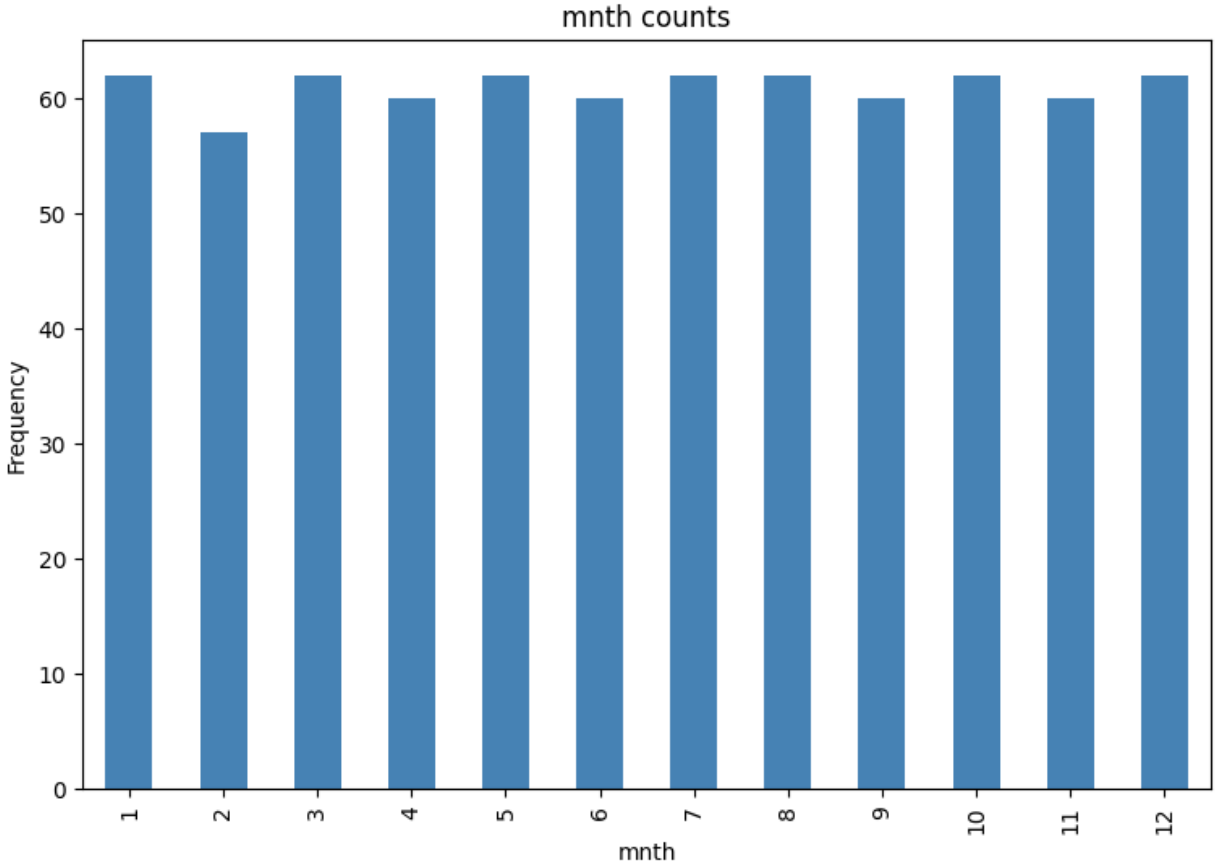

hum
25
20
15
10
5
0
0.0
0.2
0.4
0.6
0.8
1.0

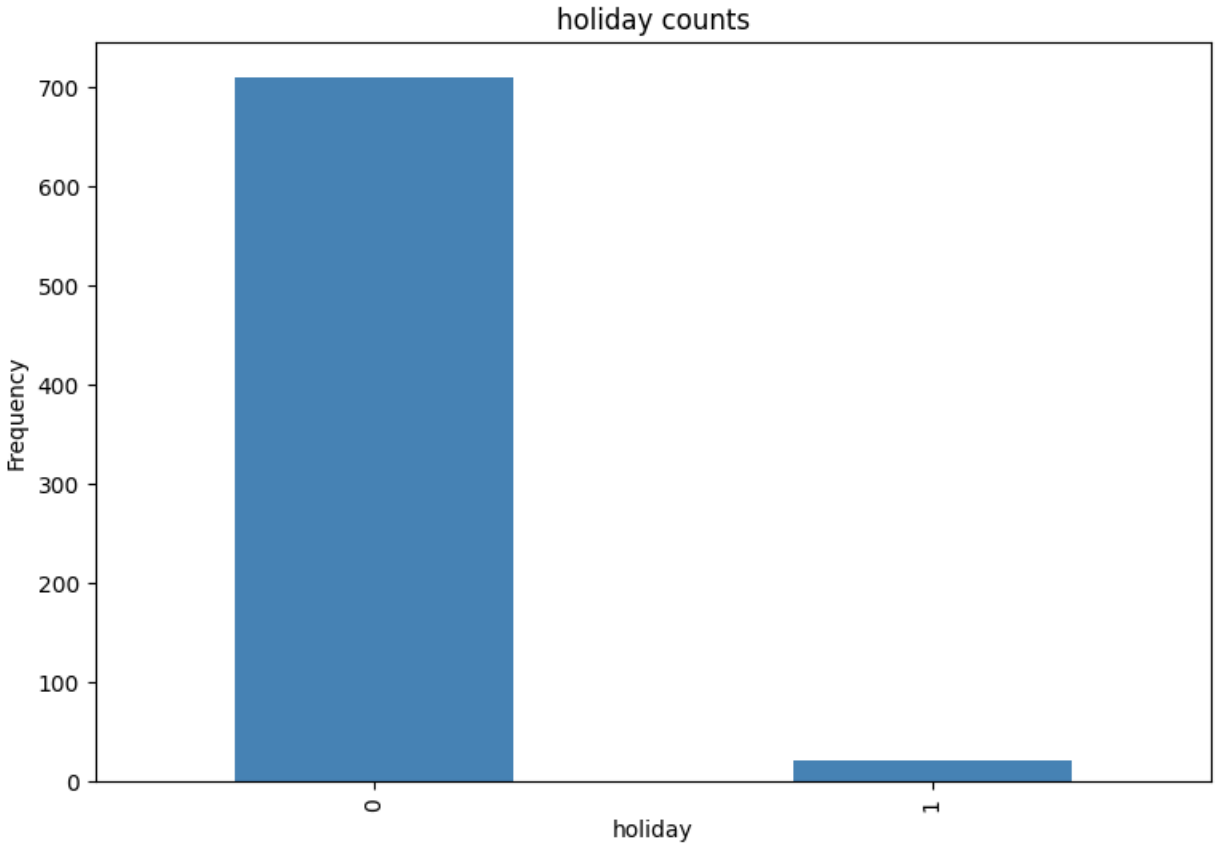


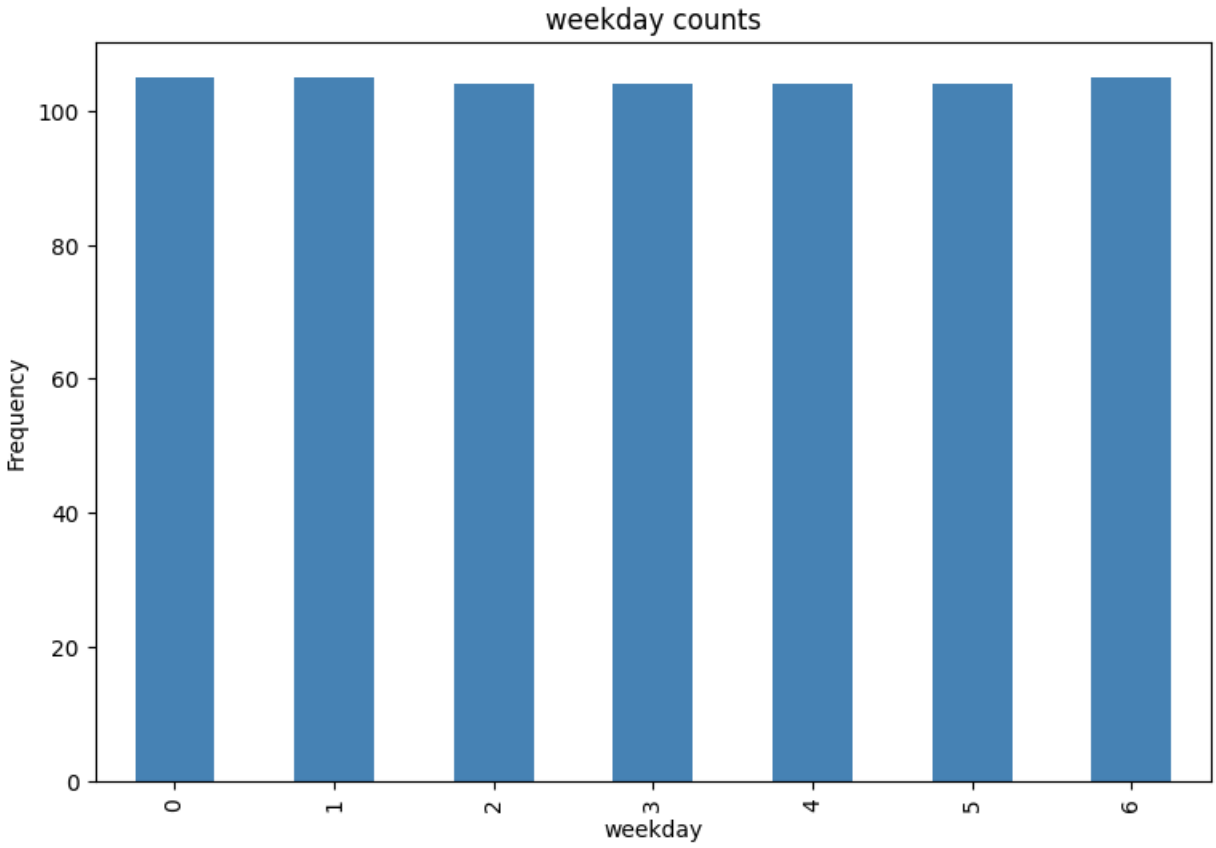
Trazar un gráfico de barras para cada recuento de características categóricas

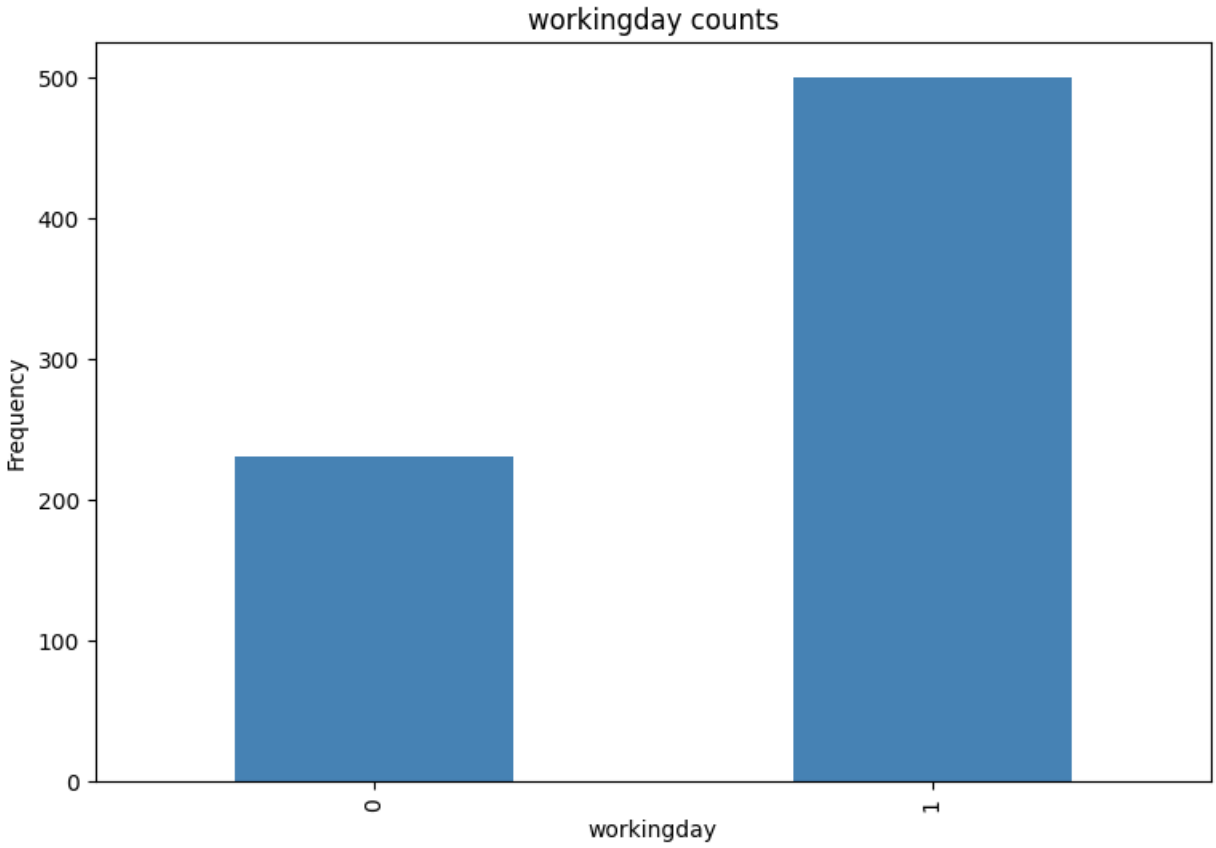
In [ ]:
```
import numpy as np

#
categorical_features = ['season','mnth','holiday','weekday','workingday','weathersit',

for col in categorical_features:
    counts = bike_data[col].value_counts().sort_index()
    fig = plt.figure(figsize=(9, 6))
    ax = fig.gca()
    counts.plot.bar(ax = ax, color='steelblue')
    ax.set_title(col + ' counts')
    ax.set_xlabel(col)
    ax.set_ylabel("Frequency")
plt.show()
```

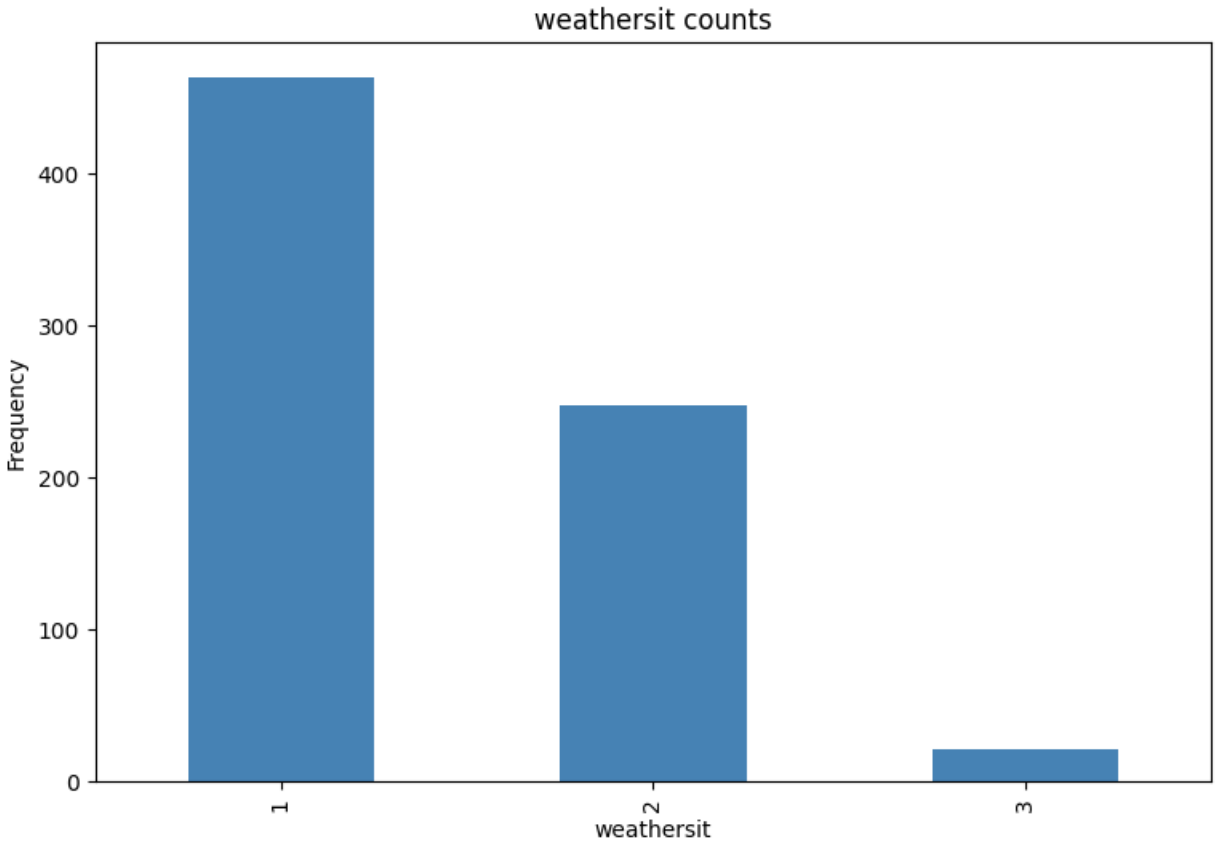

season counts
Frequency
0
25
50
75
100
125
150
175
1
2
3
4
season

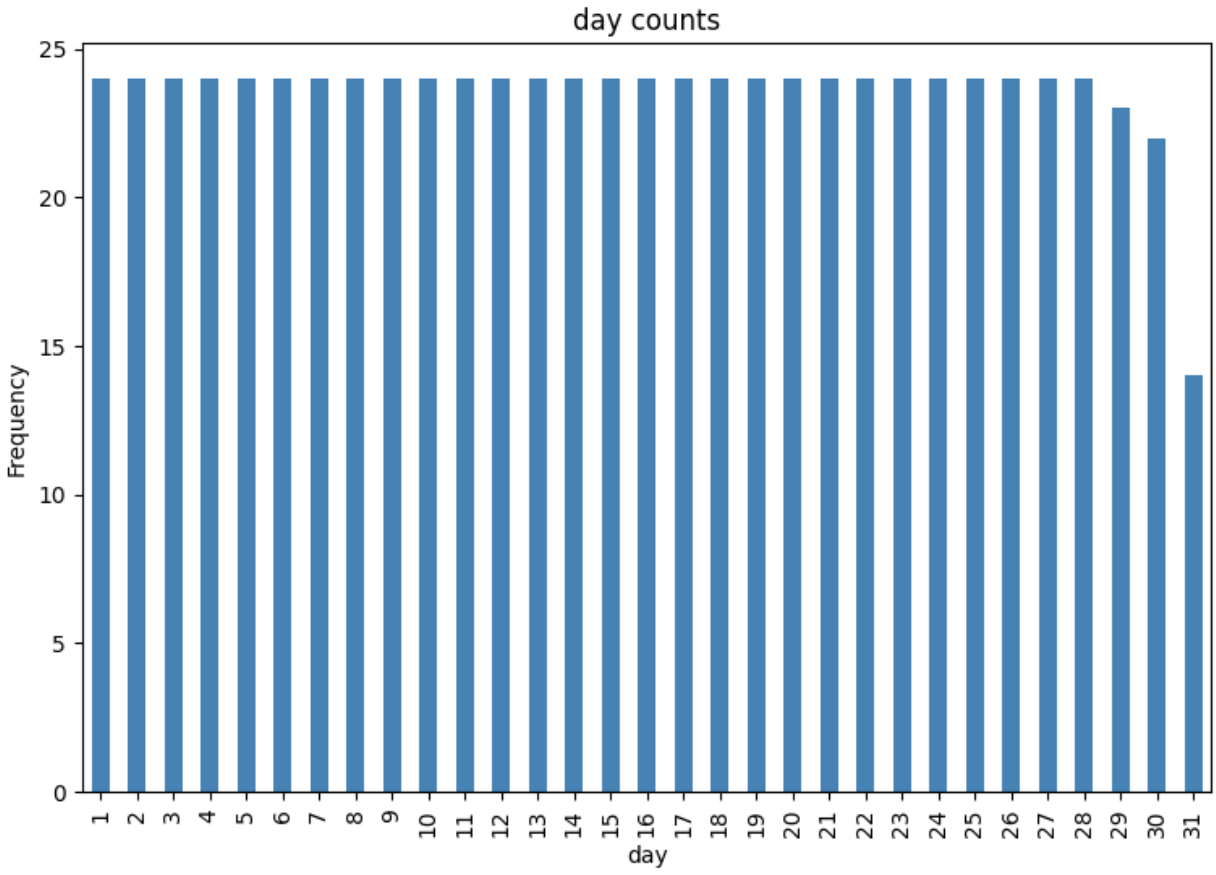

mnth counts
Frequency
0
10
20
30
40
50
60
1
2
3
4
5
6
7
8
9
10
11
12
mnth


holiday counts
Frequency
0
100
200
300
400
500
600
700
0
1
holiday

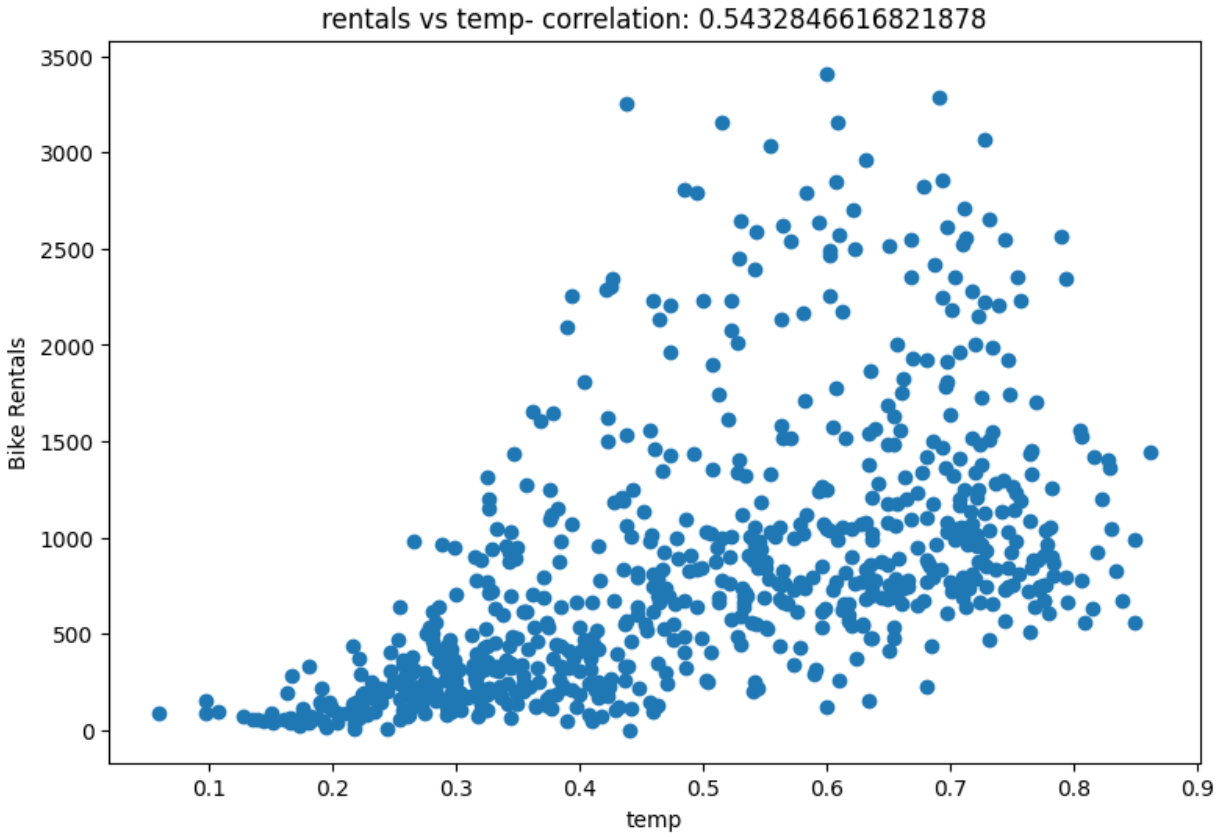

weekday counts
Frequency
0
20
40
60
80
100
0
1
2
3
4
5
6
weekday

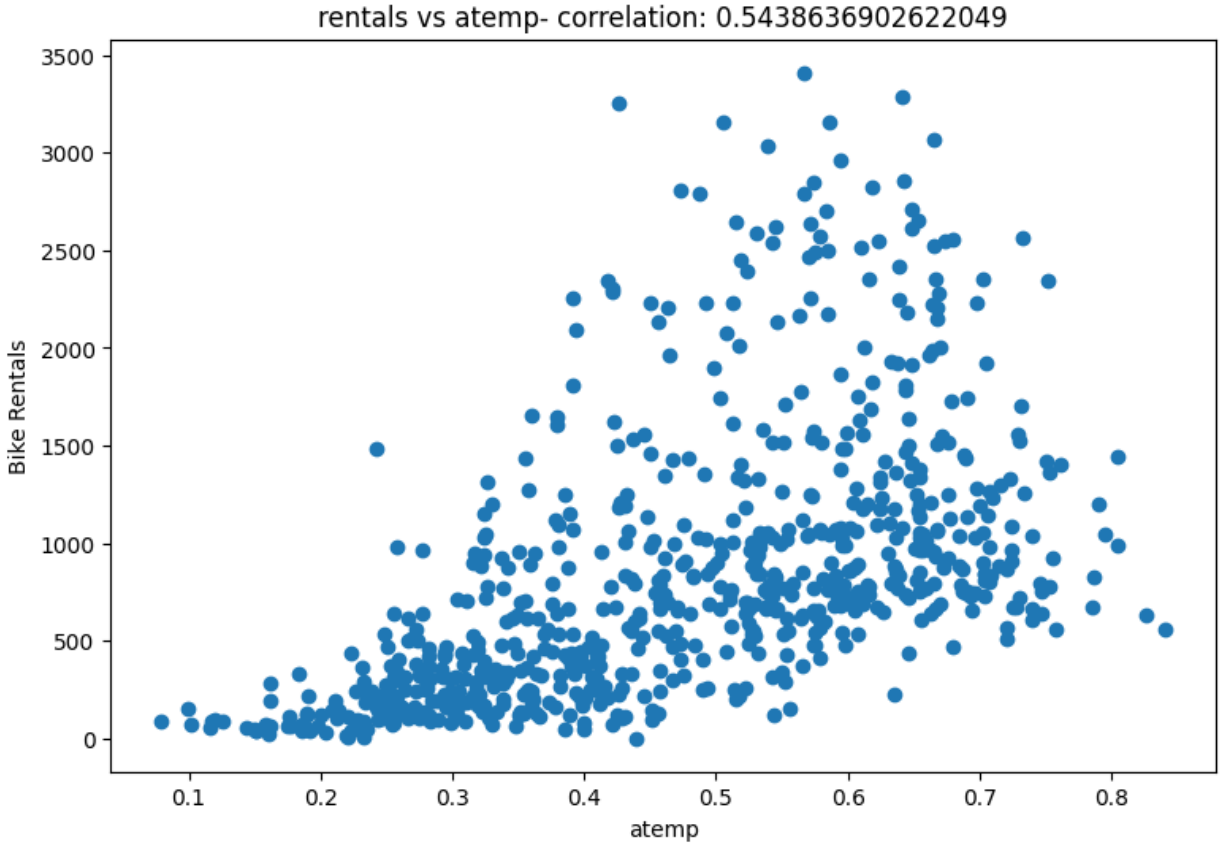

workingday counts
Frequency
0
100
200
300
400
500
0
1
workingday

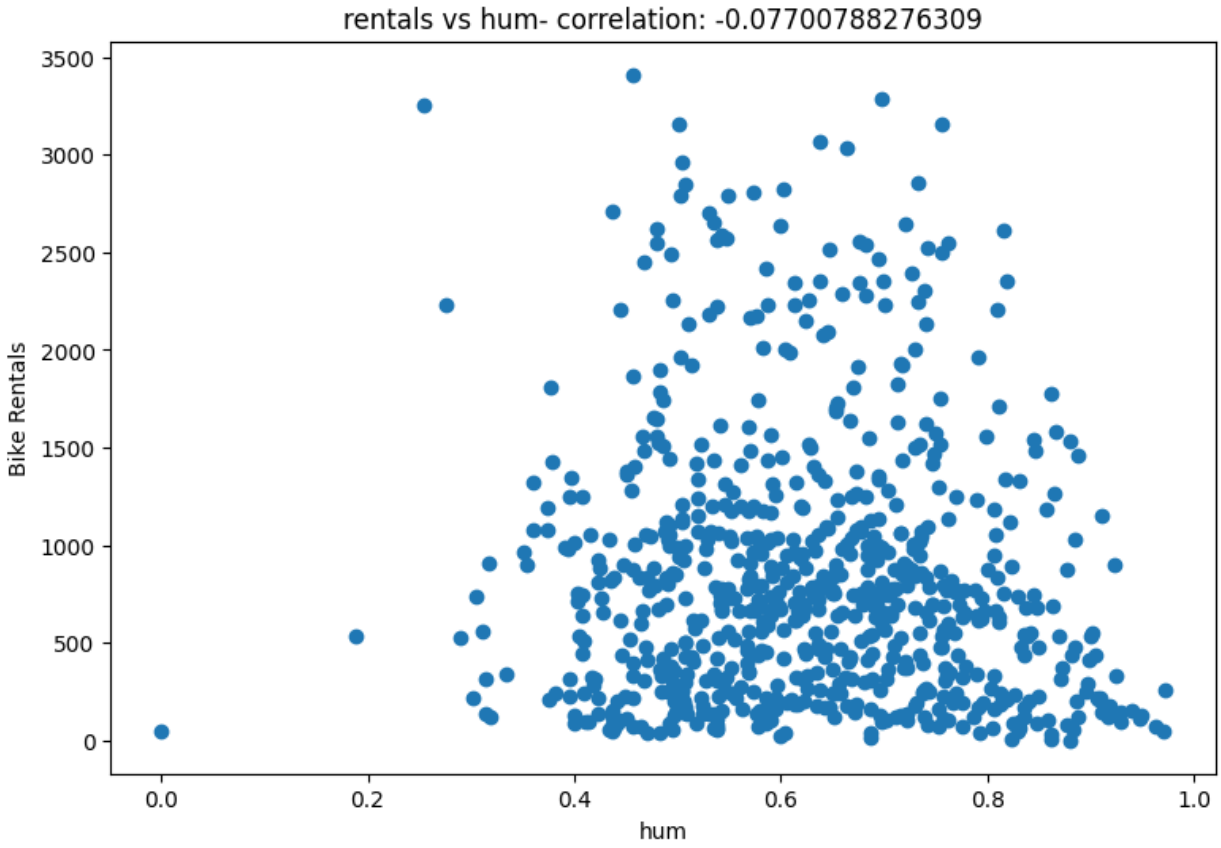

weathersit counts
Frequency
0
100
200
300
400
1
2
3
weathersit

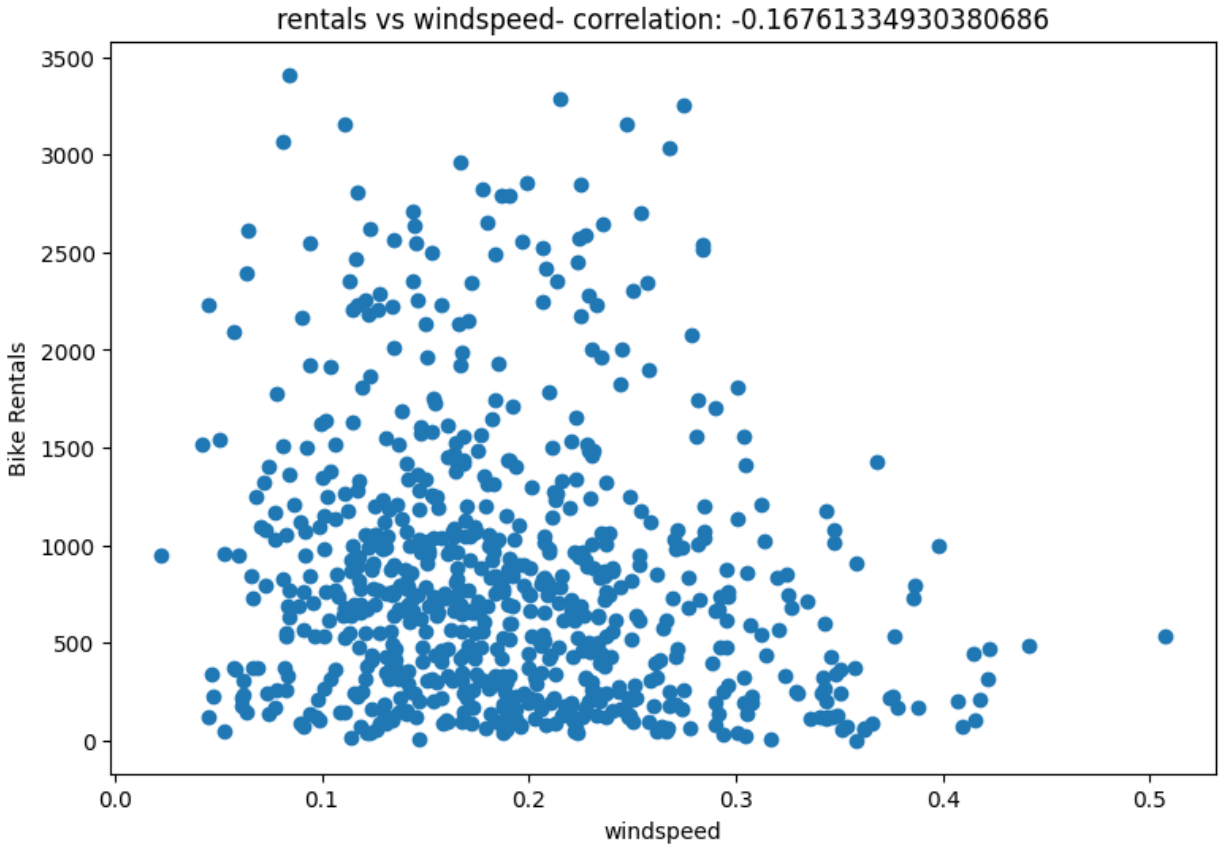


In [ ]:
```
for col in numeric_features:
    fig = plt.figure(figsize=(9, 6))
    ax = fig.gca()
    feature = bike_data[col]
    label = bike_data['rentals']
    correlation = feature.corr(label)
    plt.scatter(x=feature, y=label)
    plt.xlabel(col)
    plt.ylabel('Bike Rentals')
    ax.set_title('rentals vs ' + col + '- correlation: ' + str(correlation))
plt.show()
```


rentals vs temp- correlation: 0.5432846616821878
Bike Rentals
temp
0
500
1000
1500
2000
2500
3000
3500
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

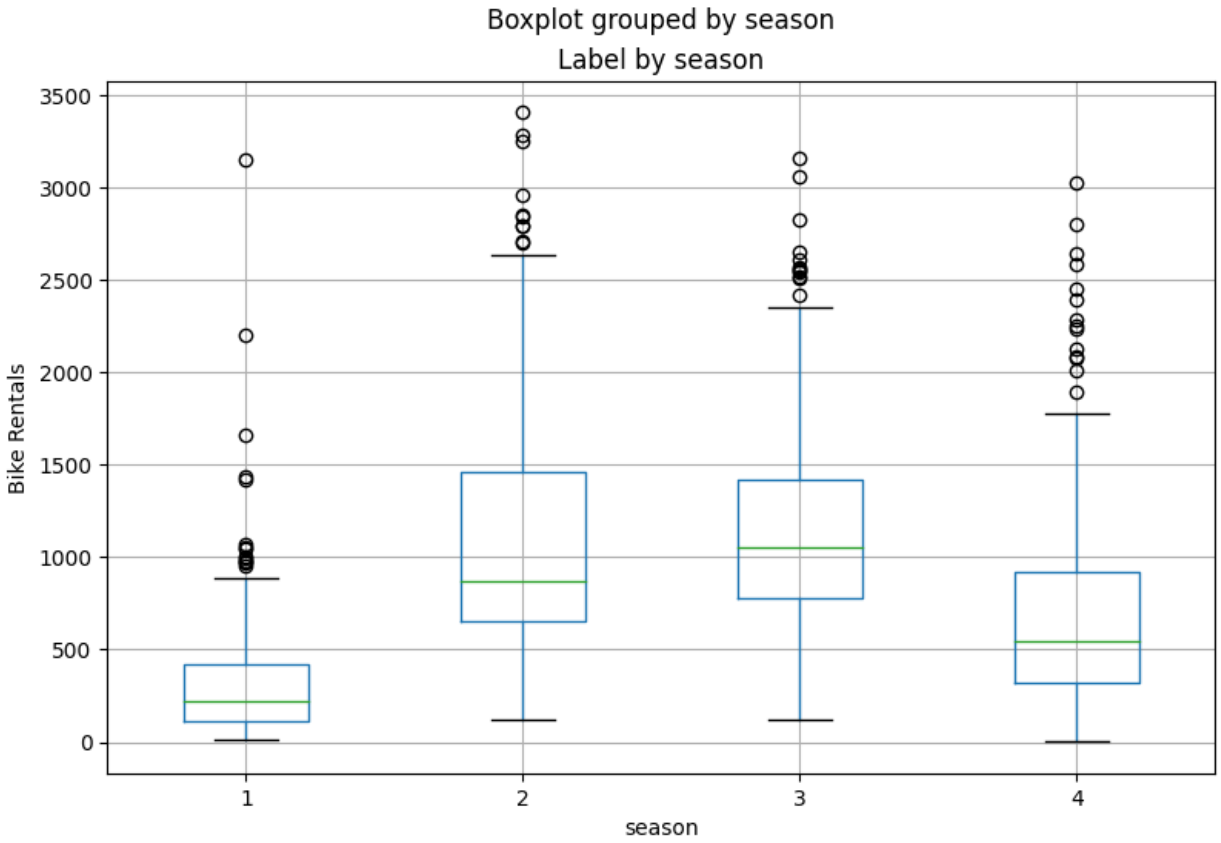

rentals vs atemp- correlation: 0.5438636902622049
Bike Rentals
atemp
0
500
1000
1500
2000
2500
3000
3500
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

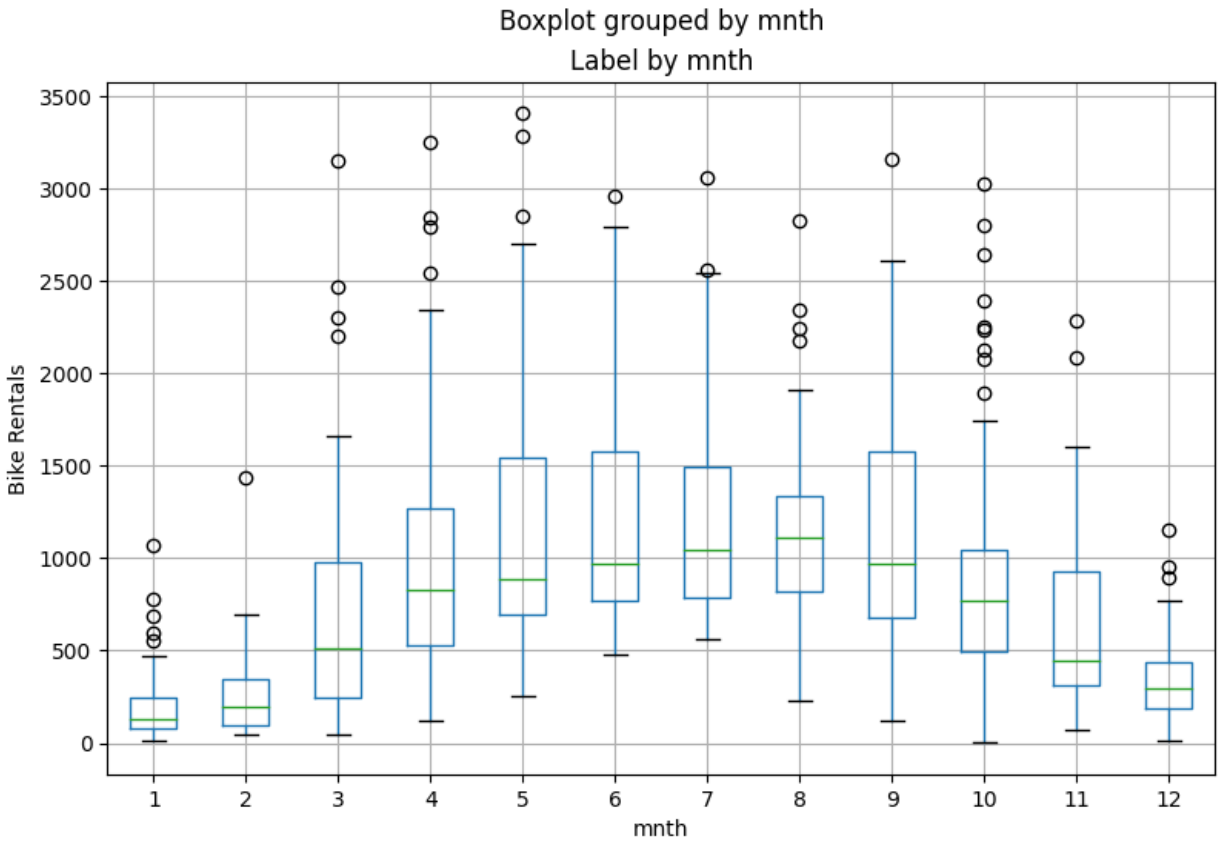


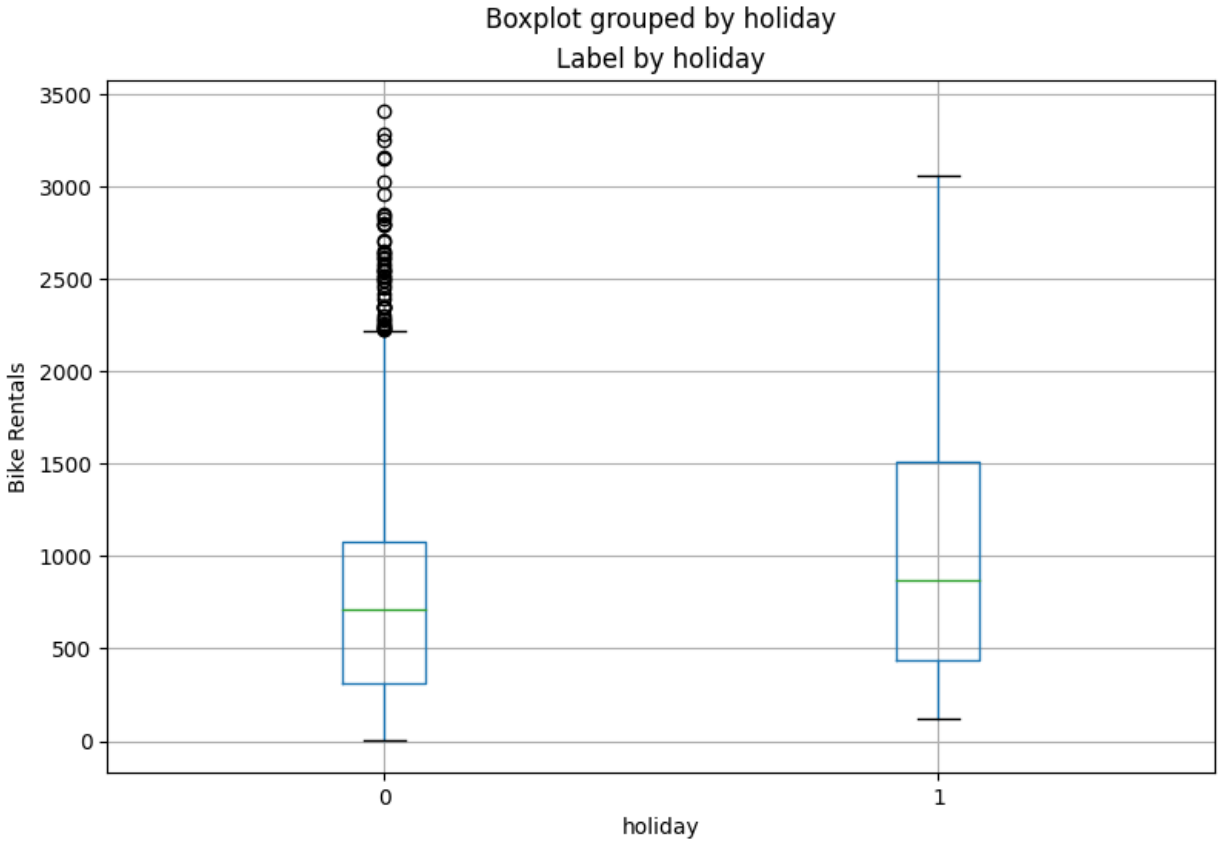


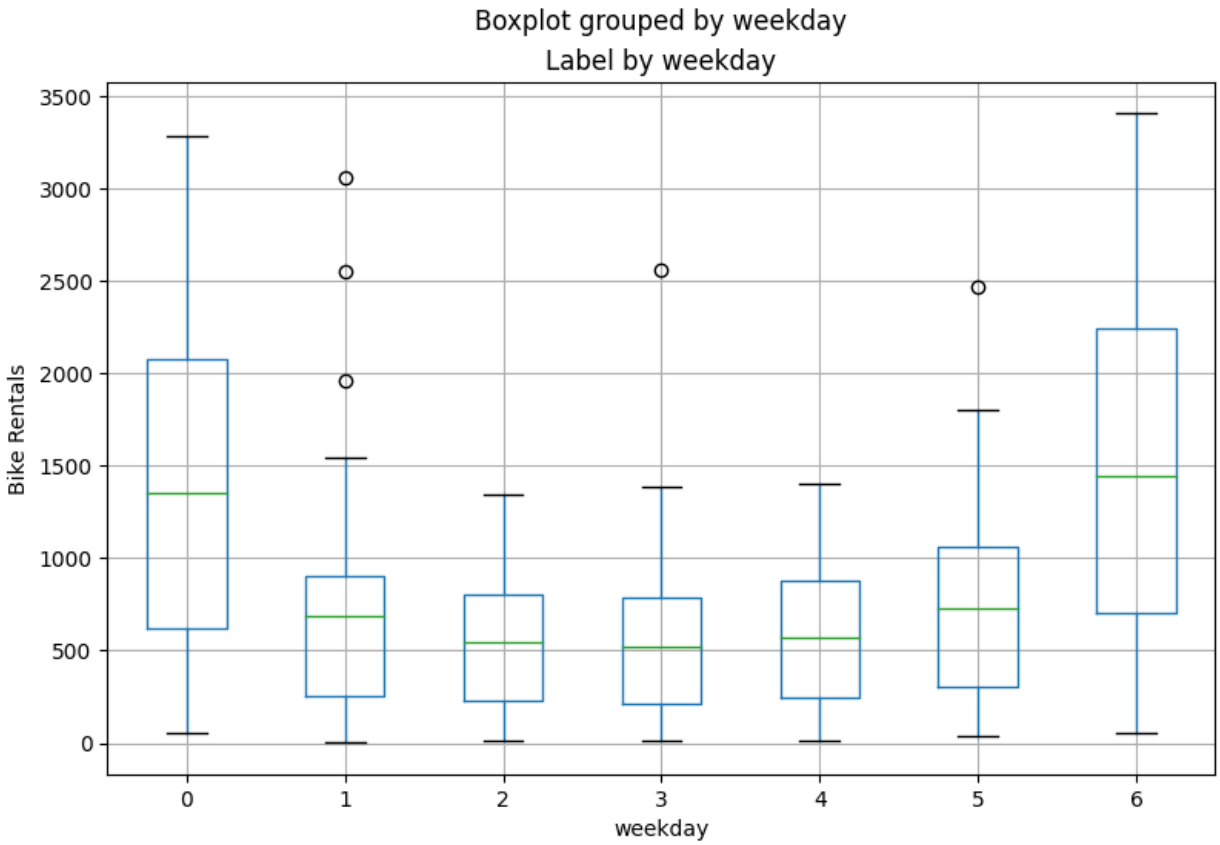
Trazar un diagrama de caja para la etiqueta por cada característica categórica

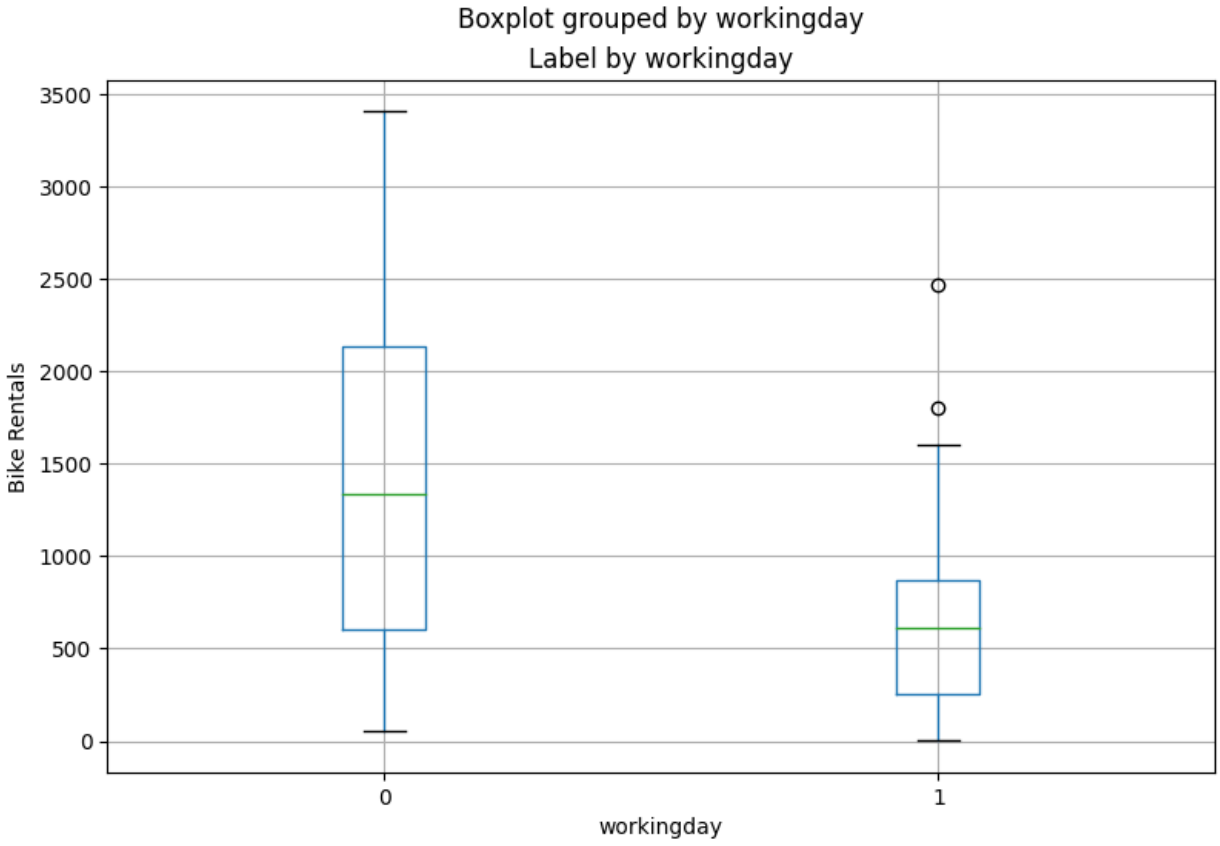
In [ ]:
```
#
for col in categorical_features:
    fig = plt.figure(figsize=(9, 6))
    ax = fig.gca()
    bike_data.boxplot(column = 'rentals', by = col, ax = ax)
    ax.set_title('Label by ' + col)
    ax.set_ylabel("Bike Rentals")
plt.show()
```

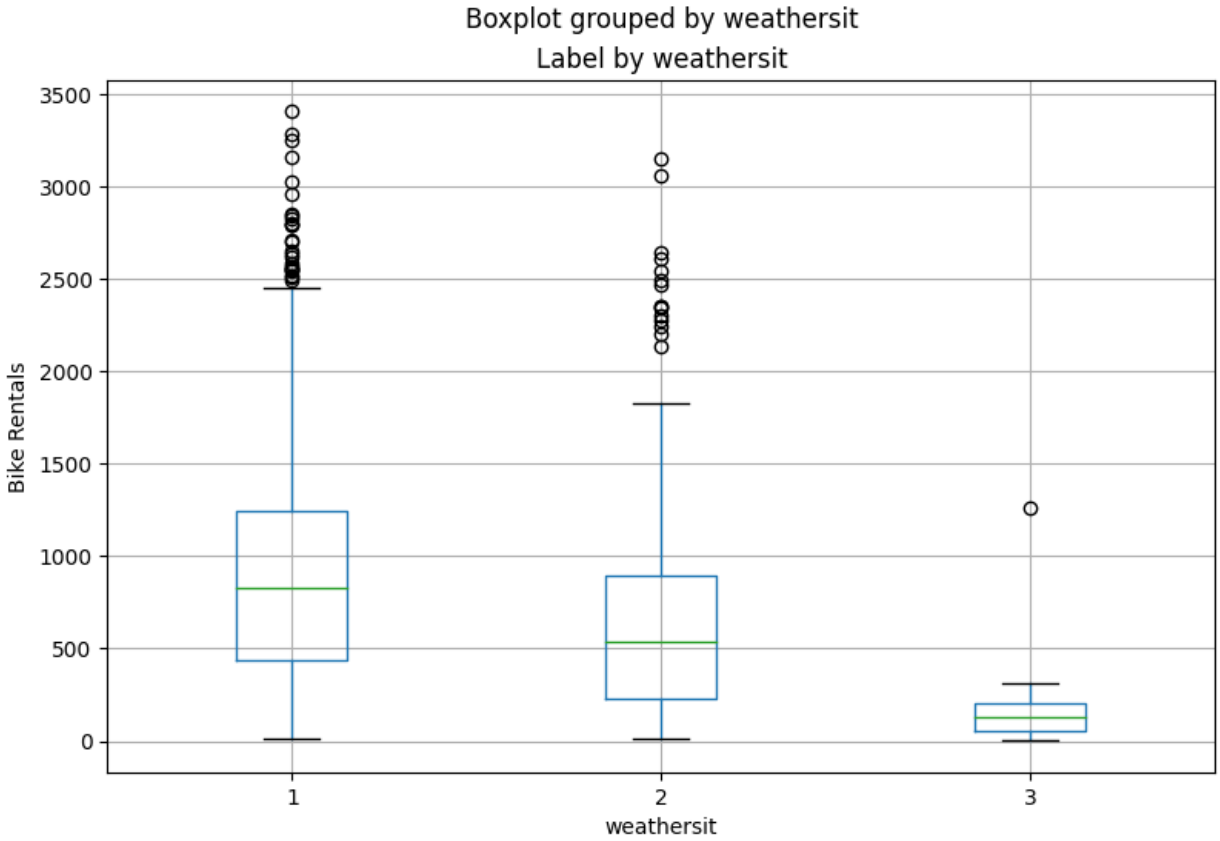


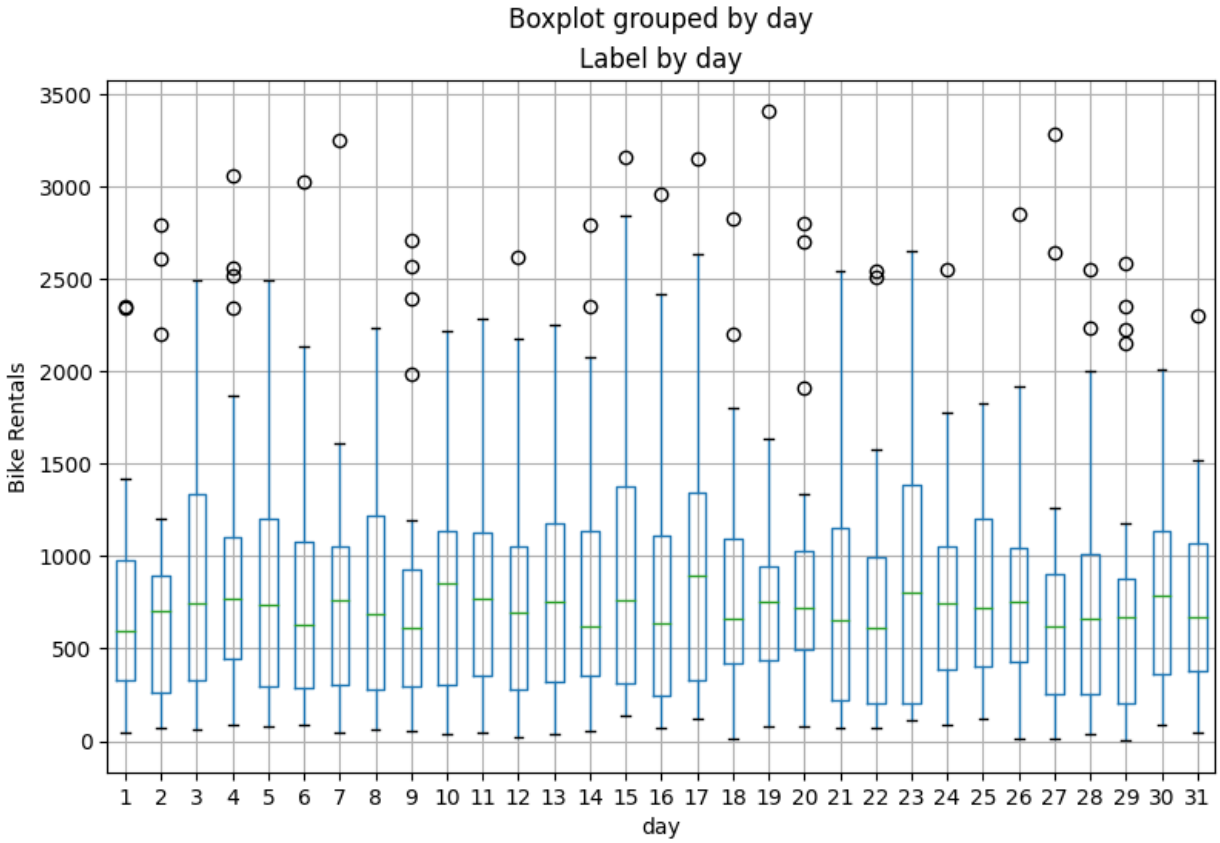

Boxplot grouped by mnth
Label by mnth
Bike Rentals
mnth
0
500
1000
1500
2000
2500
3000
3500
1
2
3
4
5
6
7
8
9
10
11
12


Boxplot grouped by holiday
Label by holiday
Bike Rentals
holiday
0
500
1000
1500
2000
2500
3000
3500
0
1


Boxplot grouped by weekday
Label by weekday
Bike Rentals
3500
3000
2500
2000
1500
1000
500
0
0
1
2
3
4
5
6
weekday


Boxplot grouped by workingday
Label by workingday
Bike Rentals
3500
3000
2500
2000
1500
1000
500
0
0
1
workingday





Características y etiquetas separadas

In [ ]:
```
#
X, y = bike_data[['season','mnth', 'holiday','weekday','workingday','weathersit','temp
```

```
print('Features:',X[:10], '\nLabels:', y[:10], sep='\n')
```

```
Features:
[[1.        1.        0.        6.        0.        2.        0.344167
  0.363625  0.805833  0.160446 ]
 [1.        1.        0.        0.        0.        2.        0.363478
  0.353739  0.696087  0.248539 ]
 [1.        1.        0.        1.        1.        1.        0.196364
  0.189405  0.437273  0.248309 ]
 [1.        1.        0.        2.        1.        1.        0.2
  0.212122  0.590435  0.160296 ]
 [1.        1.        0.        3.        1.        1.        0.226957
  0.22927   0.436957  0.1869   ]
 [1.        1.        0.        4.        1.        1.        0.204348
  0.233209  0.518261  0.0895652]
 [1.        1.        0.        5.        1.        2.        0.196522
  0.208839  0.498696  0.168726 ]
 [1.        1.        0.        6.        0.        2.        0.165
  0.162254  0.535833  0.266804 ]
 [1.        1.        0.        0.        0.        1.        0.138333
  0.116175  0.434167  0.36195  ]
 [1.        1.        0.        1.        1.        1.        0.150833
  0.150888  0.482917  0.223267 ]]

Labels:
[331 131 120 108  82  88 148  68  54  41]
```

Divida los datos entre el 70% y el 30% en el conjunto de entrenamiento y el conjunto de prueba

In [ ]:
```
from sklearn.model_selection import train_test_split

#
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state

print ('Training Set: %d rows\nTest Set: %d rows' % (X_train.shape[0], X_test.shape[0]
```

```
Training Set: 511 rows
Test Set: 220 rows
```

Entrenar el modelo

Ajustar un modelo de regresión lineal en el conjunto de entrenamiento

In [ ]:
```
#
from sklearn.linear_model import LinearRegression

#
model = LinearRegression().fit(X_train, y_train)
print (model)
```

```
LinearRegression()
```

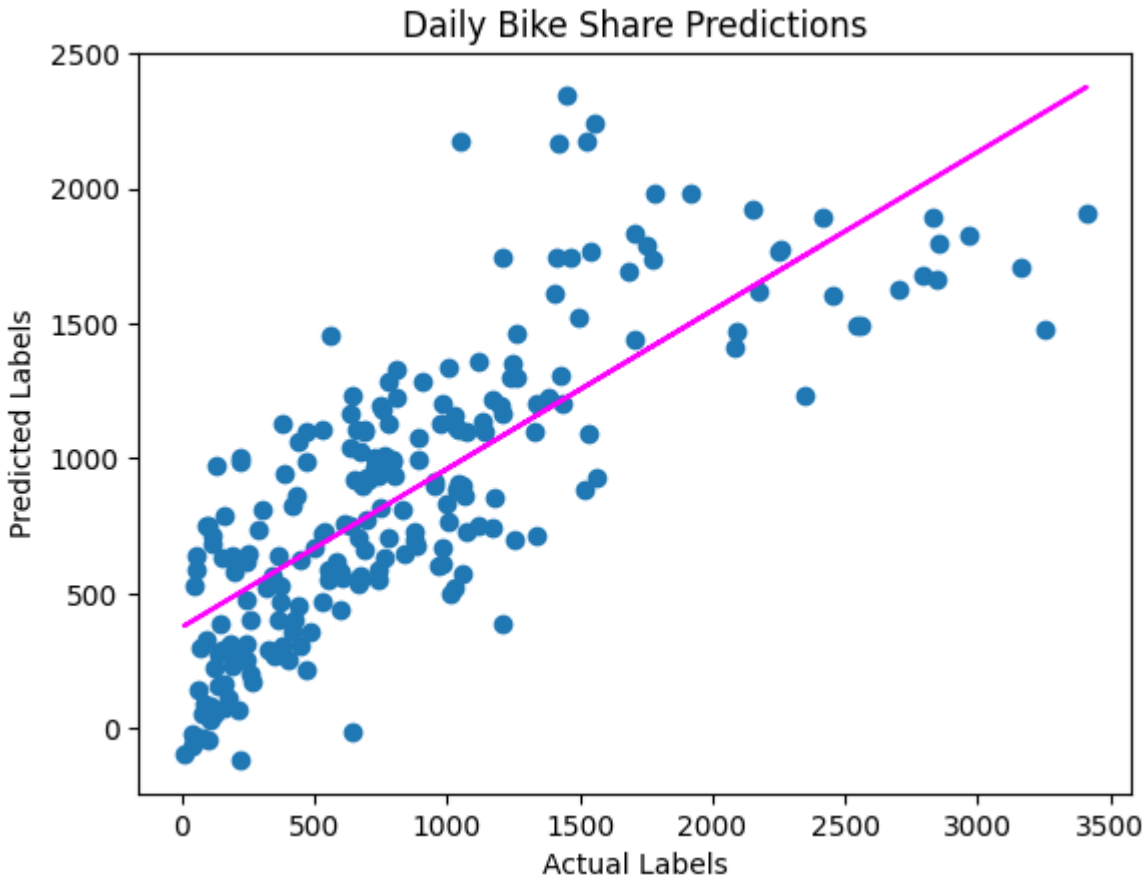
In [ ]:
```
import numpy as np
```

```
predictions = model.predict(X_test)
np.set_printoptions(suppress=True)
print('Predicted labels: ', np.round(predictions)[:10])
print('Actual labels   : ' ,y_test[:10])
```

```
Predicted labels:  [1896. 1184. 1007.  -28.  314.  385.  475.  590. 1476.  -22.]
Actual labels   :  [2418  754  222   47  244  145  240  555 3252   38]
```

Superponer la línea de regresión

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

%matplotlib inline

plt.scatter(y_test, predictions)
plt.xlabel('Actual Labels')
plt.ylabel('Predicted Labels')
plt.title('Daily Bike Share Predictions')
#
z = np.polyfit(y_test, predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test), color='magenta')
plt.show()
```



In [ ]:
```
from sklearn.metrics import mean_squared_error, r2_score

mse = mean_squared_error(y_test, predictions)
print("MSE:", mse)
```

```
rmse = np.sqrt(mse)
print("RMSE:", rmse)

r2 = r2_score(y_test, predictions)
print("R2:", r2)
```

```
MSE: 201972.55947035595
RMSE: 449.4135728595165
R2: 0.604045473691919
```