In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import seaborn as sns
```

In [2]:
```
#Load the dataset
bike = pd.read_csv('daily-bike-share.csv')
```

In [3]:
```
bike.head()
```

Out[3]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	a
0	1	1/1/2011	1	0	1	0	6	0	2	0.344167	0.3
1	2	1/2/2011	1	0	1	0	0	0	2	0.363478	0.3
2	3	1/3/2011	1	0	1	0	1	1	1	0.196364	0.1
3	4	1/4/2011	1	0	1	0	2	1	1	0.200000	0.2
4	5	1/5/2011	1	0	1	0	3	1	1	0.226957	0.2

- **instant**: A unique row identifier
- **dteday**: The date on which the data was observed - in this case, the data was collected daily; so there's one row per date.
- **season**: A numerically encoded value indicating the season (1:winter, 2:spring, 3:summer, 4:fall)
- **yr**: The year of the study in which the observation was made (the study took place over two years - year 0 represents 2011, and year 1 represents 2012)
- **mnth**: The calendar month in which the observation was made (1:January ... 12:December)
- **holiday**: A binary value indicating whether or not the observation was made on a public holiday)
- **weekday**: The day of the week on which the observation was made (0:Sunday ... 6:Saturday)
- **workingday**: A binary value indicating whether or not the day is a working day (not a weekend or holiday)
- **weathersit**: A categorical value indicating the weather situation (1:clear, 2:mist/cloud, 3:light rain/snow, 4:heavy rain/hail/snow/fog)
- **temp**: The temperature in celsius (normalized)
- **atemp**: The apparent ("feels-like") temperature in celsius (normalized)
- **hum**: The humidity level (normalized)
- **windspeed**: The windspeed (normalized)
- **rentals**: The number of bicycle rentals recorded.

In [4]:
```
#day extraction
bike['day'] = pd.DatetimeIndex(bike['dteday']).day
bike.head()
```

Out[4]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	a
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

In [5]:
```
bike.head(31)
```

Out[5]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	
0	1	1/1/2011	1	0	1	0	6	0	2	0.344167	0
1	2	1/2/2011	1	0	1	0	0	0	2	0.363478	0
2	3	1/3/2011	1	0	1	0	1	1	1	0.196364	0
3	4	1/4/2011	1	0	1	0	2	1	1	0.200000	0
4	5	1/5/2011	1	0	1	0	3	1	1	0.226957	0
5	6	1/6/2011	1	0	1	0	4	1	1	0.204348	0
6	7	1/7/2011	1	0	1	0	5	1	2	0.196522	0
7	8	1/8/2011	1	0	1	0	6	0	2	0.165000	0
8	9	1/9/2011	1	0	1	0	0	0	1	0.138333	0
9	10	1/10/2011	1	0	1	0	1	1	1	0.150833	0
10	11	1/11/2011	1	0	1	0	2	1	2	0.169091	0
11	12	1/12/2011	1	0	1	0	3	1	1	0.172727	0
12	13	1/13/2011	1	0	1	0	4	1	1	0.165000	0
13	14	1/14/2011	1	0	1	0	5	1	1	0.160870	0
14	15	1/15/2011	1	0	1	0	6	0	2	0.233333	0
15	16	1/16/2011	1	0	1	0	0	0	1	0.231667	0
16	17	1/17/2011	1	0	1	1	1	0	2	0.175833	0
17	18	1/18/2011	1	0	1	0	2	1	2	0.216667	0
18	19	1/19/2011	1	0	1	0	3	1	2	0.292174	0
19	20	1/20/2011	1	0	1	0	4	1	2	0.261667	0
20	21	1/21/2011	1	0	1	0	5	1	1	0.177500	0
21	22	1/22/2011	1	0	1	0	6	0	1	0.059130	0
22	23	1/23/2011	1	0	1	0	0	0	1	0.096522	0
23	24	1/24/2011	1	0	1	0	1	1	1	0.097391	0
24	25	1/25/2011	1	0	1	0	2	1	2	0.223478	0
25	26	1/26/2011	1	0	1	0	3	1	3	0.217500	0
26	27	1/27/2011	1	0	1	0	4	1	1	0.195000	0
27	28	1/28/2011	1	0	1	0	5	1	2	0.203478	0
28	29	1/29/2011	1	0	1	0	6	0	1	0.196522	0
29	30	1/30/2011	1	0	1	0	0	0	1	0.216522	0
30	31	1/31/2011	1	0	1	0	1	1	2	0.180833	0

In [6]:
```
bike.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 731 entries, 0 to 730
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   instant     731 non-null    int64  
 1   dteday      731 non-null    object 
 2   season      731 non-null    int64  
 3   yr          731 non-null    int64  
 4   mnth        731 non-null    int64  
 5   holiday     731 non-null    int64  
 6   weekday     731 non-null    int64  
 7   workingday  731 non-null    int64  
 8   weathersit  731 non-null    int64  
 9   temp        731 non-null    float64
 10  atemp       731 non-null    float64
 11  hum         731 non-null    float64
 12  windspeed   731 non-null    float64
 13  rentals     731 non-null    int64  
 14  day         731 non-null    int64  
dtypes: float64(4), int64(10), object(1)
memory usage: 85.8+ KB
```

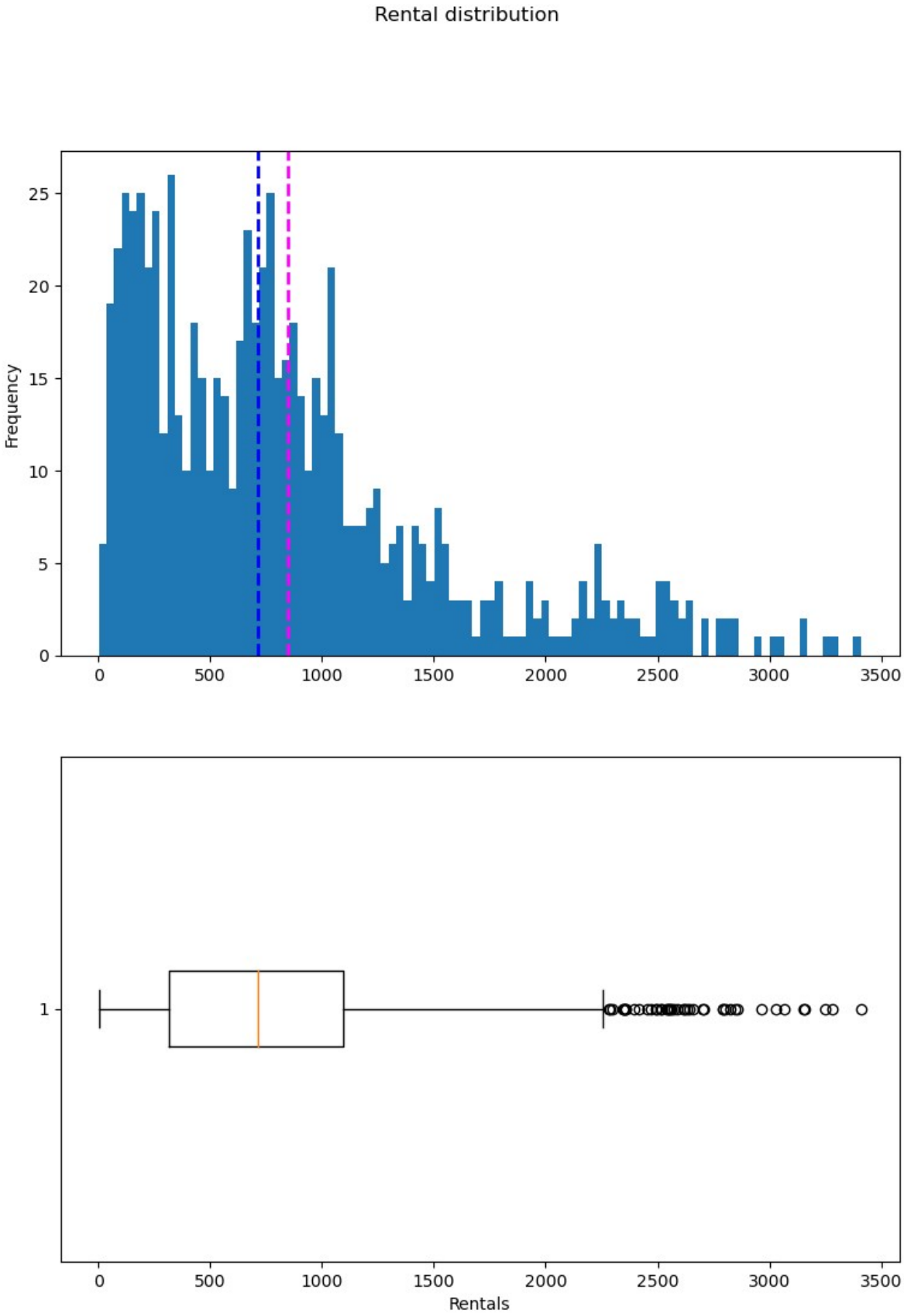
In [7]:
```
numeric_features = ['temp', 'atemp', 'hum', 'windspeed']
bike[numeric_features+['rentals']].describe()
```

Out[7]:

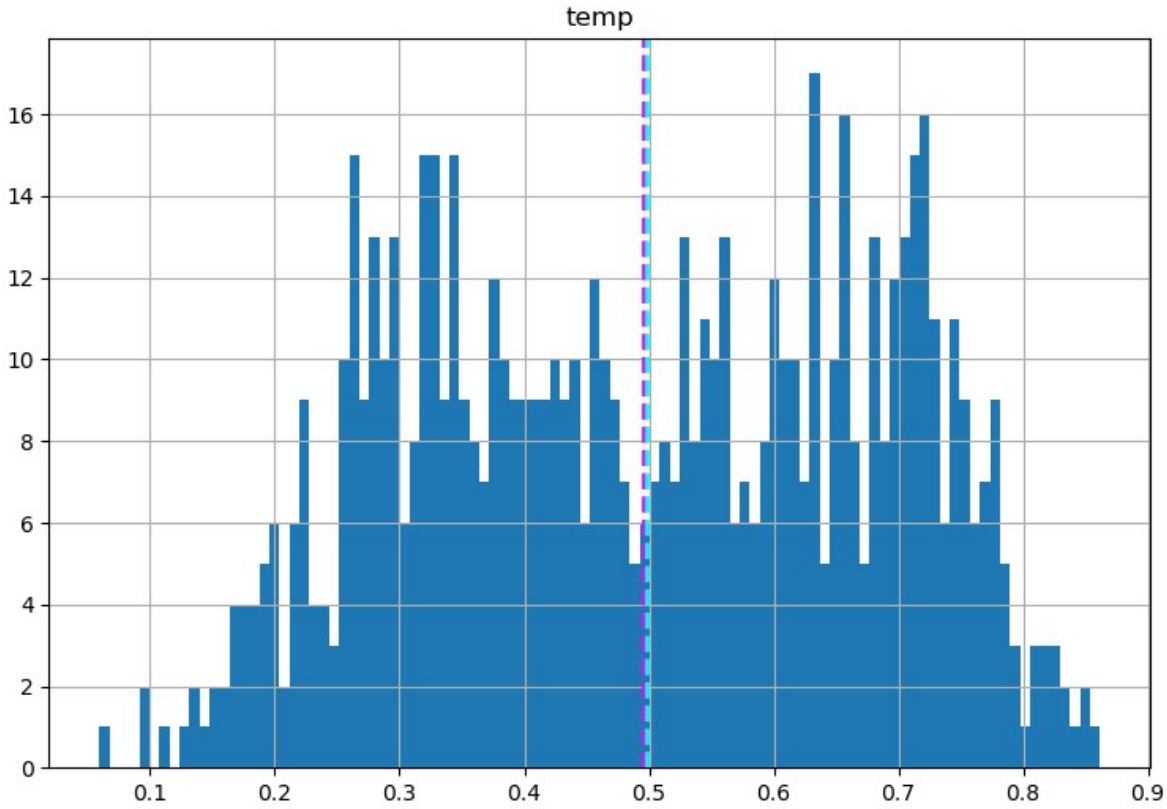
	temp	atemp	hum	windspeed	rentals
count	731.000000	731.000000	731.000000	731.000000	731.000000
mean	0.495385	0.474354	0.627894	0.190486	848.176471
std	0.183051	0.162961	0.142429	0.077498	686.622488
min	0.059130	0.079070	0.000000	0.022392	2.000000
25%	0.337083	0.337842	0.520000	0.134950	315.500000
50%	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

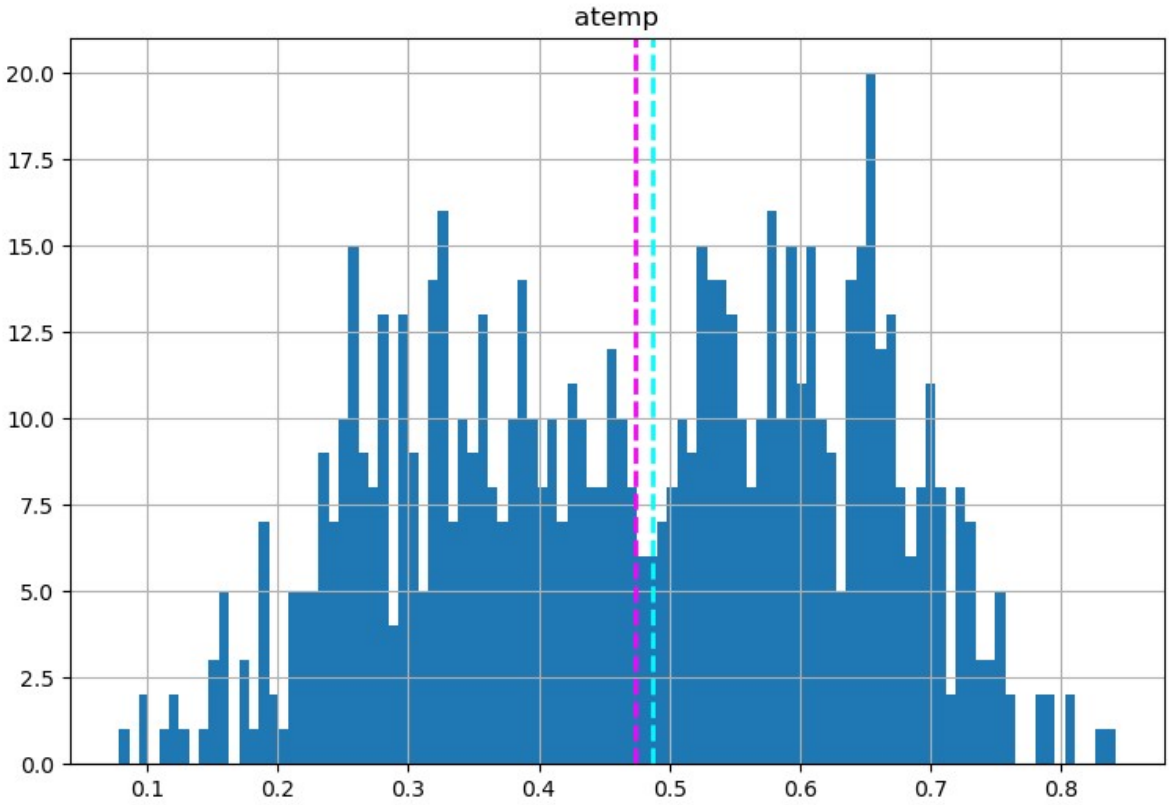
In [8]:
```
label = bike['rentals']

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

#histogram
ax[0].hist(label, bins=100)
ax[0].set_ylabel('Frequency')
#mean, median and mode lines
ax[0].axvline(label.mean(), c='magenta',ls='dashed', lw=2)
ax[0].axvline(label.median(), c='blue',ls='dashed', lw=2)

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

fig.suptitle('Rental distribution')
fig.show()
```

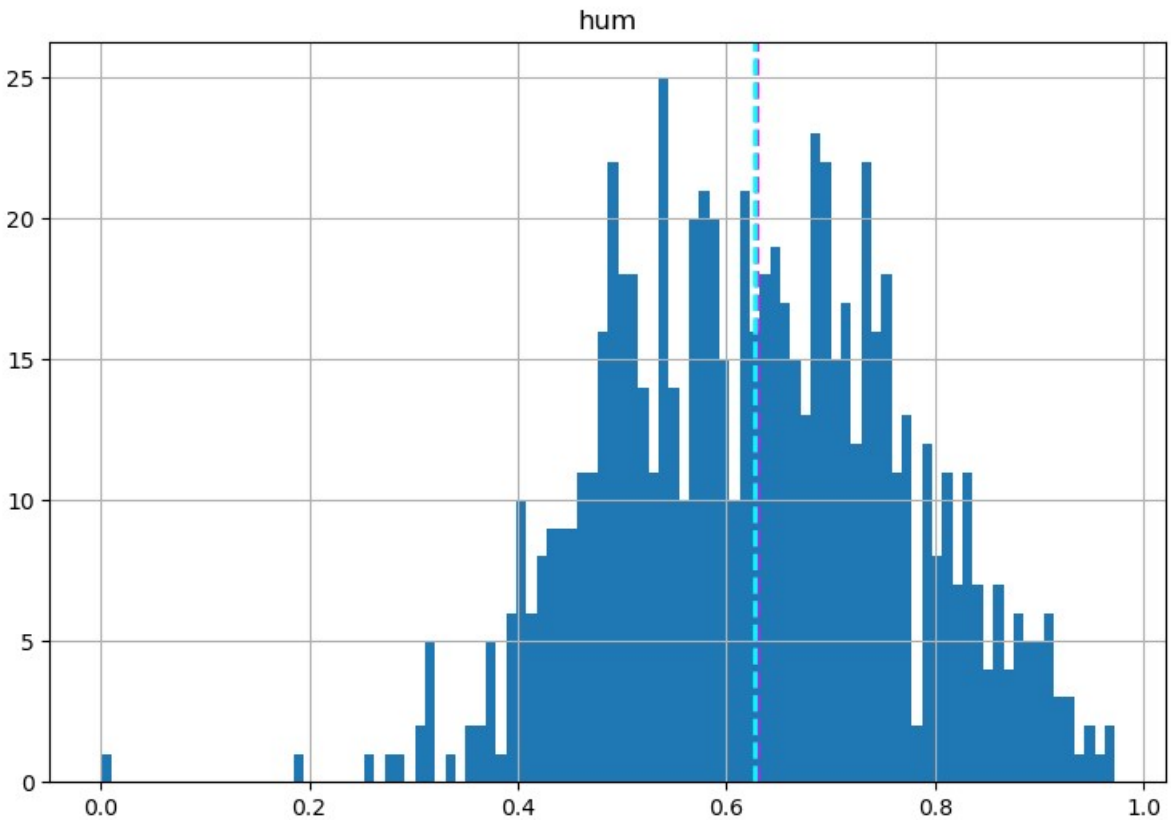
```
C:\Users\steph\AppData\Local\Temp\ipykernel_23048\3744000061.py:17: UserWarni
ng: Matplotlib is currently using module://matplotlib_inline.backend_inline,
which is a non-GUI backend, so cannot show the figure.
  fig.show()
```


Rental distribution
Frequency
0
5
10
15
20
25
0
500
1000
1500
2000
2500
3000
3500
1
Rentals

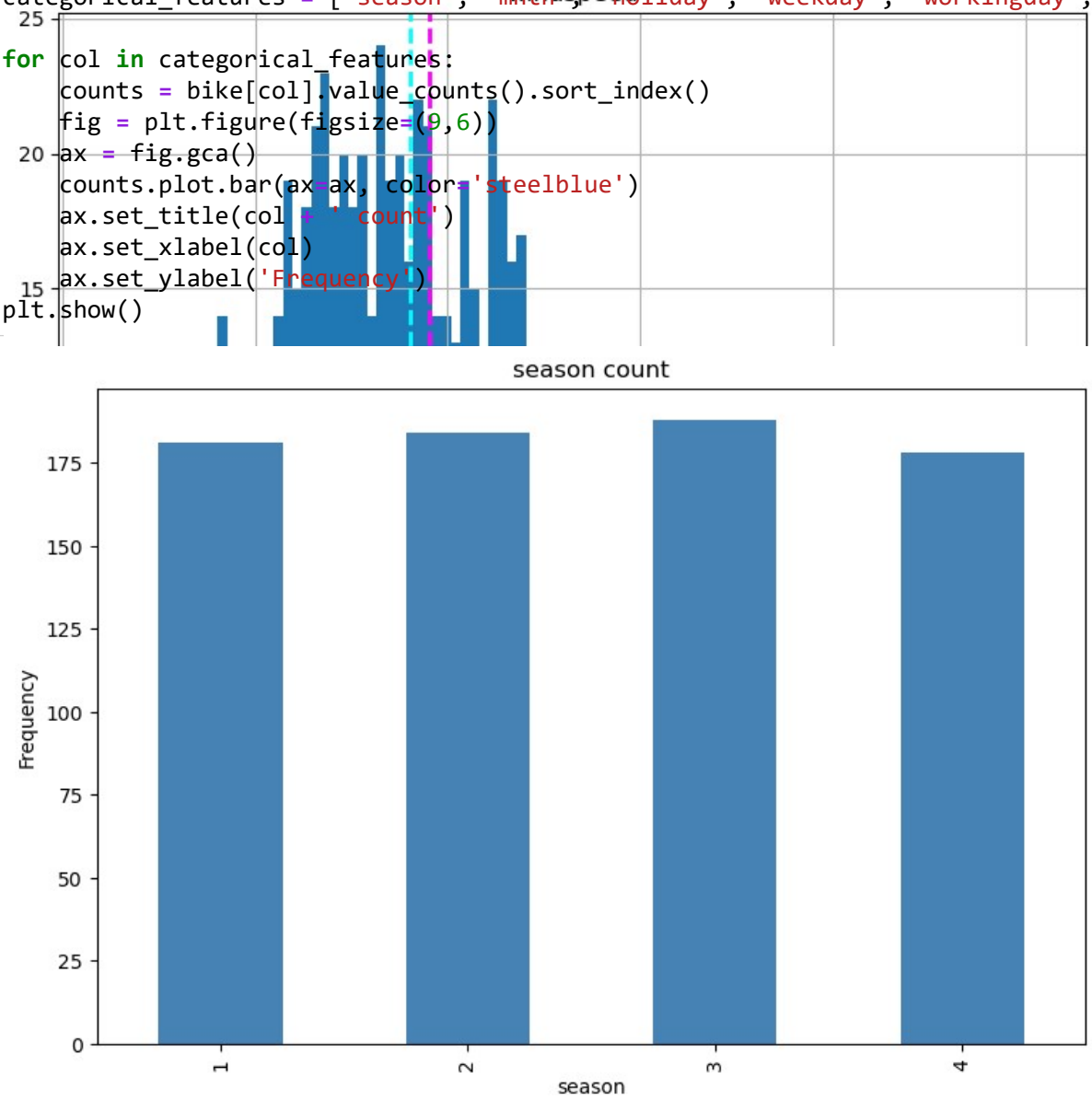
In [9]:
```
#numeric feature plot
for col in numeric_features:
    fig = plt.figure(figsize=(9,6))
    ax = fig.gca()
    feature = bike[col]
    feature.hist(bins=100, ax=ax)
    ax.axvline(feature.mean(), c='magenta', ls='dashed', lw=2)
    ax.axvline(feature.median(), c='cyan', ls='dashed', lw=2)
    ax.set_title(col)
plt.show()
```

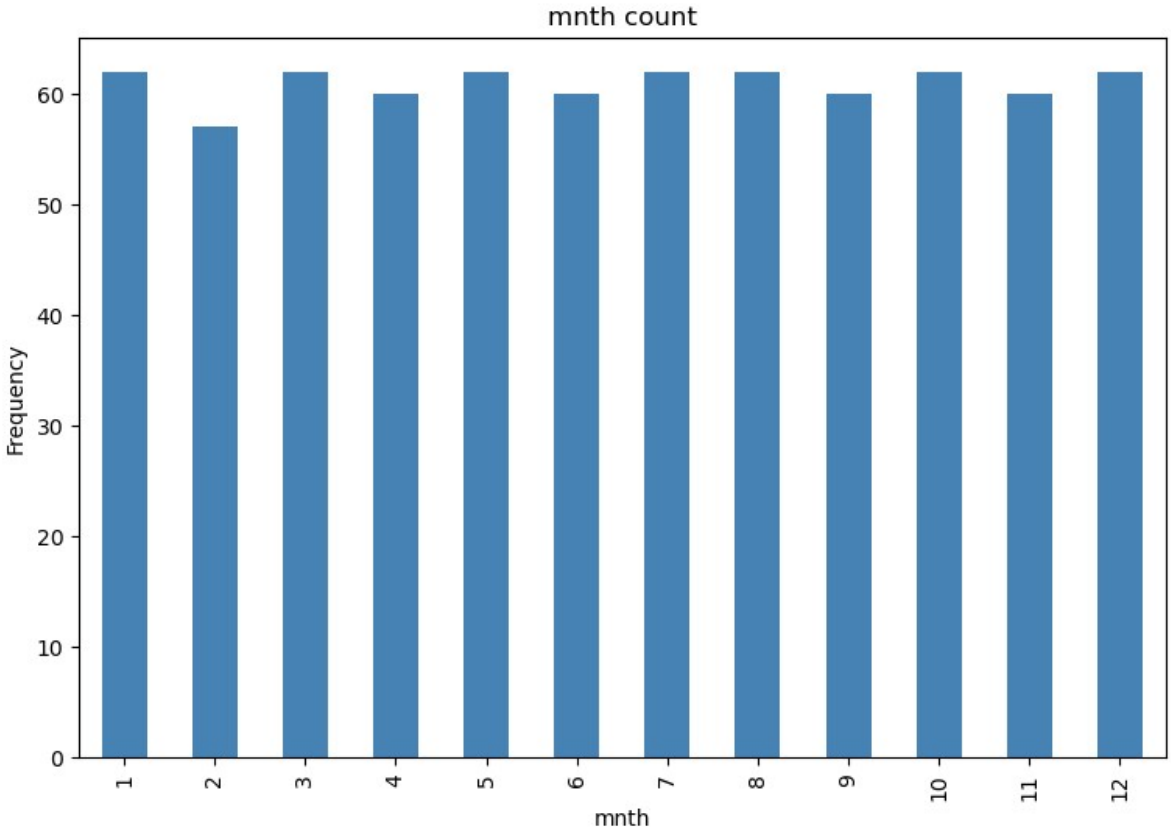


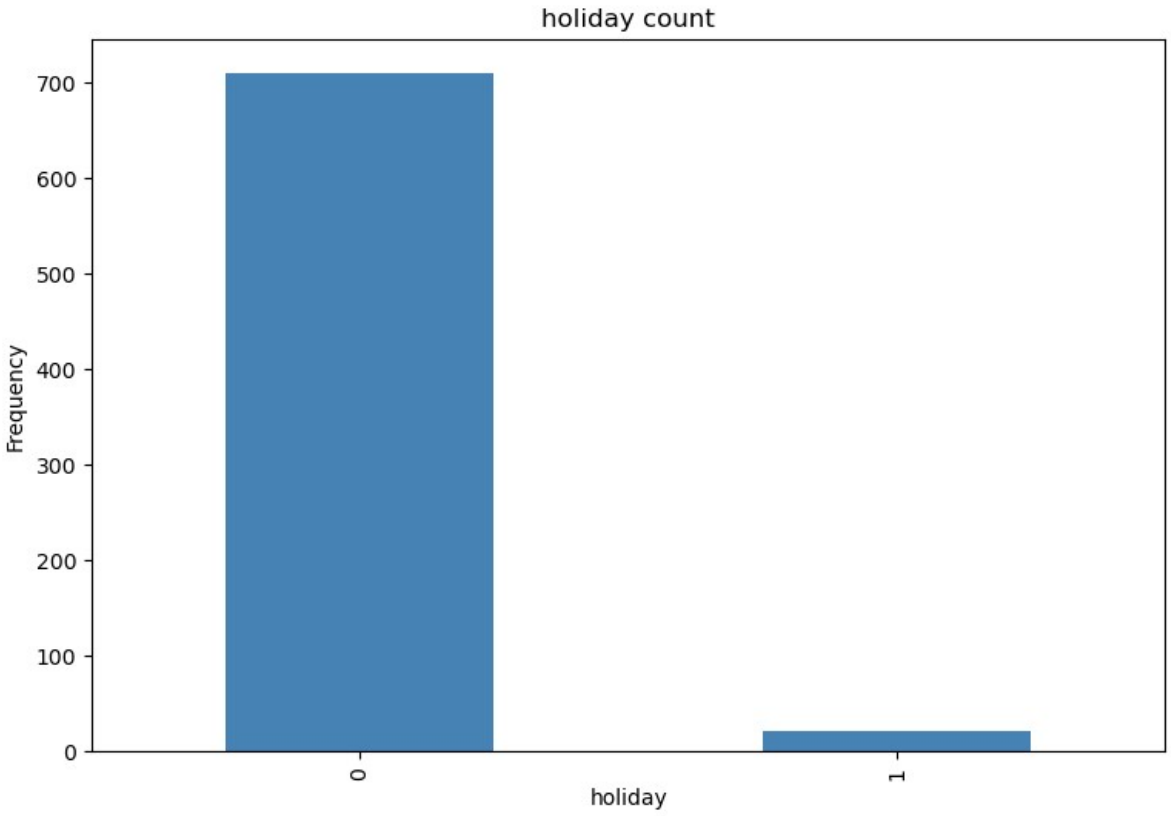

atemp
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

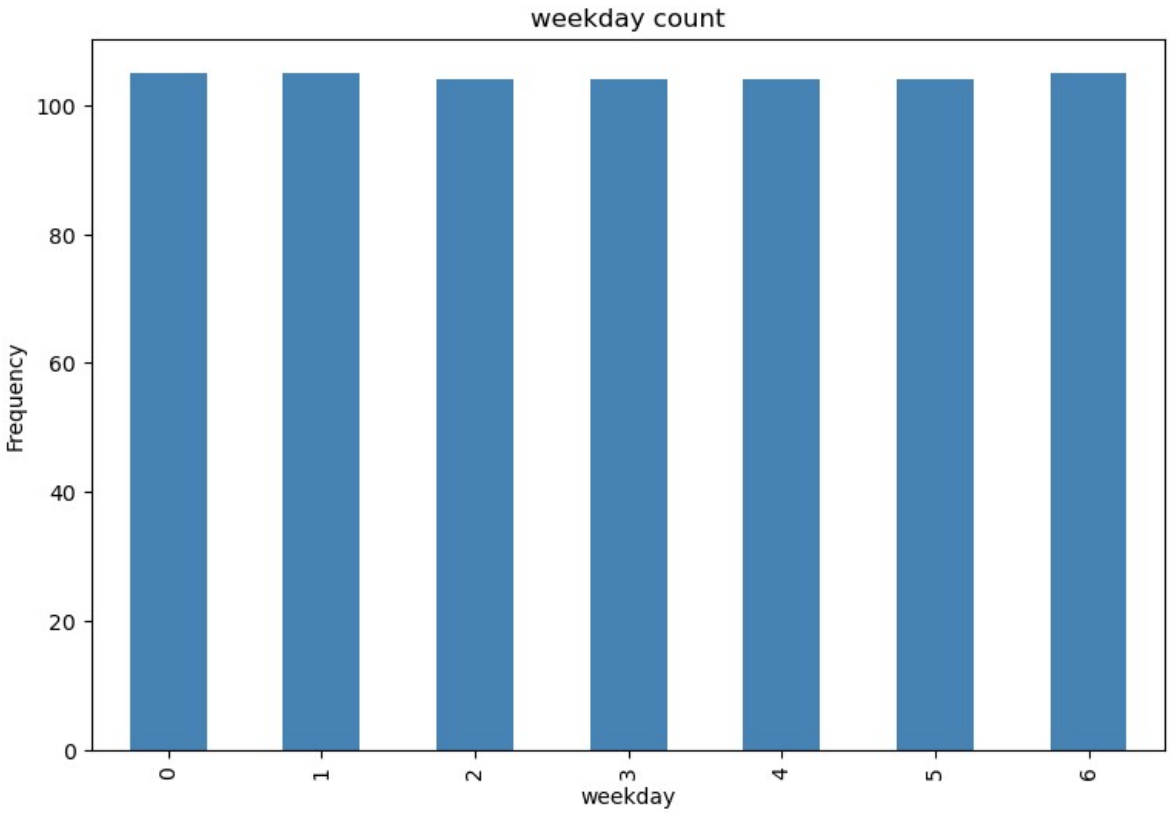

hum
25
20
15
10
5
0
0.0
0.2
0.4
0.6
0.8
1.0

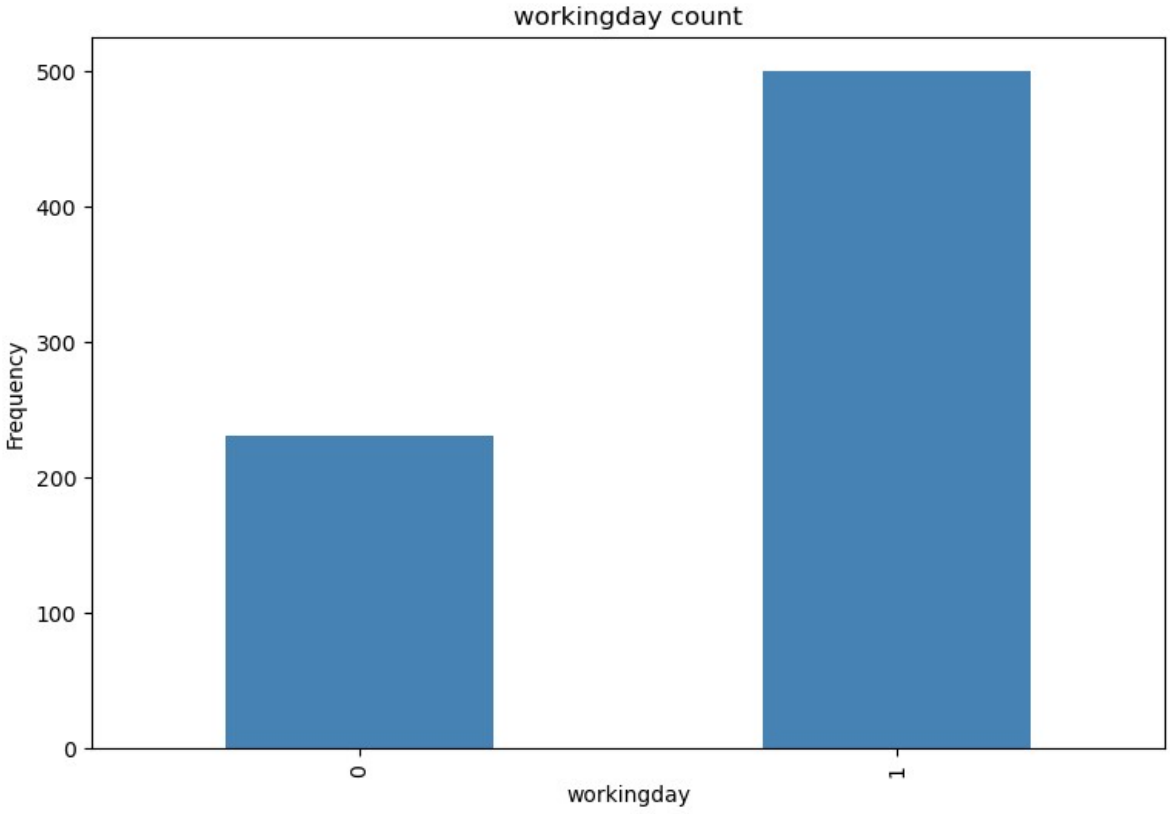
In [10]:
```
categorical_features = ['season', 'mnth', 'holiday', 'weekday', 'workingday',
for col in categorical_features:
    counts = bike[col].value_counts().sort_index()
    fig = plt.figure(figsize=(9,6))
    ax = fig.gca()
    counts.plot.bar(ax=ax, color='steelblue')
    ax.set_title(col + ' count')
    ax.set_xlabel(col)
    ax.set_ylabel('Frequency')
plt.show()
```

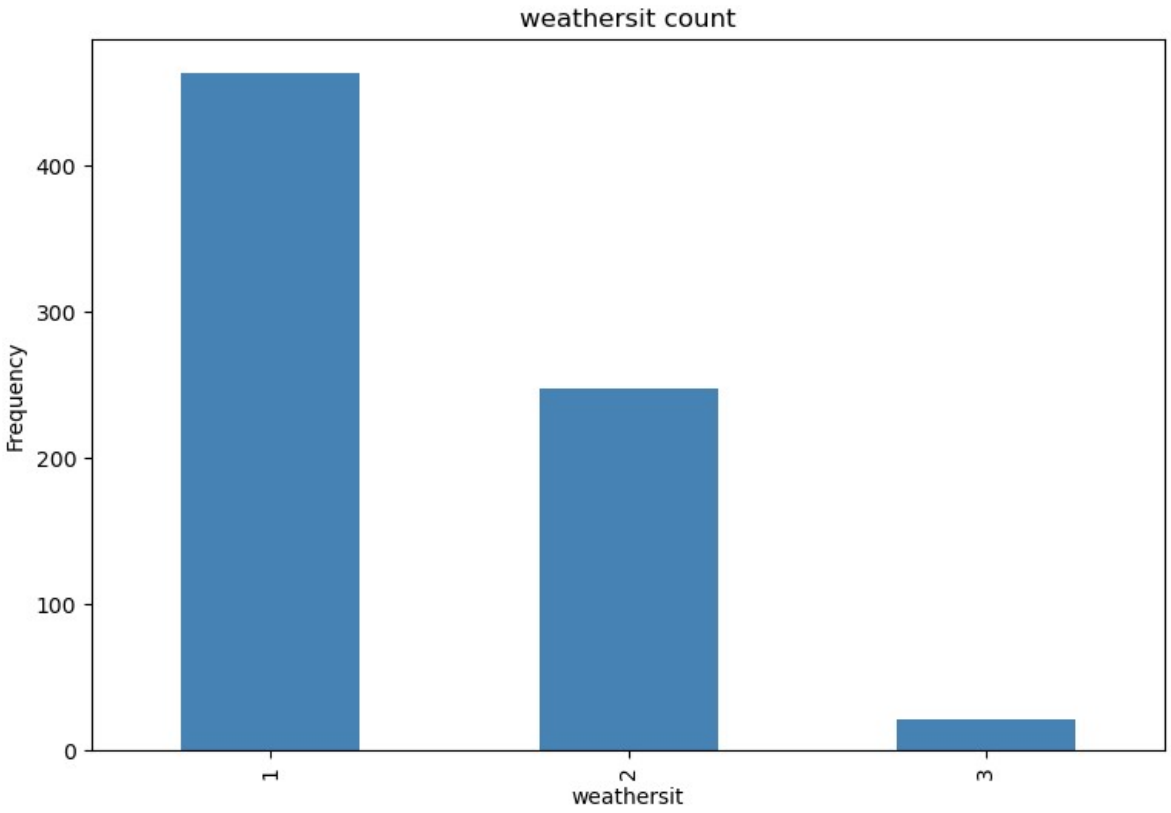


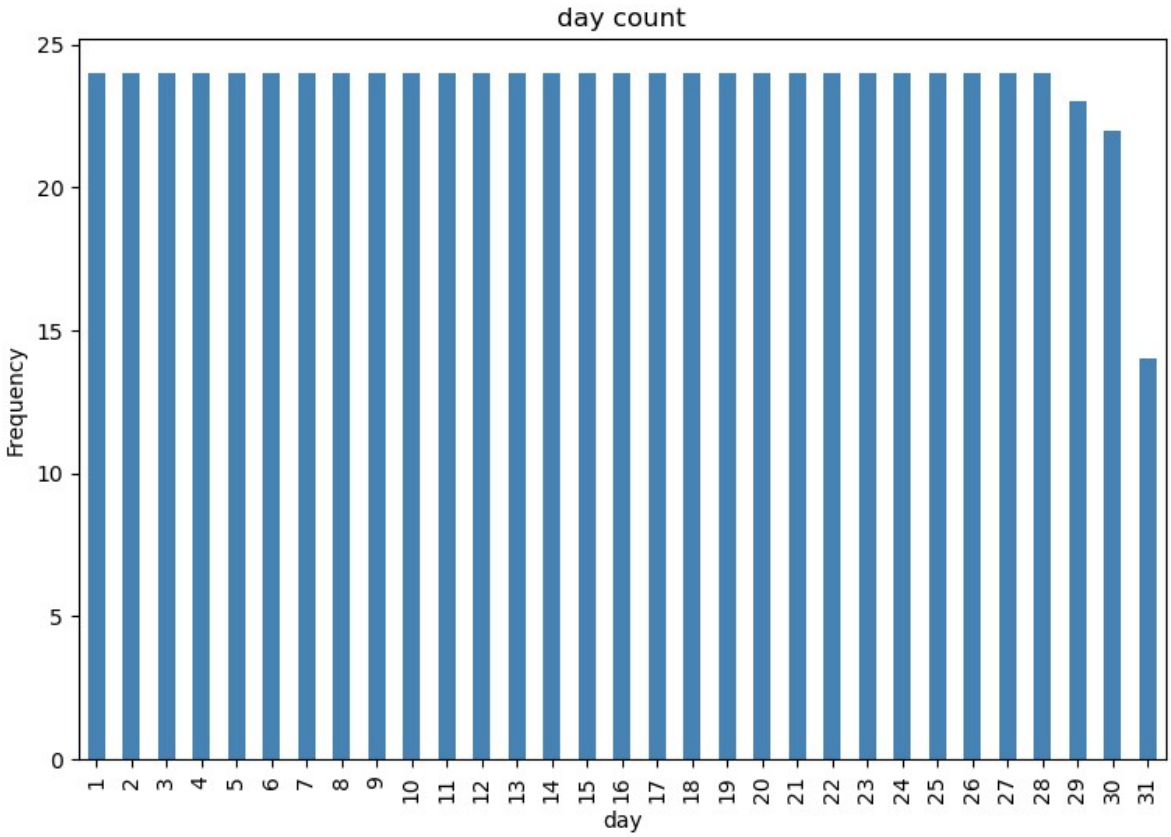

mnth count
Frequency
0
10
20
30
40
50
60
1
2
3
4
5
6
7
8
9
10
11
12
mnth


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

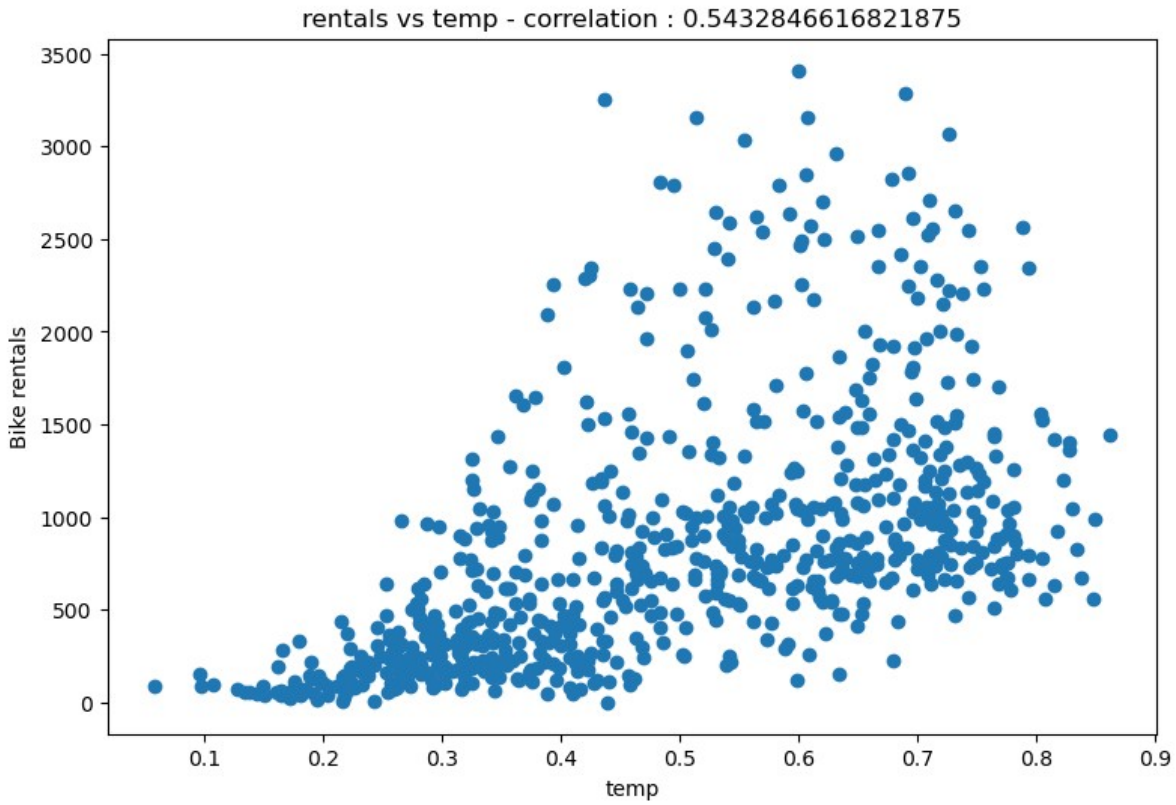

weekday count
Frequency
0
20
40
60
80
100
0
1
2
3
4
5
6
weekday

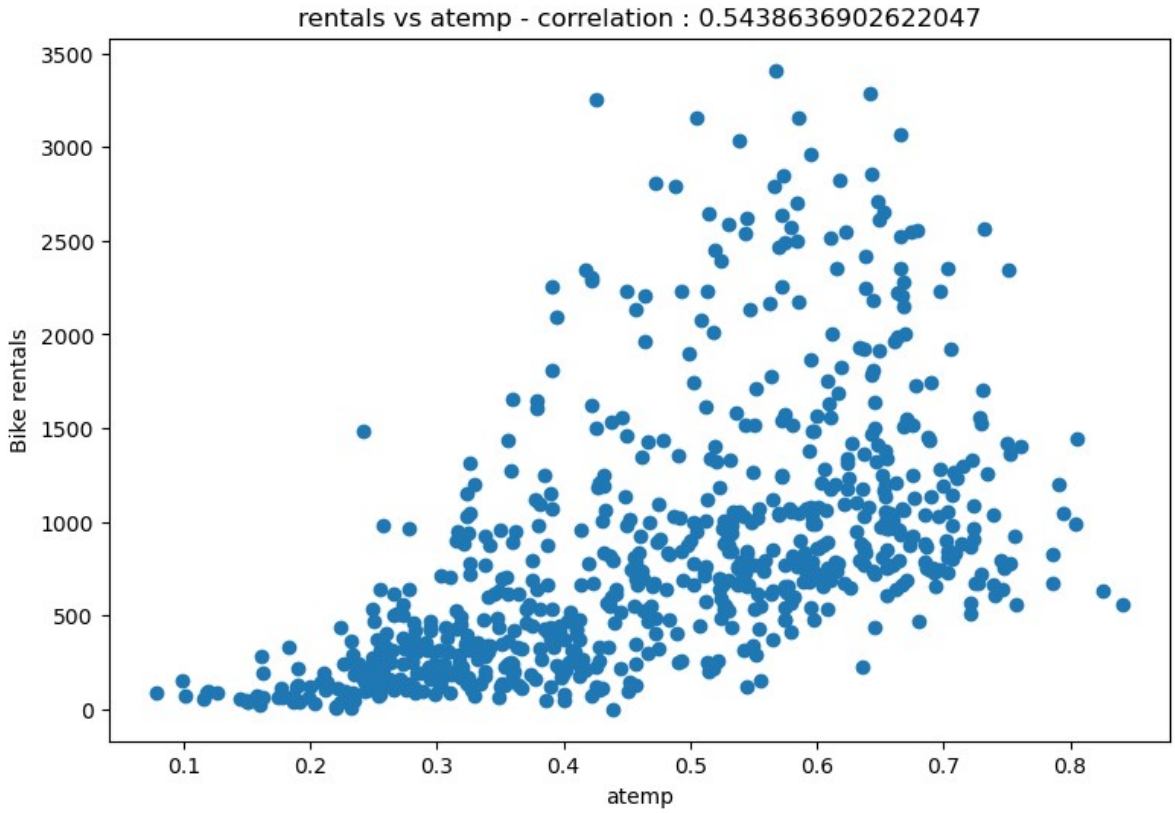

workingday count
Frequency
0
100
200
300
400
500
0
1
workingday

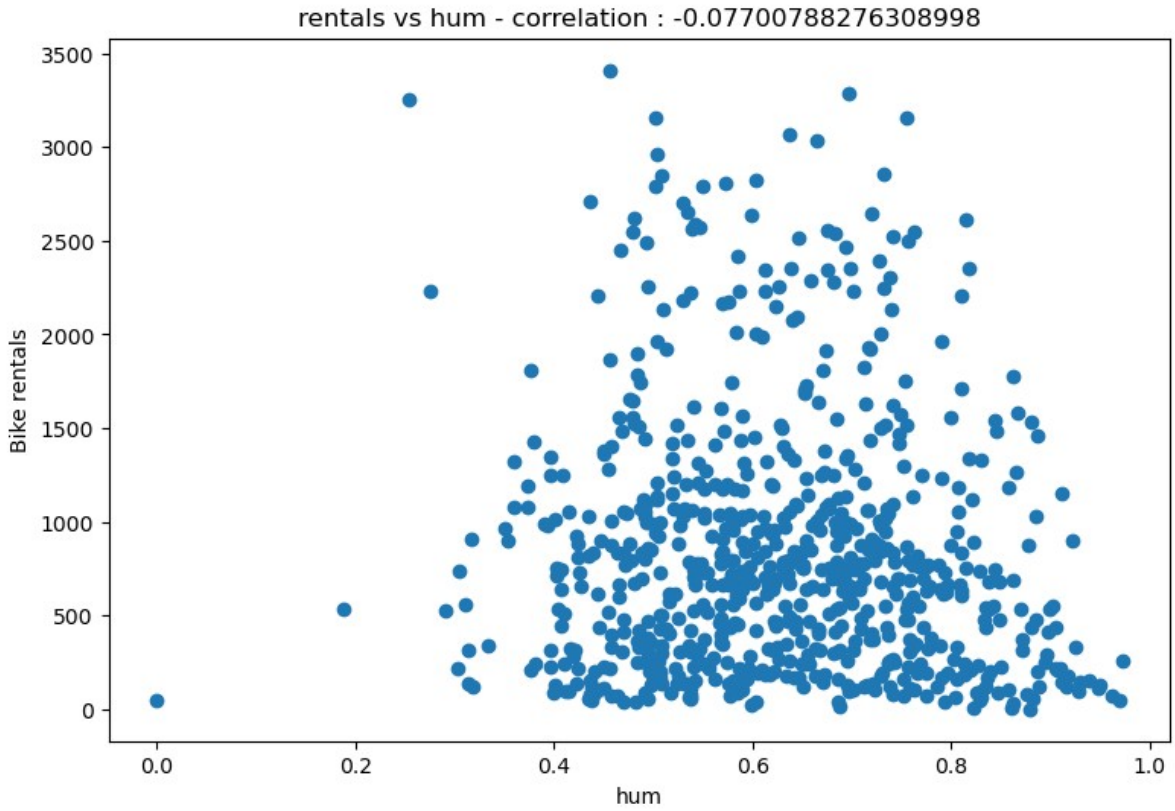

weathersit count
Frequency
0
100
200
300
400
1
2
3
weathersit


day count
Frequency
0
5
10
15
20
25
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
day

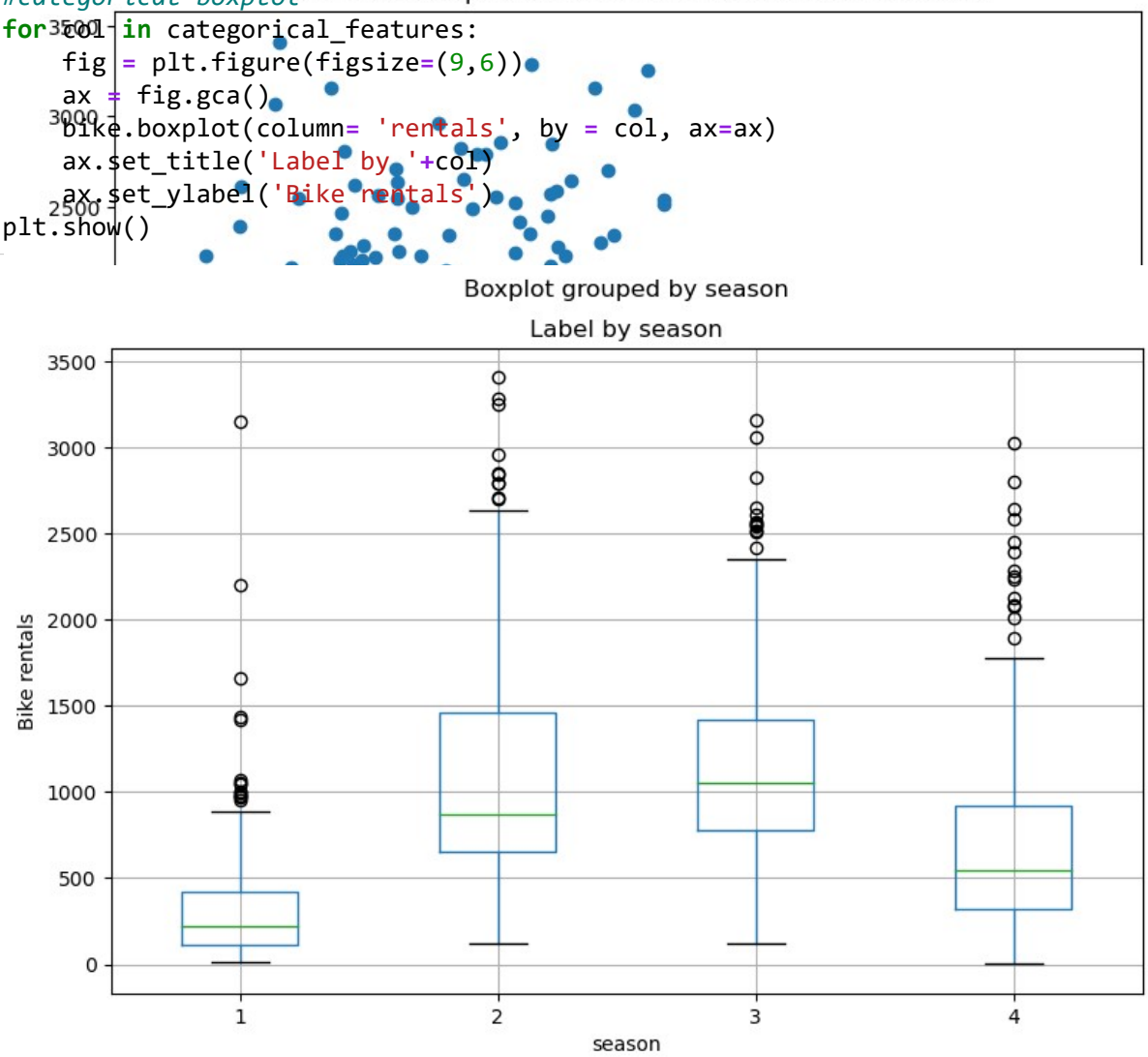
In [11]:
```
for col in numeric_features:
    fig = plt.figure(figsize=(9,6))
    ax = fig.gca()
    feature = bike[col]
    label = bike['rentals']
    correlation =feature.corr(label)
    plt.scatter(x=feature, y=label)
    plt.xlabel(col)
    plt.ylabel('Bike rentals')
    ax.set_title(f"rentals vs {col} - correlation : {correlation}" )
plt.show()
```

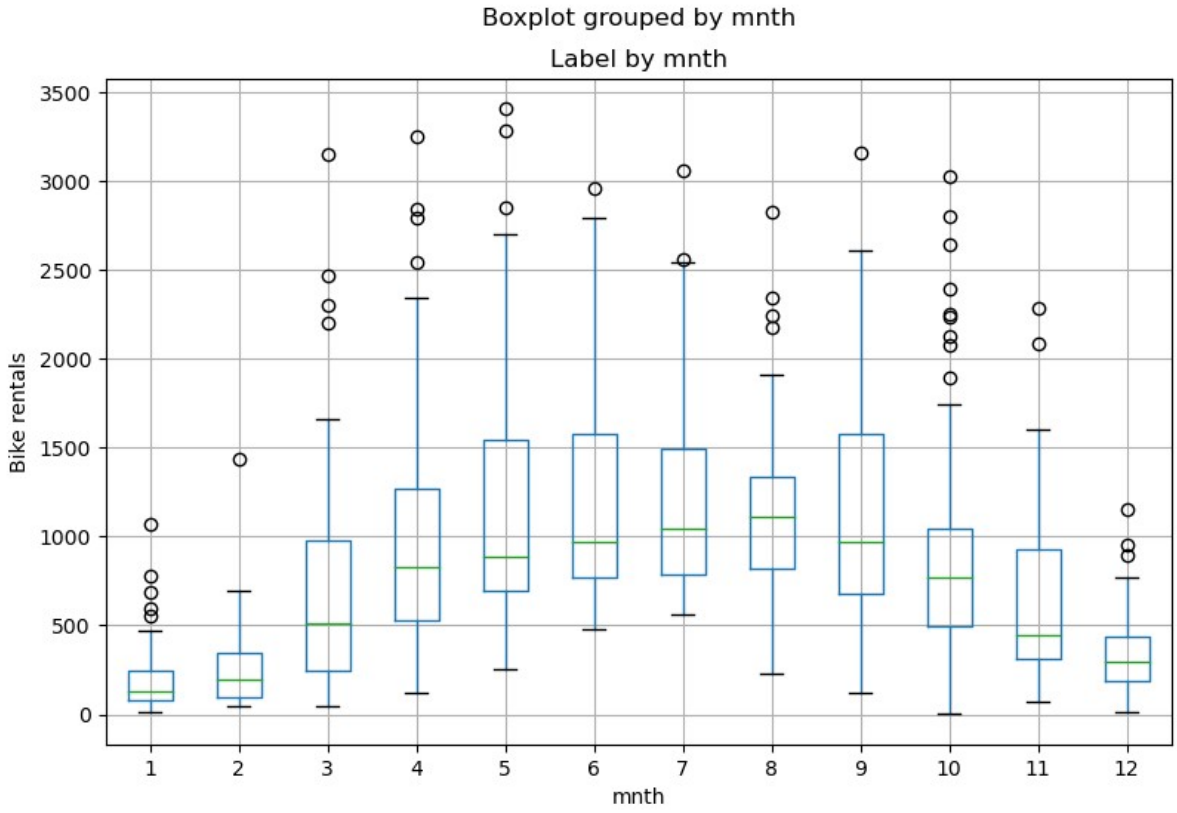


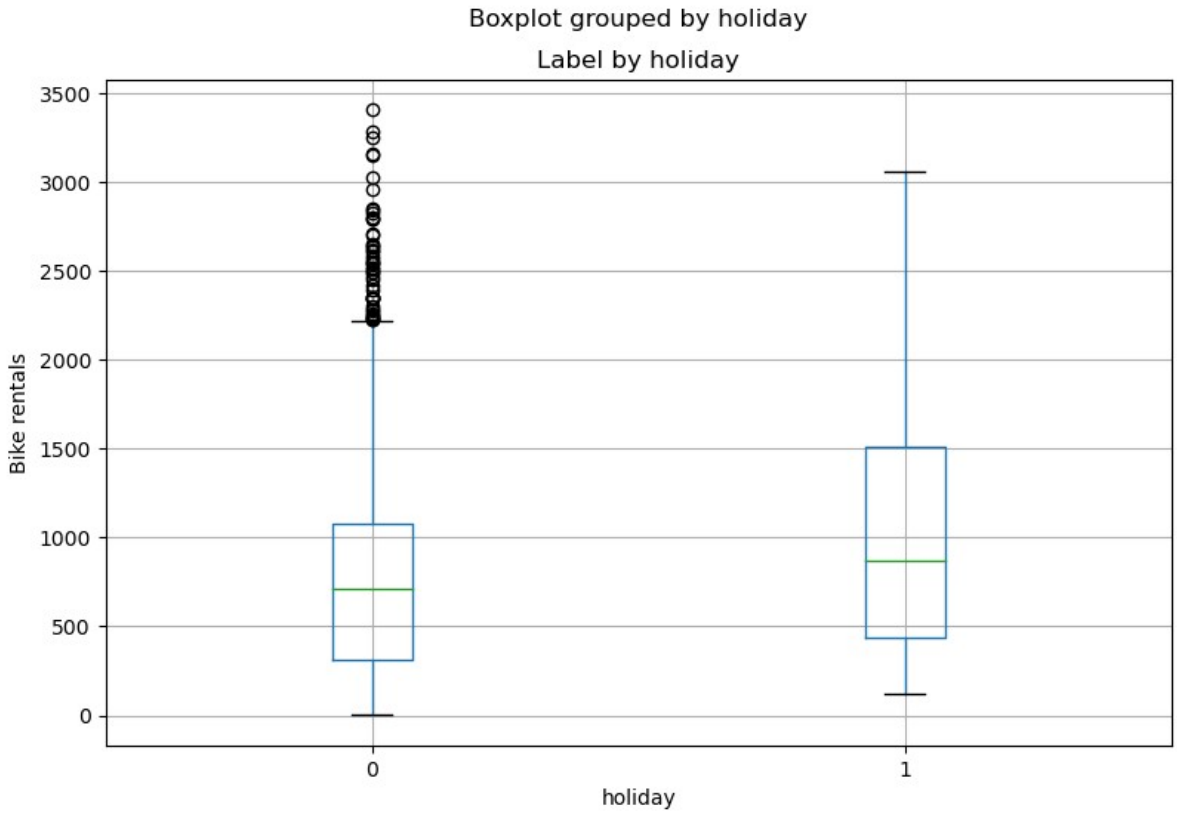

rentals vs atemp - correlation : 0.5438636902622047
Bike rentals
atemp
0
500
1000
1500
2000
2500
3000
3500
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

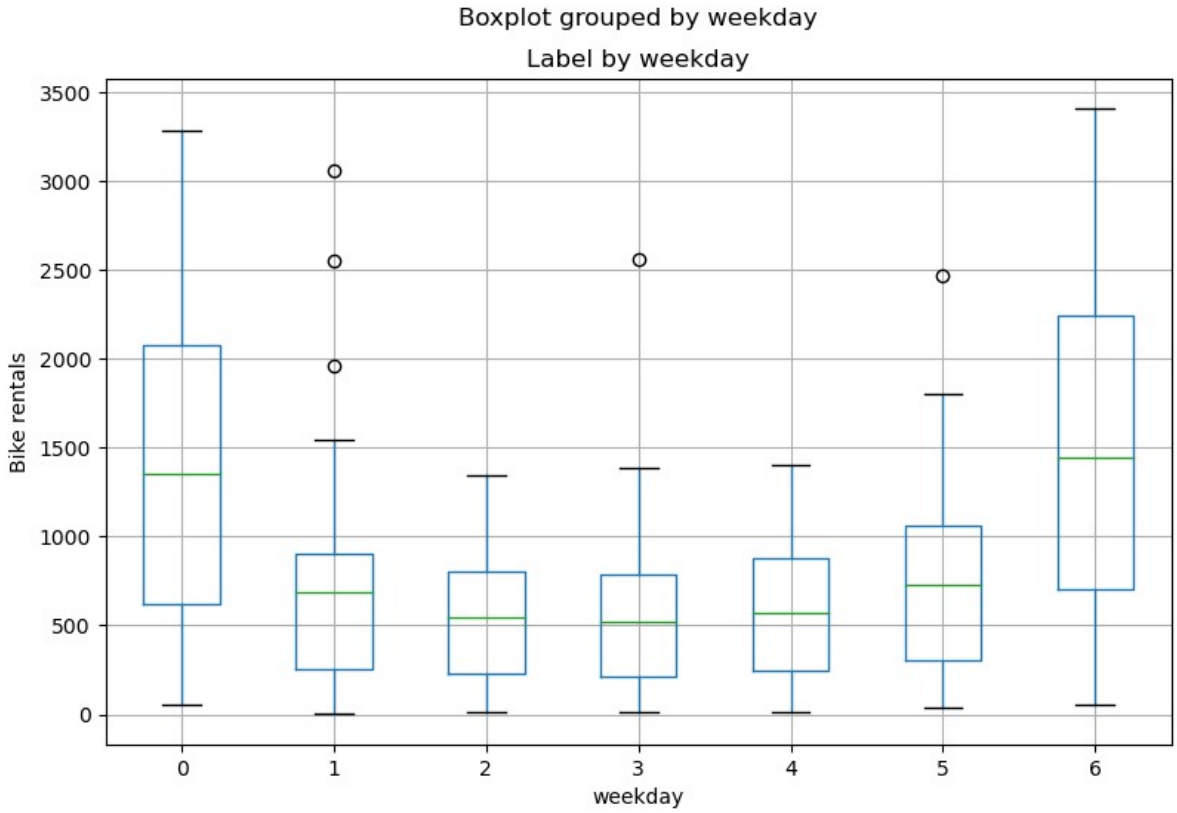

rentals vs hum - correlation : -0.07700788276308998
Bike rentals
hum
0
500
1000
1500
2000
2500
3000
3500
0.0
0.2
0.4
0.6
0.8
1.0

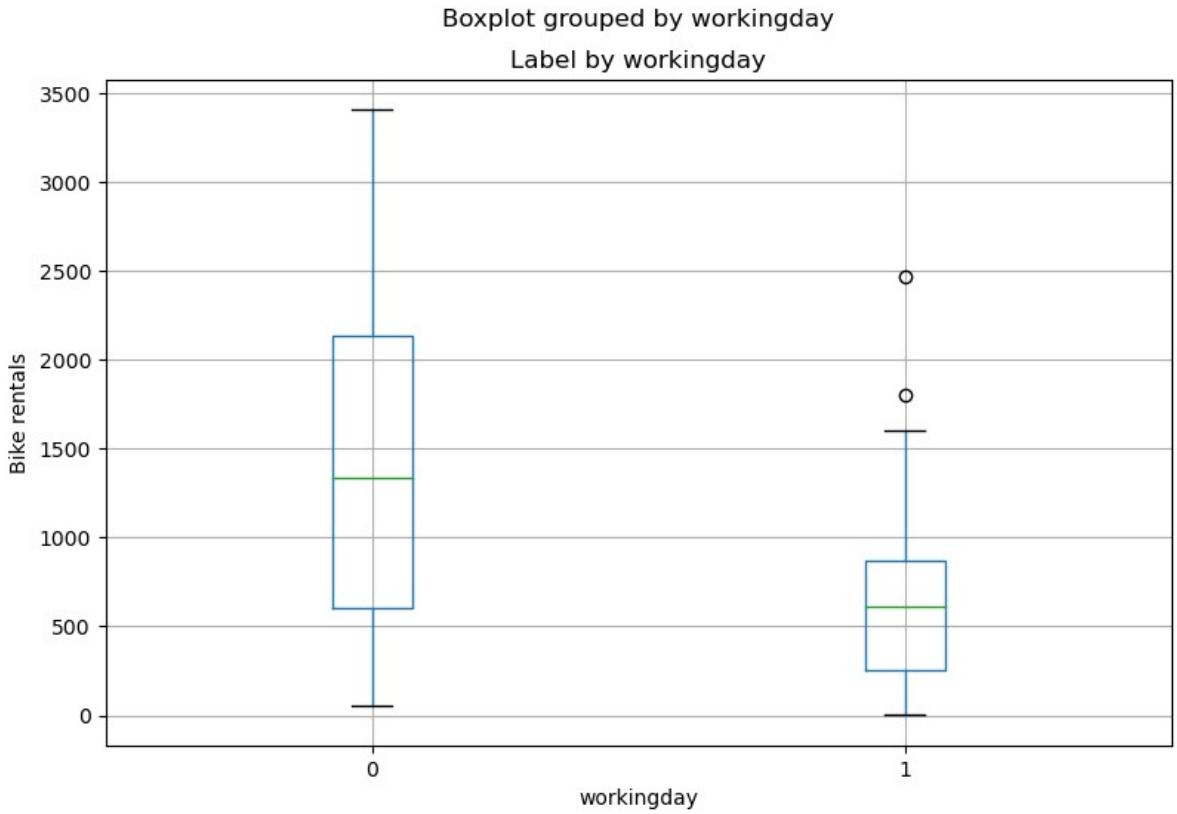
In [12]:
```
#categorical boxplot
for col in categorical_features:
    fig = plt.figure(figsize=(9,6))
    ax = fig.gca()
    bike.boxplot(column= 'rentals', by = col, ax=ax)
    ax.set_title('Label by '+col)
    ax.set_ylabel('Bike rentals')
plt.show()
```

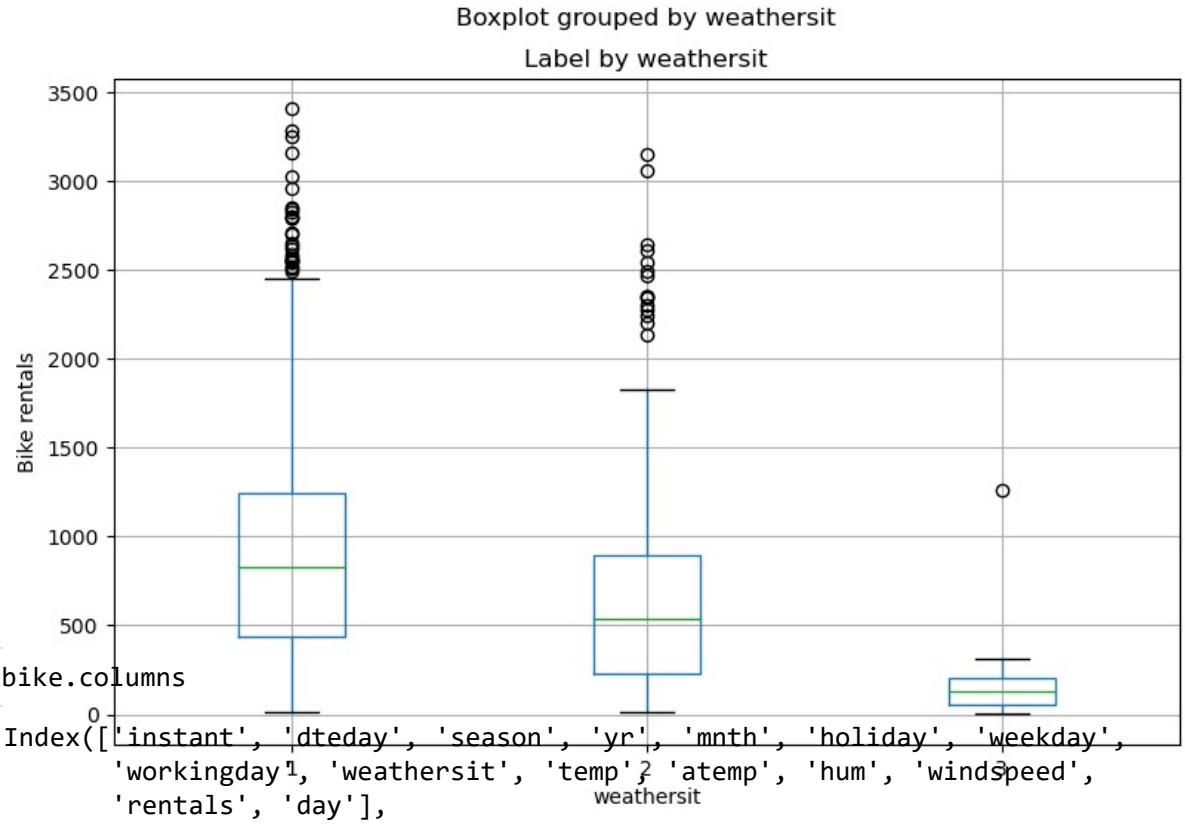



Boxplot grouped by mnth
Label by mnth
Bike rentals
mnth
0
500
1000
1500
2000
2500
3000
3500
1
2
3
4
5
6
7
8
9
10
11
12


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

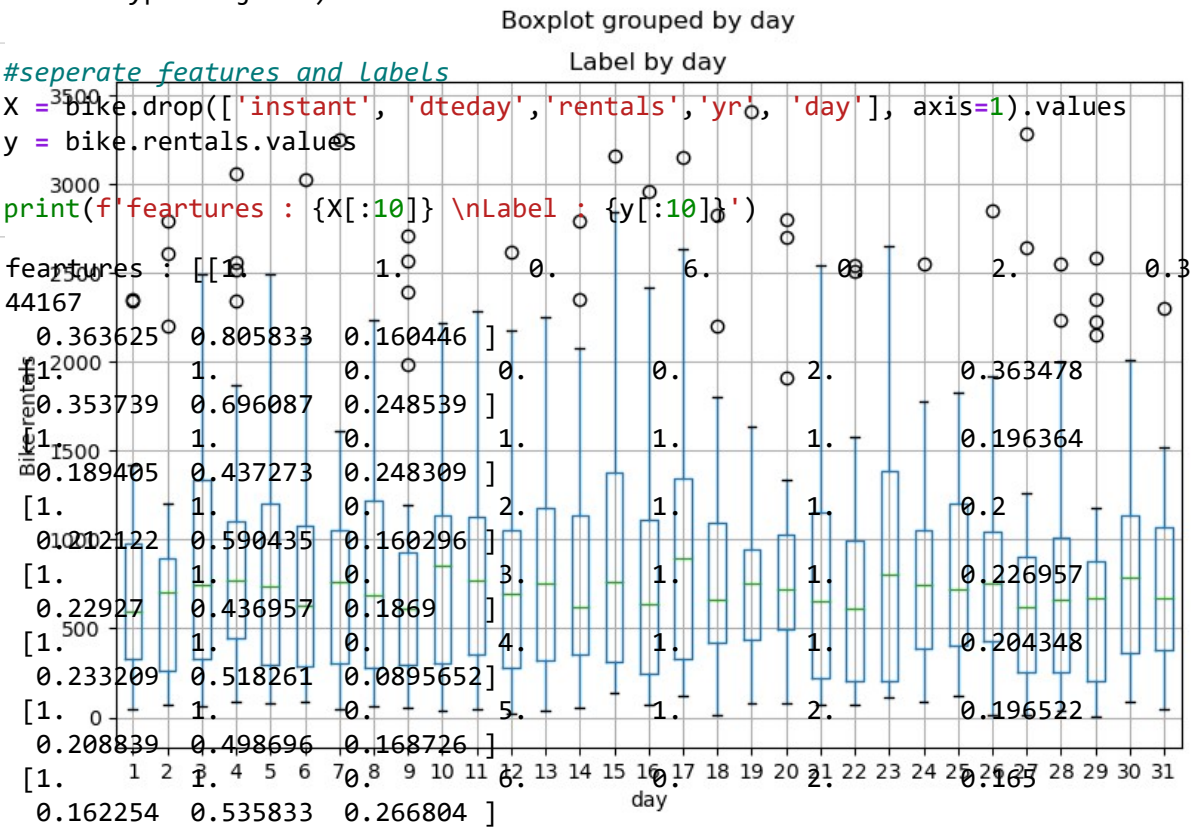

Boxplot grouped by weekday
Label by weekday
Bike rentals
3500
3000
2500
2000
1500
1000
500
0
0
1
2
3
4
5
6
weekday


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



In [13]:
```
bike.columns
```

Out[13]:
```
Index(['instant', 'dteday', 'season', 'yr', 'mnth', 'holiday', 'weekday',
       'workingday', 'weathersit', 'temp', 'atemp', 'hum', 'windspeed',
       'rentals', 'day'],
      dtype='object')
```



In [14]:
```
#seperate features and labels
X = bike.drop(['instant', 'dteday','rentals','yr', 'day'], axis=1).values
y = bike.rentals.values

print(f'feartures : {X[:10]} \nLabel : {y[:10]}')
```

```
feartures : [[1.       1.       0.       6.       0.       2.       0.3
44167
  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 ]]
Label : [331 131 120 108  82  88 148  68  54  41]
```

In [15]:
```
#training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, rand

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

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

In [16]:
```
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[16]:
```
▾ LinearRegression

LinearRegression()
```

Evaluate the Trained Model

In [17]:
```
predictions =model.predict(X_test)
np.set_printoptions(suppress=True)

print(f"Predicted label: {np.round(predictions)[:10]} \nActual labels: {y_test
```

```
Predicted label: [1896. 1184. 1007.  -28.  314.  385.  475.  590. 1476.  -2
2.]
Actual labels: [2418  754  222   47  244  145  240  555 3252   38]
```

In [18]:
```
plt.scatter(y_test, predictions)
plt.xlabel('Actual Labels')
plt.ylabel('Predicted Labels')
plt.title('Daily Bike share Predictins')

#overlay the regz = np.polyfit(y_test, predictions, 1)
p = np.poly1d(z)
plt.plot(y_test, p(y_test), color='magenta')ression line

plt.show()
```

```
  Cell In[18], line 8
    plt.plot(y_test, p(y_test), color='magenta')ression line
                                                ^
SyntaxError: invalid syntax
```

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

mse = mean_squared_error(y_test, predictions)

rmse = np.sqrt(mse)

r2 = r2_score(y_test, predictions)

print(f"MSE : {mse}\nRMSE : {rmse}\nR2 : {r2}")
```

In [ ]:
```

```

KPOVIESSI O. A. Stéphane