DATA VISUALIZATION IN PYTHON FOR ABSOLUTE BEGINNERS

Visualizing Seoul Public Bike Sharing

Vatsal Vinay Parikh

This project covers the basics of how to create an interactive plot using Plotly. We will visualize Seoul bike sharing data using bar plots, scatter plots, and line plots using Plotly as well as DataCamp Workspace's no-code chart cell. In the process, we'll tease out how Seoul weather is impacting bike sharing trends.

Load in required packages

```
import pandas as pd
from datetime import datetime, timedelta
import plotly.express as px
```

Load and clean the data

The dataset consists of the number of public bikes rented in Seoul's bike sharing system at each hour. It also includes information about the weather and the time, such as whether it was a public holiday. Source of dataset.

```
# Import CSV with renamed columns
df = pd.read_csv('data/seoul_bike_data_renamed.csv')
df
```

	d...	n_rente...		temperature_cel...	humi...	wind_sp...	visibility...	dew_
0	01/12/2017	254	0	-5.2	37	2.2	2000	
1	01/12/2017	204	1	-5.5	38	0.8	2000	
2	01/12/2017	173	2	-6	39	1	2000	
3	01/12/2017	107	3	-6.2	40	0.9	2000	
4	01/12/2017	78	4	-6	36	2.3	2000	
5	01/12/2017	100	5	-6.4	37	1.5	2000	
6	01/12/2017	181	6	-6.6	35	1.3	2000	
7	01/12/2017	460	7	-7.4	38	0.9	2000	
8	01/12/2017	930	8	-7.6	37	1.1	2000	
9	01/12/2017	490	9	-6.5	27	0.5	1928	
10	01/12/2017	339	10	-3.5	24	1.2	1996	
11	01/12/2017	360	11	-0.5	21	1.3	1936	
12	01/12/2017	449	12	1.7	23	1.4	2000	
13	01/12/2017	451	13	2.4	25	1.6	2000	
14	01/12/2017	447	14	3	26	2	2000	
15	01/12/2017	463	15	2.1	36	3.2	2000	

Rows: 7,142 ⚠ truncated from 8,760 rows

```
# Clean up some columns
df["date"] = pd.to_datetime(df["date"], format="%d/%m/%Y")
df["datetime"] = df.apply(
    lambda row: row["date"] + timedelta(hours=row["hour"]), axis=1
)
df["is_holiday"] = df["is_holiday"].map({"No Holiday": False, "Holiday": True})

# Print out the result
df
```

	date	n_rente...		temperature_cel...	humi...	wind_sp...	visibility
0	2017-12-01T00:00:00.000	254	0	-5.2	37	2.2	
1	2017-12-01T00:00:00.000	204	1	-5.5	38	0.8	
2	2017-12-01T00:00:00.000	173	2	-6	39	1	
3	2017-12-01T00:00:00.000	107	3	-6.2	40	0.9	
4	2017-12-01T00:00:00.000	78	4	-6	36	2.3	
5	2017-12-01T00:00:00.000	100	5	-6.4	37	1.5	
6	2017-12-01T00:00:00.000	181	6	-6.6	35	1.3	
7	2017-12-01T00:00:00.000	460	7	-7.4	38	0.9	
8	2017-12-01T00:00:00.000	930	8	-7.6	37	1.1	
9	2017-12-01T00:00:00.000	490	9	-6.5	27	0.5	
10	2017-12-01T00:00:00.000	339	10	-3.5	24	1.2	
11	2017-12-01T00:00:00.000	360	11	-0.5	21	1.3	
12	2017-12-01T00:00:00.000	449	12	1.7	23	1.4	
13	2017-12-01T00:00:00.000	451	13	2.4	25	1.6	
14	2017-12-01T00:00:00.000	447	14	3	26	2	
15	2017-12-01T00:00:00.000	463	15	2.1	36	3.2	

Rows: 6,666 ⚠ truncated from 8,760 rows

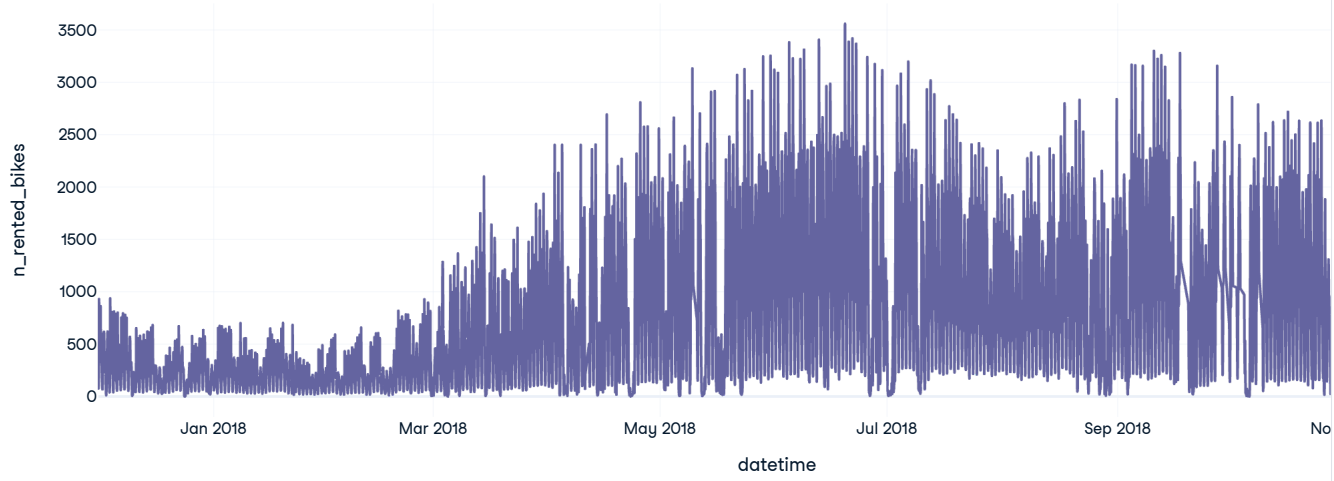
```
# Similar to is_holiday, map is_functioning to True and False
df["is_functioning"] = df["is_functioning"].map({"No": False, "Yes": True})

# Only keep observations where the system is functioning
df = df.query('is_functioning')
df
```

	date	n_rente...		temperature_cel...	humi...	wind_sp...	visibility
0	2017-12-01T00:00:00.000	254	0	-5.2	37	2.2	
1	2017-12-01T00:00:00.000	204	1	-5.5	38	0.8	
2	2017-12-01T00:00:00.000	173	2	-6	39	1	
3	2017-12-01T00:00:00.000	107	3	-6.2	40	0.9	
4	2017-12-01T00:00:00.000	78	4	-6	36	2.3	
5	2017-12-01T00:00:00.000	100	5	-6.4	37	1.5	
6	2017-12-01T00:00:00.000	181	6	-6.6	35	1.3	
7	2017-12-01T00:00:00.000	460	7	-7.4	38	0.9	
8	2017-12-01T00:00:00.000	930	8	-7.6	37	1.1	
9	2017-12-01T00:00:00.000	490	9	-6.5	27	0.5	
10	2017-12-01T00:00:00.000	339	10	-3.5	24	1.2	
11	2017-12-01T00:00:00.000	360	11	-0.5	21	1.3	
12	2017-12-01T00:00:00.000	449	12	1.7	23	1.4	
13	2017-12-01T00:00:00.000	451	13	2.4	25	1.6	
14	2017-12-01T00:00:00.000	447	14	3	26	2	
15	2017-12-01T00:00:00.000	463	15	2.1	36	3.2	

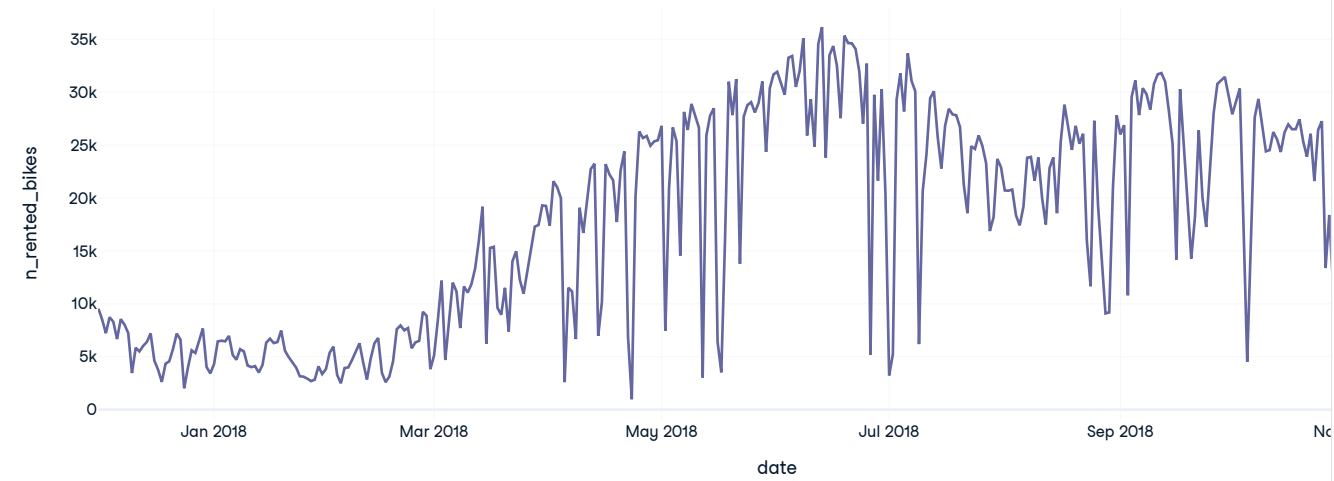
Rows: 6,666 ⚠ truncated from 8,465 rows

Visualize bike rentals over time

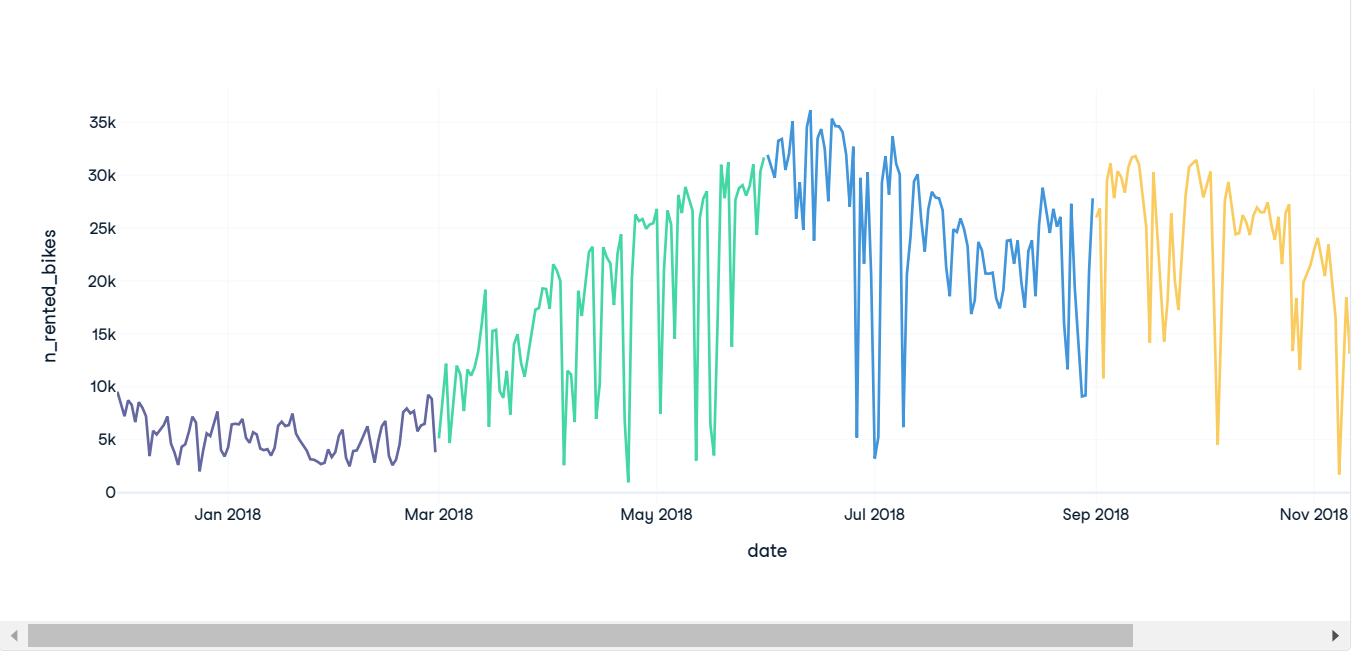
```
# Create a line plot of rented bikes over time
px.line(df, x = 'datetime', y = 'n_rented_bikes')
```




```
# Calculate the total number of rented bikes per day
by_day = df \
    .groupby(by="date", as_index=False) \
    .sum("n_rented_bikes") \
    [["date", "n_rented_bikes"]]

# Create a line plot showing total number of bikes per day over time
px.line(by_day, x = 'date', y = 'n_rented_bikes')
```

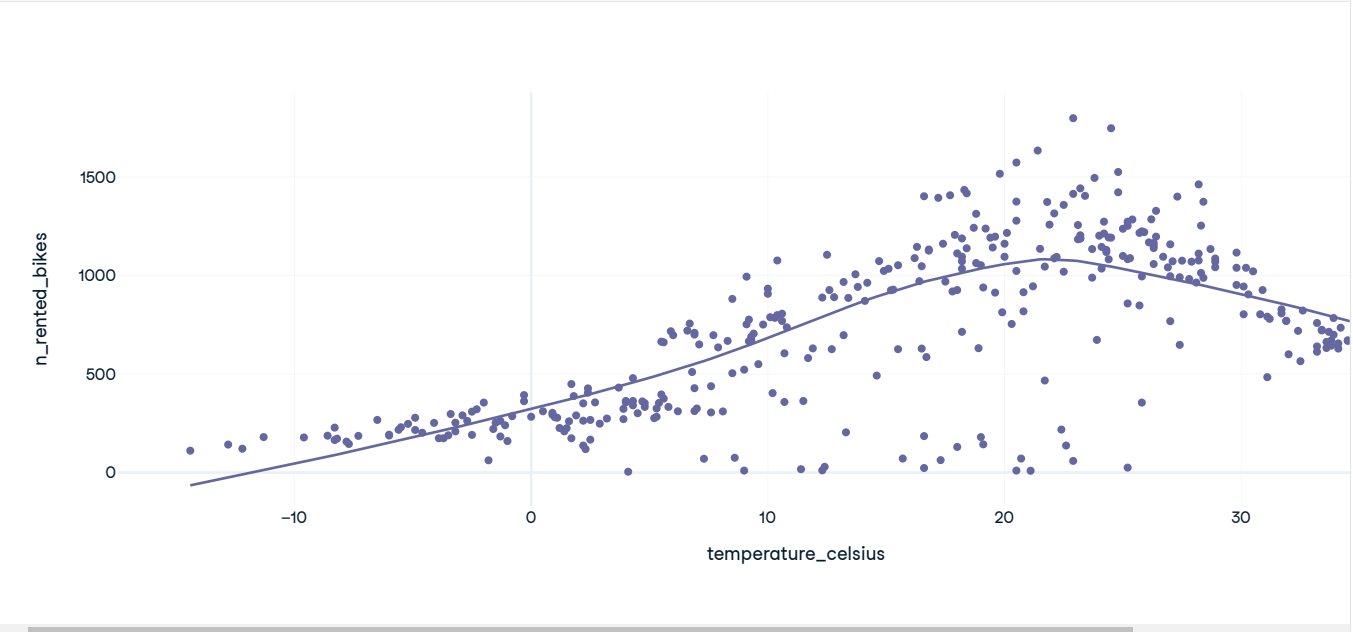


```
# Copy the previous chain of manipulations and add season as a variable to group by
by_season = df \
    .groupby(by=["date","season"], as_index=False) \
    .sum("n_rented_bikes") \
    [["date", "season","n_rented_bikes"]]

# Copy the code for the previous line plot and map season to color
px.line(by_season, x = 'date', y = 'n_rented_bikes', color = "season")
```

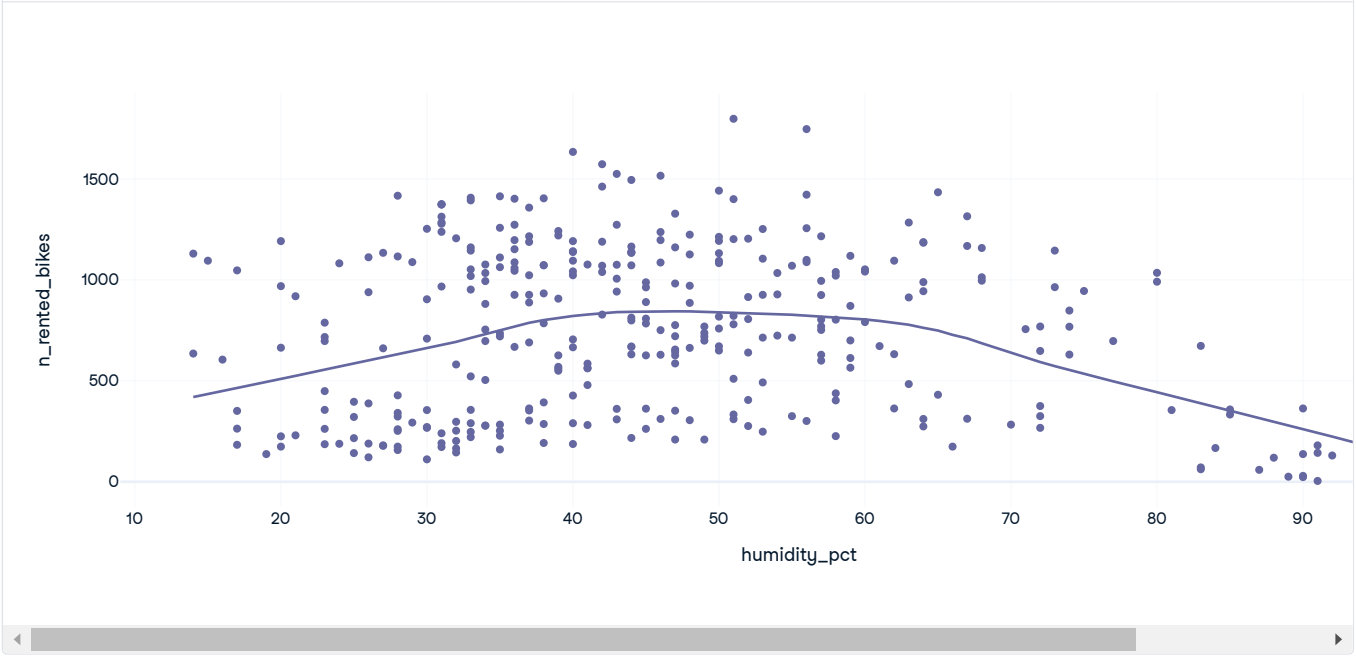


Explore the relation between weather and rentals

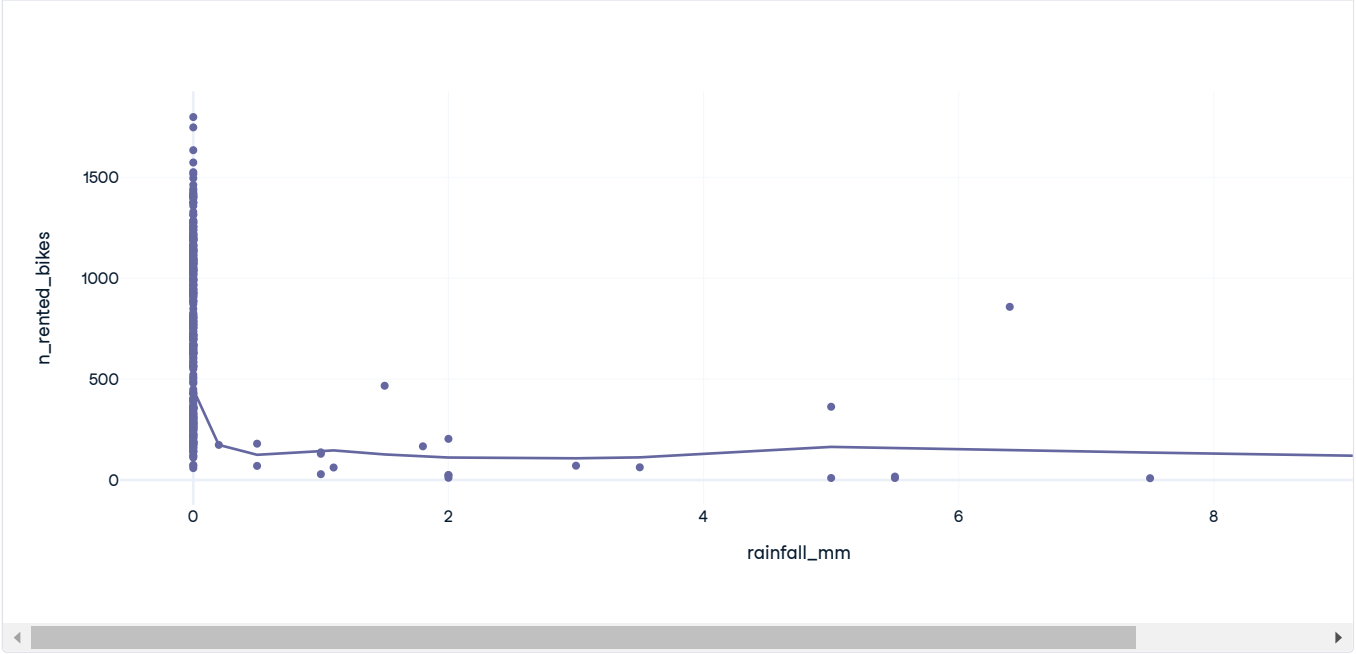
```
# Query df to only keep observations at noon
noon_rides = df.query('hour == 12')

# Create a scatter plot showing temperature against number of rented bikes
# Add a trendline if you feel like it
px.scatter(noon_rides, x = 'temperature_celsius', y = 'n_rented_bikes', trendline = 'lowess')
```



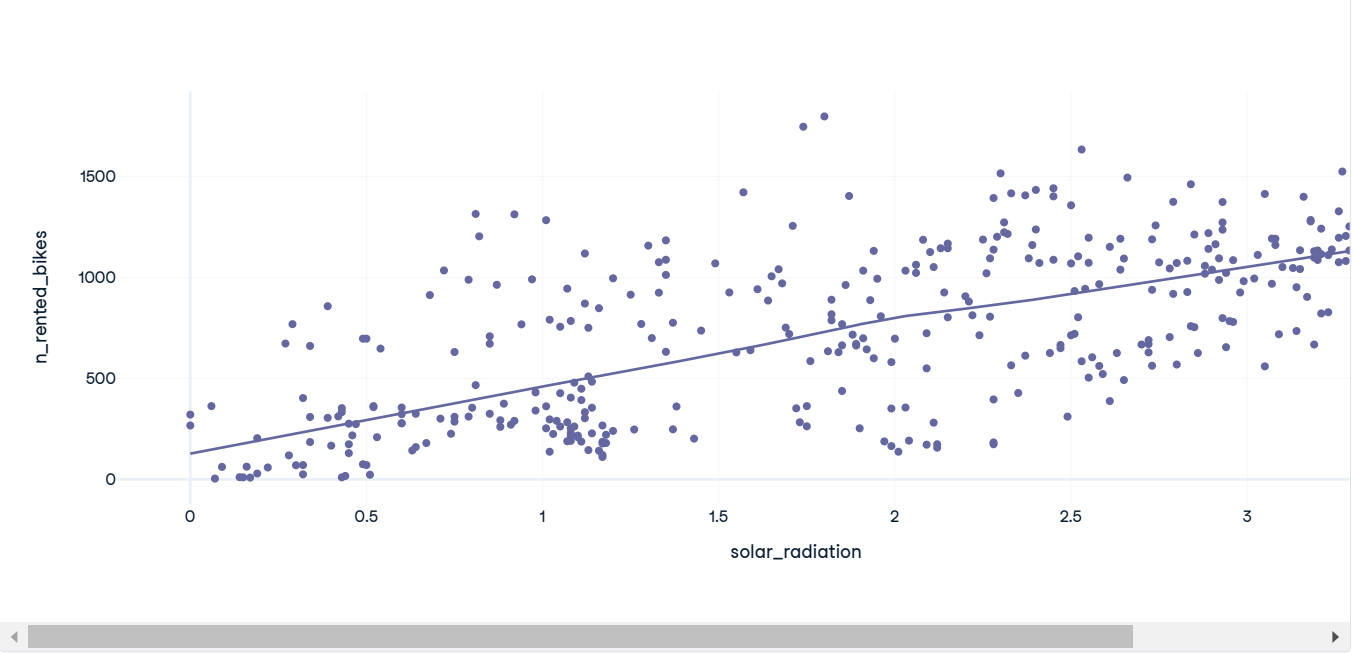
```
# Copy and update the code for the previous scatter plot
# to investigate relation with other weather parameters
px.scatter(noon_rides, x = 'humidity_pct', y = 'n_rented_bikes', trendline = 'lowess')
```




```
px.scatter(noon_rides, x = 'rainfall_mm', y = 'n_rented_bikes', trendline = 'lowess')
```

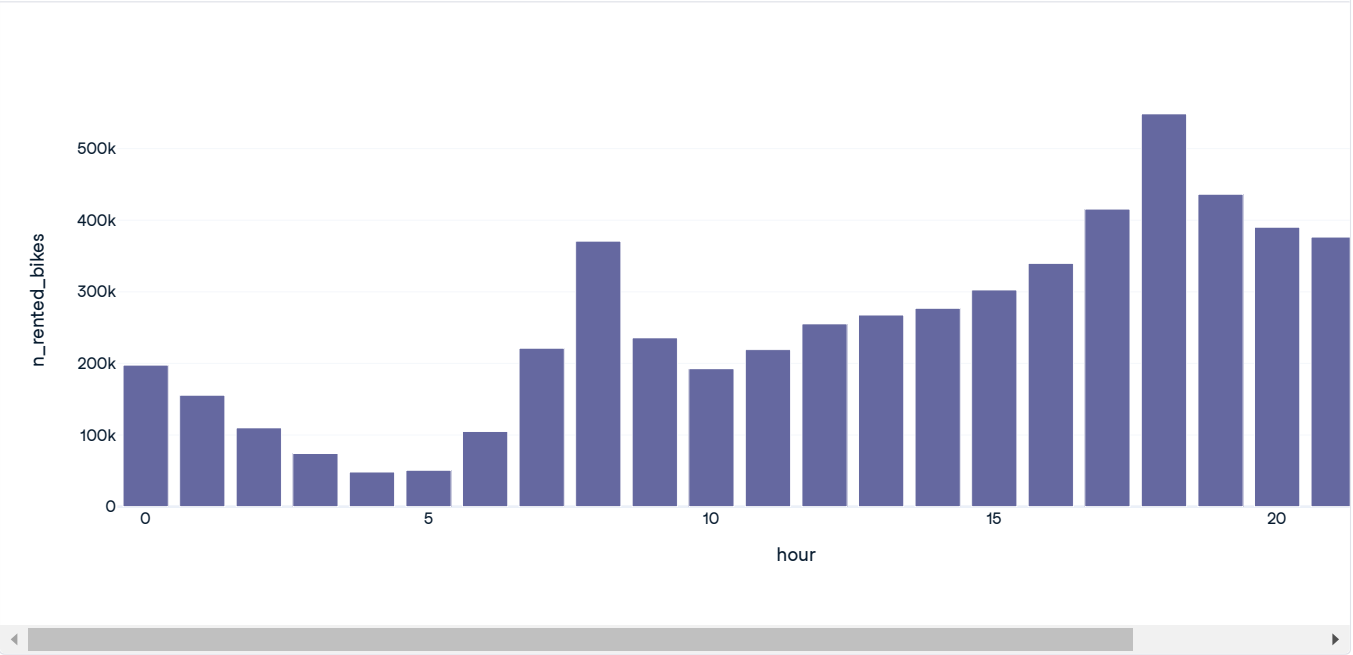


```
px.scatter(noon_rides, x = 'solar_radiation', y = 'n_rented_bikes', trendline = 'lowess')
```

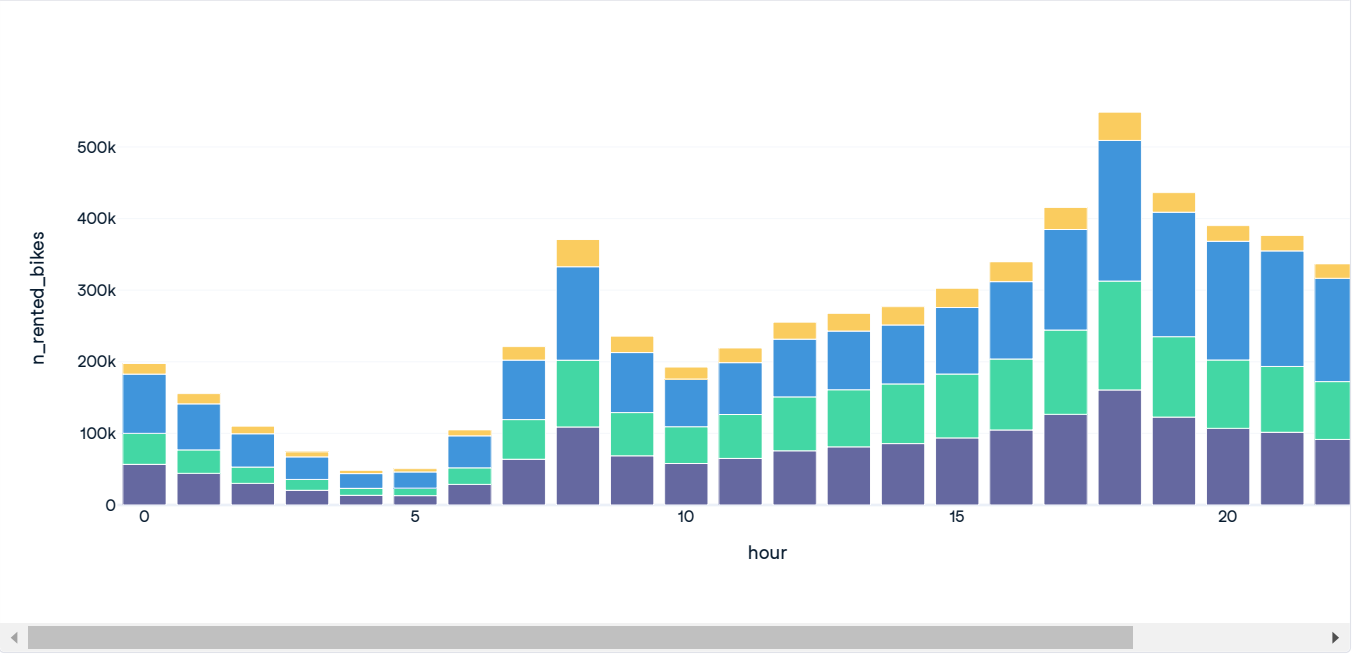


Explore typical daily usage pattern

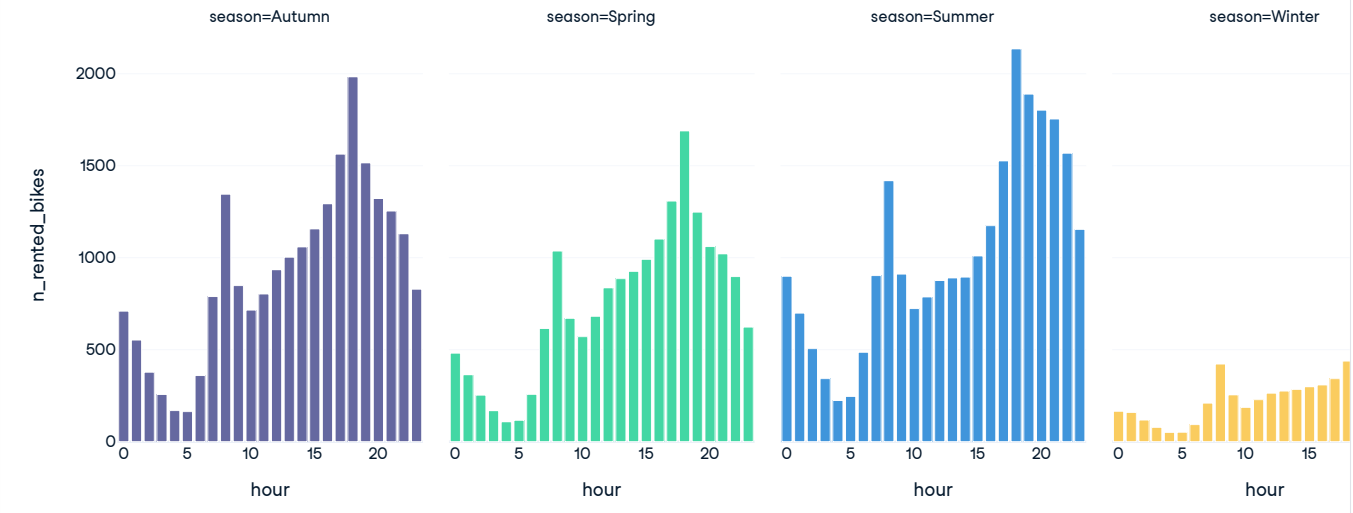
```
# Calculate the average number of rented bikes per hour
time_of_day = df \
    .groupby(by="hour", as_index=False) \
    .sum("n_rented_bikes") \
    [["hour", "n_rented_bikes"]]

# Create a bar chart showing the usage pattern
px.bar(time_of_day, x = 'hour', y = 'n_rented_bikes')
```



```
# Copy and adapt the previous query to take into account the season
time_of_day_season = df \
    .groupby(by=["hour" , "season"], as_index=False) \
    .sum("n_rented_bikes") \
    [["hour", "season", "n_rented_bikes"]]

# Copy and adapt the code for the previous bar chart to show usage pattern per season
px.bar(time_of_day_season, x = 'hour', y = 'n_rented_bikes', color= "season" )
```




```
time_of_day_season = df \
    .groupby(by=["hour" , "season"], as_index=False) \
    .mean("n_rented_bikes") \
    [["hour", "season", "n_rented_bikes"]]

px.bar(time_of_day_season, x = 'hour', y = 'n_rented_bikes', color= "season", facet_col= "season")
```



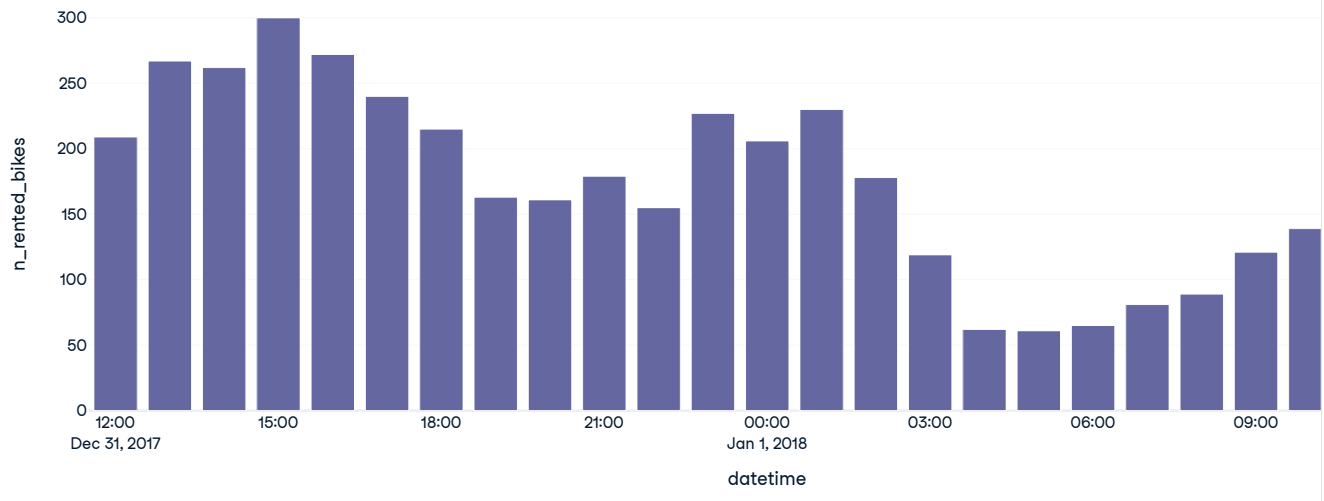
Is New Year's Eve different?

```
# New Years dates
new_years_start = datetime(2017, 12, 31, 12)
new_years_end = datetime(2018, 1, 1, 12)

# Create data frame with new year's data
(df['datetime'] >= new_years_start) & (df['datetime'] <= new_years_end)
```

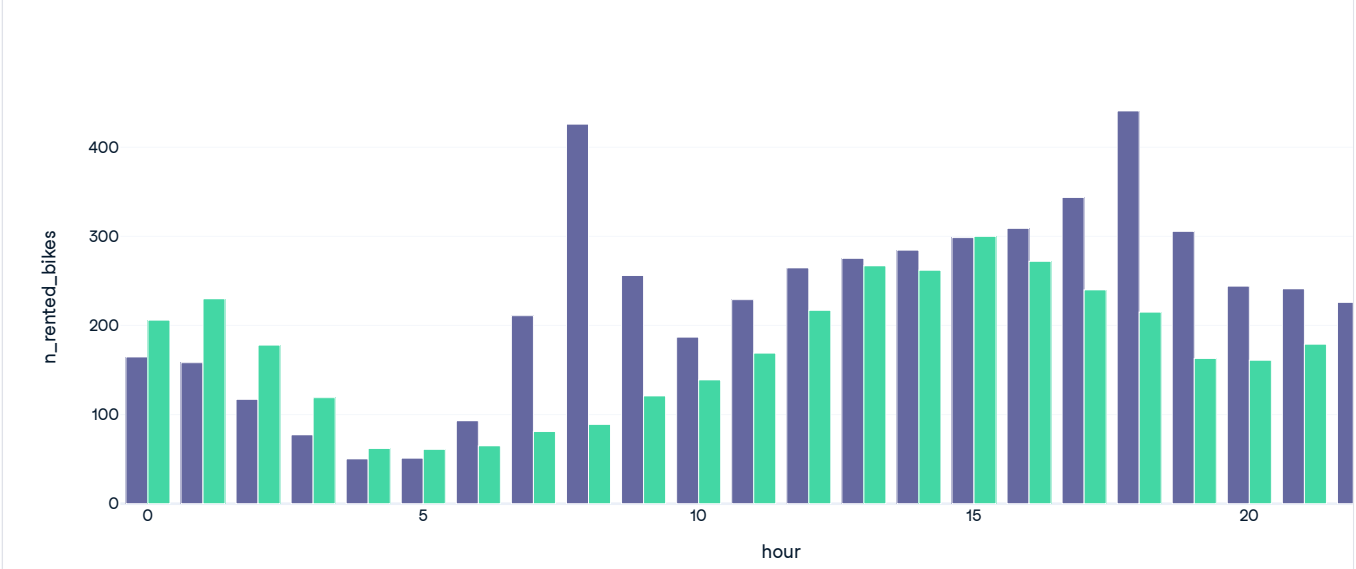
0	False
1	False
2	False
3	False
4	False
5	False
6	False
7	False
8	False
9	False
10	False
11	False
12	False
13	False
14	False
15	False
16	False

Rows: 8,465

```
# Show usage pattern
new_year = df[(df['datetime'] >= new_years_start) & (df['datetime'] <= new_years_end)]
px.bar(new_year, x = 'datetime', y = 'n_rented_bikes')
```



```
# Create a new column indicating whether the rental is on New Year's Eve
df['is_nye'] = (df['datetime'] >= new_years_start) & (df['datetime'] <= new_years_end)

# Create a DataFrame comparing winter usage with New Year's Eve usage
time_of_day = df \
        .query("season == 'Winter'") \
        .groupby(by = ["hour", "is_nye"], as_index = False) \
        .mean("n_rented_bikes") \
        [["hour", "is_nye", "n_rented_bikes"]]

# Build a bar plot that compares New Year's usage with standard winter usage
px.bar(time_of_day, x = 'hour', y = 'n_rented_bikes', color = "is_nye", barmode = 'group')
```




```
px.bar(time_of_day, x = 'hour', y = 'n_rented_bikes', facet_row="is_nye",color = "is_nye")
```