To compare bike rentals on holidays and their adjacent days, we will:

1. **Ensure accurate data filtering**.
2. **Aggregate the rental data for holidays, the day before, and the day after**.
3. **Visualize the comparison**.

Let's start by correctly filtering the data for holidays and their adjacent days, and then aggregate the rental data for comparison.

**Step 1: Filter Data for Holidays and Adjacent Days**

Ensure the filtering logic correctly isolates the required dates.

**Step 2: Aggregate Rental Data**

Calculate total rentals for holidays, the day before, and the day after.

**Step 3: Visualization**

Create a visualization to compare the rentals.

Let's proceed with these steps.

python

Copy code

# Re-filter the data for holidays and adjacent days

holiday\_adjacent\_days\_set = set()

# Collect all relevant days into a set

for holiday in holidays:

day\_before = holiday - pd.Timedelta(days=1)

day\_after = holiday + pd.Timedelta(days=1)

holiday\_adjacent\_days\_set.update([day\_before, holiday, day\_after])

# Filter the dataframe for these days

filtered\_data = data[data['Date\_only'].isin(holiday\_adjacent\_days\_set)]

filtered\_data = filtered\_data[['Date', 'Rented Bike Count', 'Date\_only', 'Holiday']]

# Aggregate rental data by Date\_only and Holiday

aggregated\_data = filtered\_data.groupby(['Date\_only', 'Holiday']).agg({'Rented Bike Count': 'sum'}).reset\_index()

# Display the aggregated data

aggregated\_data.head()

Let's execute this to get the aggregated rental data.

Analyzed

python

Always show details

Copy code

# Aggregate rental data by Date\_only and Holiday

aggregated\_data = filtered\_data.groupby(['Date\_only', 'Holiday']).agg({'Rented Bike Count': 'sum'}).reset\_index()

# Display the aggregated data

import ace\_tools as tools; tools.display\_dataframe\_to\_user(name="Aggregated Holiday and Adjacent Days Rental Data", dataframe=aggregated\_data)

aggregated\_data.head()

Result

Date\_only Holiday Rented Bike Count

0 2017-12-21 No Holiday 5734

1 2017-12-22 Holiday 7184

2 2017-12-23 No Holiday 6624

3 2017-12-24 Holiday 2014

4 2017-12-25 Holiday 3966

Aggregated Holiday and Adjacent Days Rental Data

| **Date\_only** | **Holiday** | **Rented Bike Count** |
| --- | --- | --- |
| 2018-09-11 | No Holiday | 0 |
| 2018-09-22 | No Holiday | 26398 |
| 2018-09-23 | Holiday | 20060 |
| 2018-09-24 | Holiday | 17259 |
| 2018-09-25 | Holiday | 23350 |
| 2018-09-26 | No Holiday | 28018 |

The aggregated data for holiday and adjacent days has been successfully filtered and displayed. The next step is to create a visualization to compare the rentals on holidays and their adjacent days.

**Visualization of Rentals on Holidays and Adjacent Days**

We'll create a bar chart to visualize the total rentals for:

* Holidays
* Day before the holiday
* Day after the holiday

Let's proceed with creating this visualization.

python

Copy code

import matplotlib.pyplot as plt

# Create a bar chart for the aggregated rental data

plt.figure(figsize=(14, 7))

for holiday in holidays:

day\_before = holiday - pd.Timedelta(days=1)

day\_after = holiday + pd.Timedelta(days=1)

holiday\_rentals = aggregated\_data[aggregated\_data['Date\_only'] == holiday]['Rented Bike Count'].values[0]

day\_before\_rentals = aggregated\_data[aggregated\_data['Date\_only'] == day\_before]['Rented Bike Count'].values[0]

day\_after\_rentals = aggregated\_data[aggregated\_data['Date\_only'] == day\_after]['Rented Bike Count'].values[0]

plt.bar(f'{day\_before}\nDay Before', day\_before\_rentals, color='blue', label='Day Before')

plt.bar(f'{holiday}\nHoliday', holiday\_rentals, color='green', label='Holiday')

plt.bar(f'{day\_after}\nDay After', day\_after\_rentals, color='red', label='Day After')

plt.xlabel('Date')

plt.ylabel('Total Rentals')

plt.title('Bike Rentals on Holidays and Adjacent Days')

plt.xticks(rotation=45)

plt.legend(['Day Before', 'Holiday', 'Day After'])

plt.tight\_layout()

plt.show()

Let's generate the visualization to compare the bike rentals on holidays and their adjacent days. ​​

Analyzed

python

Always show details

Copy code

import matplotlib.pyplot as plt

# Create a bar chart for the aggregated rental data

plt.figure(figsize=(14, 7))

# Collect the rental counts for visualization

dates = []

rental\_counts = []

labels = []

for holiday in holidays:

day\_before = holiday - pd.Timedelta(days=1)

day\_after = holiday + pd.Timedelta(days=1)

# Ensure each day exists in the aggregated data

if holiday in aggregated\_data['Date\_only'].values:

holiday\_rentals = aggregated\_data[aggregated\_data['Date\_only'] == holiday]['Rented Bike Count'].values[0]

dates.append(f'{holiday}\nHoliday')

rental\_counts.append(holiday\_rentals)

labels.append('Holiday')

if day\_before in aggregated\_data['Date\_only'].values:

day\_before\_rentals = aggregated\_data[aggregated\_data['Date\_only'] == day\_before]['Rented Bike Count'].values[0]

dates.append(f'{day\_before}\nDay Before')

rental\_counts.append(day\_before\_rentals)

labels.append('Day Before')

if day\_after in aggregated\_data['Date\_only'].values:

day\_after\_rentals = aggregated\_data[aggregated\_data['Date\_only'] == day\_after]['Rented Bike Count'].values[0]

dates.append(f'{day\_after}\nDay After')

rental\_counts.append(day\_after\_rentals)

labels.append('Day After')

# Plot the rentals

colors = {'Day Before': 'blue', 'Holiday': 'green', 'Day After': 'red'}

plt.bar(dates, rental\_counts, color=[colors[label] for label in labels])

plt.xlabel('Date')

plt.ylabel('Total Rentals')

plt.title('Bike Rentals on Holidays and Adjacent Days')

plt.xticks(rotation=90)

plt.legend(labels=colors.keys())

plt.tight\_layout()

plt.show()

Bike Rentals on Holidays and Adjacent Days

Total Rentals

by

Date

A screenshot of a graph

Description automatically generated

The bar chart above visualizes the total bike rentals on holidays and their adjacent days (day before and day after the holiday). The colors represent:

* **Blue**: Day Before
* **Green**: Holiday
* **Red**: Day After

**Summary**

* **Holidays** tend to have varying impacts on bike rentals, with some holidays showing a significant decrease in rentals compared to adjacent days.
* **Day Before and Day After**: Generally, these days can show different patterns, often with higher or similar rentals compared to the holiday itself.

This visualization provides a clear comparison of bike rentals around holidays, highlighting trends and patterns that can inform decisions and strategies for the transportation department.

If you need further analysis, additional visualizations, or more specific insights, please let me know! ​