```
# Complete the following cell with your details and run to produce your personalised header for this assignment

from IPython.display import HTML

first_name = 'Chayanich'
last_name = 'Apaipakdee'
student_number = 'N12087289'

personal_header = f"<h1>{first_name} {last_name}</h1>"
HTML(personal_header)
```

Chayanich Apaipakdee

Data for both Part A and Part B

This assignment uses data from the Queensland Government Open Data Portal. Both parts will use data on Queensland Wave Monitoring. Part B will also use data on Storm tide monitoring You should familiarise yourself with the information on this site to understand the context for the data.

For this assignment, you will use the Coastal Data System - Near real time wave data and for Part B you will add Coastal Data System – Near real time storm tide data. Note that this data will change over the time period of the assignment.

Part A

QUESTION:

What can we learn from the wave height data for South East Queensland, and how might this data be used strategically during a major weather event?

[Q1] Read the data

- Open the CSV version of the file. Open directly from the URL into a pandas dataframe.
- Make a note of your first access of the data
- Identify an appropriate index, and make a note of the columns.

```
# Your answer here

# 1.1 Open the CSV version of the file. Open directly from the URL into a pandas dataframe.
import pandas as pd

# Open a CSV file from URL into a new dataframe
NRT_wave_data = "https://www.data.qld.gov.au/datastore/dump/2bbef99e-9974-49b9-a316-57402b00609c?bom=True"
pd.read_csv(NRT_wave_data)
```

	_id	Site	SiteNumber	Seconds	DateTime	Latitude	Longitude	Hsig	Hmax	Tp	Tz	SST	Direction	Current Speed	Current Direction
0	1	Caloundra	54	1754661600	2025-08-09T00:00:00	-26.84620	153.15585	1.913	3.13	10.00	5.556	20.00	91.40	-99.9	-99.9
1	2	Caloundra	54	1754663400	2025-08-09T00:30:00	-26.84620	153.15589	1.795	2.57	11.76	5.882	19.80	88.60	-99.9	-99.9
2	3	Caloundra	54	1754665200	2025-08-09T01:00:00	-26.84619	153.15587	1.886	3.33	11.76	5.714	19.90	94.20	-99.9	-99.9
3	4	Caloundra	54	1754667000	2025-08-09T01:30:00	-26.84619	153.15592	1.665	2.87	10.00	6.061	19.70	90.00	-99.9	-99.9
4	5	Caloundra	54	1754668800	2025-08-09T02:00:00	-26.84620	153.15587	1.593	2.68	11.11	5.797	19.20	84.40	-99.9	-99.9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6774	6775	Hay Point TriAxys	4740tx	1755297600	2025-08-16 08:40:00	-21.27770	149.32270	0.230	0.36	6.70	3.500	20.82	84.25	-99.9	-99.9
6775	6776	Hay Point TriAxys	4740tx	1755298800	2025-08-16 09:00:00	-21.27780	149.32270	0.220	0.38	6.70	3.500	20.83	90.25	-99.9	-99.9
6776	6777	Hay Point TriAxys	4740tx	1755300000	2025-08-16 09:20:00	-21.27780	149.32270	0.220	0.38	6.90	3.900	20.85	92.25	-99.9	-99.9
6777	6778	Hay Point TriAxys	4740tx	1755301200	2025-08-16 09:40:00	-21.27790	149.32270	0.240	0.39	10.50	4.400	20.89	69.25	-99.9	-99.9
6778	6779	Hay Point TriAxys	4740tx	1755302400	2025-08-16 10:00:00	0.00000	0.00000	0.200	0.32	3.80	3.600	20.97	94.25	-99.9	-99.9

6779 rows × 15 columns

[Q1] Findings from first acccess

1.2 Make a note of your first access of the data

- If the URL appears on the Wave Data page were used the format of the csv file has one extra row indicates the date and time the csv file was extracted (timestamp 18 Mar 2025 9:33am)
- The URL used in my code can be found by clicking an arrow pointing down next to the "go to resource" on the top right cornor.

1.3 Identify an appropriate index, and make a note of the columns.

- the appropriate index column for this data frame is ***'_id'***.

> **The data consist of 15 columns as follows:**
>
> - **_id**: An index column for wave data
> - **Site**: Wave monitoring site name
> - **SiteNumber**: Unique number represents each site name
> - **Seconds**: Unix timestamp represents the number of seconds since January 1, 1970 (UTC)
> - **DataTime**: Date and time of record.
> - **Latitude**: Monitoring site latitude.
> - **Longitude**: Monitoring site longitude.
> - **Hsig**: Average height of the tallest one-third of waves (significant wave) during one record period.
> - **Hmax**: The maximum wave height in the record.
> - **Tp**: The peak energy wave period.
> - **Tz**: The zero upcrossing wave period.
> - **SST**: Approximation of sea surface temperature.
> - **Direction**: Direction (related to true north) from which the peak period waves are coming.
> - **CurrentSpeed**: Current speed of the wave
> - **CurrentDirection**: Current Direction of the wave

- In the analysis of wave height, I focus on column ***'Site'*** (identify the location of the monitoring site) and ***'Hsig'*** (average height of the significant wave).

[Q2] Save the data

- Transform your grouped data into a dataframe
- Save the dataframe as a CSV file with the date that reflects the URL access in Q1

```
# Your answer here

# 2.1 Transform your grouped data into a dataframe
NRT_wave_df = pd.read_csv(NRT_wave_data,index_col = '_id')
NRT_wave_df
```

_id	Site	SiteNumber	Seconds	DateTime	Latitude	Longitude	Hsig	Hmax	Tp	Tz	SST	Direction	Current Speed	Current Direction
1	Caloundra	54	1743429600	2025-04-01T00:00:00	-26.84629	153.15553	1.050	1.87	9.09	4.545	25.90	106.90	-99.9	-99.9
2	Caloundra	54	1743431400	2025-04-01T00:30:00	-26.84622	153.15570	1.172	1.81	8.33	4.444	25.90	99.80	-99.9	-99.9
3	Caloundra	54	1743433200	2025-04-01T01:00:00	-26.84621	153.15581	1.225	2.02	9.09	4.545	25.90	106.90	-99.9	-99.9
4	Caloundra	54	1743435000	2025-04-01T01:30:00	-26.84620	153.15585	1.168	1.89	9.09	4.762	25.80	102.70	-99.9	-99.9
5	Caloundra	54	1743436800	2025-04-01T02:00:00	-26.84620	153.15589	1.134	2.21	8.33	4.651	25.80	91.40	-99.9	-99.9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7467	Hay Point TriAxys	4740tx	1744093200	2025-04-08 16:20:00	-21.27830	149.32270	0.810	1.24	4.30	3.600	27.73	124.26	-99.9	-99.9
7468	Hay Point TriAxys	4740tx	1744094400	2025-04-08 16:40:00	-21.27830	149.32270	0.910	1.75	4.10	3.600	27.74	127.26	-99.9	-99.9
7469	Hay Point TriAxys	4740tx	1744095600	2025-04-08 17:00:00	-21.27830	149.32260	0.830	1.38	4.30	3.700	27.72	116.26	-99.9	-99.9
7470	Hay Point TriAxys	4740tx	1744096800	2025-04-08 17:20:00	-21.27830	149.32270	0.900	1.48	4.40	3.800	27.72	115.26	-99.9	-99.9
7471	Hay Point TriAxys	4740tx	1744098000	2025-04-08 17:40:00	-21.27830	149.32270	0.950	2.06	4.50	3.600	27.71	104.26	-99.9	-99.9

7471 rows × 14 columns

```
# 2.2 Save the dataframe as a CSV file with the date that reflects the URL access in Q1
# Create new csv file in data folder
file_name = "NRT_wave.csv"
path = "data"
NRT_wave_df.to_csv(f"{path}/{file_name}")
```

Read the data from a file

- To read the same data back in (rather than up-to-date data), write code here to read in your file from Q2

```
# Your answer here
# Read csv file
NRT_wave_file_df = pd.read_csv(f"{path}/{file_name}", index_col = "_id")
NRT_wave_file_df
```

_id	Site	SiteNumber	Seconds	DateTime	Latitude	Longitude	Hsig	Hmax	Tp	Tz	SST	Direction	Current Speed	Current Direction
1	Caloundra	54	1743429600	2025-04-01T00:00:00	-26.84629	153.15553	1.050	1.87	9.09	4.545	25.90	106.90	-99.9	-99.9
2	Caloundra	54	1743431400	2025-04-01T00:30:00	-26.84622	153.15570	1.172	1.81	8.33	4.444	25.90	99.80	-99.9	-99.9
3	Caloundra	54	1743433200	2025-04-01T01:00:00	-26.84621	153.15581	1.225	2.02	9.09	4.545	25.90	106.90	-99.9	-99.9
4	Caloundra	54	1743435000	2025-04-01T01:30:00	-26.84620	153.15585	1.168	1.89	9.09	4.762	25.80	102.70	-99.9	-99.9
5	Caloundra	54	1743436800	2025-04-01T02:00:00	-26.84620	153.15589	1.134	2.21	8.33	4.651	25.80	91.40	-99.9	-99.9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7467	Hay Point TriAxys	4740tx	1744093200	2025-04-08 16:20:00	-21.27830	149.32270	0.810	1.24	4.30	3.600	27.73	124.26	-99.9	-99.9
7468	Hay Point TriAxys	4740tx	1744094400	2025-04-08 16:40:00	-21.27830	149.32270	0.910	1.75	4.10	3.600	27.74	127.26	-99.9	-99.9
7469	Hay Point TriAxys	4740tx	1744095600	2025-04-08 17:00:00	-21.27830	149.32260	0.830	1.38	4.30	3.700	27.72	116.26	-99.9	-99.9
7470	Hay Point TriAxys	4740tx	1744096800	2025-04-08 17:20:00	-21.27830	149.32270	0.900	1.48	4.40	3.800	27.72	115.26	-99.9	-99.9
7471	Hay Point TriAxys	4740tx	1744098000	2025-04-08 17:40:00	-21.27830	149.32270	0.950	2.06	4.50	3.600	27.71	104.26	-99.9	-99.9

7471 rows × 14 columns

[Q3] Analyse the data

- Filter the data to include only sites for the South East coast (from Gold Coast to Sunshine Coast)
- Group the filtered data by site
- Obtain an appropriate aggregate for the groups (e.g. Sum, Mean, etc)
- Save the grouped data as a new dataframe

```
# Your answer here

# 3.1 Filter the data to include only sites for the South East coast (from Gold Coast to Sunshine Coast)

# Get all the site name from Gold Coast to Sunshine Coast
# Refering to https://www.qld.gov.au/environment/coasts-waterways/beach/monitoring/waves-sites
SE_coast = ['Mooloolaba','North Moreton Bay','Caloundra','Tweed Offshore','Brisbane Mk4','Gold Coast Mk4','Palm Beach Mk4','Bilinga','Tweed Heads Mk4']

# Filter data to include only site from SE coast
NRT_wave_SEQ_df = NRT_wave_file_df[NRT_wave_file_df['Site'].isin(SE_coast)]
NRT_wave_SEQ_df

#Using isin instead of typing 'Site' == #site name#
```

_id	Site	SiteNumber	Seconds	DateTime	Latitude	Longitude	Hsig	Hmax	Tp	Tz	SST	Direction	Current Speed	Current Direction
1	Caloundra	54	1743429600	2025-04-01T00:00:00	-26.84629	153.15553	1.050	1.87	9.09	4.545	25.90	106.90	-99.90	-99.90
2	Caloundra	54	1743431400	2025-04-01T00:30:00	-26.84622	153.15570	1.172	1.81	8.33	4.444	25.90	99.80	-99.90	-99.90
3	Caloundra	54	1743433200	2025-04-01T01:00:00	-26.84621	153.15581	1.225	2.02	9.09	4.545	25.90	106.90	-99.90	-99.90
4	Caloundra	54	1743435000	2025-04-01T01:30:00	-26.84620	153.15585	1.168	1.89	9.09	4.762	25.80	102.70	-99.90	-99.90
5	Caloundra	54	1743436800	2025-04-01T02:00:00	-26.84620	153.15589	1.134	2.21	8.33	4.651	25.80	91.40	-99.90	-99.90
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6540	Bilinga	4224	1744086600	2025-04-08T14:30:00	-28.14270	153.51328	0.600	1.21	9.09	3.930	25.33	74.83	0.09	201.14
6541	Bilinga	4224	1744088400	2025-04-08T15:00:00	-28.14274	153.51326	0.530	0.95	9.52	3.670	25.28	77.20	0.09	168.09
6542	Bilinga	4224	1744090200	2025-04-08T15:30:00	-28.14274	153.51328	0.540	0.96	10.00	3.720	25.28	81.77	0.07	179.08
6543	Bilinga	4224	1744092000	2025-04-08T16:00:00	-28.14274	153.51328	0.540	0.89	10.00	3.740	25.24	76.24	0.10	180.04
6544	Bilinga	4224	1744093800	2025-04-08T16:30:00	-28.14274	153.51330	0.570	0.98	10.53	3.800	25.19	81.42	0.07	168.26

3327 rows × 14 columns

```
# 3.2 Group the filtered data by site
NRT_wave_SEQ_df.groupby(['Site']).size()
```

```
Site
Bilinga              370
Brisbane Mk4         369
Caloundra            369
Gold Coast Mk4       371
Mooloolaba           370
North Moreton Bay    368
Palm Beach Mk4       370
Tweed Heads Mk4      370
Tweed Offshore       370
dtype: int64
```

```
# 3.3 Obtain an appropriate aggregate for the groups (e.g. Sum, Mean, etc)
# 3.4 Save the grouped data as a new dataframe

# Will focus on variables for wave heights Hsig (average wave height)
wave_hight_col = ['Hsig']

NRT_wave_SEQ_mean_df = NRT_wave_SEQ_df.groupby(['Site'])[wave_hight_col].mean()
NRT_wave_SEQ_mean_df
```

Site	Hsig
Bilinga	0.822622
Brisbane Mk4	1.571924
Caloundra	0.820314
Gold Coast Mk4	1.120135
Mooloolaba	1.125635
North Moreton Bay	0.837109
Palm Beach Mk4	0.977892
Tweed Heads Mk4	1.214351
Tweed Offshore	1.651892

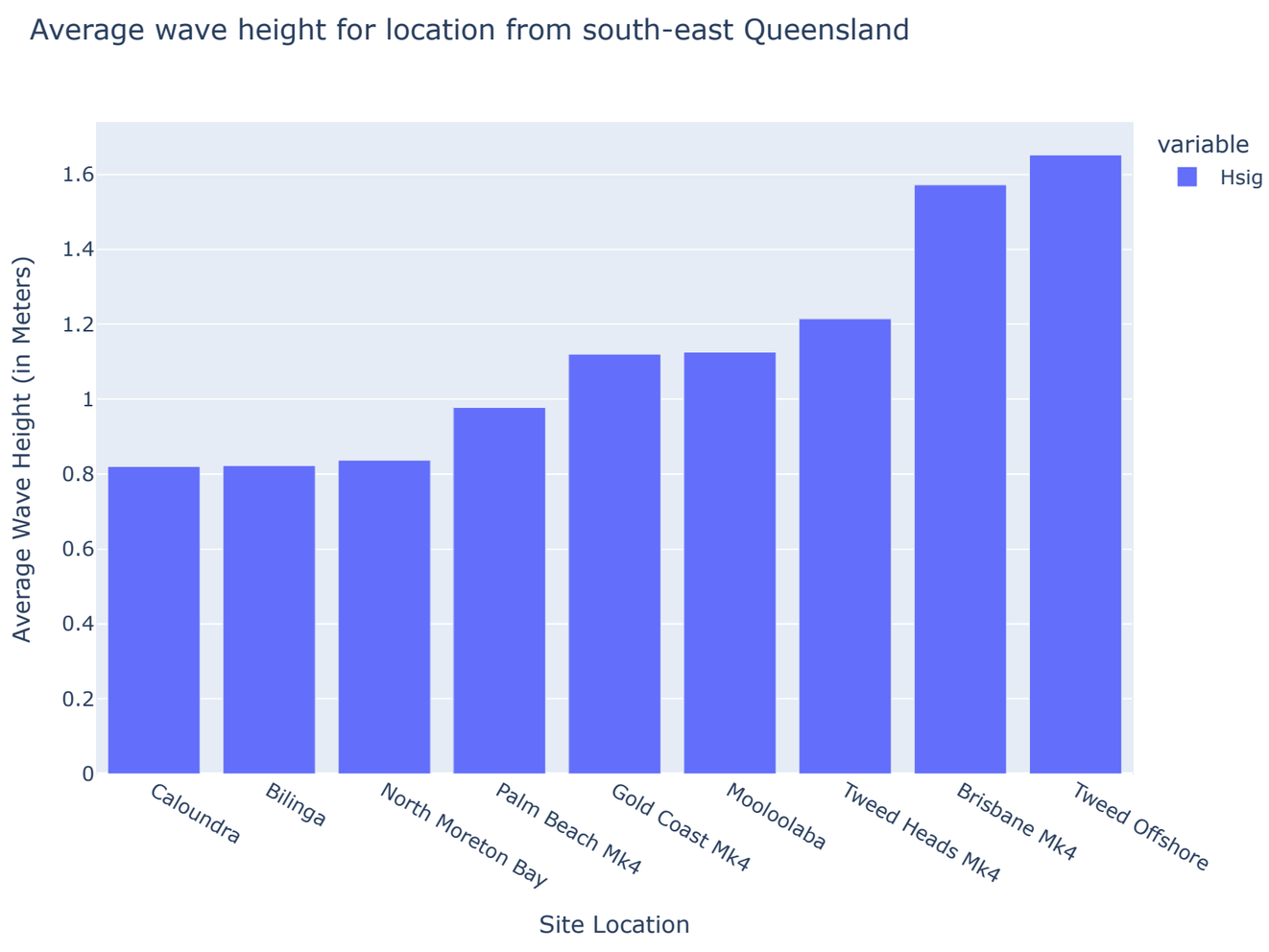
```
NRT_wave_SEQ_mean_df_sorted = NRT_wave_SEQ_mean_df.sort_values(by='Hsig')
NRT_wave_SEQ_mean_df_sorted
```

Site	Hsig
Caloundra	0.820314
Bilinga	0.822622
North Moreton Bay	0.837109
Palm Beach Mk4	0.977892
Gold Coast Mk4	1.120135
Mooloolaba	1.125635
Tweed Heads Mk4	1.214351
Brisbane Mk4	1.571924
Tweed Offshore	1.651892

[Q4] Visualise the data

- Visualise the grouped data with an appropriate chart
- Ensure X and Y axes are labelled appropriately
- Add an appropriate title for the chart

```
# Your answer here
import plotly.express as px

fig = px.bar(NRT_wave_SEQ_mean_df_sorted,
             labels={'Site': 'Site Location', 'value': 'Average Wave Height (in Meters)'},
             title = "Average wave height for location from south-east Queensland",
             width=800, height=600)
fig.show()
```



[Q5] Extract Insights

- Referring back to the question, what can we learn from this data (analysed and visualised above)?
- Thinking about the kind of data, how might it be used strategically in a major weather event?
- Who might benefit most from strategic use of this data.

Your answer here:

Referring back to the question, what can we learn from this data (analysed and visualised above)?

1. Average wave height for each location in southeast Queensland.
2. Location prone to coastline erosion due to more powerful waves.

Thinking about the kind of data, how might it be used strategically in a major weather event?

1. Warning & evacuation priority for each location.
2. Flood prevention facility designs. (how high should the sandbag wall be for each location)

Who might benefit most from strategic use of this data?

1. Bureau of Meteorology for weather and flood warnings since they are the primary responsible agency. They can integrate this data with other available information to improve their predictions.
2. Local Government for infrastructure planning to prevent erosions along the coast with strong waves.
3. Surf club/school for surfer camp location plan for surfers with different skill levels so they can

---

Part B - creating a narrative to answer significant questions

SCENARIO: You are responsible for providing helpful analysis to a variety of government departments (Including Police and Emergency Services). The responsible government ministers have asked your team to make data analysis plans to assist during a major weather emergency in South East Queensland. You have been tasked (within your team) to focus on waves and storm tide monitoring. Your team has also suggested that you consider flood maps for relevant areas (See: Local government flood maps and data)

ESSENTIAL REQUIREMENTS: Your task as a data analyst is to:

- Ensure that you use the techniques and libraries/packages that have been used in class
- Identify high quality questions that when answered may be helpful in addressing the scenario above (You may choose to focus on just 1 site that you use as a proof of concept for multiple sites)
- Clean and filter the data as appropriate
- Analyse the data in a way that answers your questions and ultimately addresses the concern in the scenario
- Visualise your results in a meaningful way that is helpful in making visible key findings
- Provide a detailed summary of the insights found and how they address the original questions and scenario

AUTHENTICITY AND INTEGRITY: You will be marked on (a) *HOW* you undertake the task todgether; with (b) detail of *WHY* you made various decisions involved in the tasks; and (c) acknowledgement of **WHERE** you used material that is not directly yours. Therefore, you must document your thinking and approach throughout the notebook using the Markdown cells, and give credit to other resources as appropriate. You are encouraged to use the `Exemplars` PDF to help write your code. You may use online resources including `GenAI tools` and `stackoverflow` to help you write your code, however you must acknowledge that you are using these resources in the markdown cells explaining your analysis. Note that you do not need to use formal referencing for this.

---

1. QUESTION:

I considered the dynamic nature of the data, which is always changing over time. Capturing a pattern from a single moment in time might not give a complete picture of the situation because tide levels and wave heights vary frequently. Rather, I decided that ongoing monitoring might provide a more useful insight. By tracking these patterns over time, we can build a monitoring tool that provides ongoing insights, making it valuable for long-term flood risk assessments and infrastructure planning.

Concern: Waves and storm tides pose significant risks to coastal communities and natural environments, contributing to flooding and erosion.

> **Question:** How do near real-time wave height and tide level data vary at major coastal locations in South East Queensland, and what patterns can be observed through continuous monitoring and how can those information help with flood warnings and infrastructure planning.

2. DATA:

To answer the question, I will use both near real-time `Wave Data` and `Storm Tide Data`. In this part, I perform data cleaning, including formatting, before identifying relevant data fields that would help answer the question.

2.1 Wave Data

For `Wave Data`, continuing from part A, I clean the data further by changing the format of a `DateTime` field. In the beginning, python detects this as an object type since it is in a long string with *"YYYY-MM-DDTHH:MM:SS"* format.

I convert the `DateTime` fields from object type to `datetime64` before creating a new column with the date-only format as I want to monitor the conditions by daily average.

```
#Inspect data field in dataframe
NRT_wave_SEQ_df
```

_id	Site	SiteNumber	Seconds	DateTime	Latitude	Longitude	Hsig	Hmax	Tp	Tz	SST	Direction	Current Speed	Current Direction
1	Caloundra	54	1743429600	2025-04-01T00:00:00	-26.84629	153.15553	1.050	1.87	9.09	4.545	25.90	106.90	-99.90	-99.90
2	Caloundra	54	1743431400	2025-04-01T00:30:00	-26.84622	153.15570	1.172	1.81	8.33	4.444	25.90	99.80	-99.90	-99.90
3	Caloundra	54	1743433200	2025-04-01T01:00:00	-26.84621	153.15581	1.225	2.02	9.09	4.545	25.90	106.90	-99.90	-99.90
4	Caloundra	54	1743435000	2025-04-01T01:30:00	-26.84620	153.15585	1.168	1.89	9.09	4.762	25.80	102.70	-99.90	-99.90
5	Caloundra	54	1743436800	2025-04-01T02:00:00	-26.84620	153.15589	1.134	2.21	8.33	4.651	25.80	91.40	-99.90	-99.90
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6540	Bilinga	4224	1744086600	2025-04-08T14:30:00	-28.14270	153.51328	0.600	1.21	9.09	3.930	25.33	74.83	0.09	201.14
6541	Bilinga	4224	1744088400	2025-04-08T15:00:00	-28.14274	153.51326	0.530	0.95	9.52	3.670	25.28	77.20	0.09	168.09
6542	Bilinga	4224	1744090200	2025-04-08T15:30:00	-28.14274	153.51328	0.540	0.96	10.00	3.720	25.28	81.77	0.07	179.08
6543	Bilinga	4224	1744092000	2025-04-08T16:00:00	-28.14274	153.51328	0.540	0.89	10.00	3.740	25.24	76.24	0.10	180.04
6544	Bilinga	4224	1744093800	2025-04-08T16:30:00	-28.14274	153.51330	0.570	0.98	10.53	3.800	25.19	81.42	0.07	168.26

3327 rows × 14 columns

```
#Inspect data type for each field
NRT_wave_SEQ_df.dtypes
```

```
Site                 object
SiteNumber           object
Seconds               int64
DateTime             object
Latitude            float64
Longitude           float64
Hsig                float64
Hmax                float64
Tp                  float64
Tz                  float64
SST                 float64
Direction           float64
Current Speed       float64
Current Direction   float64
dtype: object
```

```
#Convert DateTime into correct type (datatime64)
NRT_wave_SEQ_df['DateTime'] = pd.to_datetime(NRT_wave_SEQ_df['DateTime'], errors='coerce', format='%Y-%m-%dT%H:%M:%S')
NRT_wave_SEQ_df.dtypes
```

```
/tmp/ipykernel_292/921785337.py:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

Site                          object
SiteNumber                    object
Seconds                        int64
DateTime             datetime64[ns]
Latitude                    float64
Longitude                   float64
Hsig                        float64
Hmax                        float64
Tp                          float64
Tz                          float64
SST                         float64
Direction                   float64
Current Speed               float64
Current Direction           float64
dtype: object
```

I created a new datafield called `Date` which derive from the original `DateTime` field. I use the GenerativeAI tool (ChatGPT) to help me with the code.

Prompt Used:

> "Suggest me a code that can extract only date-month-year information from datafield with date-month-year and time."

```
#Create new column for date (no time) to use as time stamp
NRT_wave_SEQ_df['Date'] = NRT_wave_SEQ_df['DateTime'].dt.date
```

```
/tmp/ipykernel_292/1163174838.py:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

After the data is cleaned, I consider each data field to get more understanding by reading a Data Glossary. I also use ChatGPT to help explain how each piece of information integrates for insights (`Hsig`, `Hmax`, `Tp`, `Tz`, `SST`) since most of the data fields are in specific terms.

Prompt Used:

> Please help explain to me what Hsig, Hmax, Tp, Tz, SST mean and how are they used in coastal monitoring.

I find `Hsig` and `Tp` particularly interesting, so I followed up with ChatGPT to understand how they combine to offer valuable insights. According to ChatGPT `Hsig` and `TP` together can identify the wave strength.

Summary of Response

> Hsig (significant wave height) and Tp (peak period) are two key wave parameters that, when analysed together, provide valuable insights into coastal conditions. Hsig reflects the average height of the largest waves, giving a sense of overall wave energy, while Tp indicates the time between the most energetic waves and helps distinguish between local wind-driven waves and distant swell. Together, they can be used to estimate wave power and understand wave behavior—*long-period swells with high Hsig carry significantly more energy and pose a greater threat to coastal infrastructure through erosion and overtopping*. This combination is essential for accurate coastal monitoring, forecasting, and planning.

Therefore I decided to includes both `Hsig` and `Tp` in my wave analysis.

The data fields included for further analysis are as follows:

- `Site`: Monitoring Site Name to determine the location known by public, use in the group by
- `Date`: Monitoring date and time to identify time series, use in groups by
- `Latitude`: Coordinate used for potting location on the map
- `Longitude`: Coordinate used for potting location on the map
- `Hsig`: Significant wave height during the monitoring period as a representative of wave height
- `Tp`: Peak Energy Wave Period, time in seconds between the most powerful waves in a set

```
#Define column to include in Analysis
wave_col = ['Latitude','Longitude','Hsig','Tp']
```

2.2 Storm Tide Data

Following the same approach I used for the `Wave Data`, I began by importing the `Storm Tide Data` from the link. To make data handling more efficient, I saved a local copy of the CSV file.

Having the file locally allows for quicker access and makes it easier to review the dataset as a whole, especially when I want to get a quick overview of the table structure and the data it contains.

```
# Open a CSV file from URL into a new dataframe
NRT_tide_data = "https://www.data.qld.gov.au/datastore/dump/7afe7233-fae0-4024-bc98-3a72f05675bd?bom=True"
pd.read_csv(NRT_tide_data)

NRT_tide_df = pd.read_csv(NRT_tide_data,index_col = '_id')
NRT_tide_df

# Create new csv file in data folder
file_name = "NRT_tide.csv"
path = "data"
NRT_tide_df.to_csv(f"{path}/{file_name}")

NRT_tide_file_df = pd.read_csv(NRT_tide_data,index_col = '_id')
NRT_tide_file_df
```

_id	Site	Seconds	DateTime	Water Level	Prediction	Residual	Latitude	Longitude
1	abellpoint	1743429600	2025-04-01T00:00	3.440	3.358	0.082	-20.2608	148.7103
2	abellpoint	1743430200	2025-04-01T00:10	3.447	3.372	0.075	-20.2608	148.7103
3	abellpoint	1743430800	2025-04-01T00:20	3.456	3.374	0.082	-20.2608	148.7103
4	abellpoint	1743431400	2025-04-01T00:30	3.449	3.366	0.083	-20.2608	148.7103
5	abellpoint	1743432000	2025-04-01T00:40	3.408	3.346	0.062	-20.2608	148.7103
...	...	...	...	...	...	...	...	...
64782	whyteislandnx	1744096800	2025-04-08T17:20	1.683	1.602	0.081	-27.4017	153.1574
64783	whyteislandnx	1744097400	2025-04-08T17:30	1.735	1.647	0.088	-27.4017	153.1574
64784	whyteislandnx	1744098000	2025-04-08T17:40	1.771	1.690	0.081	-27.4017	153.1574
64785	whyteislandnx	1744098600	2025-04-08T17:50	-99.000	1.730	-99.000	-27.4017	153.1574
64786	whyteislandnx	1744099200	2025-04-08T18:00	-99.000	1.767	-99.000	-27.4017	153.1574

64786 rows × 8 columns

To prepare the `Storm Tide Data` for analysis, I focused on locations within the South East Queensland (SEQ) region. To ensure that only data from sites located within the same region as the wave monitoring locations are included, I used the minimum and maximum `Latitude` values from the Wave dataset as a filter.

```
#Filter Storm tide Data from site (use latitude from Wave Data to filter)
NRT_wave_SEQ_df['Latitude'].describe()
```

```
count    3327.000000
mean      -27.599578
std         0.626237
min       -28.214360
25%       -28.142660
50%       -27.964220
75%       -26.899370
max       -26.565150
Name: Latitude, dtype: float64
```

```
#Filter only sites between max and min latitude in wave data (only site in South East Queensland)
NRT_tide_SEQ_df = NRT_tide_file_df[(NRT_tide_file_df['Latitude'] < -26.565220) & (NRT_tide_file_df['Latitude'] > -28.214620)]
NRT_tide_SEQ_df
```

_id	Site	Seconds	DateTime	Water Level	Prediction	Residual	Latitude	Longitude
1118	bananabank	1743429600	2025-04-01T00:00	2.833	2.858	-0.025	-27.5411	153.3368
1119	bananabank	1743430200	2025-04-01T00:10	2.810	2.824	-0.014	-27.5411	153.3368
1120	bananabank	1743430800	2025-04-01T00:20	2.772	2.782	-0.010	-27.5411	153.3368
1121	bananabank	1743431400	2025-04-01T00:30	2.721	2.731	-0.010	-27.5411	153.3368
1122	bananabank	1743432000	2025-04-01T00:40	2.662	2.672	-0.010	-27.5411	153.3368
...	...	...	...	...	...	...	...	...
64782	whyteislandnx	1744096800	2025-04-08T17:20	1.683	1.602	0.081	-27.4017	153.1574
64783	whyteislandnx	1744097400	2025-04-08T17:30	1.735	1.647	0.088	-27.4017	153.1574
64784	whyteislandnx	1744098000	2025-04-08T17:40	1.771	1.690	0.081	-27.4017	153.1574
64785	whyteislandnx	1744098600	2025-04-08T17:50	-99.000	1.730	-99.000	-27.4017	153.1574
64786	whyteislandnx	1744099200	2025-04-08T18:00	-99.000	1.767	-99.000	-27.4017	153.1574

23457 rows × 8 columns

To make sure each data field is in the correct type. I changed the `DateTime` columns to `datetime64` format.

Following this conversion, I made a new `Date` column by extracting only the date, month, and year to use as a timestamp for daily monitoring.

```
NRT_tide_SEQ_df.dtypes
```

```
Site            object
Seconds          int64
DateTime        object
Water Level    float64
Prediction     float64
Residual       float64
Latitude       float64
Longitude      float64
dtype: object
```

```
NRT_tide_SEQ_df['DateTime'] = pd.to_datetime(NRT_tide_SEQ_df['DateTime'], errors='coerce', format='%Y-%m-%dT%H:%M')

#Create new column for date (no time) to use as time stamp
NRT_tide_SEQ_df['Date'] = NRT_tide_SEQ_df['DateTime'].dt.date #Chat GPT
NRT_tide_SEQ_df
```

```
/tmp/ipykernel_292/2079727820.py:1: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/tmp/ipykernel_292/2079727820.py:4: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

_id	Site	Seconds	DateTime	Water Level	Prediction	Residual	Latitude	Longitude	Date
1118	bananabank	1743429600	2025-04-01 00:00:00	2.833	2.858	-0.025	-27.5411	153.3368	2025-04-01
1119	bananabank	1743430200	2025-04-01 00:10:00	2.810	2.824	-0.014	-27.5411	153.3368	2025-04-01
1120	bananabank	1743430800	2025-04-01 00:20:00	2.772	2.782	-0.010	-27.5411	153.3368	2025-04-01
1121	bananabank	1743431400	2025-04-01 00:30:00	2.721	2.731	-0.010	-27.5411	153.3368	2025-04-01
1122	bananabank	1743432000	2025-04-01 00:40:00	2.662	2.672	-0.010	-27.5411	153.3368	2025-04-01
...	...	...	...	...	...	...	...	...	...
64782	whyteislandnx	1744096800	2025-04-08 17:20:00	1.683	1.602	0.081	-27.4017	153.1574	2025-04-08
64783	whyteislandnx	1744097400	2025-04-08 17:30:00	1.735	1.647	0.088	-27.4017	153.1574	2025-04-08
64784	whyteislandnx	1744098000	2025-04-08 17:40:00	1.771	1.690	0.081	-27.4017	153.1574	2025-04-08
64785	whyteislandnx	1744098600	2025-04-08 17:50:00	-99.000	1.730	-99.000	-27.4017	153.1574	2025-04-08
64786	whyteislandnx	1744099200	2025-04-08 18:00:00	-99.000	1.767	-99.000	-27.4017	153.1574	2025-04-08

23457 rows × 9 columns

As I explored the data, I found that the `Water Level` column contained noise from incorrect entries and/or missing values. I eliminated rows where the water level was either `NaN` or indicated with error values `-99` to guarantee a cleaner dataset for analysis. This stage guarantees that the analysis is founded on trustworthy data.

```
#Drop row with error data (Water Level = -99) > ChatGPT
NRT_tide_SEQ_df = NRT_tide_SEQ_df[(NRT_tide_SEQ_df['Water Level'] != -99)]

#Drop row with na value
NRT_tide_SEQ_df = NRT_tide_SEQ_df.dropna(subset=['Water Level'])

NRT_tide_SEQ_df
```

_id	Site	Seconds	DateTime	Water Level	Prediction	Residual	Latitude	Longitude	Date
1118	bananabank	1743429600	2025-04-01 00:00:00	2.833	2.858	-0.025	-27.5411	153.3368	2025-04-01
1119	bananabank	1743430200	2025-04-01 00:10:00	2.810	2.824	-0.014	-27.5411	153.3368	2025-04-01
1120	bananabank	1743430800	2025-04-01 00:20:00	2.772	2.782	-0.010	-27.5411	153.3368	2025-04-01
1121	bananabank	1743431400	2025-04-01 00:30:00	2.721	2.731	-0.010	-27.5411	153.3368	2025-04-01
1122	bananabank	1743432000	2025-04-01 00:40:00	2.662	2.672	-0.010	-27.5411	153.3368	2025-04-01
...	...	...	...	...	...	...	...	...	...
64780	whyteislandnx	1744095600	2025-04-08 17:00:00	1.603	1.507	0.096	-27.4017	153.1574	2025-04-08
64781	whyteislandnx	1744096200	2025-04-08 17:10:00	1.640	1.555	0.085	-27.4017	153.1574	2025-04-08
64782	whyteislandnx	1744096800	2025-04-08 17:20:00	1.683	1.602	0.081	-27.4017	153.1574	2025-04-08
64783	whyteislandnx	1744097400	2025-04-08 17:30:00	1.735	1.647	0.088	-27.4017	153.1574	2025-04-08
64784	whyteislandnx	1744098000	2025-04-08 17:40:00	1.771	1.690	0.081	-27.4017	153.1574	2025-04-08

15615 rows × 9 columns

After cleaning the `Storm Tide Data`, I reviewed the available fields to determine which ones would be relevant. I chose to include five important ones that would offer crucial information for both geographical and temporal research in order to visualise storm tide patterns and assist with coastal analysis.

The following data fields are included:

- `Site`: Monitoring site name to determine the location known by the public, use in the group by
- `Date`: Monitoring date and time to identify time series, use in groups by
- `Latitude`: Geographical coordinate used for mapping the monitoring site.
- `Longitude`: Geographical coordinate used for mapping the monitoring site.
- `Water Level`: The recorded tide level, the primary variable in storm tide analysis.

```
tide_col = ['Latitude','Longitude','Water Level']
```

3. Analysis

Given that the raw data is recorded every 30 minutes, I decided to group the data by site and by day. Minute-level data, while highly detailed, can be overwhelming and may contain short-term fluctuations that are not necessary for high-level monitoring.

I decided to use daily average data since it can provide meaningful trends without excessive noise. However, this design still allows flexibility. The monitoring time window can easily be adjusted by modifying the `Date` field.

I decided to compute the daily mean for each site for both `Wave Data` and `Storm Tide Data`. As a result, average conditions over time are accurately represented.

3.1 Wave Data Analysis

To analyse the impact of waves by location, my goal was to understand how wave conditions vary by site on a daily basis. Identifying locations that frequently experience higher waves or longer wave periods can help assess the patterns.

The mean values of `Wave Data` parameters like wave height (`Hsig`) and wave period (`Tp`) indicate areas at higher risk of coastal impacts like erosion or wave overtopping.

```
#Create new data frame for data related to wave strength
NRT_wave_SEQ_mean_df = NRT_wave_SEQ_df.groupby(['Site','Date'])[wave_col].mean().reset_index()
NRT_wave_SEQ_mean_df
```

	Site	Date	Latitude	Longitude	Hsig	Tp
0	Bilinga	2025-04-01	-28.142139	153.513157	0.938125	10.309583
1	Bilinga	2025-04-02	-28.142298	153.513384	1.320833	13.487500
2	Bilinga	2025-04-03	-28.141816	153.513107	0.863958	14.366458
3	Bilinga	2025-04-04	-28.142016	153.513061	0.697708	9.735833
4	Bilinga	2025-04-05	-28.142693	153.513445	0.707083	6.940208
...	...	...	...	...	...	...
67	Tweed Offshore	2025-04-04	-28.211819	153.680839	1.360625	8.323542
68	Tweed Offshore	2025-04-05	-28.213966	153.680662	1.129167	8.012917
69	Tweed Offshore	2025-04-06	-28.214252	153.680637	1.634375	8.801042
70	Tweed Offshore	2025-04-07	-28.214281	153.680612	1.395833	10.465625
71	Tweed Offshore	2025-04-08	-28.214234	153.680624	1.407353	9.449412

72 rows × 6 columns

3.2 Storm Tide Data Analysis

My goal was to analyse the effects of tides by location by concentrating on daily water level monitoring. I wanted to identify locations that often have high water levels to learn more about their characteristics. This approach allows for a better understanding of the probability of flooding and the need for preventative action in different areas.

The average daily `Water Level` from `Storm Tide Data` at each location provides information about flood risk and helps comprehend how geography affects possible effects, such as whether a high-risk area is a densely populated community or a low-lying natural area.

```
#Create new data frame for data related to wave strength
NRT_tide_SEQ_mean_df = NRT_tide_SEQ_df.groupby(['Site','Date'])[tide_col].mean().reset_index()
NRT_tide_SEQ_mean_df
```

	Site	Date	Latitude	Longitude	Water Level
0	bananabank	2025-04-01	-27.5411	153.3368	1.537222
1	bananabank	2025-04-02	-27.5411	153.3368	1.676500
2	bananabank	2025-04-03	-27.5411	153.3368	1.662653
3	bananabank	2025-04-04	-27.5411	153.3368	1.598181
4	bananabank	2025-04-05	-27.5411	153.3368	1.612375
...	...	...	...	...	...
107	whyteislandnx	2025-04-04	-27.4017	153.1574	1.500750
108	whyteislandnx	2025-04-05	-27.4017	153.1574	1.491326
109	whyteislandnx	2025-04-06	-27.4017	153.1574	1.471847
110	whyteislandnx	2025-04-07	-27.4017	153.1574	1.450819
111	whyteislandnx	2025-04-08	-27.4017	153.1574	1.401626

112 rows × 5 columns

4. Visualization:

Moving towards the visualisation step, I started by identifying the main goal for the analysis according to the question. The aim is to convert the data into relevant insights that could guide decision-making for coastal protection and management. Keeping this in mind, I designed the visualisations around the following objective:

- **Identify High-Risk Locations:** The visualization should help pinpoint sites that are most at risk from waves and storms, which can translate to coastal impacts such as erosion or flooding. By highlighting these locations, the analysis can support more targeted risk management and preventive actions, enabling decision-makers to prioritize resources and plan effective implementation.
- **Inspect Patterns:** The visualisation should highlight trends in coastal conditions over time to help identify recurring patterns and potential risks, which can support early warnings and better planning for coastal infrastructure.
- **Geographical Insight:** To complement the wave and tide analysis, the map's visualisation should help pinpoint the exact locations of the monitoring sites, providing context about each site's surroundings, such as proximity to populated areas or coastal features. Visualising this data geographically will help better understand how location influences the severity of coastal impacts, helping in infrastructure planning and risk prioritisation.

4.1 Wave Data

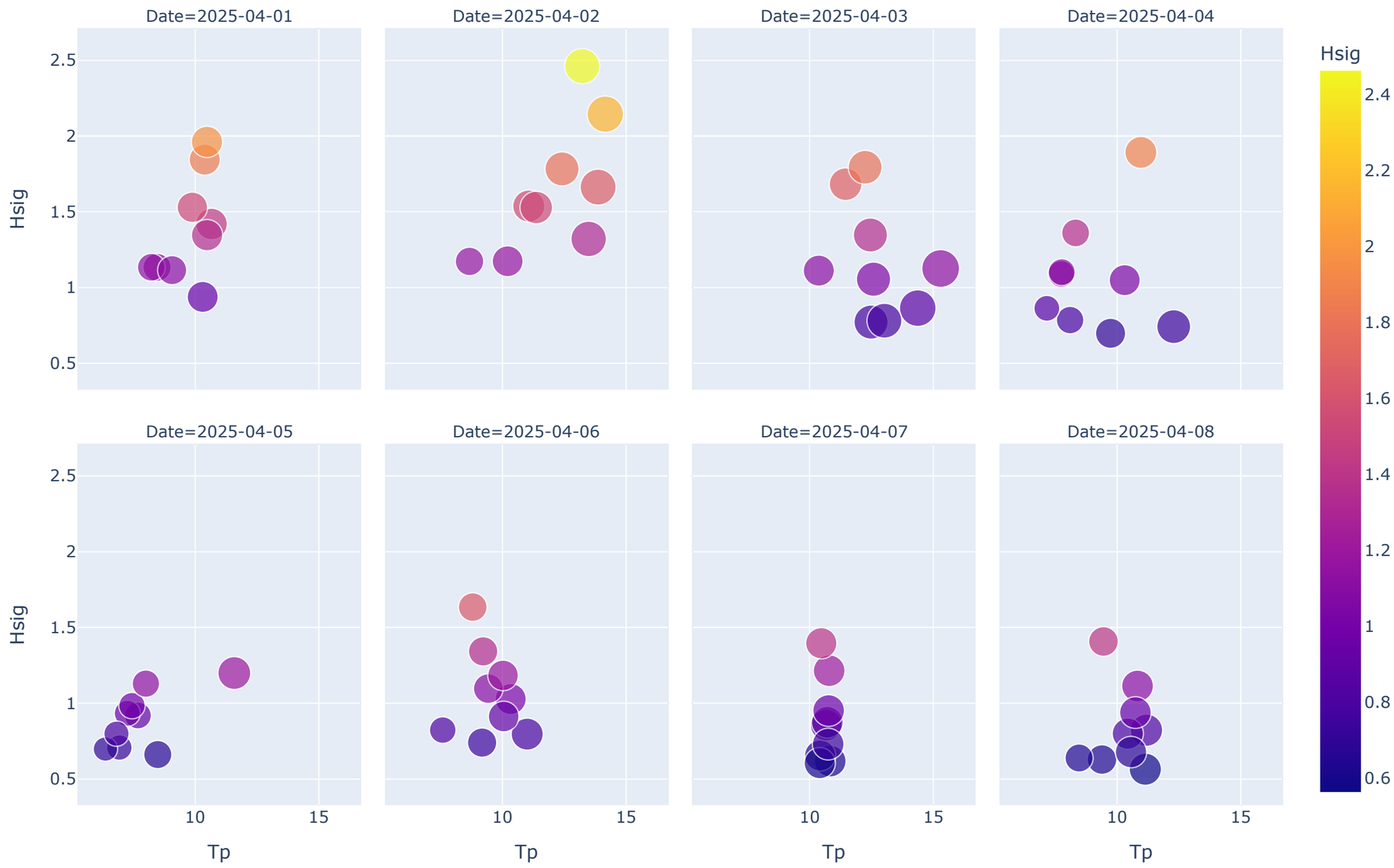
To support the analysis, I plan to create a time series plot showing the relationship between `Hsig` and `Tp` per site, which may reveal patterns like high-energy swell events over time.

I created a scatter plot to show how wave conditions varied throughout the monitoring locations. The goal is to plot wave height (Hsig) and wave period (Tp) over time.

The code below shows the relative wave height and duration of wave periods by plotting `Hsig` on the Y-axis and colour while mapping `Tp` to the X-axis and marker size. High-energy wave occurrences can be identified visually by colour and size. Each point shows the daily mean readings from each monitoring site.

The use of a faceted layout, where each subplot represents wave conditions for a specific date, allows temporal comparison of wave energy and behaviour through out the week.

```
#pin point high risk location
wave_scatter = px.scatter(NRT_wave_SEQ_mean_df,
                          x="Tp",
                          y="Hsig",
                          size="Tp",
                          color="Hsig",
                          facet_col="Date",
                          facet_col_wrap=4,
                          height = 720,
                          width = 1080,
                          )
wave_scatter.update_traces(textposition='top center')
wave_scatter.show()
```



After generating the initial scatter plot, I noticed that the size of each point was barely noticeable due to the relatively small range of `Tp` values. To address this, I chose to normalise the `Tp` columns, ensuring that variations in wave period are clearly visible.

I also get the min-max value for `Hsig` in order to use it as a range in future visualisation.

```
# Normalize the 'Tp' values to a 0-1 range
min_Tp = NRT_wave_SEQ_mean_df['Tp'].min()
max_Tp = NRT_wave_SEQ_mean_df['Tp'].max()
NRT_wave_SEQ_mean_df['normalized_Tp'] = (NRT_wave_SEQ_mean_df['Tp'] - min_Tp) / (max_Tp - min_Tp)

# Normalize the 'Hsig' values to a 0-1 range
min_Hsig = NRT_wave_SEQ_mean_df['Hsig'].min()
max_Hsig = NRT_wave_SEQ_mean_df['Hsig'].max()
```

The scatter plot helps highlight sites experiencing the most powerful waves. To make it easier for users to identify the location with the strongest waves each day, I decided to add some custom labelling, showing the site with the strongest waves in each day.

To identify the site with the strongest waves, I asked ChatGPT for guidance on how to calculate wave strength using `Hsig` and `Tp` values.

Prompt Used:

> If I have Hsig and Tp data how can I calculate wave strength. Explain to me how those information are used in the industry and how to calculate.

Summary of Response

- **Wave Energy** ($E = (1/8) * \rho * g * Hsig^2$): Shows the amount of energy stored in the wave field at a given moment.
  - Good for understanding the potential impact on structures (like erosion force), not necessarily how much energy is transferred.
- **Wave Power** ($P \approx 0.49 * Hsig^2 * Tp$): The rate at which energy is being transported by the waves (energy flow per meter of wave crest).
  - Good for estimating how destructive or dynamic the wave environment is over time (like overtopping risk or potential for wave energy extraction).

> Wave Energy is useful for estimating potential impact on coastal structures, while Wave Power is more relevant for evaluating processes like erosion or overtopping. Depending on the goal, these metrics can be used separately or combined into a custom wave strength index for a more comprehensive analysis.

I decided to use a combined method that integrates `Wave Energy` and `Wave Power` to create a custom index, which would help identify the site with the strongest waves each day.

The code for this implementation was provided by ChatGPT.

```
#Calculate Wave Energy and Wave Power
# Constants
g = 9.81  # gravity (m/s²)
rho = 1025  # density of seawater (kg/m³)

# Energy per unit area
NRT_wave_SEQ_mean_df["Wave_Energy"] = (1/8) * rho * g * NRT_wave_SEQ_mean_df["Hsig"]**2

# Wave power per unit crest length (approximate)
NRT_wave_SEQ_mean_df["Wave_Power"] = 0.49 * (NRT_wave_SEQ_mean_df["Hsig"]**2) *NRT_wave_SEQ_mean_df["Tp"]

# Create a new column "text_label" that is blank by default.
NRT_wave_SEQ_mean_df['text_label'] = ""

#Create a custom index
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
NRT_wave_SEQ_mean_df[["Wave_Energy_scaled", "Wave_Power_scaled"]] = scaler.fit_transform(NRT_wave_SEQ_mean_df[["Wave_Energy", "Wave_Power"]])
NRT_wave_SEQ_mean_df["Wave_Strength_Index"] = (NRT_wave_SEQ_mean_df["Wave_Energy_scaled"] + NRT_wave_SEQ_mean_df["Wave_Power_scaled"]) / 2

#Label the site with the highest index for each date
for date, group in NRT_wave_SEQ_mean_df.groupby("Date"):
    idx = group["Wave_Strength_Index"].idxmax()  # Get the index of the row with the max index
    NRT_wave_SEQ_mean_df.loc[idx, "text_label"] = NRT_wave_SEQ_mean_df.loc[idx, "Site"]
```

This is the final version of the scatter plot, designed to visualize the sites with the strongest waves each day.

In this plot, each point represents a monitoring site, with the x-axis and size showing the peak wave period (`Tp`) and the y-axis and colour representing the significant wave height (`Hsig`). To highlight the locations with the highest wave index, I added a custom `text_label` to annotate these sites' names.

I adjusted the colour scale from the default to `Spectral_r` because the range of colours provides a clearer visualisation of wave intensity. The red shades represent higher wave heights, signalling potentially strong waves in the areas that may require attention, while the blue hues represent lower wave heights, indicating calmer conditions.

Moreover, according to the explanation from ChatGPT, the size of the plot reflects the `Tp` values, which, when combined with wave height, can indicate if the wave is stronger or not. A longer `Tp` along with a higher `Hsig` may suggest that the waves are influenced by a storm occurring far from the shore. This is because waves generated by distant storms tend to travel long distances with relatively consistent energy. As they move away from the storm source, shorter-period waves lose energy and disappear, while longer-period waves continue to travel, arriving at the shore with higher wave heights and longer periods. This pattern is different from locally generated wind waves, which typically have shorter wave periods and are more irregular.

Therefore, when both `Tp` and `Hsig` are high, it's a strong signal of a powerful offshore weather event affecting the coastline. Being able to detect this combination can help with early warning systems, coastal planning, and safety.

The visualization, with colour showing `Hsig` and size representing `Tp` allows users to quickly identify these conditions. The plot also includes facets for each day (`Date`), allowing for clear daily comparisons.

The following code generates this final version of the scatter plot.

```
#Create scatter plot with the name of the site with strongest wave (highest wave index)
wave_scatter = px.scatter(NRT_wave_SEQ_mean_df,
                          x="Tp",
                          y="Hsig",
                          range_y = [0, max_Hsig+0.5], #Equal to normalized
                          size="normalized_Tp",
                          color="Hsig",
                          facet_col="Date",
                          color_continuous_scale="Spectral_r",
                          facet_col_wrap=4,
                          height = 720,
                          width = 1080,
                          text="text_label") #Pin point high risk location

wave_scatter.update_traces(textposition='top center')
wave_scatter.show()
```

To explore more ways of visualising the data, I created this chart, which follows the same concept as the previous one but presents the information from a different perspective.

This chart shows the wave height (`Hsig`) for each site over time, with the x-axis representing the `Date` and the y-axis representing wave height (`Hsig`). The size of the points corresponds to the normalised wave period (`Tp`), and the colour indicates the magnitude of wave height.

Additionally, I added a line representing the *Average Wave Height* across all sites for each day as a benchmark to help identify sites with wave heights exceeding the average.

To further enhance the visualisation, I explored the **Animation Feature** in `Plotly`. After reviewing examples on the Plotly Website, I decided to incorporate animation into this chart, which allows users to see how wave heights change over time for each site.

```
min_Date = NRT_wave_SEQ_mean_df['Date'].min()
max_Date = NRT_wave_SEQ_mean_df['Date'].max()

sites = sorted(NRT_wave_SEQ_mean_df['Site'].unique()) #Sorted by site name

import plotly.graph_objects as go

# Create the animated faceted scatter plot.
fig = px.scatter(
    NRT_wave_SEQ_mean_df,
    x='Date',
    y="Hsig",
    range_x=[min_Date, max_Date],
    range_y=[0, NRT_wave_SEQ_mean_df['Hsig'].max() + 0.5],
    color="Hsig",
    size="normalized_Tp",
    size_max=30,
    color_continuous_scale="Spectral_r",
    range_color=[NRT_wave_SEQ_mean_df['Hsig'].min(), NRT_wave_SEQ_mean_df['Hsig'].max()],
    facet_col='Site',
    facet_col_wrap=3,
    animation_frame="Date",
    title="Daily Wave Height & Wave Period (Wave Strength) by Location"
)

# Precompute the global average Hsig by Date.
df_avg = NRT_wave_SEQ_mean_df.groupby("Date", as_index=False)["Hsig"].mean()
df_avg = df_avg.sort_values("Date")
```

```
facet_mapping = {}
for i, site in enumerate(sites):
    xaxis = "x" if i == 0 else f"x{i+1}"
    yaxis = "y" if i == 0 else f"y{i+1}"
    facet_mapping[site] = (xaxis, yaxis)

# Helper function to create the average line trace for a given facet.
def create_avg_line_trace(site):
    xaxis_name, yaxis_name = facet_mapping.get(site, ("x", "y"))
    return go.Scatter(
        x=df_avg['Date'],
        y=df_avg['Hsig'],
        mode='lines',
        name='Avg Hsig',
        line=dict(color='grey', width=2),
        xaxis=xaxis_name,
        yaxis=yaxis_name,
        showlegend=False,
        hoverinfo='skip'
    )

# Add the average trace to the base figure for each facet.
for site in sites:
    fig.add_trace(create_avg_line_trace(site))

# Add copies of the average trace to every animation frame.
for frame in fig.frames:
    current_traces = list(frame.data)
    additional_traces = [create_avg_line_trace(site) for site in sites]
    frame.data = tuple(current_traces + additional_traces)
```

```
fig.update_layout(height=800,width = 960)
fig.show()
```

I then explored visualising the data on a map, which I believe is especially helpful for users who may not be familiar with the site names.

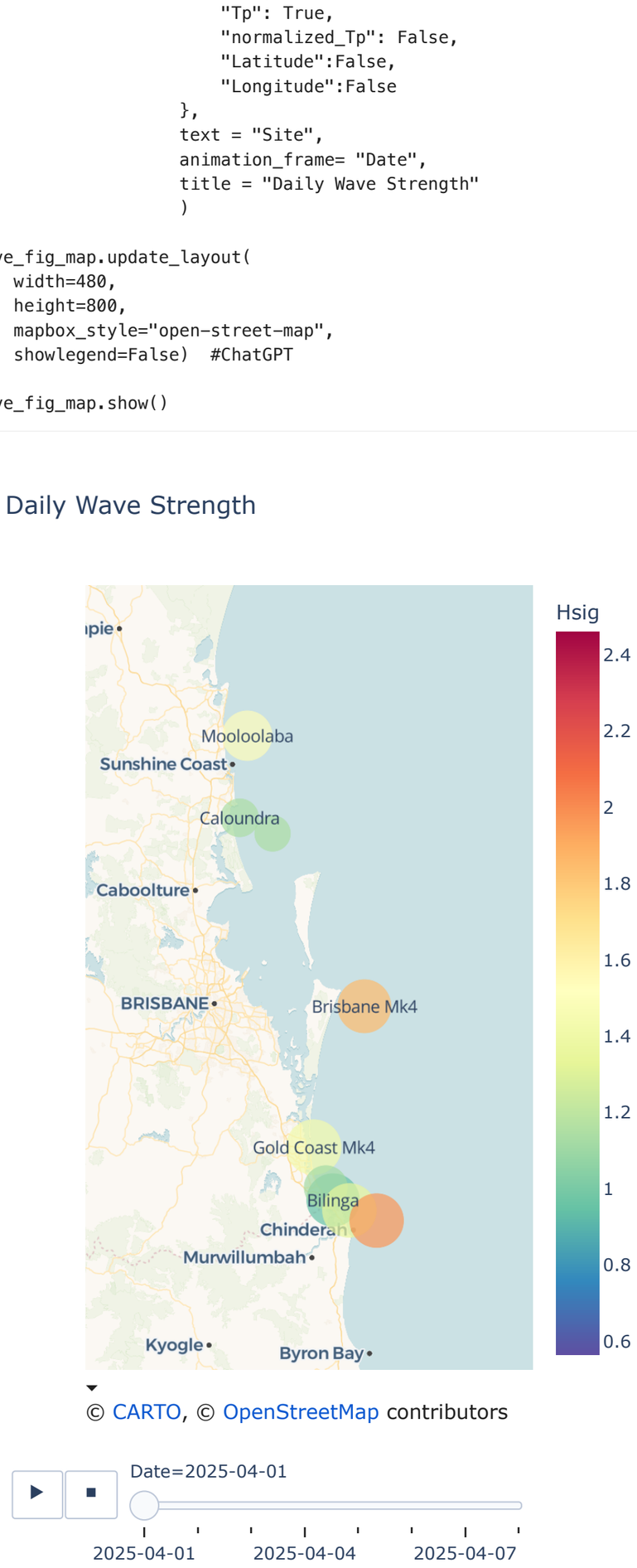
Unlike a list of names, a map provides geographic context, allowing us to see each site's physical location and surrounding characteristics. This view can reveal patterns such as the influence of nearby sites on one another or whether a site is located close to densely populated areas or further offshore.

These insights are valuable for decision-making, particularly in infrastructure planning. By identifying high-risk sites and understanding their surrounding environments, such as urban areas versus protected zones like national parks, we can better prioritize and allocate resources based on the potential impact in each region.

For this part, I used the code from the tutorial together with example code from Plotly Website and made a few refinements, such as adding **Animation feature** and customising the **Hover Details**, based on suggestions from ChatGPT.

```
# finding the central point
wave_lat_centre = (max(NRT_wave_SEQ_mean_df['Latitude']) + min(NRT_wave_SEQ_mean_df['Latitude']))/2
wave_lon_centre = (max(NRT_wave_SEQ_mean_df['Longitude']) + min(NRT_wave_SEQ_mean_df['Longitude']))/2
```

```
#from https://plotly.com/python/plotly-express/
wave_fig_map = px.scatter_map(NRT_wave_SEQ_mean_df, lat="Latitude", lon="Longitude",
                    color = "Hsig",
                    size = "normalized_Tp", #Support Hsig > if red and big then that wave is very strong
                    size_max = 40, zoom = 7,
                    center = {'lat':wave_lat_centre, 'lon': wave_lon_centre},
                    range_color = [min_Hsig, max_Hsig],
                    color_continuous_scale="Spectral_r", #ChatGPT >> Change because red represent high wave which tends to be strong wave better
                    hover_name = "Site",  # Primary tooltip text
                    hover_data = {
                        "Site": False,
                        "Date": True,
                        "Hsig": True,
                        "Tp": True,
                        "normalized_Tp": False,
                        "Latitude":False,
                        "Longitude":False
                    },
                    text = "Site",
                    animation_frame= "Date",
                    title = "Daily Wave Strength"
                    )

wave_fig_map.update_layout(
    width=480,
    height=800,
    mapbox_style="open-street-map",
    showlegend=False)  #ChatGPT

wave_fig_map.show()
```



4.2 Storm Tide Data

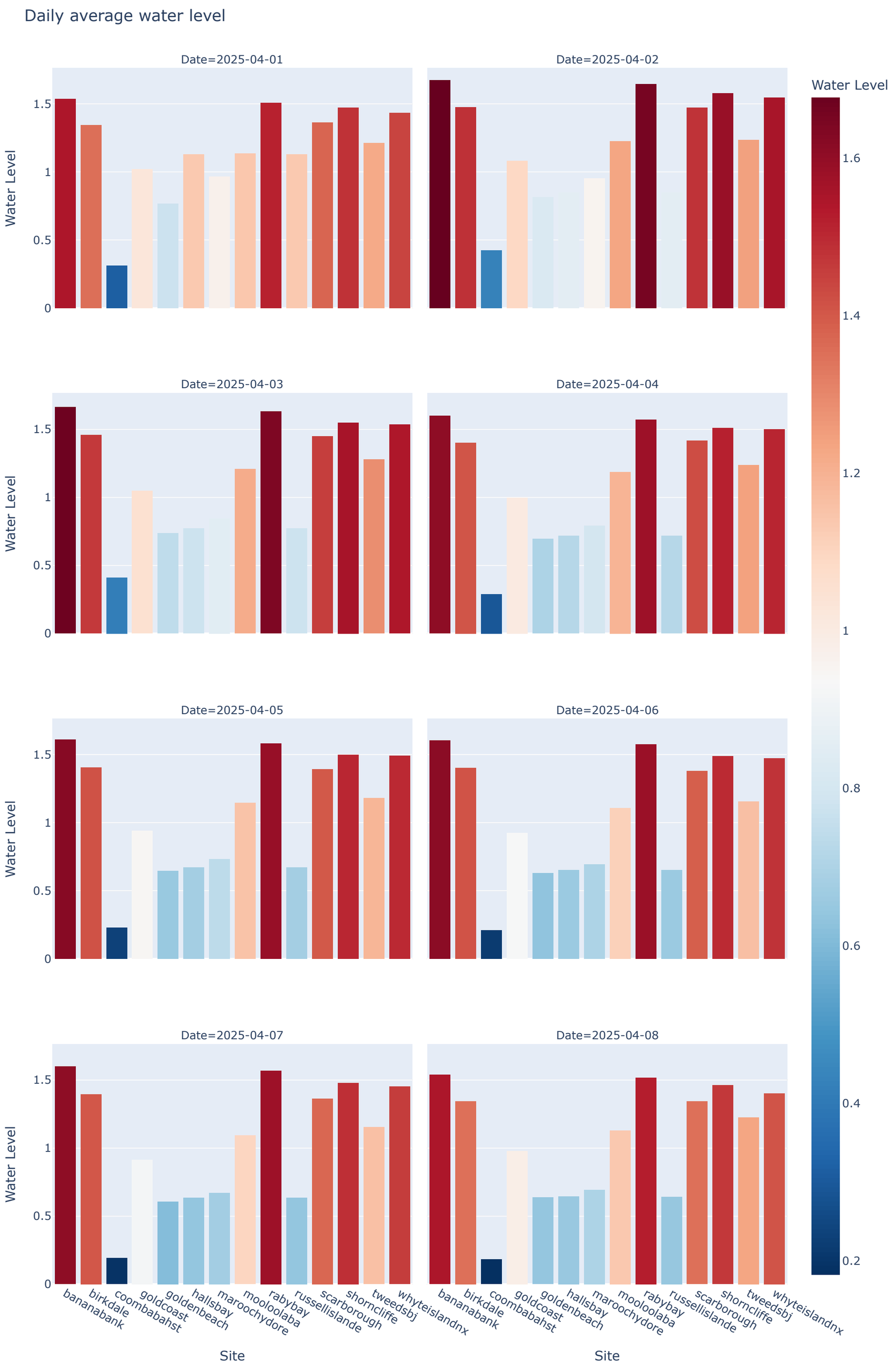
To support the tide analysis, I plan to create a bar chart showing `Water Level` at each site.

The code below plots the `Water Level` on the Y-axis, with each bar representing the daily mean water level at each site. This allows for easy comparison of water levels across locations, highlighting sites with higher water levels, which could indicate higher risks for flooding.

This bar chart allows us to quickly identify differences in water levels across the sites, facilitating better decision-making for flood risk management. Each bar represents the average daily reading, allowing for comparisons across the sites.

For the `Storm Tide Data`, I decided to customise the colour scale for the same reason I did in the wave section visualisation. Water levels can indicate flood risk, and red and blue colours can effectively convey a sense of urgency. Therefore, I chose the `RdBu_r` colour scale to reflect this.

```
NRT_tide_SEQ_mean_df_bar = px.bar(NRT_tide_SEQ_mean_df, x="Site", y="Water Level",
                                  labels={"Water Level (Meter)"},
                                  color = "Water Level",
                                  title= "Daily average water level",
                                  facet_col="Date",
                                  facet_col_wrap=2,
                                  color_continuous_scale="RdBu_r",
                                  height = 1500,
                                  width = 1000,
                                  #text="text_label"
                                 )
NRT_tide_SEQ_mean_df_bar.update_traces(marker_line_width=0)
NRT_tide_SEQ_mean_df_bar.show()
```

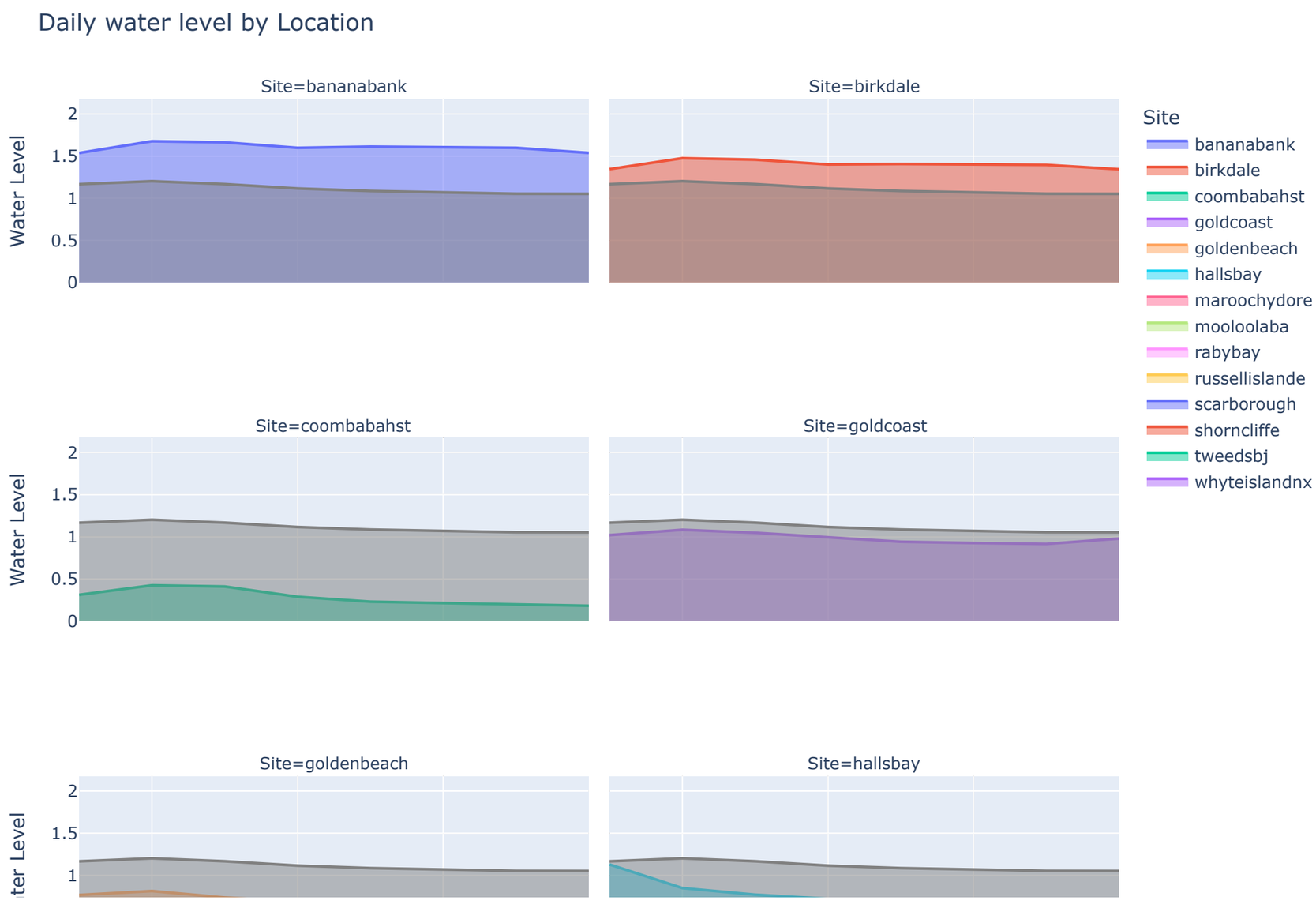


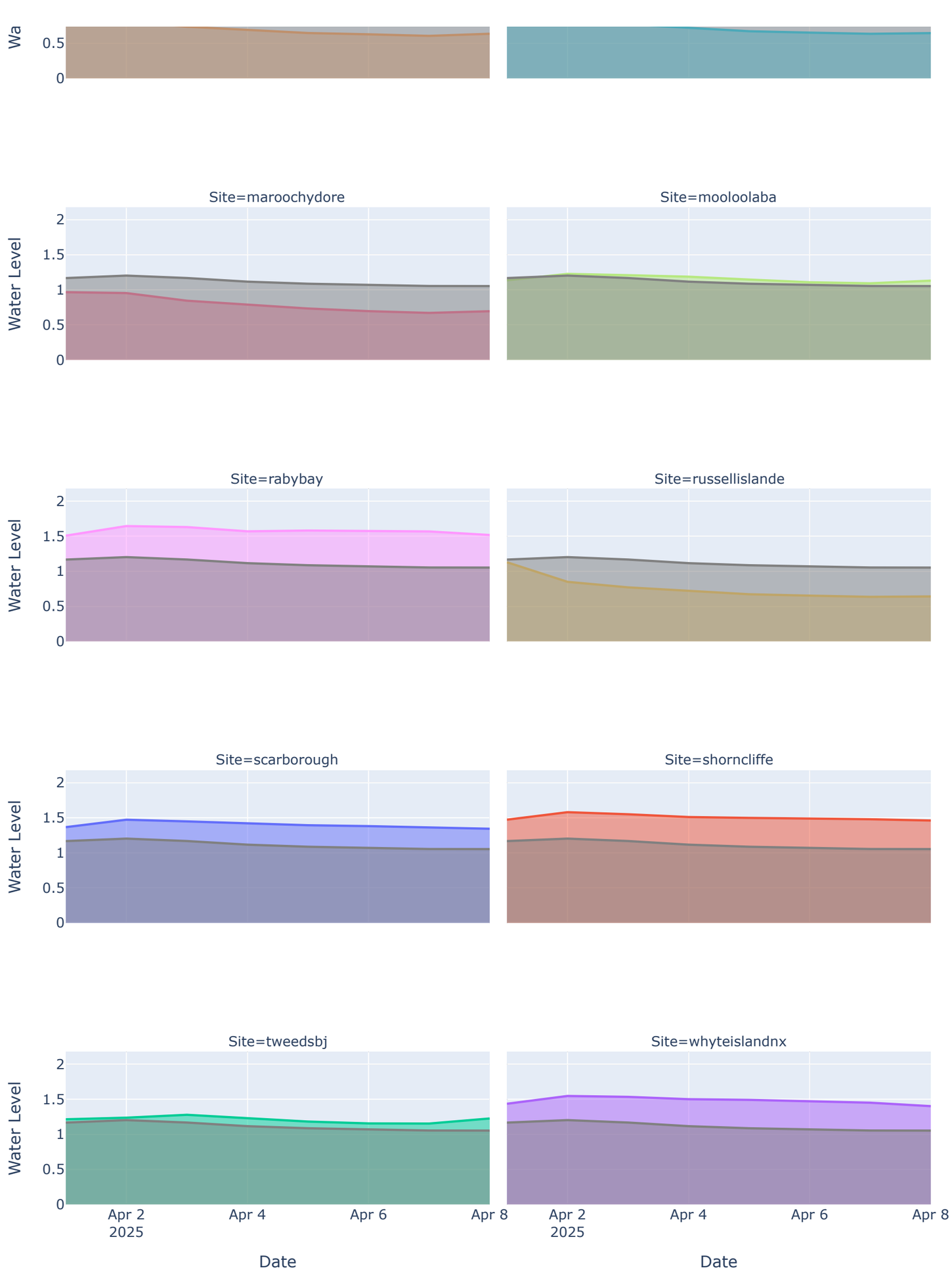
Using the same concept as with the `Wave Data`, I created a chart that provides a different perspective.

This area chart visualizes the water level at each site over time, with the x-axis representing the `Date` and the y-axis showing the `Water Level`. The area under each graph reflects the water level, making it easier to compare levels across locations. A benchmark line for the **Average Water Level** across all sites helps users identify sites with water levels above the average, highlighting those at higher risk. This chart gives a clear view of how water levels fluctuate through time.

I chose an area chart because it effectively shows how water levels change over time, with the filled areas offering an intuitive comparison. It resembles water filling a jug, helping to visualise the differences. Adding a benchmark line makes it easy to spot sites with higher-than-average water levels, making it easier to assess risk and guide decisions.

```
min_Date = NRT_tide_SEQ_mean_df['Date'].min()
max_Date = NRT_tide_SEQ_mean_df['Date'].max()

sites = sorted(NRT_tide_SEQ_mean_df['Site'].unique())

import plotly.graph_objects as go

# Create the animated faceted scatter plot.
fig = px.area(
    NRT_tide_SEQ_mean_df,
    x='Date',
    y="Water Level",
    range_x=[min_Date, max_Date],
    range_y=[0, NRT_tide_SEQ_mean_df['Water Level'].max() + 0.5],
    color="Site",
    #color_continuous_scale="RdBu_r",
    #range_color=[NRT_tide_SEQ_mean_df['Water Level'].min(), NRT_tide_SEQ_mean_df['Water Level'].max()],
    facet_col='Site',
    facet_col_wrap=2,
    #animation_frame="Date",
    title="Daily water level by Location"
)

# Precompute the global average Hsig by Date.
df_avg = NRT_tide_SEQ_mean_df.groupby("Date", as_index=False)["Water Level"].mean()
df_avg = df_avg.sort_values("Date")

# Build a manual facet mapping.
# Here we assume that the order of facets in the layout matches the order in 'sites'.
facet_mapping = {}
for i, site in enumerate(sites):
    xaxis = "x" if i == 0 else f"x{i+1}"
    yaxis = "y" if i == 0 else f"y{i+1}"
    facet_mapping[site] = (xaxis, yaxis)

# Helper function to create the average line trace for a given facet.
def create_avg_line_trace(site):
    xaxis_name, yaxis_name = facet_mapping.get(site, ("x", "y"))
    return go.Scatter(
        x=df_avg['Date'],
        y=df_avg['Water Level'],
        mode='lines',
        fill = 'tozeroy',
        name='Avg Water Level',
        line=dict(color='grey', width=2),
        xaxis=xaxis_name,
        yaxis=yaxis_name,
        showlegend=False,
        hoverinfo='skip'
    )

# Add the average trace to the base figure for each facet.
for site in sites:
    fig.add_trace(create_avg_line_trace(site))

# Add copies of the average trace to every animation frame.
for frame in fig.frames:
    current_traces = list(frame.data)
    additional_traces = [create_avg_line_trace(site) for site in sites]
    frame.data = tuple(current_traces + additional_traces)

fig.update_layout(height=1800, width = 1000)
fig.show()
```





With the same thought process when visualising the `Wave Data`, the map format allows for a better risk assessment.

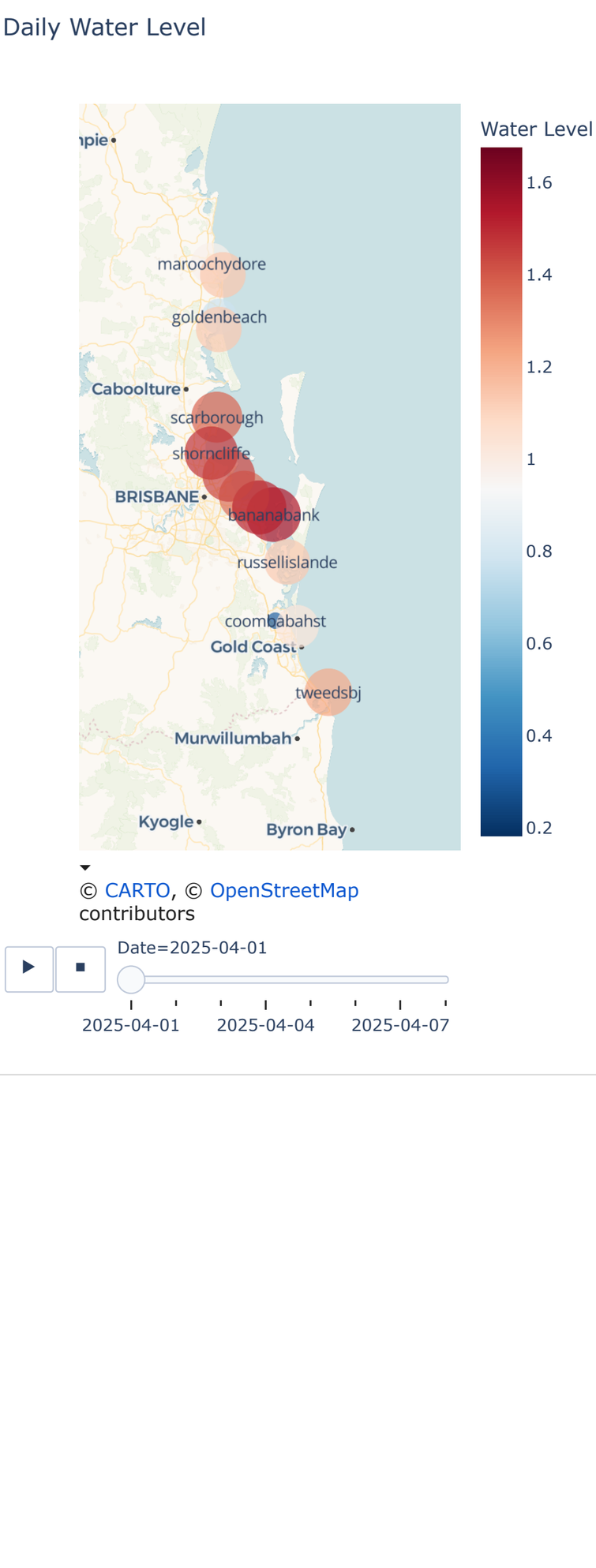
Similar to the `Wave Data`, the map provides insight into the physical location and surrounding characteristics of each site, highlighting patterns such as the proximity to densely populated areas or nearby sites that could influence conditions. These insights are essential for making informed decisions, particularly when prioritising areas for infrastructure development based on flood risk.

For this visualisation, I used the same code structure as the map for `Wave Data`, which I adjusted to accommodate `Storm Tide Data` instead.

For the size of the plot, with the same idea for `Tp` in `Wave Data`, I normalised `Water Level` to be used as a size indicator so it is easier to detect.

```
#Normalised water level to user as size
# Normalize the 'Tp' values to a 0-1 range
min_WL = NRT_tide_SEQ_mean_df['Water Level'].min()
max_WL = NRT_tide_SEQ_mean_df['Water Level'].max()
NRT_tide_SEQ_mean_df['normalized_WL'] = (NRT_tide_SEQ_mean_df['Water Level'] - min_WL) / (max_WL - min_WL)
```

```
# finding the central point
tide_lat_centre = (max(NRT_tide_SEQ_mean_df['Latitude']) + min(NRT_tide_SEQ_mean_df['Latitude']))/2
tide_lon_centre = (max(NRT_tide_SEQ_mean_df['Longitude']) + min(NRT_tide_SEQ_mean_df['Longitude']))/2

min_WL = NRT_tide_SEQ_mean_df['Water Level'].min()
max_WL = NRT_tide_SEQ_mean_df['Water Level'].max()

#from https://plotly.com/python/plotly-express/
tide_fig_map = px.scatter_map(NRT_tide_SEQ_mean_df, lat="Latitude", lon="Longitude",
                    color = "Water Level",
                    size = "normalized_WL", #Support Hsig > if red and big then that wave is very strong
                    size_max = 30, zoom = 7,
                    center = {'lat':tide_lat_centre, 'lon': tide_lon_centre},
                    range_color = [min_WL, max_WL],
                    color_continuous_scale="RdBu_r", #ChatGPT >> Change because red represent high wave which tends to be strong wave better
                    hover_name = "Site",  # Primary tooltip text
                    hover_data = {
                        "Site": False,
                        "Date": True,
                        "Water Level":True,
                        "Latitude":False,
                        "Longitude":False
                    },
                    text = "Site",
                    animation_frame= "Date",
                    title = "Daily Water Level"
                    )

tide_fig_map.update_layout(
    width=480,
    height=800,
    mapbox_style="open-street-map",
    showlegend=True)  #ChatGPT

tide_fig_map.show()
```



5.Summary Insight

I look through the analysis and visualisations and connect them to the initial question in order to gain insights:

> **Question**: How do near real-time wave height and tide level data vary at major coastal locations in South East Queensland, and what patterns can be observed through continuous monitoring? How can this information help with flood warnings and infrastructure planning?

Findings:

Wave height and tide data vary by site and time, with clear patterns linked to the geographical location and surrounding environmental features

- **Stronger Waves** (`Hsig` x `Tp`) tend to occur at sites further from the shore or those facing the open ocean, indicating an increased risk for coastal impacts like erosion and flooding.
- **High tide levels** (`Water Level`) are more common near river mouths or locations with complex waterway systems, which are often in proximity to neighbourhoods and vulnerable infrastructure.

The most important finding is that areas with consistently strong waves and/or high tides can be classified as high-risk areas for coastal erosion and flooding.

This insight can help prioritise where coastal and/or flood protection infrastructure should be developed first according to urgency and estimated impact, which will help define budget allocation and help with the infrastructure design based on site-specific conditions.

Further Analysis:

While working on this assignment, I noticed that wave height and water level patterns might be influenced by daily weather conditions. For instance, on 29th March, there was a noticeable increase in wave activity during rain events, which aligns with weather reports from 29th – 30th March indicating rainfall. This suggests that integrating weather data can add important context to wave and tide analysis (Screenshots of a graph and weather observation from BOM Website below).

Incorporating weather forecast data could improve the ability to predict wave and tide conditions, especially during extreme weather events. This would enhance flood and erosion risk assessments in real time.

Additionally, exploring seasonal trends and incorporating climate change projections would strengthen long-term planning and provide a more comprehensive understanding of coastal risks.

Data Limitations and How to Address Them:

- Near real-time data provides timely updates but is limited to showing only the last 7 days, lacking the broader historical context needed to understand long-term trends. This limitation can be addressed by integrating historical datasets to identify recurring patterns and trends over time.
- The different site locations of these datasets prevent them from being fully combined. Initially, I aimed to create a `scatter_map` to show both datasets simultaneously, allowing users to distinguish between `Storm Tide Data` and `Wave Data`. However, `Plotly Express`'s scatter_map feature does not support using different scatter shapes for two variables. As a result, the data needs to be displayed on separate maps.
- Currently, the best approach is to combine all charts into a single dashboard, where users can view and analyze the charts together to monitor the overall coastal situation.

Applications and Recommendations:

- **Near Real-Time Monitoring Dashboard:** Facilitates continuous monitoring and timely flood warnings for high-risk coastal areas.
- **Disaster Management & Early Warning Systems:** Supports better preparation and response during coastal hazard events.
- **Coastal Preservation Infrastructure Planning:** Helps prioritise resource allocation in areas vulnerable to flooding or erosion.

Key Recommendations for Decision-Makers:

- Prioritize **offshore-facing sites** for *erosion control*.
- Monitor high **tide-prone sites** near rivers for *flood prevention*.
- *Focus resources* on **densely populated areas with high risk**.