Royal Caribbean International WTD Analytics & Trend Tracker

A comprehensive analysis of booking patterns and revenue optimization strategies based on Week-To-Date (WTD) performance metrics.

JL

Jiaman Li

June 19, 2025

Table of Contents

PART I

PART II

PART I Objectives

To support smarter revenue management, we need to understand how bookings build week-to-date and compare that to realistic targets.

What is the expected WTD ratio for each product?

How do bookings build by weekday and sailing period?

How can these insights help managers make timely pricing or inventory decisions?

PART I Tools & Methodology

Step 1: Exploratory Data Analysis

Clean and analyze historical booking data using Python(pandas, seaborn, matplotlib) tools to identify patterns.

Step 2: WTD Analytics & Trend Tracker Dashboard

Build interactive visualizations to track real-time booking performance.

Step 3: Business Analytics & Insights

Derive actionable revenue optimization strategies from the data.

Step 1: Exploratory Data Analysis (EDA)

Tools Used:

Python ecosystem with pandas, matplotlib.pyplot, and seaborn for data manipulation and visualization.

In [12]:

```
# Correct data format
df['sail_date'] = pd.to_datetime(df['sail_date'])
df['meta_product_code'] = df['meta_product_code'].astype('category')

# Add 'weekday' column based on run_date
df['weekday'] = df['run_date'].dt.day_name()

# Total records by weekday
weekday_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday
weekday_counts = df['weekday'].value_counts().reindex(weekday_order, fill_value=0)
print(weekday_counts)
```

```
Monday       16251
Tuesday      16251
Wednesday    16251
Thursday     16252
Friday       16251
Saturday         0
Sunday       13963
Name: weekday, dtype: int64
```

Key Step:

Added 'weekday' column based on run_date to identify daily patterns.

In [14]:

```
# Show cleaned data
df_clean.head(10)
```

Out[14]:

	run_date	calendar_week	ship	sail_date	sail_year	sail_month	meta_product_code	pax_build	weekday
1	2024-12-08	2024-12-13	SY	2025-03-02 00:00:00+00:00	2025	3	7N CARIBBEAN	0.525550	Sunday
2	2024-12-08	2024-12-13	UT	2025-07-18 00:00:00+00:00	2025	7	SHORT CARIBBEAN	3.986292	Sunday
3	2024-12-08	2024-12-13	EN	2025-09-04 00:00:00+00:00	2025	9	SHORT CARIBBEAN	1.099793	Sunday
4	2024-12-08	2024-12-13	IC	2027-04-10 00:00:00+00:00	2027	4	7N CARIBBEAN	0.730949	Sunday
6	2024-12-08	2024-12-13	UT	2024-12-30 00:00:00+00:00	2024	12	SHORT CARIBBEAN	4.608766	Sunday
7	2024-12-08	2024-12-13	ID	2025-07-20 00:00:00+00:00	2025	7	EUROPE	6.258185	Sunday
8	2024-12-08	2024-12-13	UT	2025-09-05 00:00:00+00:00	2025	9	SHORT CARIBBEAN	-1.875163	Sunday
9	2024-12-08	2024-12-13	RH	2024-12-15 00:00:00+00:00	2024	12	7N CARIBBEAN	2.853934	Sunday
10	2024-12-08	2024-12-13	HM	2025-06-15 00:00:00+00:00	2025	6	7N CARIBBEAN	23.430896	Sunday
11	2024-12-08	2024-12-13	EN	2026-01-31 00:00:00+00:00	2026	2	SHORT CARIBBEAN	0.194890	Sunday

Note: Saturday shown as 0 due to no run activity captured on that day.

Step 1: Exploratory Data Analysis (EDA)

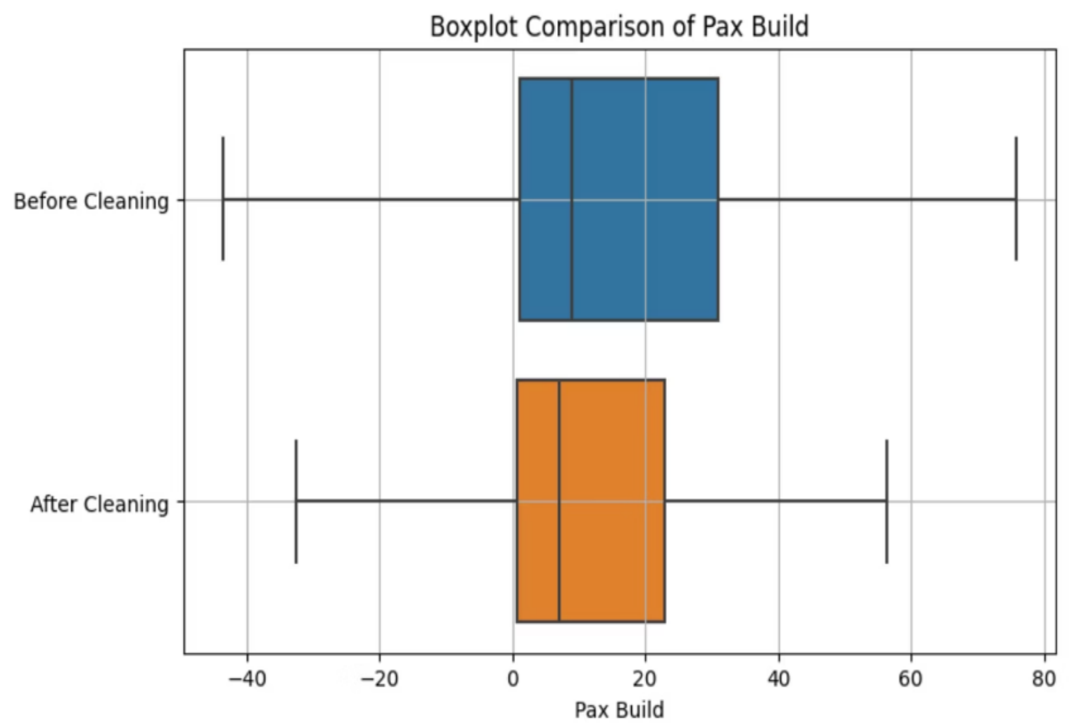
Key Step:

Extreme outliers in passenger build were identified and removed using Interquartile Range (IQR) method.

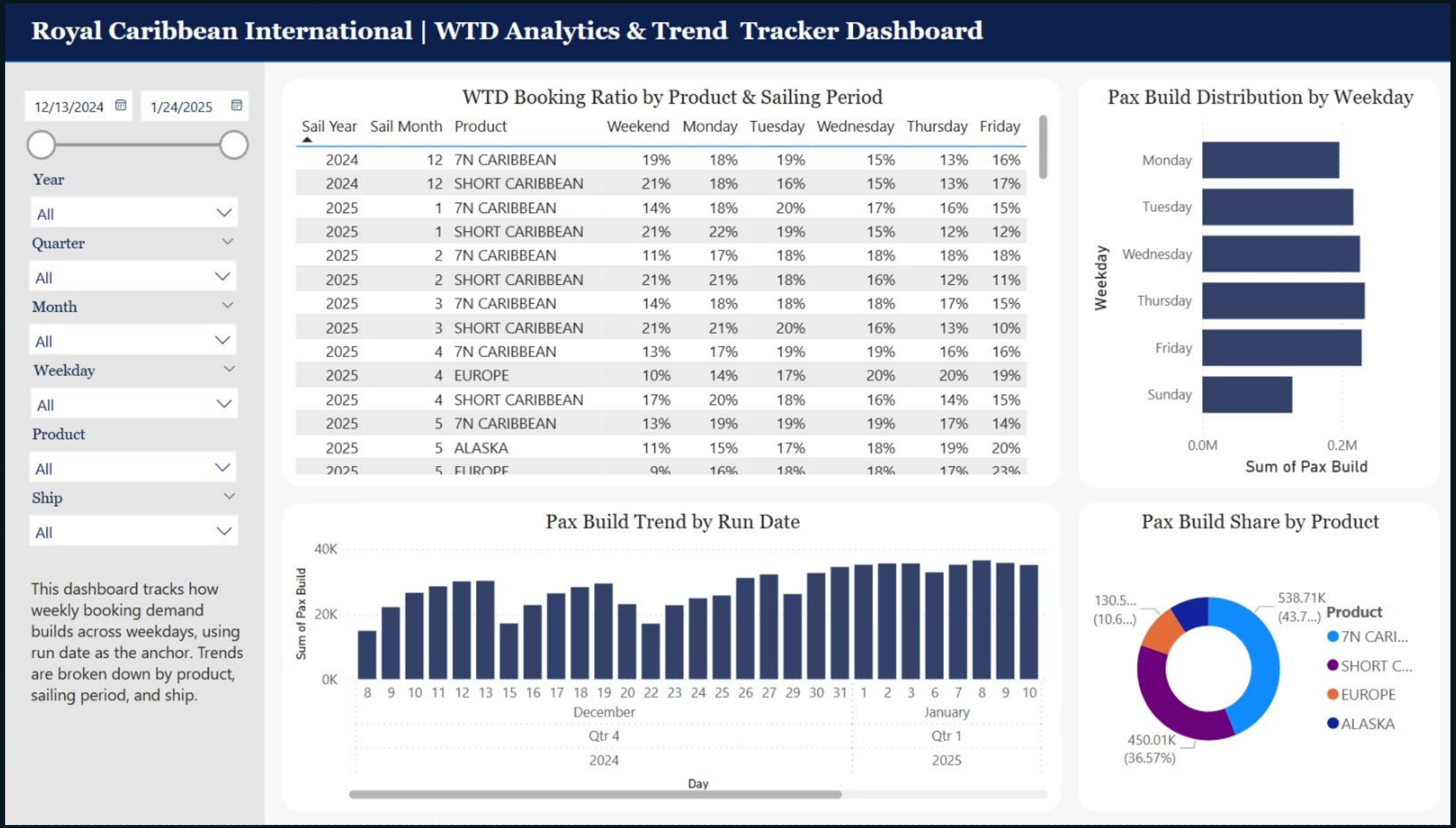
This improved data quality and ensured more accurate trend analysis.

```
In [8]: # Remove outliers from 'pax_build' using the IQR method
        Q1 = df['pax_build'].quantile(0.25)
        Q3 = df['pax_build'].quantile(0.75)
        IQR = Q3 - Q1

        # Filter the dataframe to exclude outliers
        df_clean = df[(df['pax_build'] >= Q1 - 1.5 * IQR) & (df['pax_build'] <= Q3 + 1.5 * IQR)]
        original_count = len(df)
        cleaned_count = len(df_clean)
        removed_count = original_count - cleaned_count
        removed_count_ratio = removed_count/original_count*100

        print("Outlier Removal Summary:")
        print(f"Total records before cleaning: {original_count}")
        print(f"Total records after cleaning:  {cleaned_count}")
        print(f"Total outliers removed:        {removed_count}")
        print(f"Total outliers %:        {removed_count_ratio:.2f}%")

        Outlier Removal Summary:
        Total records before cleaning: 95219
        Total records after cleaning:  86819
        Total outliers removed:        8400
        Total outliers %:        8.82%
```

```
In [10]: # Boxplot comparison of pax_build before and after cleanning
         plt.figure(figsize=(8, 5))
         sns.boxplot(data=[df['pax_build'], df_clean['pax_build']], orient='h', showfliers=False)
         plt.yticks([0, 1], ['Before Cleaning', 'After Cleaning'])
         plt.title('Boxplot Comparison of Pax Build')
         plt.xlabel('Pax Build')
         plt.grid(True)
         plt.show()
```



Step 2: WTD Analytics & Trend Tracker Dashboard



Interactive Visualization

Real-time tracking of booking performance against WTD targets across products.

Temporal Analysis

Daily and weekly booking patterns displayed for strategic pricing decisions.

Product Segmentation

Separate tracking for different cruise types to identify product-specific trends.

Step 3: Business Analytics & Insights

Power BI

Insight I - Risk Impact of Inaccurate WTD Expectations

Back to report

WTD BOOKING RATIO BY PRODUCT & SAILING PERIOD

Sail Year	Sail Month	Product	Weekend	Monday	Tuesday	Wednesday	Thursday	Friday
2024	12	7N CARIBBEAN	19%	18%	19%	15%	13%	16%
2024	12	SHORT CARIBBEAN	21%	18%	16%	15%	13%	17%
2025	1	7N CARIBBEAN	14%	18%	20%	17%	16%	15%
2025	1	SHORT CARIBBEAN	21%	22%	19%	15%	12%	12%
2025	2	7N CARIBBEAN	11%	17%	18%	18%	18%	18%
2025	2	SHORT CARIBBEAN	21%	21%	18%	16%	12%	11%
2025	3	7N CARIBBEAN	14%	18%	18%	18%	17%	15%
2025	3	SHORT CARIBBEAN	21%	21%	20%	16%	13%	10%
2025	4	7N CARIBBEAN	13%	17%	19%	19%	16%	16%
2025	4	EUROPE	10%	14%	17%	20%	20%	19%
2025	4	SHORT CARIBBEAN	17%	20%	18%	16%	14%	15%
2025	5	7N CARIBBEAN	13%	19%	19%	19%	17%	14%
2025	5	ALASKA	11%	15%	17%	18%	19%	20%
2025	5	EUROPE	9%	16%	18%	18%	17%	23%

Overestimated WTD

Inventory held too long, resulting in last-minute discounting and missed revenue opportunities.

Underestimated WTD

Premature discounting leads to reduced revenue per passenger and margin erosion.

Recommendation

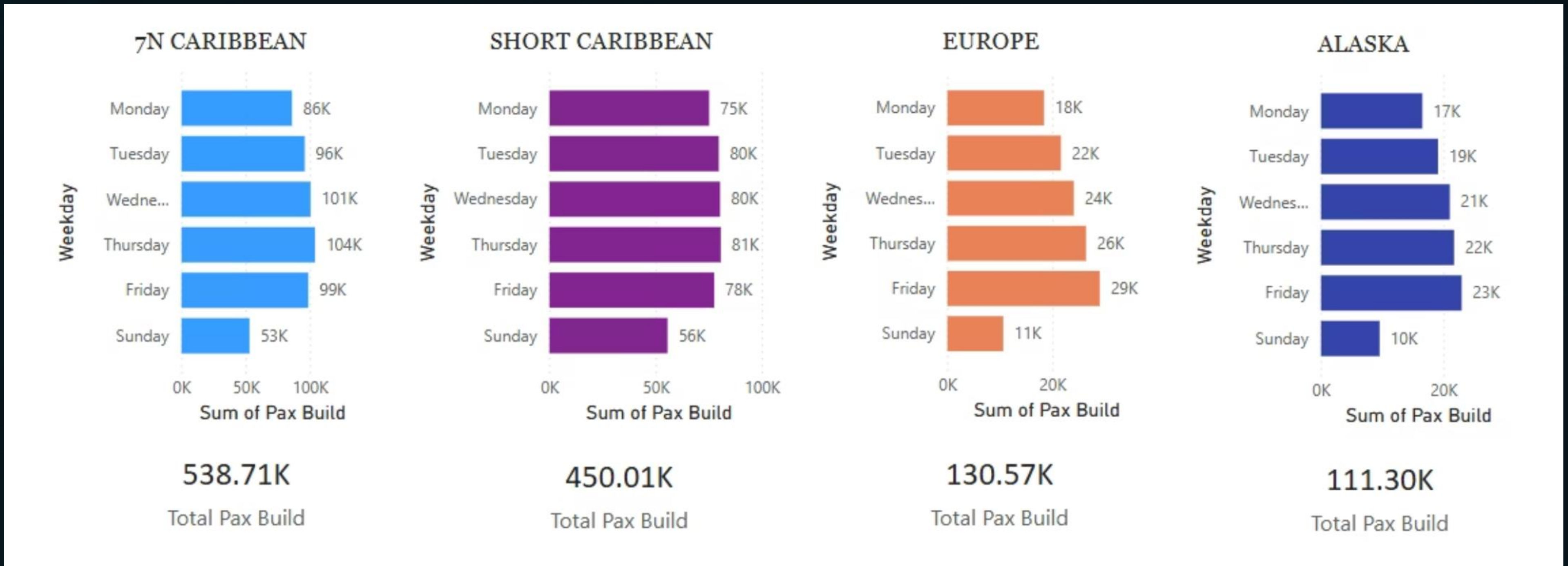
Use WTD benchmarks to guide pricing. Monitor pacing weekly to optimize fill rates and margins.

CSV

Click to download the 'WTD Booking Ratio by Product & Sailing Period.csv'

Overview: Total Pax Build varies significantly across products.

Insight II - Differences in weekday booking patterns reflect operational setups (e.g., RM cadence), market behaviors (e.g., EMEA time zones), and product characteristics (e.g., B2C vs B2B demand mix).



Caribbean products show strong midweek activity (Tues–Fri), reflecting higher B2C volume and structured Revenue Management workflows.

Europe builds gradually toward Friday, possibly due to EMEA time zone lag and regional consumer booking habits.

Alaska shows a flatter, low-volume trend, suggesting B2B-oriented demand or fewer sailing opportunities.

Sunday has the lowest activity across all products; Saturday data is structurally missing.

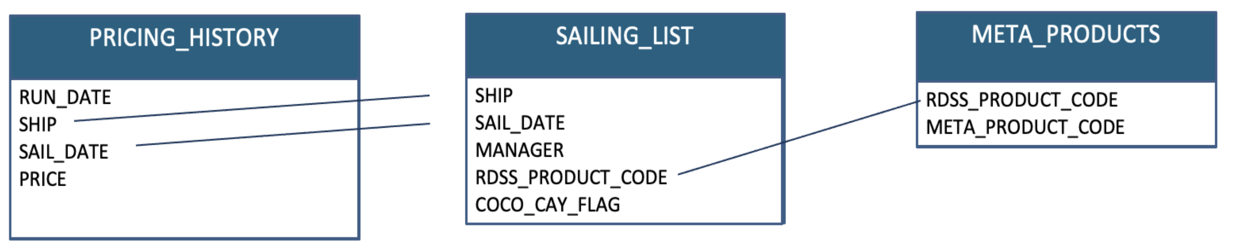
PART II Appendix: SQL Queries

1. Write a query to see how many Meta_Products and how many sailings each Manager has for future sailings

```sql
SELECT
  s.manager,
  COUNT(DISTINCT m.meta_product_code) AS meta_products,
  COUNT(*) AS sailings
FROM sailing_list s
JOIN meta_products m
  ON s.rdss_product_code = m.rdss_product_code
WHERE s.sail_date > CURRENT_DATE
GROUP BY s.manager
ORDER BY s.manager;
```

2. Write a query that shows the ship that had the most price changes

```sql
WITH price_change_log AS (
  SELECT
    ship_code,
    sail_date,
    run_date,
    price,
    LAG(price) OVER (
      PARTITION BY ship_code, sail_date
      ORDER BY run_date
    ) AS prev_price
  FROM pricing_history
),
price_changes AS (
  SELECT
    ship_code,
    COUNT(*) AS change_count
  FROM price_change_log
  WHERE price != prev_price
  GROUP BY ship_code
)
SELECT *
FROM price_changes
WHERE change_count = (
  SELECT MAX(change_count) FROM price_changes
)
ORDER BY ship_code;
```



PRICING_HISTORY SAMPLE

RUN_DAT	SHIP_COD	SAIL_DAT	PRICE
5/2/24	FE	8/16/25	$ 120
5/3/24	FE	8/16/25	$ 123
5/4/24	FE	8/16/25	$ 123
5/2/24	TH	6/10/25	$ 187
5/3/24	TH	6/10/25	$ 187

SAILING_LIST SAMPLE

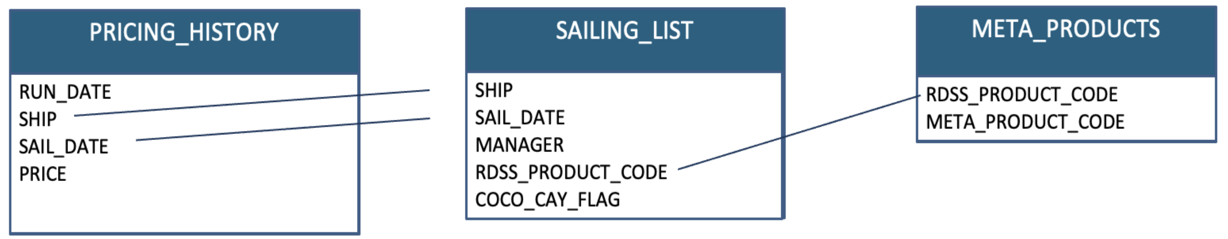
SHIP_COD	SAILING_DAT	MANAGE	RDSS_PRODUCT_CO	COCO_CAY_FLA
FE	8/16/25	ALAN	CARIB4	Y
FE	8/20/25	ALAN	CARIB3	N
TH	6/10/25	DEBBIE	CARIBEST	Y
TH	6/17/25	DEBBIE	CARIBEST	Y

META_PRODUCTS SAMPLE

RDSS_PRODUCT_CO	META_PRODUCT_CO
CARIB4	SHORT CARIBBEAN
CARIB3	SHORT CARIBBEAN
CARIBEST	7N CARIBBEAN

Note: The column ship_code is used consistently in place of ship to align with the sample data schema and maintain naming clarity across all tables.

3. Write a query to get the average of all price changes made over the past 5 days for each sailing (*ignore $0 changes*)



PRICING_HISTORY SAMPLE

RUN_DAT	SHIP_COD	SAIL_DAT	PRICE
5/2/24	FE	8/16/25	$ 120
5/3/24	FE	8/16/25	$ 123
5/4/24	FE	8/16/25	$ 123
5/2/24	TH	6/10/25	$ 187
5/3/24	TH	6/10/25	$ 187

SAILING_LIST SAMPLE

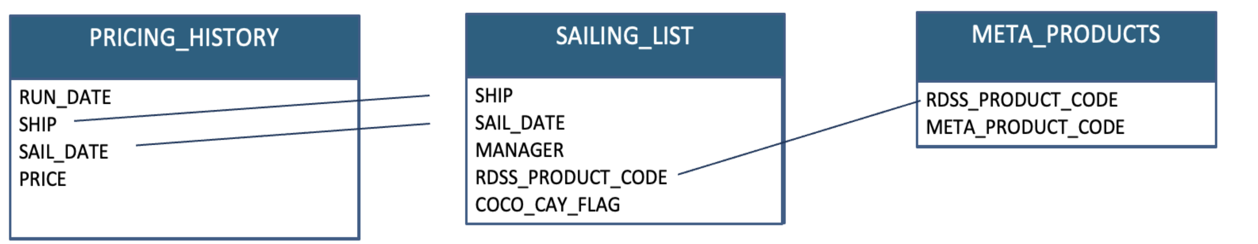
SHIP_COD	SAILING_DAT	MANAGE	RDSS_PRODUCT_CO	COCO_CAY_FLA
FE	8/16/25	ALAN	CARIB4	Y
FE	8/20/25	ALAN	CARIB3	N
TH	6/10/25	DEBBIE	CARIBEST	Y
TH	6/17/25	DEBBIE	CARIBEST	Y

META_PRODUCTS SAMPLE

RDSS_PRODUCT_CO	META_PRODUCT_CO
CARIB4	SHORT CARIBBEAN
CARIB3	SHORT CARIBBEAN
CARIBEST	7N CARIBBEAN

```
WITH price_change_log AS (
  SELECT
    ship_code,
    sail_date,
    run_date,
    price,
    LAG(price) OVER (
      PARTITION BY ship_code, sail_date
      ORDER BY run_date
    ) AS prev_price
  FROM pricing_history
)
SELECT
  ship_code,
  sail_date,
  AVG(price - prev_price) AS avg_price_change_last_5_days
FROM price_change_log
WHERE run_date >= CURRENT_DATE - INTERVAL '5 days'
  AND price != prev_price
GROUP BY ship_code, sail_date
ORDER BY ship_code;
```

4. Write a query that would allow us to see which manager hasn't made a price change in the longest time



PRICING_HISTORY SAMPLE

RUN_DAT	SHIP_COD	SAIL_DAT	PRICE
5/2/24	FE	8/16/25	$ 120
5/3/24	FE	8/16/25	$ 123
5/4/24	FE	8/16/25	$ 123
5/2/24	TH	6/10/25	$ 187
5/3/24	TH	6/10/25	$ 187

SAILING_LIST SAMPLE

SHIP_COD	SAILING_DAT	MANAGE	RDSS_PRODUCT_COI	COCO_CAY_FLA
FE	8/16/25	ALAN	CARIB4	Y
FE	8/20/25	ALAN	CARIB3	N
TH	6/10/25	DEBBIE	CARIBEST	Y
TH	6/17/25	DEBBIE	CARIBEST	Y

META_PRODUCTS SAMPLE

RDSS_PRODUCT_COI	META_PRODUCT_COI
CARIB4	SHORT CARIBBEAN
CARIB3	SHORT CARIBBEAN
CARIBEST	7N CARIBBEAN

```
WITH price_change_log AS (
  SELECT
    ship_code,
    sail_date,
    run_date,
    price,
    LAG(price) OVER (
      PARTITION BY ship_code, sail_date
      ORDER BY run_date
    ) AS prev_price
  FROM pricing_history
),
price_changes AS (
  SELECT *
  FROM price_change_log
  WHERE price != prev_price
),
manager_last_change AS (
  SELECT
    s.manager,
    MAX(p.run_date) AS last_price_change_date
  FROM price_changes p
  JOIN sailing_list s
    ON p.ship_code = s.ship_code
   AND p.sail_date = s.sail_date
  GROUP BY s.manager
)
SELECT
  manager,
  CURRENT_DATE - last_price_change_date AS
days_since_last_price_change
FROM manager_last_change
WHERE last_price_change_date = (
  SELECT MIN(last_price_change_date)
  FROM manager_last_change
);
```

```
price_change_gaps AS (
  SELECT
    p.ship_code,
    p.sail_date,
    p.run_date,
    p.price,
    p.prev_price,
    p.run_date - LAG(p.run_date) OVER
(
      PARTITION BY p.ship_code,
p.sail_date
      ORDER BY p.run_date
    ) AS day_gap
  FROM price_changes p
),

price_gaps_with_manager AS (
  SELECT
    s.manager,
    g.day_gap
  FROM price_change_gaps g
  JOIN sailing_list s
    ON g.ship_code = s.ship_code
   AND g.sail_date = s.sail_date
  WHERE g.day_gap IS NOT NULL
)

SELECT
  manager,
  MAX(day_gap) AS max_gap_days
FROM price_gaps_with_manager
GROUP BY manager
ORDER BY max_gap_days DESC;
```

Reference Click Link to Download!

Original File & Data:

Track Analyst Case Study.docx

Track_case_study_data.xlsx

Submissions:

wtd_data_cleaning_eda.ipynb

WTD_Tracking_Dashboard.pbix

cleaned_pax_build_data.csv

WTD Booking Ratio by Product & Sailing Period.csv

Thank You !