RESTAURANT SALES & ORDER ANALYSIS DASHBOARD

(Metabase + PostgreSQL)

Project Overview and Objective

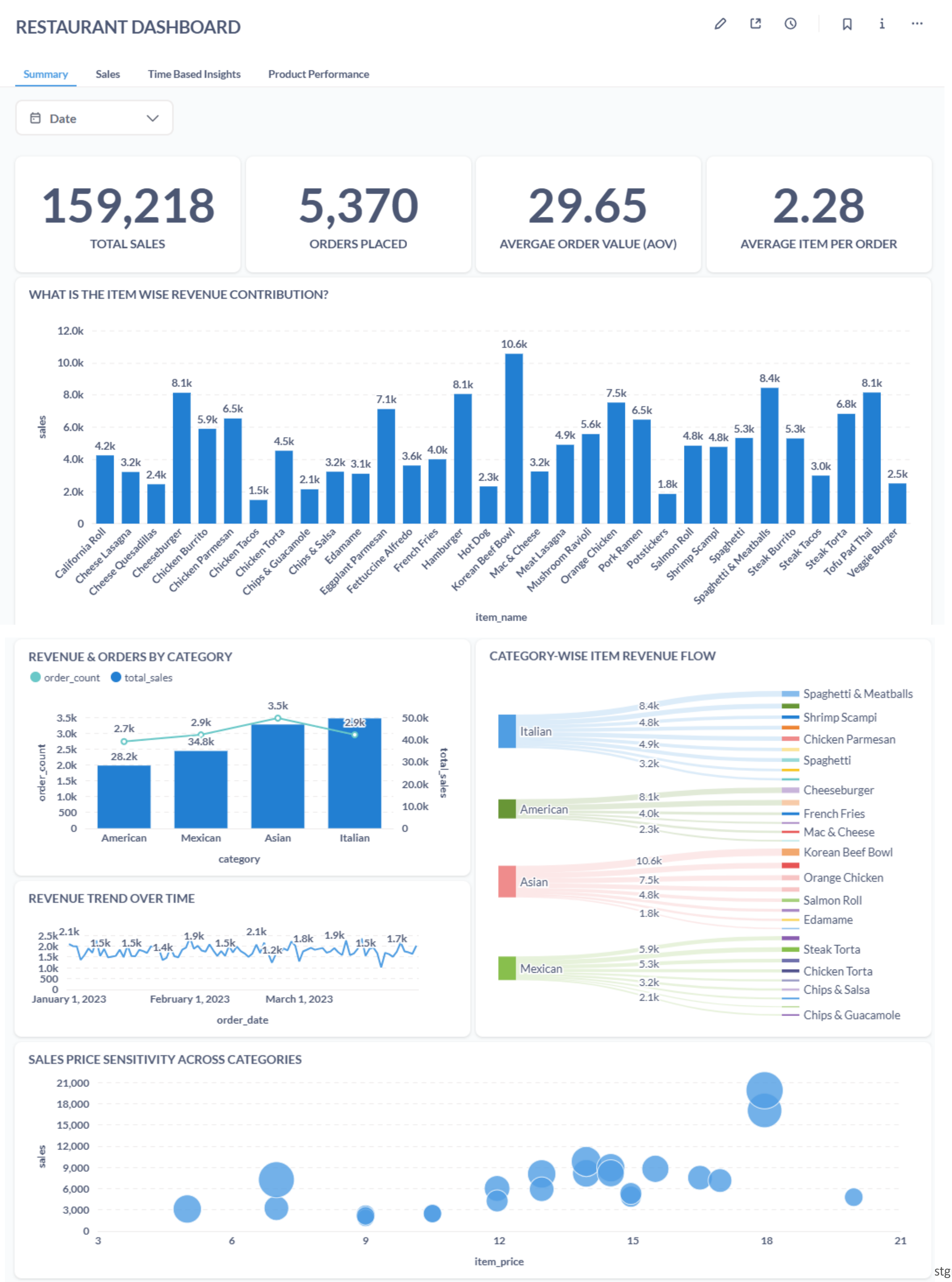
This project presents a comprehensive analytical overview of the restaurant's sales and order patterns using data hosted in **Supabase** and visualized through **Metabase**, powered by **PostgreSQL queries**. The aim is to transform raw transactional data into actionable insights that inform key business decisions across sales performance, product performance and time based insights for better revenue optimization for future.

The analysis focuses on breaking down and understanding **what sells, when it sells, and how it contributes** to the restaurant's overall performance. The dashboard provides a multi-dimensional view of sales data, uncovering trends across **time, product categories, price segments, and customer ordering patterns**.

Key objectives of the project include:

- Analyzing the **breadth of the restaurant menu**, including unique food items and their categorization.
- Tracking **revenue and order volume** at both product and category levels.
- Identifying **top-performing and underperforming items**, both in terms of order frequency and revenue generation.
- Exploring **temporal sales trends**, such as daily and hourly order patterns, running totals, and moving averages.
- Understanding the **contribution of different price segments** to overall revenue.
- Highlighting **customer footfall patterns** and **order trends** across time slots (morning, noon, evening, night).
- Establishing a basis for **ranking and forecasting**, including future sales estimations and performance comparisons.

By leveraging data-driven insights, this analysis equips restaurant stakeholders a deeper understanding of restaurant performance through real-time, data-backed insights. It serves as a strategic tool for monitoring operational trends, enhancing product offerings and supporting evidence-based decision-making across marketing, inventory, resource and menu planning efforts.



RESTAURANT DASHBOARD

159,218
TOTAL SALES

5,370
ORDERS PLACED

29.65
AVERGAE ORDER VALUE (AOV)

2.28
AVERAGE ITEM PER ORDER

WHAT IS THE ITEM WISE REVENUE CONTRIBUTION?

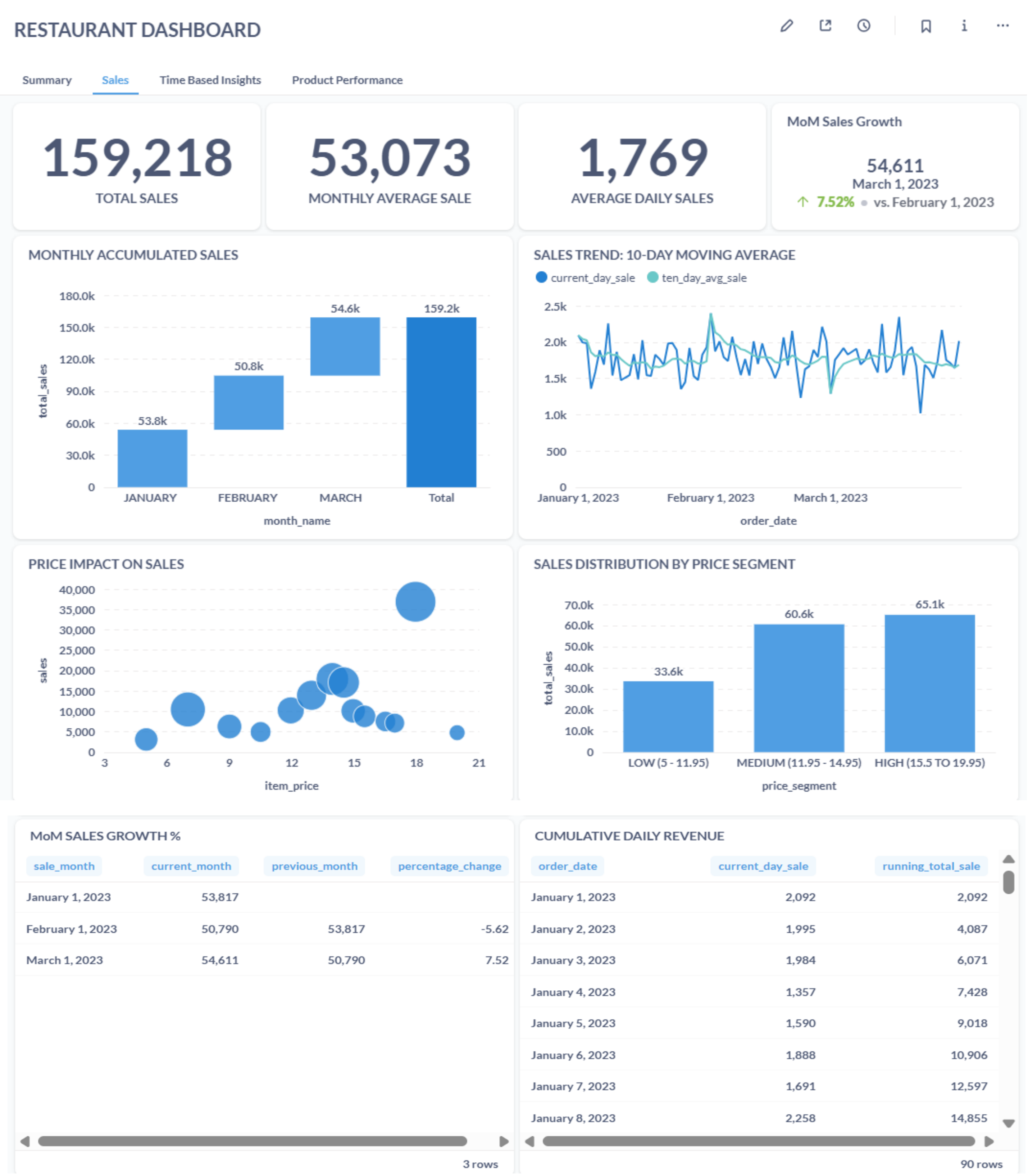
REVENUE & ORDERS BY CATEGORY

CATEGORY-WISE ITEM REVENUE FLOW

REVENUE TREND OVER TIME

SALES PRICE SENSITIVITY ACROSS CATEGORIES

stgreSQL)
By Md. Ishtiak Azim
http://linkedin.com/in/azim-ishtiak/



RESTAURANT DASHBOARD

159,218
TOTAL SALES

53,073
MONTHLY AVERAGE SALE

1,769
AVERAGE DAILY SALES

MoM Sales Growth
54,611
March 1, 2023
↑ 7.52% • vs. February 1, 2023

MONTHLY ACCUMULATED SALES

SALES TREND: 10-DAY MOVING AVERAGE

PRICE IMPACT ON SALES

SALES DISTRIBUTION BY PRICE SEGMENT

MoM SALES GROWTH %

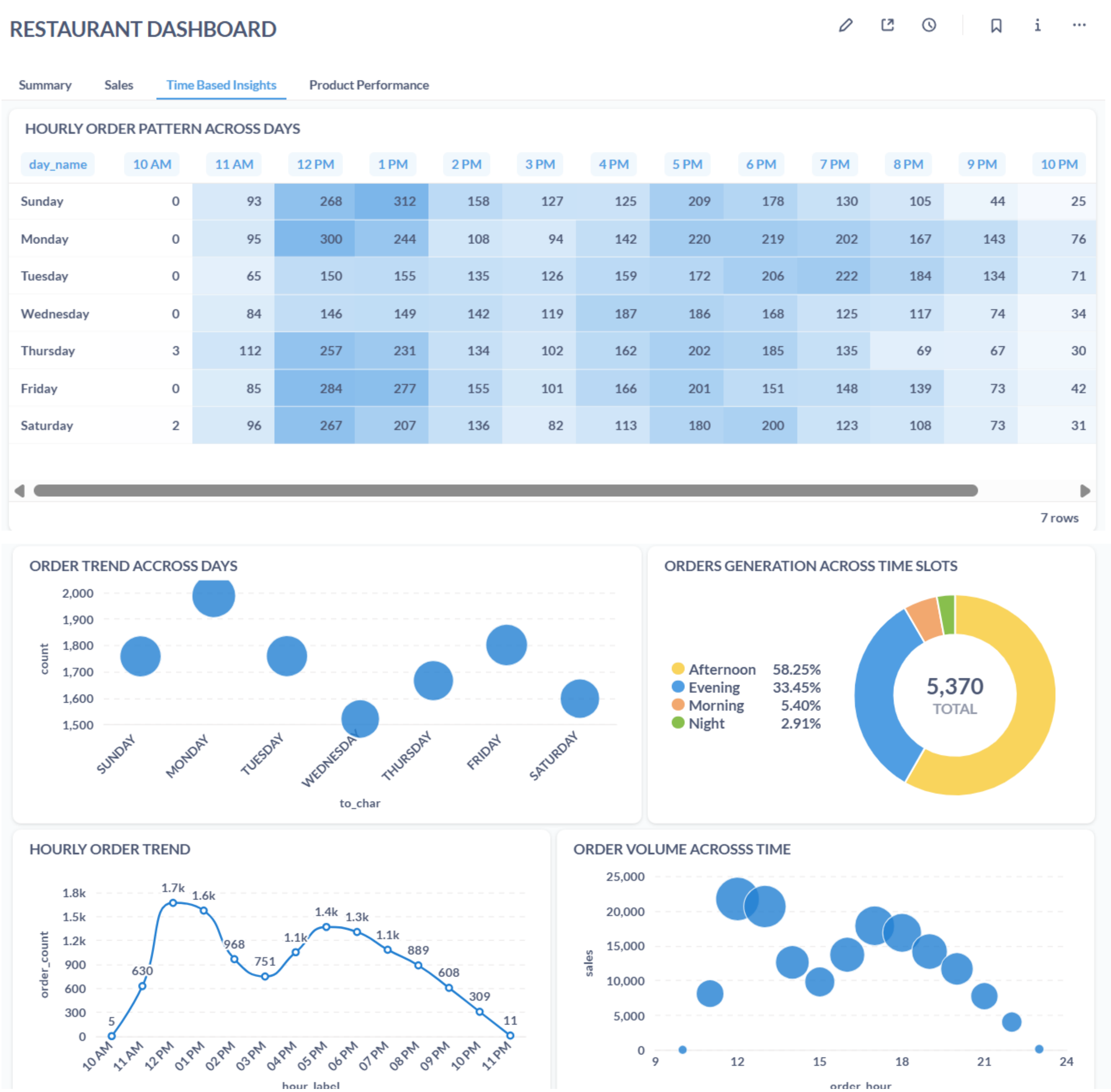
sale_month	current_month	previous_month	percentage_change
January 1, 2023	53,817		
February 1, 2023	50,790	53,817	-5.62
March 1, 2023	54,611	50,790	7.52

3 rows

CUMULATIVE DAILY REVENUE

order_date	current_day_sale	running_total_sale
January 1, 2023	2,092	2,092
January 2, 2023	1,995	4,087
January 3, 2023	1,984	6,071
January 4, 2023	1,357	7,428
January 5, 2023	1,590	9,018
January 6, 2023	1,888	10,906
January 7, 2023	1,691	12,597
January 8, 2023	2,258	14,855

90 rows



RESTAURANT DASHBOARD

HOURLY ORDER PATTERN ACROSS DAYS

day_name	10 AM	11 AM	12 PM	1 PM	2 PM	3 PM	4 PM	5 PM	6 PM	7 PM	8 PM	9 PM	10 PM
Sunday	0	93	268	312	158	127	125	209	178	130	105	44	25
Monday	0	95	300	244	108	94	142	220	219	202	167	143	76
Tuesday	0	65	150	155	135	126	159	172	206	222	184	134	71
Wednesday	0	84	146	149	142	119	187	186	168	125	117	74	34
Thursday	3	112	257	231	134	102	162	202	185	135	69	67	30
Friday	0	85	284	277	155	101	166	201	151	148	139	73	42
Saturday	2	96	267	207	136	82	113	180	200	123	108	73	31

7 rows

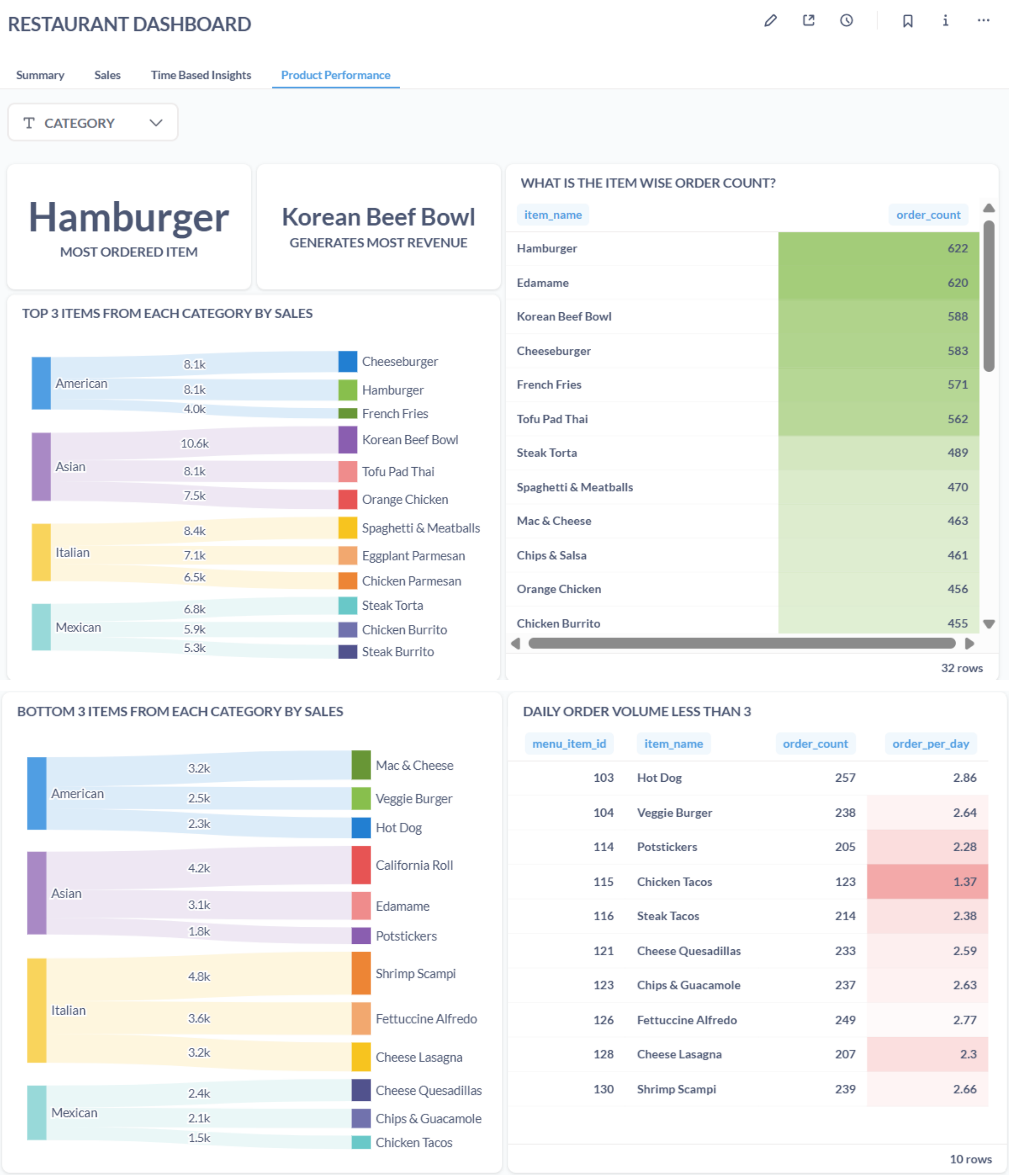
ORDER TREND ACCROSS DAYS

ORDERS GENERATION ACROSS TIME SLOTS

● Afternoon	58.25%
● Evening	33.45%
● Morning	5.40%
● Night	2.91%

HOURLY ORDER TREND

ORDER VOLUME ACROSSS TIME



RESTAURANT DASHBOARD

Hamburger

MOST ORDERED ITEM

Korean Beef Bowl

GENERATES MOST REVENUE

TOP 3 ITEMS FROM EACH CATEGORY BY SALES

WHAT IS THE ITEM WISE ORDER COUNT?

item_name	order_count
Hamburger	622
Edamame	620
Korean Beef Bowl	588
Cheeseburger	583
French Fries	571
Tofu Pad Thai	562
Steak Torta	489
Spaghetti & Meatballs	470
Mac & Cheese	463
Chips & Salsa	461
Orange Chicken	456
Chicken Burrito	455

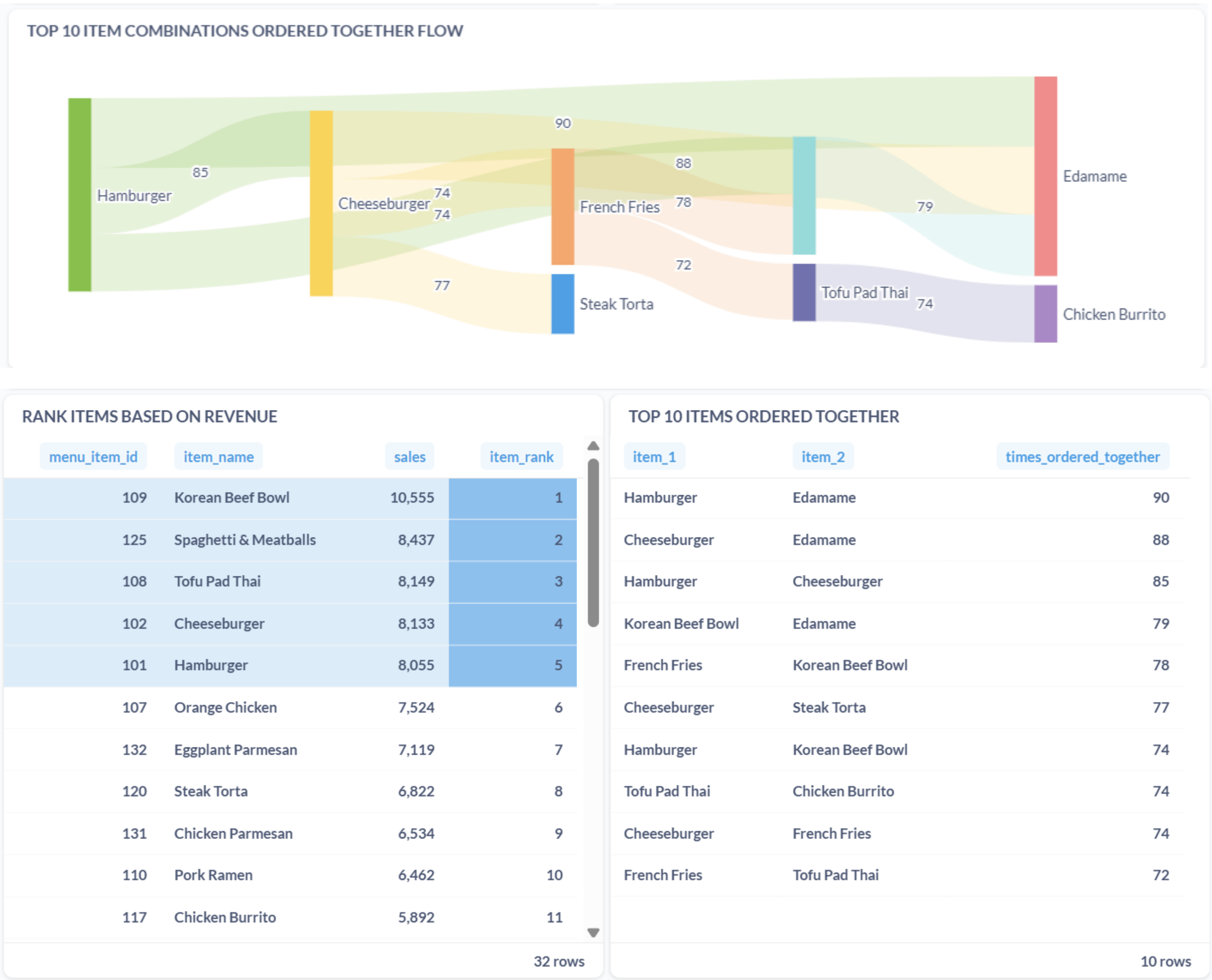
32 rows

BOTTOM 3 ITEMS FROM EACH CATEGORY BY SALES

DAILY ORDER VOLUME LESS THAN 3

menu_item_id	item_name	order_count	order_per_day
103	Hot Dog	257	2.86
104	Veggie Burger	238	2.64
114	Potstickers	205	2.28
115	Chicken Tacos	123	1.37
116	Steak Tacos	214	2.38
121	Cheese Quesadillas	233	2.59
123	Chips & Guacamole	237	2.63
126	Fettuccine Alfredo	249	2.77
128	Cheese Lasagna	207	2.3
130	Shrimp Scampi	239	2.66

10 rows



TOP 10 ITEM COMBINATIONS ORDERED TOGETHER FLOW

RANK ITEMS BASED ON REVENUE

menu_item_id	item_name	sales	item_rank
109	Korean Beef Bowl	10,555	1
125	Spaghetti & Meatballs	8,437	2
108	Tofu Pad Thai	8,149	3
102	Cheeseburger	8,133	4
101	Hamburger	8,055	5
107	Orange Chicken	7,524	6
132	Eggplant Parmesan	7,119	7
120	Steak Torta	6,822	8
131	Chicken Parmesan	6,534	9
110	Pork Ramen	6,462	10
117	Chicken Burrito	5,892	11

32 rows

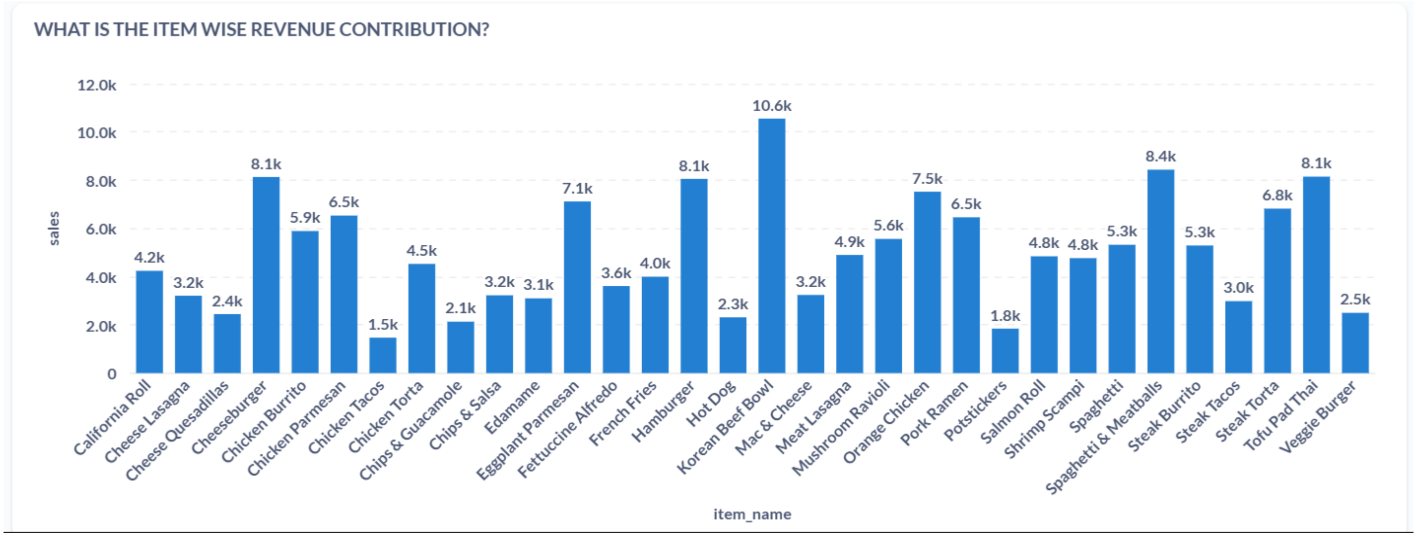
TOP 10 ITEMS ORDERED TOGETHER

item_1	item_2	times_ordered_together
Hamburger	Edamame	90
Cheeseburger	Edamame	88
Hamburger	Cheeseburger	85
Korean Beef Bowl	Edamame	79
French Fries	Korean Beef Bowl	78
Cheeseburger	Steak Torta	77
Hamburger	Korean Beef Bowl	74
Tofu Pad Thai	Chicken Burrito	74
Cheeseburger	French Fries	74
French Fries	Tofu Pad Thai	72

10 rows

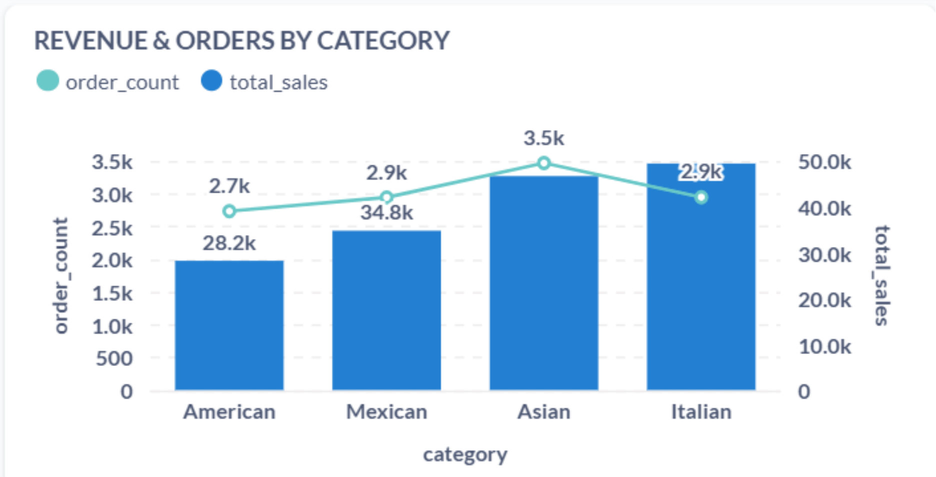
ITEM WISE REVENUE CONTRIBUTION

```
SELECT
     menu_items.item_name,
     ROUND(SUM(menu_items.price),0) AS TOTAL_SALES
FROM menu_items
LEFT JOIN order_details ON order_details.item_id = menu_items.menu_item_id
GROUP BY menu_items.item_name;
```



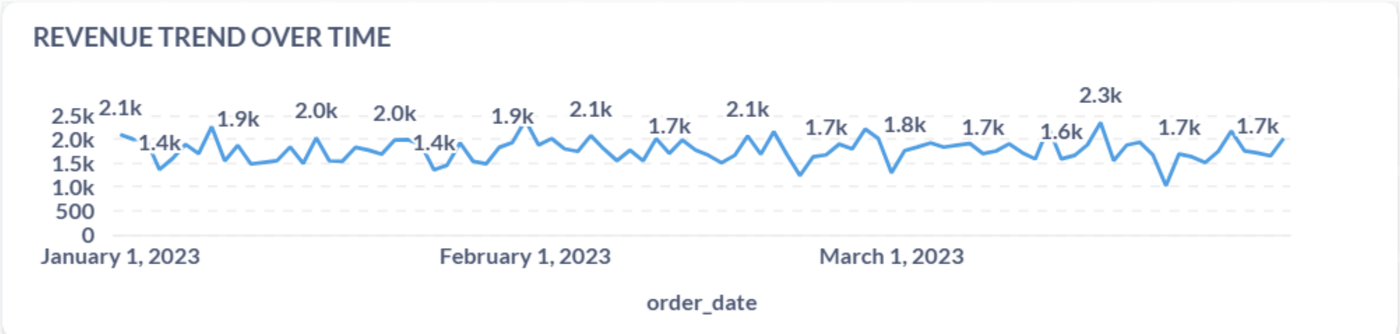
REVENUE AND ORDERS BY CATEGORY

```
SELECT
     menu_items.category,
     COUNT(order_details.order_id) AS ORDER_COUNT,
     ROUND(SUM(menu_items.price),0) AS TOTAL_SALES
FROM menu_items
LEFT JOIN order_details ON order_details.item_id = menu_items.menu_item_id
GROUP BY 1;
```



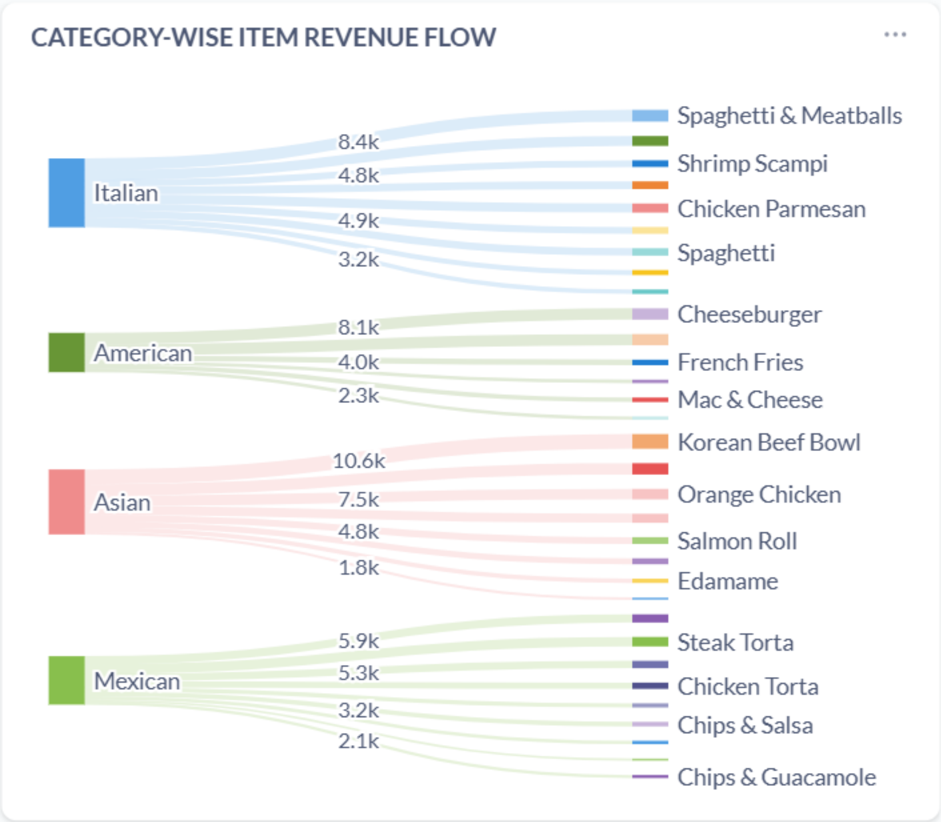
REVENUE TREND OVER TIME

```
SELECT
     order_details.order_date,
     ROUND(SUM(menu_items.price),0) AS TOTAL_SALES
FROM order_details
LEFT JOIN menu_items ON order_details.item_id = menu_items.menu_item_id
GROUP BY 1;
```



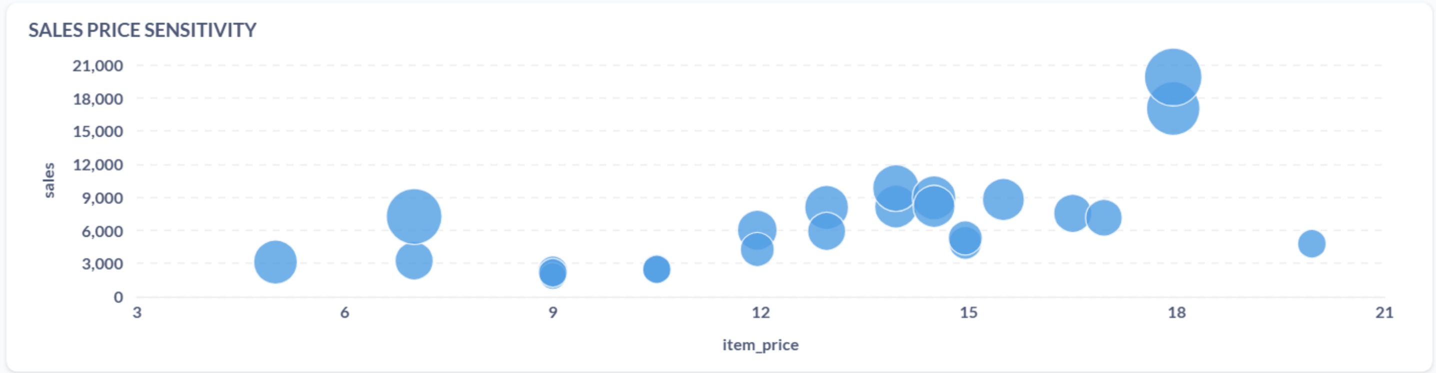
CATEGORY-WISE ITEM REVENUE FLOW

```
SELECT
     menu_items.category,
     menu_items.item_name,
     ROUND(SUM(menu_items.price),0) AS TOTAL_SALES
FROM menu_items
LEFT JOIN order_details ON order_details.item_id = menu_items.menu_item_id
GROUP BY 1, 2;
```



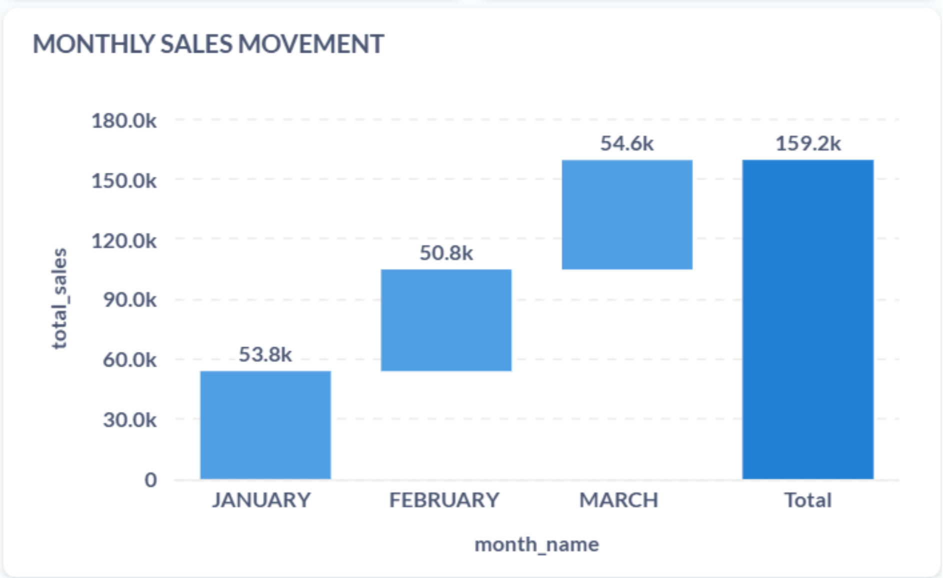
SALES PRICE SENSITIVITY

```
SELECT
     menu_items.price AS ITEM_PRICE,
     COUNT(order_details.order_id) AS ORDER_COUNT,
     ROUND(SUM(menu_items.price),0) AS SALES
FROM menu_items
LEFT JOIN order_details ON order_details.item_id = menu_items.menu_item_id
GROUP BY 1
ORDER BY 1;
```



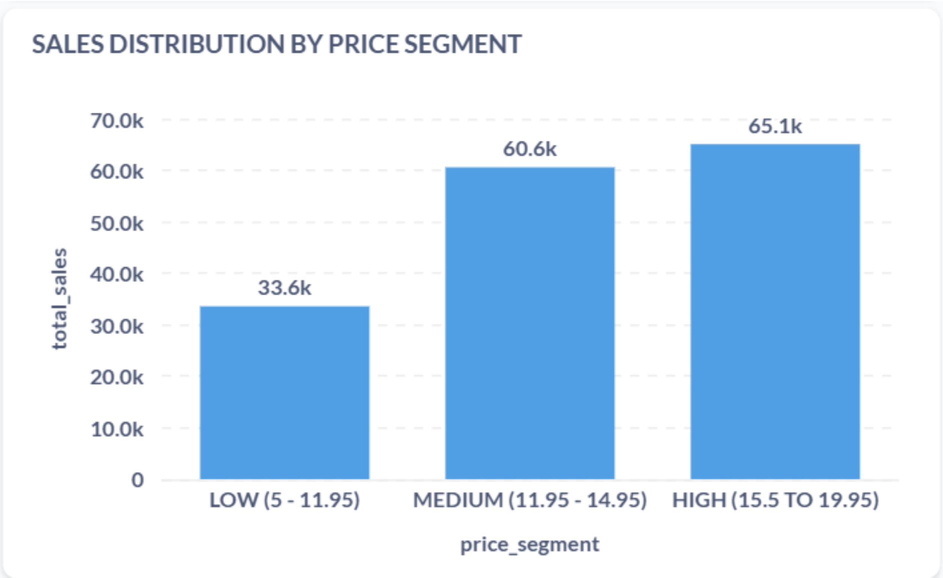
MONTHLY SALES ACCUMULATION

```
SELECT
     TO_CHAR(order_details.order_date,'MONTH') as MONTH_NAME,
     ROUND(SUM(menu_items.price),0) AS TOTAL_SALES
FROM order_details
LEFT JOIN menu_items ON order_details.item_id = menu_items.menu_item_id
GROUP BY 1, EXTRACT(MONTH FROM order_details.order_date)
ORDER BY EXTRACT(MONTH FROM order_details.order_date);
```



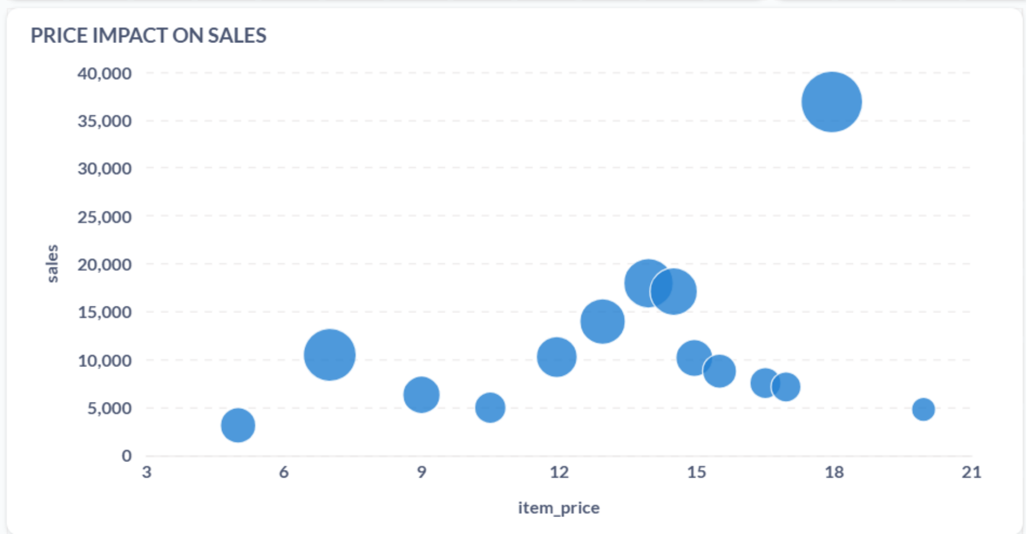
SALES DISTRIBUTION BY PRICE SEGMENT

```sql
WITH PRICE_SEGMMENTATION AS (
SELECT
    menu_item_id,
    item_name,
    price,
    NTILE(3) OVER (ORDER BY price ASC) AS price_bucket
FROM menu_items
ORDER BY price ASC
)
SELECT
      CASE
           WHEN price_bucket = 1 THEN 'LOW (5 - 11.95)'
           WHEN price_bucket = 2 THEN 'MEDIUM (11.95 - 14.95)'
           WHEN price_bucket = 3 THEN 'HIGH (15.5 TO 19.95)'
      END AS PRICE_SEGMENT,
      ROUND(SUM(menu_items.price),0) AS TOTAL_SALES
FROM order_details
JOIN menu_items ON order_details.item_id = menu_items.menu_item_id
JOIN PRICE_SEGMMENTATION ON menu_items.menu_item_id =
PRICE_SEGMMENTATION.menu_item_id
GROUP BY PRICE_SEGMENT, price_bucket
ORDER BY price_bucket;
```



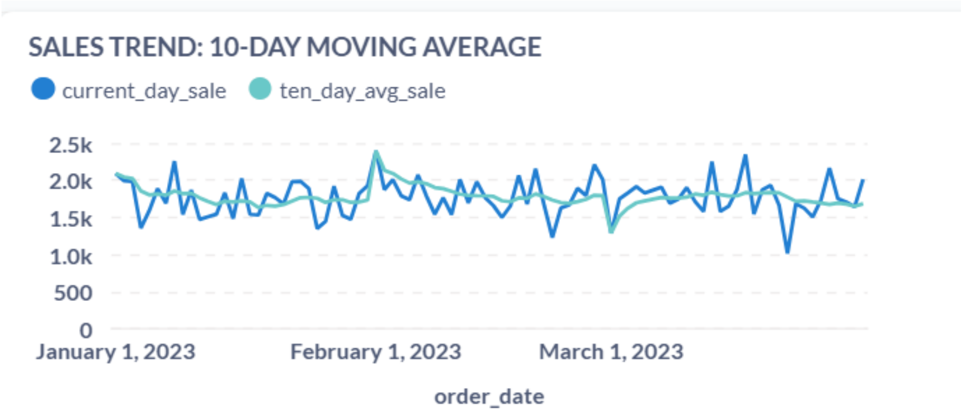
PRICE IMPACT ON SALES

```
SELECT
     menu_items.price AS ITEM_PRICE,
     COUNT(order_details.order_id) AS ORDER_COUNT,
     ROUND(SUM(menu_items.price),0) AS SALES
FROM menu_items
LEFT JOIN order_details ON order_details.item_id = menu_items.menu_item_id
GROUP BY 1
ORDER BY 1;
```



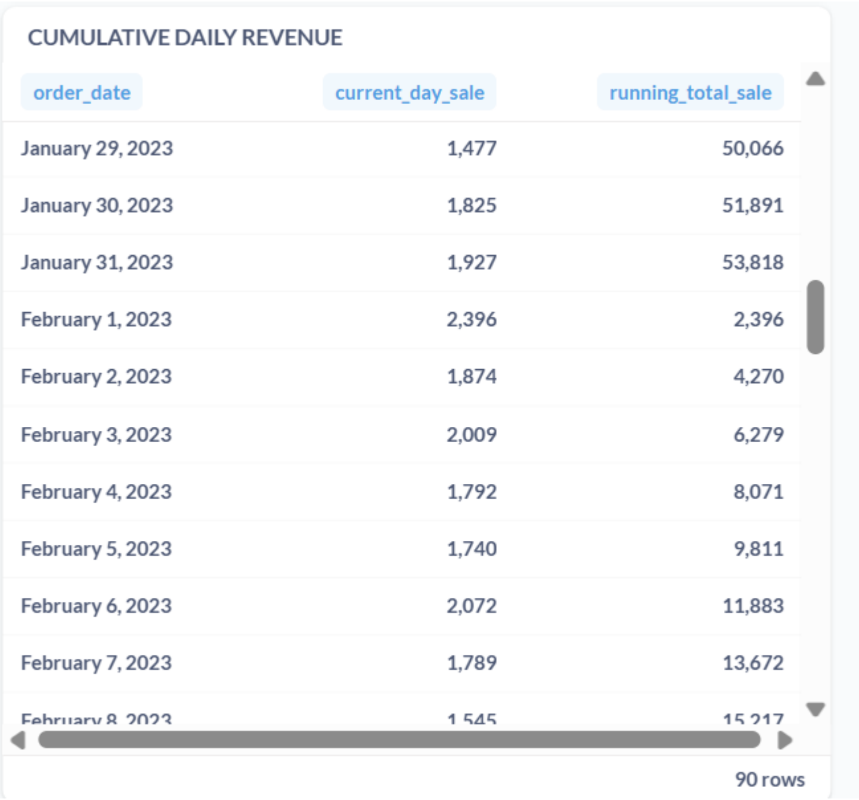
SALES TREND: 10-DAY MOVING AVERAGE

```
SELECT
     order_details.order_date,
     ROUND(SUM(menu_items.price),0) AS CURRENT_DAY_SALE,
     AVG(ROUND(SUM(menu_items.price),0)) OVER (PARTITION BY EXTRACT(MONTH FROM
order_date) ORDER BY order_details.order_date
     ROWS BETWEEN 9 PRECEDING AND CURRENT ROW
     ) AS TEN_DAY_AVG_SALE
FROM order_details
LEFT JOIN menu_items ON order_details.item_id = menu_items.menu_item_id
GROUP BY 1
ORDER BY order_details.order_date;
```



CUMULATIVE DAILY REVENUE

```
SELECT
      order_details.order_date,
      ROUND(SUM(menu_items.price),0) AS CURRENT_DAY_SALE,
      SUM(ROUND(SUM(menu_items.price),0)) OVER (PARTITION BY EXTRACT(MONTH FROM
order_date) ORDER BY order_details.order_date
      ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
      ) AS RUNNING_TOTAL_SALE
FROM order_details
LEFT JOIN menu_items ON order_details.item_id = menu_items.menu_item_id
GROUP BY 1
ORDER BY order_details.order_date;
```



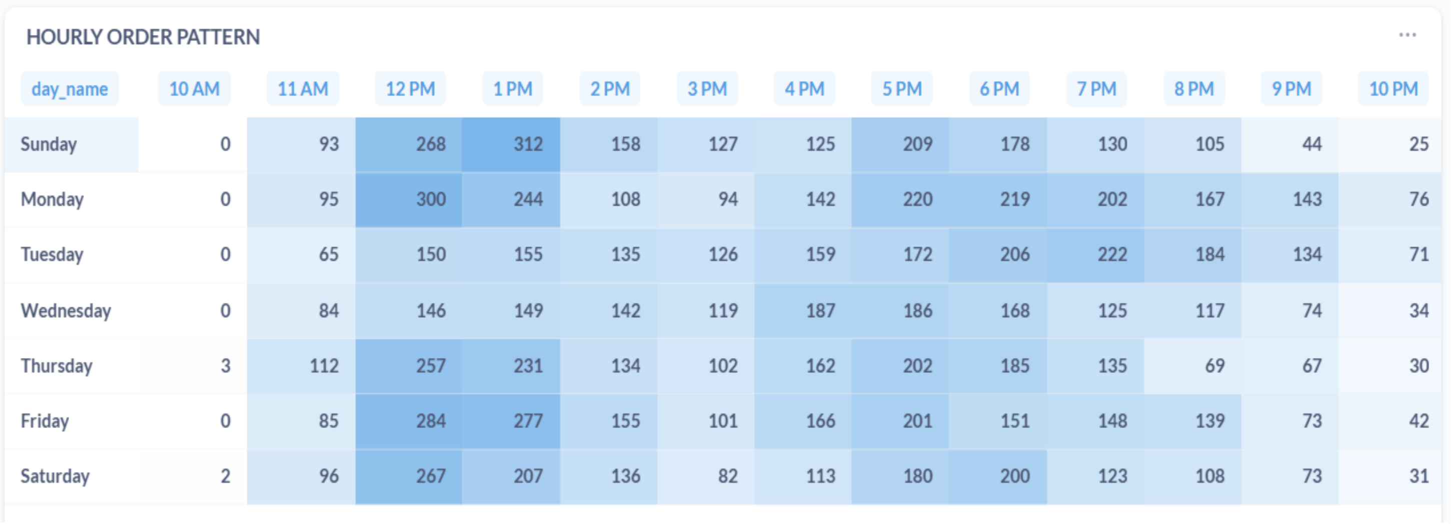
CUMULATIVE DAILY REVENUE

order_date	current_day_sale	running_total_sale
January 29, 2023	1,477	50,066
January 30, 2023	1,825	51,891
January 31, 2023	1,927	53,818
February 1, 2023	2,396	2,396
February 2, 2023	1,874	4,270
February 3, 2023	2,009	6,279
February 4, 2023	1,792	8,071
February 5, 2023	1,740	9,811
February 6, 2023	2,072	11,883
February 7, 2023	1,789	13,672
February 8, 2023	1,545	15,217

90 rows

ORDER TREND BY HOUR AND DAY

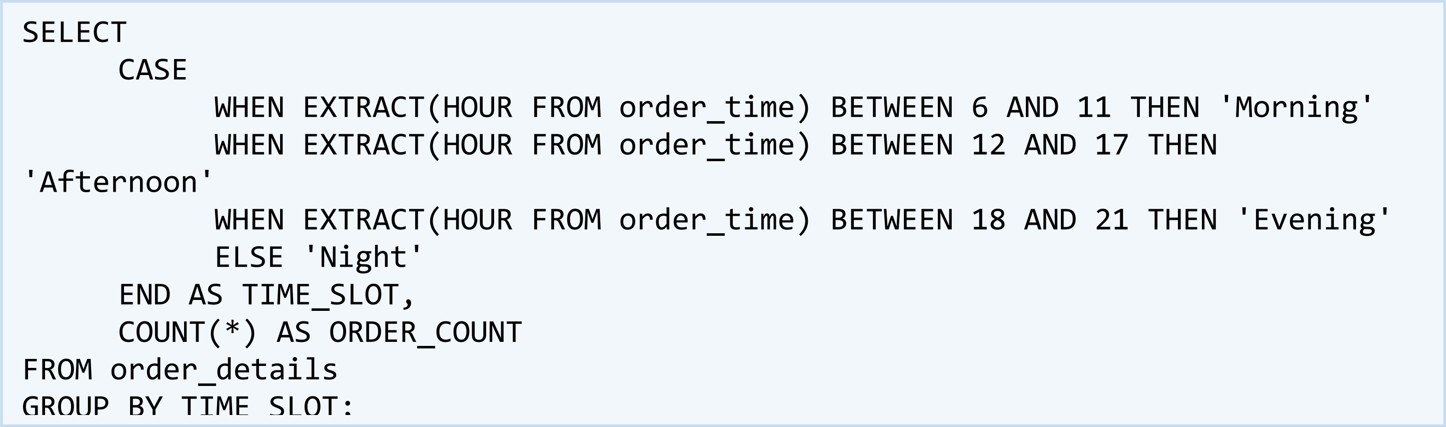
```
SELECT
    TO_CHAR(order_date, 'Day') AS DAY_NAME,
    COUNT(*) FILTER (WHERE EXTRACT(HOUR FROM order_time) = 10) AS "10 AM",
    COUNT(*) FILTER (WHERE EXTRACT(HOUR FROM order_time) = 11) AS "11 AM",
    COUNT(*) FILTER (WHERE EXTRACT(HOUR FROM order_time) = 12) AS "12 PM",
    COUNT(*) FILTER (WHERE EXTRACT(HOUR FROM order_time) = 13) AS "1 PM",
    COUNT(*) FILTER (WHERE EXTRACT(HOUR FROM order_time) = 14) AS "2 PM",
    COUNT(*) FILTER (WHERE EXTRACT(HOUR FROM order_time) = 15) AS "3 PM",
    COUNT(*) FILTER (WHERE EXTRACT(HOUR FROM order_time) = 16) AS "4 PM",
    COUNT(*) FILTER (WHERE EXTRACT(HOUR FROM order_time) = 17) AS "5 PM",
    COUNT(*) FILTER (WHERE EXTRACT(HOUR FROM order_time) = 18) AS "6 PM",
    COUNT(*) FILTER (WHERE EXTRACT(HOUR FROM order_time) = 19) AS "7 PM",
    COUNT(*) FILTER (WHERE EXTRACT(HOUR FROM order_time) = 20) AS "8 PM",
    COUNT(*) FILTER (WHERE EXTRACT(HOUR FROM order_time) = 21) AS "9 PM",
    COUNT(*) FILTER (WHERE EXTRACT(HOUR FROM order_time) = 22) AS "10 PM",
    COUNT(*) FILTER (WHERE EXTRACT(HOUR FROM order_time) = 23) AS "11 PM"
FROM order_details
GROUP BY day_name, EXTRACT(DOW FROM order_date)
ORDER BY EXTRACT(DOW FROM order_date);
```



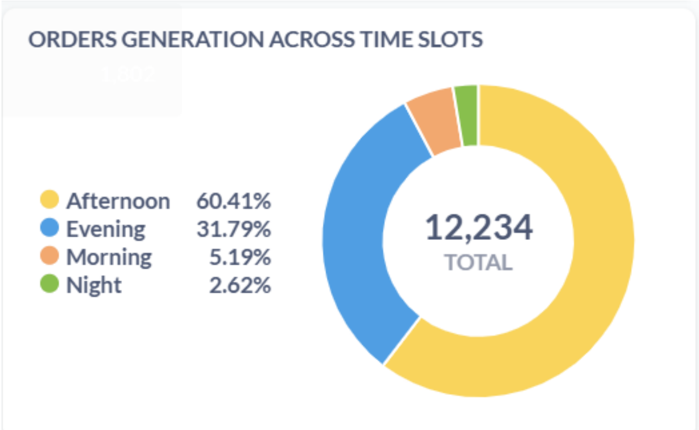
HOURLY ORDER PATTERN

day_name	10 AM	11 AM	12 PM	1 PM	2 PM	3 PM	4 PM	5 PM	6 PM	7 PM	8 PM	9 PM	10 PM
Sunday	0	93	268	312	158	127	125	209	178	130	105	44	25
Monday	0	95	300	244	108	94	142	220	219	202	167	143	76
Tuesday	0	65	150	155	135	126	159	172	206	222	184	134	71
Wednesday	0	84	146	149	142	119	187	186	168	125	117	74	34
Thursday	3	112	257	231	134	102	162	202	185	135	69	67	30
Friday	0	85	284	277	155	101	166	201	151	148	139	73	42
Saturday	2	96	267	207	136	82	113	180	200	123	108	73	31

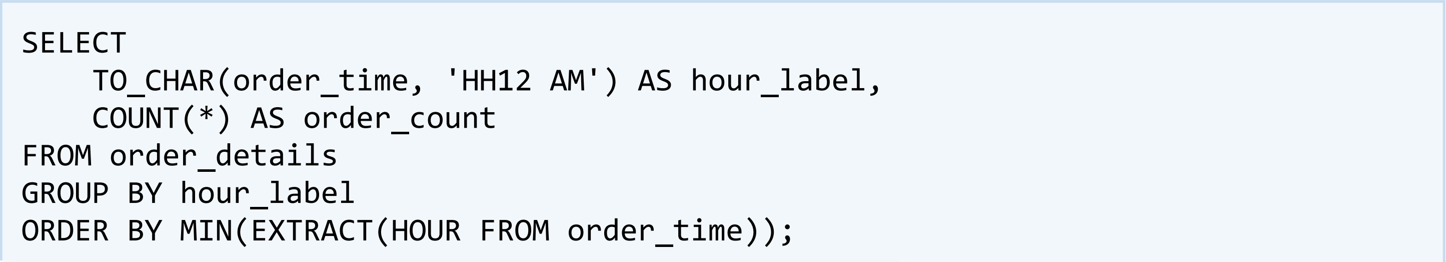
ORDER GENERATION ACROSS DIFFERENT TIME SLOTS



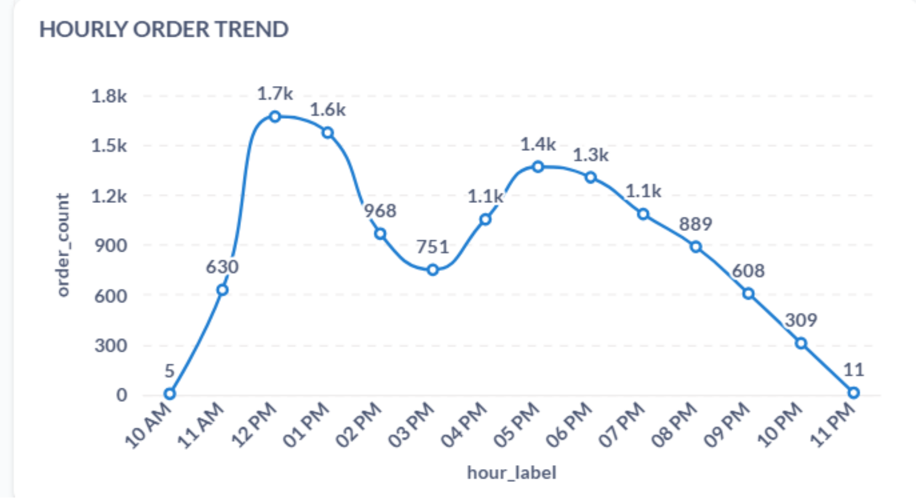
```
SELECT
      CASE
            WHEN EXTRACT(HOUR FROM order_time) BETWEEN 6 AND 11 THEN 'Morning'
            WHEN EXTRACT(HOUR FROM order_time) BETWEEN 12 AND 17 THEN
'Afternoon'
            WHEN EXTRACT(HOUR FROM order_time) BETWEEN 18 AND 21 THEN 'Evening'
            ELSE 'Night'
      END AS TIME_SLOT,
      COUNT(*) AS ORDER_COUNT
FROM order_details
GROUP BY TIME_SLOT;
```



HOURLY ORDER TREND

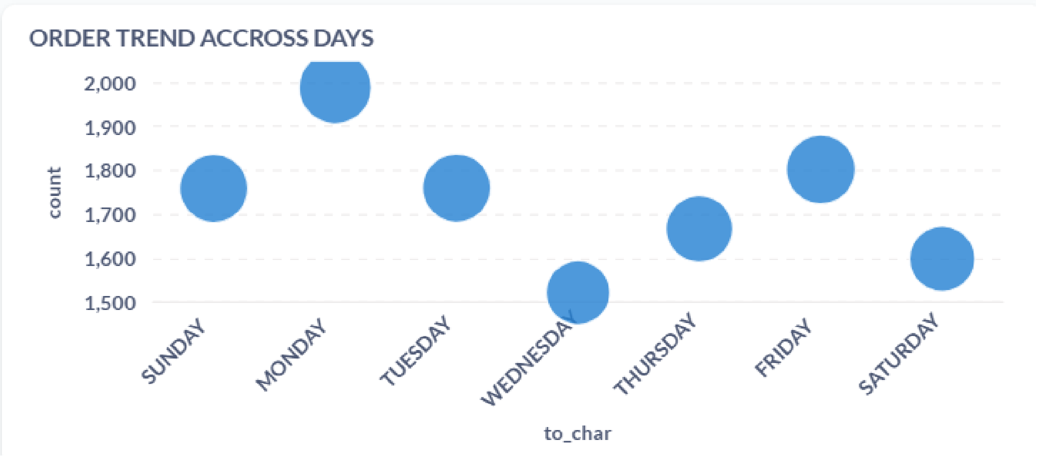


```
SELECT
    TO_CHAR(order_time, 'HH12 AM') AS hour_label,
    COUNT(*) AS order_count
FROM order_details
GROUP BY hour_label
ORDER BY MIN(EXTRACT(HOUR FROM order_time));
```



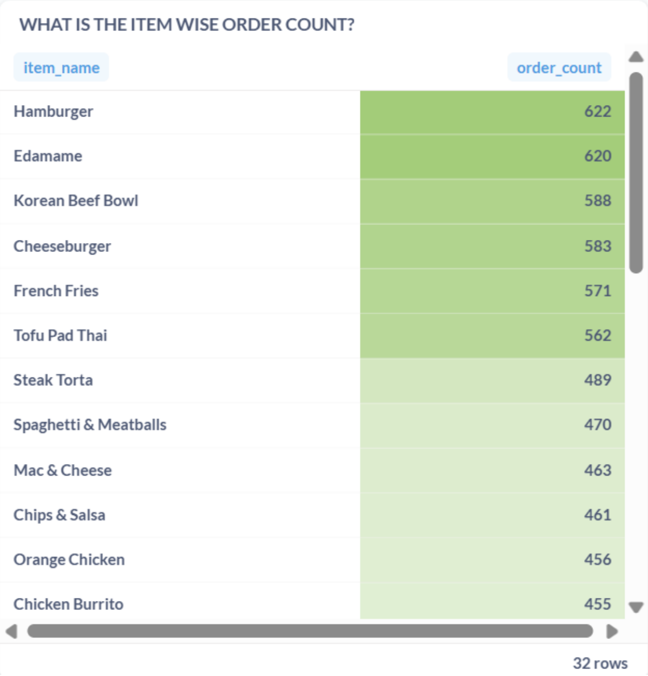
ORDER TREND ACROSS DAYS

```
SELECT
    TO_CHAR(order_time, 'HH12 AM') AS hour_label,
    COUNT(*) AS order_count
FROM order_details
GROUP BY hour_label
ORDER BY MIN(EXTRACT(HOUR FROM order_time));
```



ITEM WISE ORDER COUNT

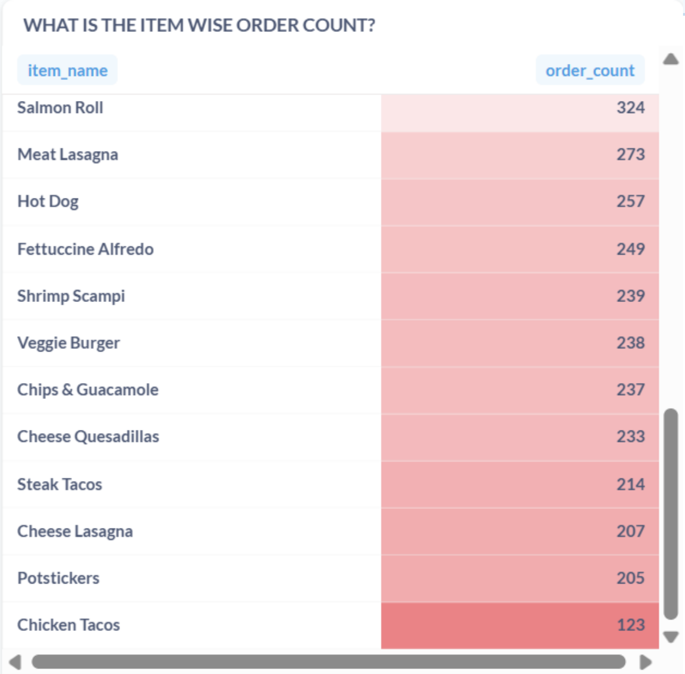
```
SELECT
     menu_items.item_name,
     COUNT(order_details.item_id) AS ORDER_COUNT
FROM menu_items
LEFT JOIN order_details ON order_details.item_id = menu_items.menu_item_id
GROUP BY 1
ORDER BY 2 DESC, 1 ASC;
```



WHAT IS THE ITEM WISE ORDER COUNT?

item_name	order_count
Hamburger	622
Edamame	620
Korean Beef Bowl	588
Cheeseburger	583
French Fries	571
Tofu Pad Thai	562
Steak Torta	489
Spaghetti & Meatballs	470
Mac & Cheese	463
Chips & Salsa	461
Orange Chicken	456
Chicken Burrito	455

32 rows

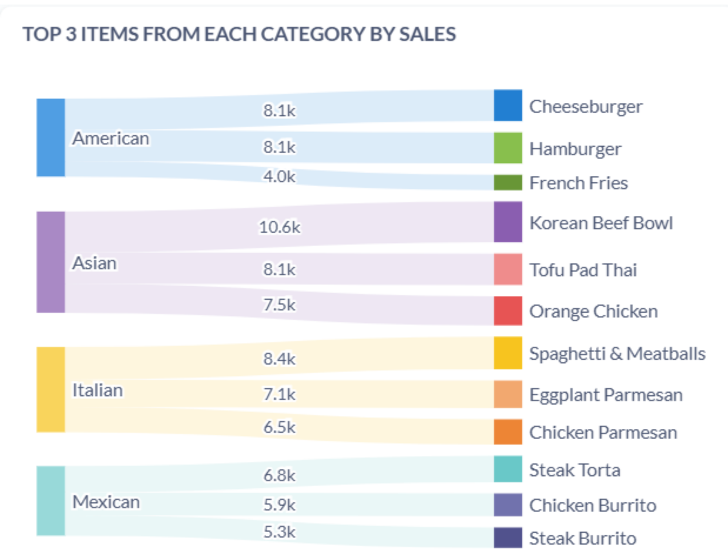


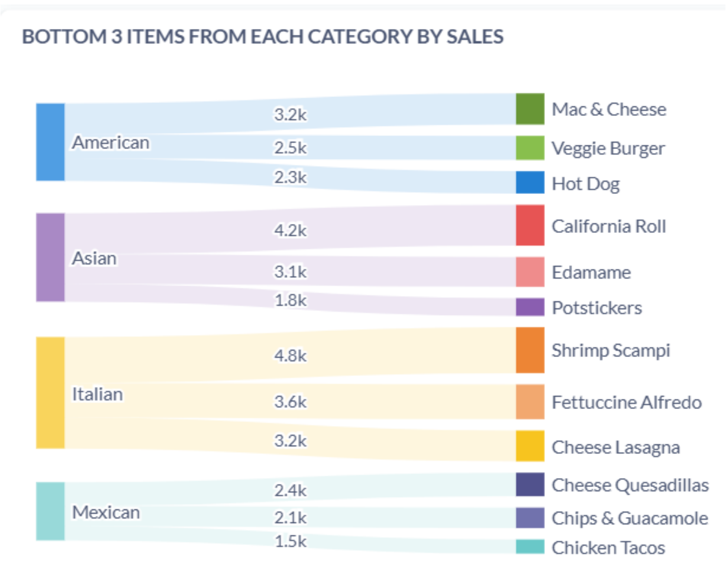
WHAT IS THE ITEM WISE ORDER COUNT?

item_name	order_count
Salmon Roll	324
Meat Lasagna	273
Hot Dog	257
Fettuccine Alfredo	249
Shrimp Scampi	239
Veggie Burger	238
Chips & Guacamole	237
Cheese Quesadillas	233
Steak Tacos	214
Cheese Lasagna	207
Potstickers	205
Chicken Tacos	123

TOP 3 AND BOTTOM 3 ITEMS FROM EACH CATEGORY BY SALES

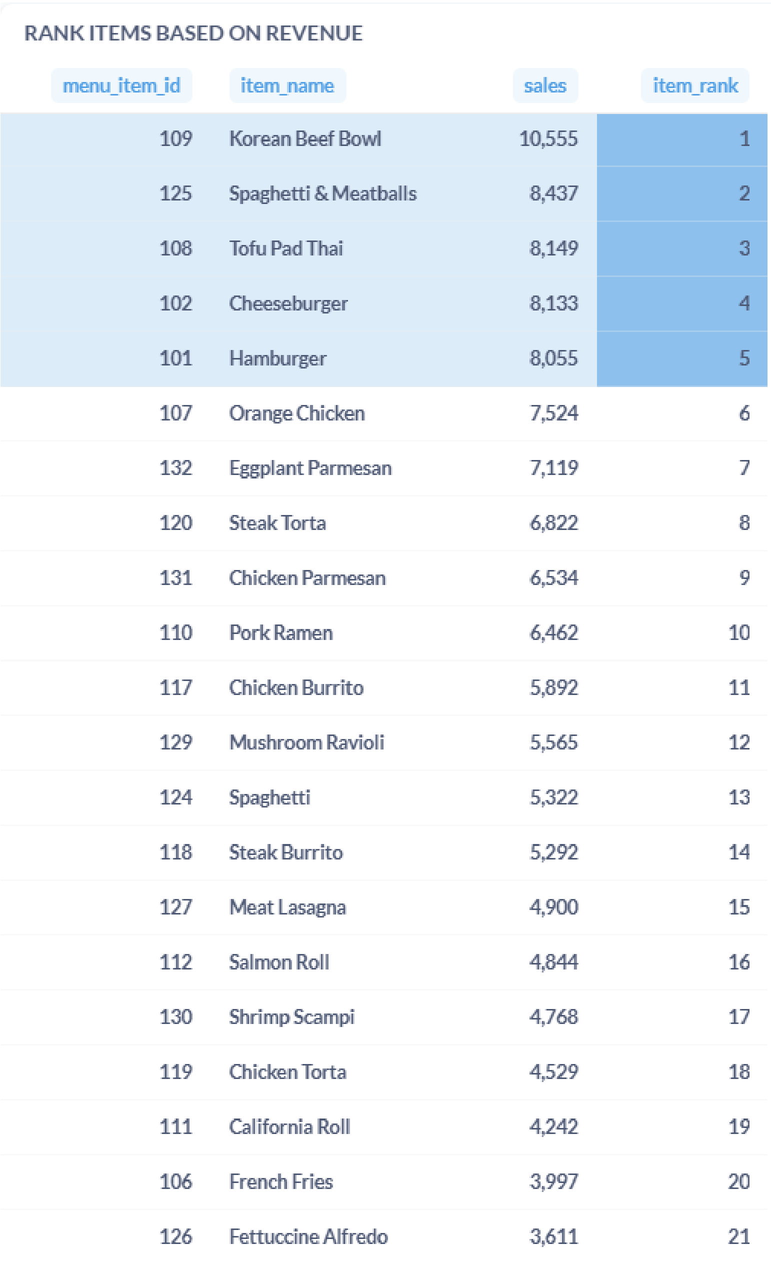
```
WITH ITEM_RANKING AS (
SELECT
     menu_items.category,
     menu_items.item_name,
     ROUND(SUM(menu_items.price),0) AS TOTAL_SALES,
     ROW_NUMBER() OVER(PARTITION BY menu_items.category ORDER BY
ROUND(SUM(menu_items.price),0) DESC) AS ITEM_RANK  -- JUST USE ASC FOR BOTTOM 3
ITEMS
FROM menu_items
LEFT JOIN order_details ON order_details.item_id = menu_items.menu_item_id
GROUP BY 1,2
)

SELECT
     category,
     item_name,
     TOTAL_SALES
FROM ITEM_RANKING
WHERE ITEM_RANK <=3;
```


TOP 3 ITEMS FROM EACH CATEGORY BY SALES


BOTTOM 3 ITEMS FROM EACH CATEGORY BY SALES

RANK ITEMS BASED ON SALES (TOP 5 ITEMS HIGHLIGHTED)

```
SELECT
     menu_items.menu_item_id,
     menu_items.item_name,
     ROUND(SUM(menu_items.price),0) AS SALES,
     RANK() OVER (ORDER BY ROUND(SUM(menu_items.price),0) DESC) AS ITEM_RANK
FROM menu_items
LEFT JOIN order_details ON order_detailS.item_id = menu_items.menu_item_id
GROUP BY 1
ORDER BY 4 ASC;
```

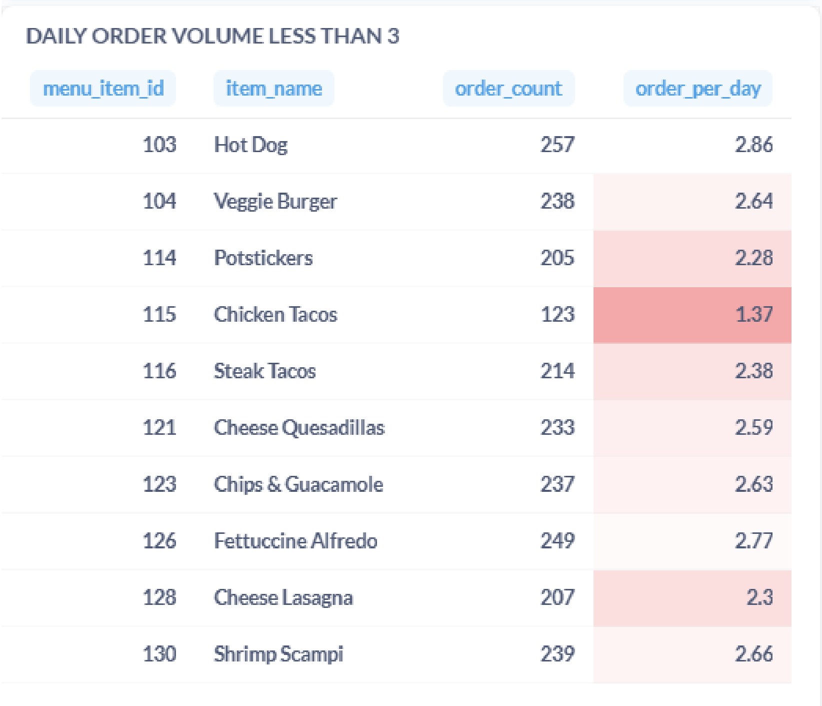


RANK ITEMS BASED ON REVENUE

menu_item_id	item_name	sales	item_rank
109	Korean Beef Bowl	10,555	1
125	Spaghetti & Meatballs	8,437	2
108	Tofu Pad Thai	8,149	3
102	Cheeseburger	8,133	4
101	Hamburger	8,055	5
107	Orange Chicken	7,524	6
132	Eggplant Parmesan	7,119	7
120	Steak Torta	6,822	8
131	Chicken Parmesan	6,534	9
110	Pork Ramen	6,462	10
117	Chicken Burrito	5,892	11
129	Mushroom Ravioli	5,565	12
124	Spaghetti	5,322	13
118	Steak Burrito	5,292	14
127	Meat Lasagna	4,900	15
112	Salmon Roll	4,844	16
130	Shrimp Scampi	4,768	17
119	Chicken Torta	4,529	18
111	California Roll	4,242	19
106	French Fries	3,997	20
126	Fettuccine Alfredo	3,611	21

LIST ITEMS WITH ORDER VOLUME LESS THAN 3 PER DAY

```
WITH DAY_COUNT AS (
SELECT
      COUNT(DISTINCT order_date) AS number_of_days
FROM order_details
)

SELECT
      menu_items.menu_item_id,
      menu_items.item_name,
      COUNT(order_details.item_id) AS ORDER_COUNT,
      COUNT(order_details.item_id):: DECIMAL / number_of_dayS AS ORDER_PER_DAY
FROM menu_items
LEFT JOIN order_details ON order_detailS.item_id = menu_items.menu_item_id
CROSS JOIN DAY_COUNT
GROUP BY menu_items.menu_item_id, DAY_COUNT.number_of_days
HAVING COUNT(order_details.item_id):: DECIMAL / number_of_dayS <3
ORDER BY 1 ASC;
```

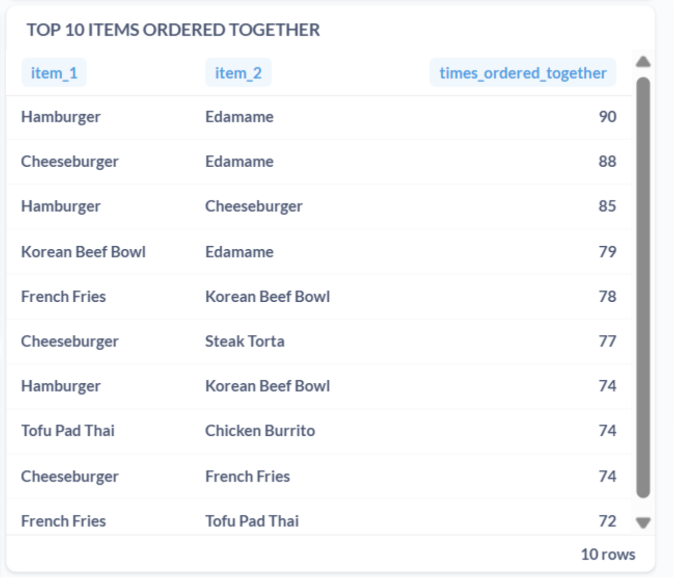


DAILY ORDER VOLUME LESS THAN 3

menu_item_id	item_name	order_count	order_per_day
103	Hot Dog	257	2.86
104	Veggie Burger	238	2.64
114	Potstickers	205	2.28
115	Chicken Tacos	123	1.37
116	Steak Tacos	214	2.38
121	Cheese Quesadillas	233	2.59
123	Chips & Guacamole	237	2.63
126	Fettuccine Alfredo	249	2.77
128	Cheese Lasagna	207	2.3
130	Shrimp Scampi	239	2.66

TOP 10 ITEMS ORDERED TOGETHER

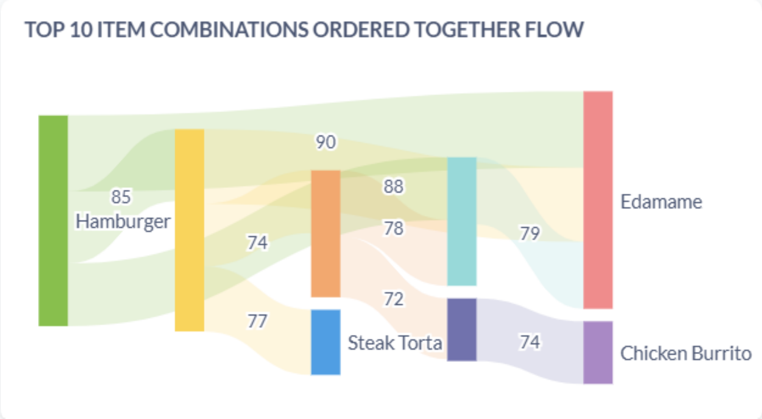
```
SELECT
    mi1.item_name AS item_1,
    mi2.item_name AS item_2,
    COUNT(*) AS times_ordered_together
FROM order_details od1
JOIN order_details od2
    ON od1.order_id = od2.order_id
    AND od1.item_id < od2.item_id  -- avoids duplicates and self-pairs
JOIN menu_items mi1 ON od1.item_id = mi1.menu_item_id
JOIN menu_items mi2 ON od2.item_id = mi2.menu_item_id
GROUP BY item_1, item_2
ORDER BY times_ordered_together DESC
LIMIT 10;
```



TOP 10 ITEMS ORDERED TOGETHER

item_1	item_2	times_ordered_together
Hamburger	Edamame	90
Cheeseburger	Edamame	88
Hamburger	Cheeseburger	85
Korean Beef Bowl	Edamame	79
French Fries	Korean Beef Bowl	78
Cheeseburger	Steak Torta	77
Hamburger	Korean Beef Bowl	74
Tofu Pad Thai	Chicken Burrito	74
Cheeseburger	French Fries	74
French Fries	Tofu Pad Thai	72

10 rows



TOP 10 ITEM COMBINATIONS ORDERED TOGETHER FLOW

INSIGHTS

Item & Category Performance

- There is a clear breakdown of sales by both **item and category**, identifying which products contribute the most to revenue.
- Certain items are ordered significantly more often than others, indicating **customer preference concentration**.
- Categories and items have been effectively ranked, and **Top 3/Bottom 3 items per category** provide actionable granularity.

Revenue Trends Over Time

- The **revenue trend line over time** highlights seasonal or daily fluctuations, useful for identifying **peak days** and **slow periods**.
- The **10-day moving average** and **daily running total** help smooth out variability and highlight sustained performance trends.

Price Sensitivity & Segment Analysis

- The segmentation of menu items into **Low**, **Medium**, and **High** price buckets reveals that revenue contribution varies by pricing tier.
- **Sales by price segment** helps identify which price range is performing best — critical for menu pricing strategy.

Time-Based Ordering Patterns

- **Hourly and daily breakdowns** show strong trends in order volumes by **time of day and day of the week**.
- Most orders seem to cluster in **specific time slots** (e.g., afternoon/evening), informing optimal staffing and promotion timings.
- Time slot segmentation (morning, afternoon, evening, night) gives a useful view of customer dining behavior.

Sales Volume Insights

- Certain items show an **average daily order frequency below 3**, which may indicate low interest or lack of visibility.
- Top 10 **item combinations** ordered together provide opportunities for bundled promotions.

RECOMMENDATIONS

Optimize Menu Based on Performance

- Consider **revising or removing underperforming items** (e.g., <3 daily orders).
- Increase promotion or visibility of **high-revenue and frequently ordered items**.
- Introduce **combo offers** for top item pairs frequently ordered together.

Leverage Time-Based Promotions

- Utilize **peak time slots** (identified by hour and day) for targeted promotions, discounts, or upselling strategies.
- For slower slots (e.g., morning or late night), explore **time-limited deals** or special menu items to increase traffic.

Refine Pricing Strategy

- Use insights from **price segmentation** to review pricing tiers — optimize for segments that generate the most revenue.
- Consider testing price adjustments for medium-performing segments to improve margins or volume.

Support Forecasting and Inventory Planning

- Use **running total and moving average trends** for planning **inventory**, **staffing**, and **procurement** more efficiently.
- Build on the available time-series data to develop a **next-month sales forecast** for operational readiness.

Enable Ongoing Monitoring

- Integrate this dashboard into routine decision-making.
- Set alerts or thresholds in Metabase for sudden spikes/drops in order volume, daily revenue, or low-performing items.