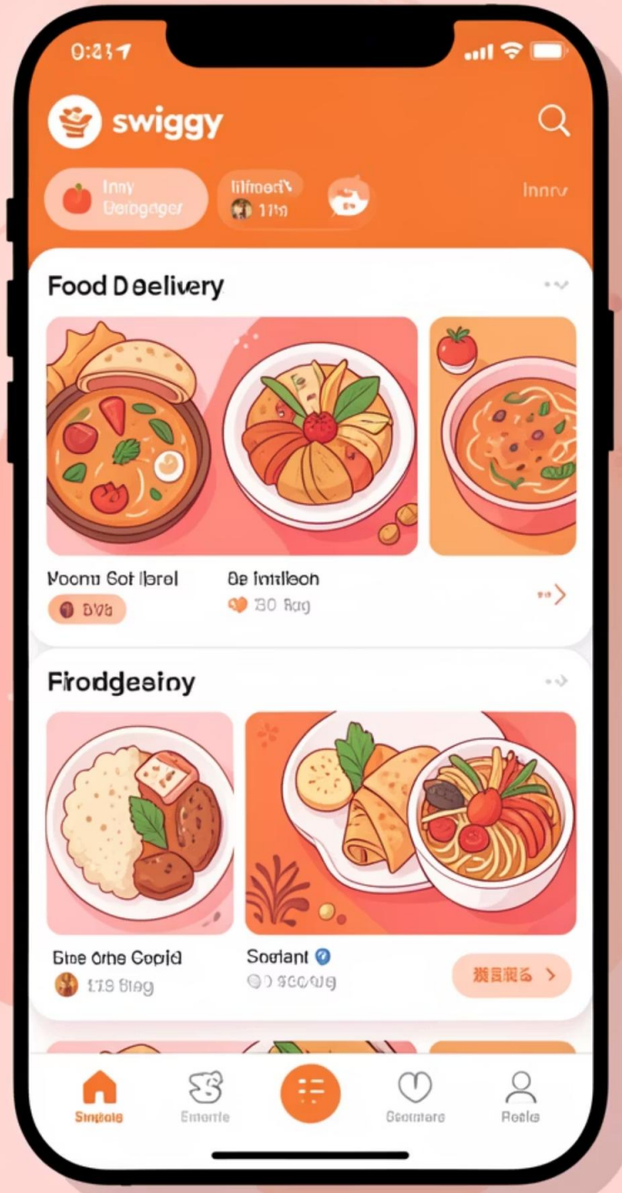


Swiggy Data Analysis

Kunal Ramrakhya

QUERY 1

Display all customers who live in 'Delhi'

This query filters the customer database to retrieve all records where the city field matches 'Delhi'. It provides a complete list of Delhi-based customers for targeted analysis.

Result Grid

name
Rohini Verma
Manish Kumar
Sonali Mishra

```
##1.    Display all customers who live in 'Delhi'.
select distinct name from customers
where city = 'Delhi';
```

QUERY 2

Find the average rating of all restaurants in 'Mumbai'

SQL Query

Calculates the mean rating across all Mumbai restaurants using the AVG function with city-based filtering.

Business Insight

Helps assess overall restaurant quality in Mumbai and benchmark performance standards.

```sql
##2.    Find the average rating of all restaurants in 'Mumbai'.
select distinct name,
avg(rating), city
from restaurants
where city = 'Mumbai' and rating is not null
group by name, city;
```

Result Grid | Filter Rows:

name	avg(rating)	city
Spice of India	4.500000	Mumbai
The Great Indian Thali	4.400000	Mumbai
Bombay Bhel	4.000000	Mumbai

QUERY 3

List all customers who have placed at least one order

This query uses a JOIN operation between customers and orders tables to identify active users. It filters out customers with no order history, providing insights into customer engagement.

```sql
##3.    List all customers who have placed at least one order.
select distinct name, count(order_id) as Order_count
from customers
join orders
using(customer_id)
group by name
having count(order_id) > 1;
```

Result Grid | Filter Rows:

name	Order_count
Amit Sharma	2
Rohini Verma	3
Rajesh Gupta	3
Sneha Mehta	2
Manish Kumar	4
Priya Singh	3
Vikas Reddy	3
Anjali Patel	3
Kavita Deshmukh	2
Vivek Bhatt	2
Meera Joshi	2
Pankaj Jain	2
Nidhi Saxena	3
Ashok Kumar	3
Deepa Rao	2
Sonali Mishra	3
Arjun Desai	2

QUERY 4

Display the total number of orders placed by each customer

Aggregation

Uses COUNT and GROUP BY to calculate order frequency per customer.

Application

Identifies high-value customers for loyalty programs and retention strategies.

```
##4.    Display the total number of orders placed by each customer.
select distinct name, count(order_id) as total_orders
from customers
join orders
using(customer_id)
group by name;
```

Result Grid | Filter Rows:

name	total_orders
Amit Sharma	2
Rohini Verma	3
Rajesh Gupta	3
Sneha Mehta	2
Manish Kumar	4
Priya Singh	3
Vikas Reddy	3
Anjali Patel	3
Suresh Nair	1
Kavita Deshmukh	2
Vivek Bhatt	2
Meera Joshi	2
Pankaj Jain	2
Nidhi Saxena	3

QUERY 5

Find the total revenue generated by each restaurant

This query aggregates order values by restaurant using SUM and GROUP BY functions. It reveals which restaurants drive the most revenue on the platform.

```sql
##5.    Find the total revenue generated by each restaurant.
select restaurants.name, sum(total_amount) as revenue
from restaurants
join orders
using (restaurant_id)
group by restaurants.name;
```

Result Grid | Filter Rows:

name	revenue
Biryani House	5300.00
Taste of Punjab	600.00
Spice of India	1100.00
Coastal Delight	2100.00
Tandoori Flames	1200.00
Curry Pot	3200.00
Veggie Delight	1600.00
Gujarat Express	2550.00
Royal Biryani	650.00
Punjabi Tadka	900.00
Flavours of Bengal	4050.00
South Treat	2950.00
The Great Indian Thali	1600.00
Rajasthani Rasoi	2100.00
Andhra Spice	4050.00
Chaat Junction	2150.00
Maharashtrian Magic	2050.00

QUERY 6

Find the top 5 restaurants with the highest average rating

Quality Leaders

Identifies best-performing restaurants based on customer satisfaction scores.

Ranking Method

Uses AVG, GROUP BY, ORDER BY DESC, and LIMIT 5 to extract top performers.

```
##6.    Find the top 5 restaurants with the highest average rating.
select distinct name, rating
from restaurants
order by rating desc
limit 5;
```

Result Grid | Filter Rows:

name	rating
Biryani House	4.80
Paradise Biryani	4.80
Lucknowi Nawabi	4.70
Royal Biryani	4.70
Awadhi Zaika	4.60

QUERY 7

Display all customers who have never placed an order

Uses a LEFT JOIN with NULL filtering to identify inactive customers. This helps target re-engagement campaigns and understand user drop-off patterns.

```sql
##7.    Display all customers who have never placed an order.
select distinct name  from customers
where customer_id not in (select customer_id from orders);
##or
select distinct name
from customers
left join orders using(customer_id)
where orders.customer_id is null;
```

Result Grid

name
Sonal Kaur
Vivek Malhotra
Divya Iyer
Rakesh Yadav
Mona Sharma
Sudha Pillai
Gaurav Khanna

QUERY 8

Find the number of orders placed by each customer in 'Mumbai'

Geographic Focus

Combines customer location filtering with order aggregation for city-specific insights.

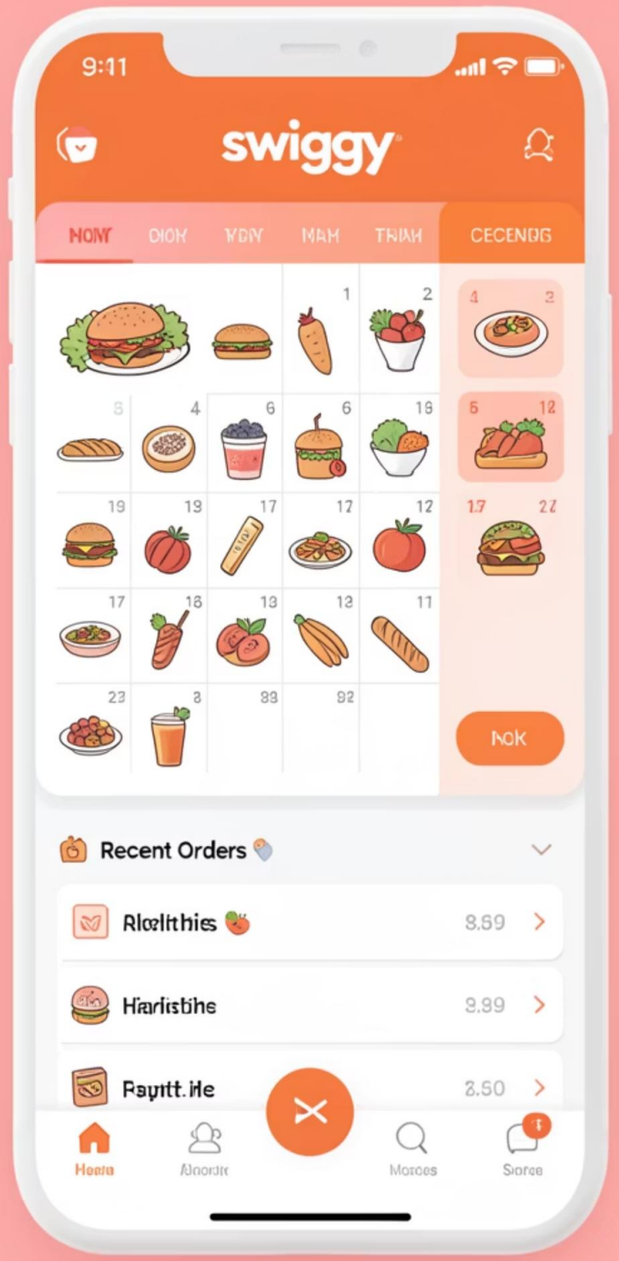
Market Analysis

Reveals ordering patterns and customer behavior specific to Mumbai market.

```
##8.    Find the number of orders placed by each customer in 'Mumbai'.
select distinct  name, count(order_id) as order_count
from orders
join customers using (customer_id)
where city = 'Mumbai'
group by name;
```

Result Grid | Filter Rows:

name	order_count
Amit Sharma	2
Rajesh Gupta	3
Arjun Desai	2
Ravi Singh	2



QUERY 9

Display all orders placed in the last 30 days

Recent Order Analysis

Focuses on time-based filtering to identify immediate sales trends, assess campaign effectiveness, and optimize inventory based on current customer demand.

```sql
##9.    Display all orders placed in the last 30 days.
SELECT *
FROM orders
WHERE order_date >= (SELECT DATE_SUB(MAX(order_date), INTERVAL 30 DAY) FROM orders);
```

Result Grid | Filter Rows: | Edit: | Export/Import:

order_id	customer_id	restaurant_id	order_date	total_amount	status
1	1	3	2024-08-01 00:00:00	750.00	Completed
2	2	5	2024-08-02 00:00:00	600.00	Completed
3	3	1	2024-08-04 00:00:00	0.00	Cancelled
4	4	7	2024-08-01 00:00:00	850.00	Completed
5	5	2	2024-08-03 00:00:00	1200.00	Completed
6	1	4	2024-08-06 00:00:00	500.00	Processing
7	6	8	2024-08-03 00:00:00	950.00	Completed
8	7	9	2024-08-08 00:00:00	700.00	Completed
9	8	6	2024-08-02 00:00:00	650.00	Completed
10	9	11	2024-08-09 00:00:00	0.00	Cancelled
11	10	12	2024-08-01 00:00:00	900.00	Completed
12	2	13	2024-08-04 00:00:00	550.00	Completed
13	3	14	2024-08-05 00:00:00	750.00	Completed
14	11	4	2024-08-06 00:00:00	800.00	Processing
15	12	15	2024-08-10 00:00:00	1100.00	Completed
16	13	10	2024-08-01 00:00:00	950.00	Completed
17	14	7	2024-08-11 00:00:00	0.00	Cancelled

QUERY 10

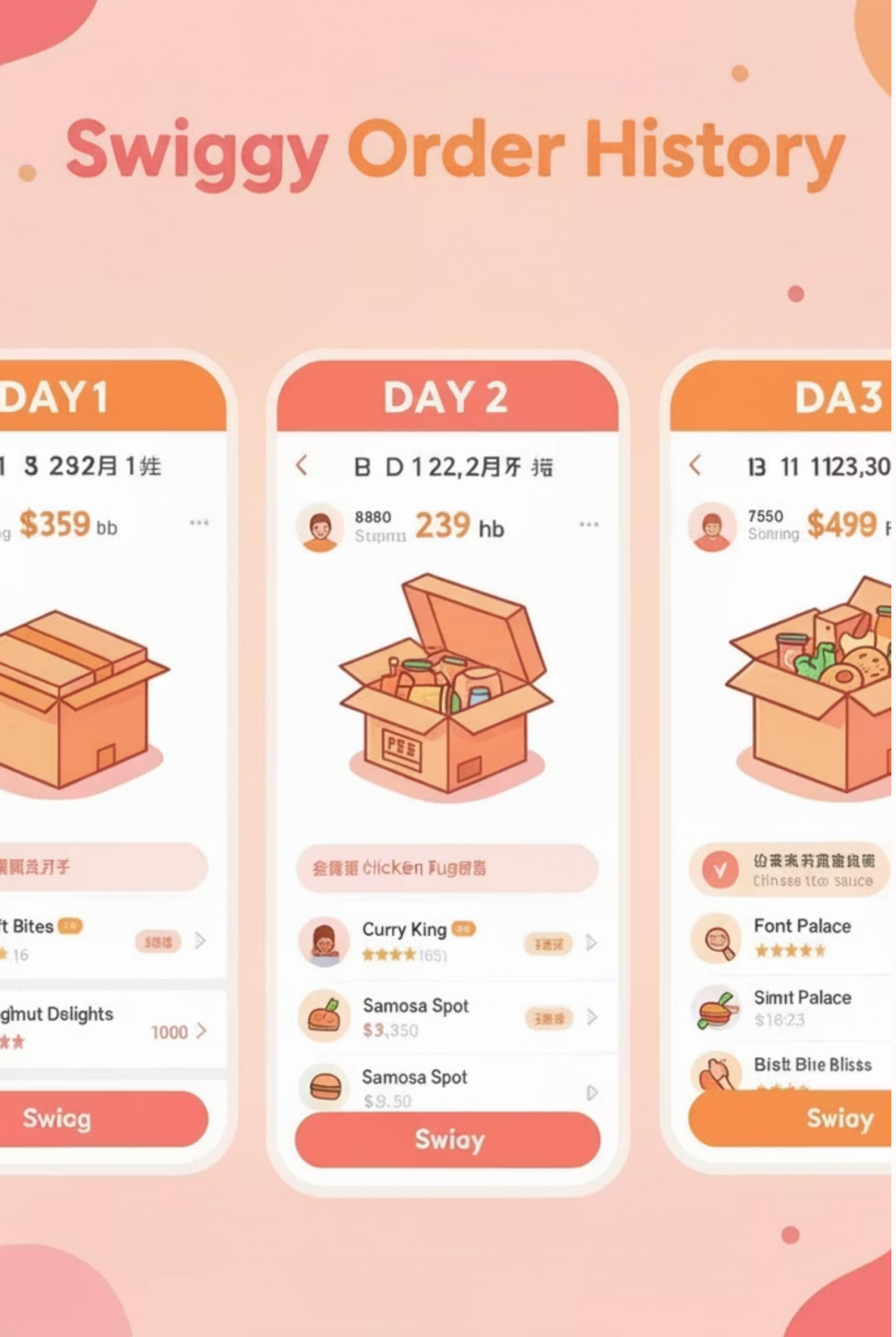
List all delivery partners who have completed more than 1 delivery

This query identifies delivery partners who have completed more than one delivery. By applying filtering and aggregation, it pinpoints active and experienced personnel. This allows for the identification of reliable partners, crucial for optimizing delivery operations and customer satisfaction.

```sql
##10.   List all delivery partners who have completed more than 1 delivery
select distinct name, count(order_delivery_id) as deliverycount
from deliverypartners
join orderdelivery
using (partner_id)
group by name
having count(order_delivery_id)  > 1
order by deliverycount desc;
```

Result Grid | Filter Rows:

name	deliverycount
Suresh Reddy	6
Ravi Kumar	5
Anita Desai	4
Rajesh Gupta	4
Priya Patel	3
Sonia Agarwal	3
Amit Sharma	2
Vikram Singh	2
Sneha Iyer	2
Reena Rao	2
Mohit Saini	2
Ritika Sharma	2



QUERY 11

Find the customers who have placed orders on exactly three different days

This query identifies customers who have demonstrated a specific ordering frequency pattern by placing orders on exactly three distinct days. This is crucial for segmenting customers based on their engagement level and ordering consistency, allowing for targeted marketing strategies or loyalty programs. Technically, it utilizes the COUNT DISTINCT aggregation on the order dates within a GROUP BY clause for customers, subsequently filtering these groups using HAVING to pinpoint those with precisely three unique order days.

Result Grid

name
Anjali Patel
Ashok Kumar
Nidhi Saxena
Priya Singh
Rohini Verma
Sonali Mishra

```sql
##11.   Find the customers who have placed orders on exactly three different days.
select distinct name
from orders
join customers
using (customer_id)
group by name
having count(distinct(order_date)) = 3;
```

QUERY 12

Find the delivery partner who has worked with the most different customers

This query identifies the delivery partner with the highest customer diversity by counting the unique customers each partner has served. Utilizing `COUNT DISTINCT` and `GROUP BY` clauses, it aggregates data to pinpoint partners with the broadest customer reach. This insight helps recognize top-performing delivery partners based on their extensive customer interactions and operational scope.

```sql
##12.   Find the delivery partner who has worked with the most different customers.
select name as delivery_partner,
COUNT(distinct customer_id) as unique_customers
from deliverypartners
join orderdelivery using (partner_id)
join orders using (order_id)
group by name
order by unique_customers desc
limit 1;
```

Result Grid | Filter Rows:

delivery_partner	unique_customers
Suresh Reddy	6

QUERY 13

Identify customers who have the same city and have placed orders at the same restaurants, but on different dates

Query Purpose

This query identifies customer groups within the same city who have patronized the same restaurants on different occasions. This reveals repeat customer patterns and restaurant loyalty across a geographic region, allowing for targeted engagement strategies. It utilizes JOIN operations and GROUP BY clauses to find customers sharing city and restaurant, while ensuring their order dates are distinct.

```sql
##13.   Identify customers who have the same city and have placed orders at the same restaurants, but on different dates.
SELECT DISTINCT c1.name AS customer1, c2.name AS customer2,
c1.city, o1.restaurant_id, r.name, o1.order_date as orderdate1, o2.order_date as orderdate2
FROM customers c1
JOIN orders o1 USING(customer_id)
JOIN customers c2 ON c1.city = c2.city AND c1.customer_id < c2.customer_id or c1.customer_id > c2.customer_id
JOIN orders o2 ON c2.customer_id = o2.customer_id
              AND o1.restaurant_id = o2.restaurant_id
              AND DATE(o1.order_date) != DATE(o2.order_date)
join restaurants r on r.restaurant_id = o1.restaurant_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer1	customer2	city	restaurant_id	name	orderdate1	orderdate2
Kavita Deshmukh	Manish Kumar	Pune	12	Flavours of Bengal	2024-08-01 00:00:00	2024-08-12 00:00:00
Kavita Deshmukh	Vikas Reddy	Pune	12	Flavours of Bengal	2024-08-01 00:00:00	2024-08-09 00:00:00
Pankaj Jain	Sneha Mehta	Surat	10	Andhra Spice	2024-08-01 00:00:00	2024-08-09 00:00:00
Nidhi Saxena	Sneha Mehta	Lucknow	7	Coastal Delight	2024-08-11 00:00:00	2024-08-01 00:00:00
Manish Kumar	Amit Sharma	Delhi	3	Biryani House	2024-08-04 00:00:00	2024-08-01 00:00:00
Manish Kumar	Sonali Mishra	Delhi	3	Biryani House	2024-08-04 00:00:00	2024-08-05 00:00:00
Karan Kapoor	Rajesh Gupta	Noida	1	Spice of India	2024-08-01 00:00:00	2024-08-04 00:00:00
Karan Kapoor	Priya Singh	Noida	1	Spice of India	2024-08-01 00:00:00	2024-08-14 00:00:00
Sonali Mishra	Amit Sharma	Delhi	3	Biryani House	2024-08-05 00:00:00	2024-08-01 00:00:00
Sonali Mishra	Manish Kumar	Delhi	3	Biryani House	2024-08-05 00:00:00	2024-08-04 00:00:00
Sonali Mishra	Anjali Patel	Delhi	3	Biryani House	2024-08-05 00:00:00	2024-08-01 00:00:00
Sonali Mishra	Manish Kumar	Delhi	3	Biryani House	2024-08-05 00:00:00	2024-08-07 00:00:00
Arjun Desai	Ravi Singh	Mumbai	8	Veggie Delight	2024-08-03 00:00:00	2024-08-09 00:00:00
Shweta Bansal	Amit Sharma	Bhopal	4	Curry Pot	2024-08-03 00:00:00	2024-08-06 00:00:00

Thank You!

Kunal Ramrakhya

ramrakhyakunal@gmail.com

https://www.linkedin.com/in/kunal-ramrakhya-229860175

https://github.com/ramrakhyakunal