Phase 1: Tasks 1 and 2

Team 2: Tsyr Rau Chen, Arailym Duisengali, Sheikh Noohery, Lo Ying Wu, Kuan Rong Yang

Task 1: Business Questions and Summary for Food Delivery

Business Summary

This database represents the operational ecosystem of a food-delivery platform, capturing the interactions among customers, restaurants, delivery partners, and orders. The domain mirrors real-world food-delivery services such as DoorDash or Uber Eats, where customers place orders from restaurants, and delivery partners fulfill those orders. The data reflects the full lifecycle of a transaction—from signup and ordering to delivery and payment—providing a comprehensive foundation for analytics, operational monitoring, and business decision-making.

The customer's entity stores user profile information, including contact details, signup dates, and home cities. Restaurants contribute essential supply-side data such as cuisine type, location, and typical preparation times. Menu items further define each restaurant’s offerings, including food categories and prices. These structures help the platform understand supply variety, geographic distribution, and menu-level economics.

The orders table forms the transactional backbone of the system. Each order links a customer to a restaurant, records the timestamp, status, and associated delivery fee, and connects to order_items, which detail the specific items and quantities purchased. This granularity supports revenue analysis, demand forecasting, basket-size insights, and menu performance evaluations.

On the fulfillment side, the delivery_partners and deliveries tables track the workforce responsible for completing deliveries. Data captured includes driver signup dates, vehicle type,

pickup/dropoff timestamps, distance traveled, and tip amounts. These fields support operational efficiency analysis, driver performance metrics, route optimization, and customer satisfaction modeling.

Overall, the domain represents a tightly integrated food-delivery marketplace where customer behavior, restaurant operations, and delivery logistics converge. This structured data enables the business to monitor service quality, revenue generation, delivery speed, geographic performance, and marketplace health across the entire platform.

Key Business Processes Represented in the Data

1. Customer Onboarding & Profile Management
    - Capturing customer identity, contact information, and signup dates
    - Tracking customer city for geographic segmentation and service availability
2. Restaurant Onboarding & Menu Management
    - Maintaining restaurant identity, cuisine, prep times, and locations
    - Managing menus, categories, and pricing for all items offered

3. Order Placement & Transaction Management
    - Customers place orders from restaurants
    - System records order time, payment-related details (delivery fee), and status
    - Order items specify the contents of each purchase

4. Delivery Fulfillment & Logistics
    - Assigning delivery partners to orders
    - Recording pickup/dropoff times, distance traveled, and tips
    - Enables tracking of delivery speed, operational efficiency, and partner compensation

5. Marketplace Operations Monitoring
    - Linking customer demand with restaurant supply
    - Measuring service quality (prep time, delivery duration, order status)
    - Supporting revenue analysis, order volume trends, and platform performance metrics

ERD

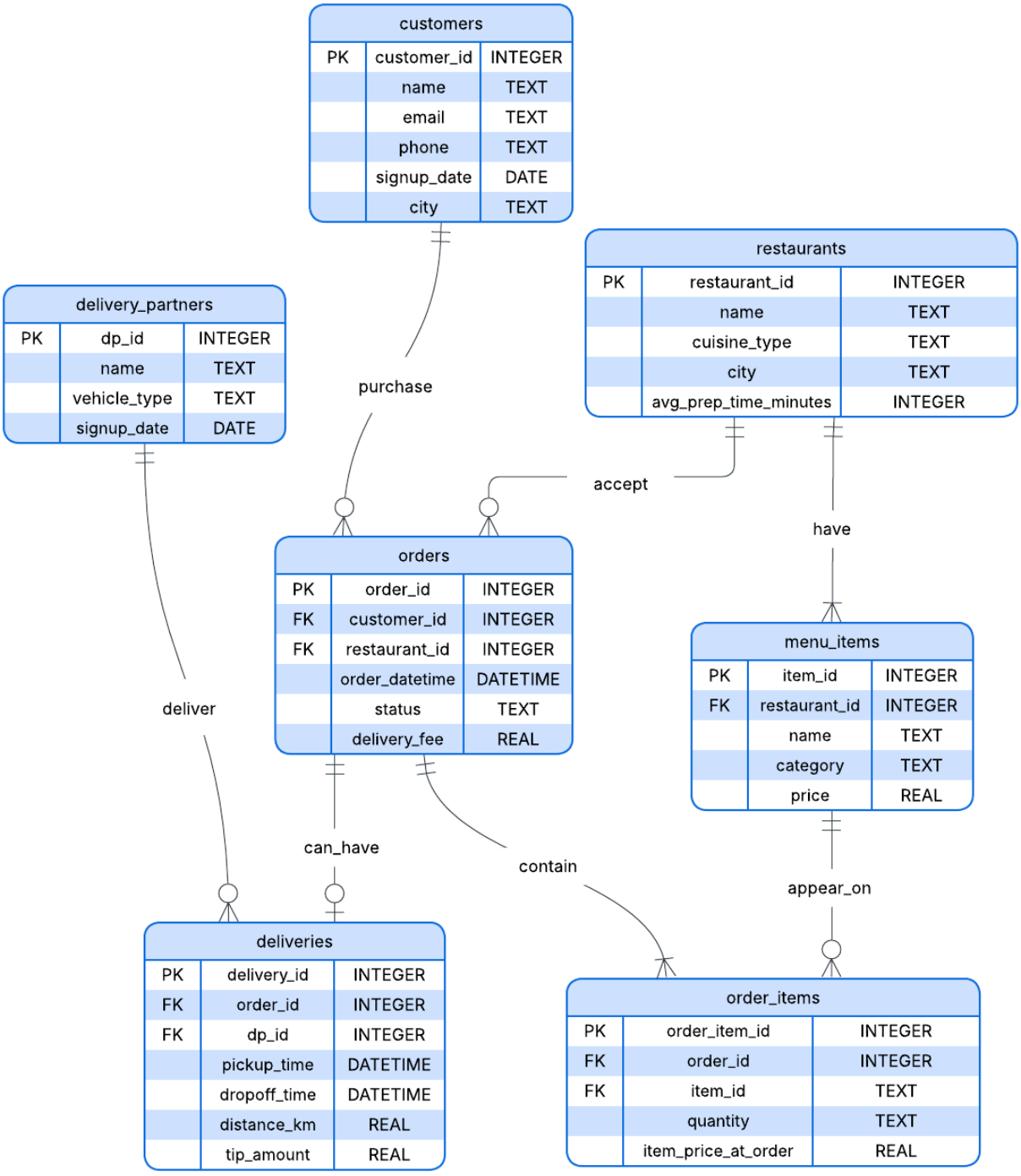


Figure 1: Entity Relationship Diagram of the food_delivery Database

Data Dictionary

Table 1: Data Dictionary for “customers” table

customers			
Field Name	**Data Type**	**Constraint**	**Description**
customer_id	integer	Primary key	Unique customer identifier
name	text	Not null	Customer full name
email	text	Unique, not null	Customer email address
phone	text		Customer phone number
signup_date	numeric	Not null	Date customer joined the platform
city	text		Customer home city

Table 2: Data Dictionary for “restaurants” table

restaurants			
Field Name	**Data Type**	**Constraint**	**Description**
restaurant_id	integer	Primary key	Unique restaurant identifier
name	text	Not null	Restaurant name
cuisine_type	text	Not null	Type of cuisine
city	text	Not null	City where restaurant in located
avg_prep_time_minutes	integer	Not null	Average food preparation time

Table 3: Data Dictionary for “menu_items” table

menu_items			
Field Name	**Data Type**	**Constraint**	**Description**
item_id	integer	Primary key	Menu item ID
restaurant_id	integer	Foreign key	Restaurant ID
name	text	Not null	Menu item name

menu_items			
Field Name	**Data Type**	**Constraint**	**Description**
category	text	Not null	Food category
price	real	Not null	Item price

Table 4: Data Dictionary for “delivery_partners” table

delivery_partners			
Field Name	**Data Type**	**Constraint**	**Description**
dp_id	integer	Primary key	Deliver partner ID
name	text	Not null	Partner name
vehicle_type	text	Not null	Driver vehicle category
signup_date	numeric	Not null	Deliver onboarding date

Table 5: Data Dictionary for “orders” table

orders			
Field Name	**Data Type**	**Constraint**	**Description**
order_id	integer	Primary key	Unique order identifier
customer_id	integer	Foreign key to customers table	Customer placing the order
restaurant_id	integer	Foreign key to restaurants table	Restaurant fulfilling the order
order_datetime	numeric		Timestamp when order was placed
status	text		Current order status (e.g., Completed, Cancelled)
delivery_fee	real		Fee charged for delivery

Table 6: Data Dictionary for “order_items” table

order_items			
Field Name	**Data Type**	**Constraint**	**Description**
order_item_id	integer	Primary key	Unique identifier for each item in an order
order_id	integer	Foreign key to orders table	Reference to the parent order
item_id	integer	Foreign key to menu_items table	Unique menu item identifier
quantity	integer		Number of units of this item ordered
item_price_at_order	real		Price per item at order time

Table 7: Data Dictionary for “deliveries” table

deliveries			
Field Name	**Data Type**	**Constraint**	**Description**
delivery_id	integer	Primary key	Unique delivery record identifier
order_id	integer	Foreign key to orders table	Delivery linked to specific order
dp_id	integer	Foreign key to delivery_partners table	Delivery partner (driver) identifier
pickup_time	numeric		Timestamp when order was picked up
dropoff_time	numeric		Timestamp when order was delivered
distance_km	real		Distance traveled during delivery (kilometers)
tip_amount	real		Tip paid to delivery partner

10 Questions

1. What are the top 10 restaurants based on delivered order quantity and their average delivery fee?
2. Who are the highest-spending customers in each city and what is their preferred cuisine?
3. Which dishes are most frequently ordered together as “combos” at each restaurant?
4. How does the average bill change depending on delivery time?
5. What is the average time interval between orders for the top three most frequent customers?
6. Which restaurants have the highest number of refunded orders and what are the characteristics of these orders?
7. Which restaurants show the strongest monthly “order-to-repeat-order” rate?
8. How does dish preparation time influence the total delivery duration?
9. What is the productivity of each delivery partner in every city, measured by deliveries per hour and average distance traveled?
10. What is the relationship between the distance of an order and its tip?

Task 2: Food Delivery SQL Queries

1 ANALYTICAL QUESTION

How does dish preparation time influence the total delivery duration?

SQL Query and Explanation

The prep_time value shows how long a restaurant's kitchen usually takes to make an order. The delivery_duration_minutes measures the actual travel and waiting time, calculated by subtracting the pickup time from the dropoff time. Using strftime('%s') ensures the timing works properly in SQLite. By grouping orders according to their prep time, we can see how different preparation speeds affect delivery results. The average of delivery_duration_minutes reveals how long customers typically waited from pickup to dropoff.

```sql
WITH delivery_stats AS (
    SELECT
        o.order_id,
        o.restaurant_id,
        r.avg_prep_time_minutes AS prep_time,
        d.pickup_time,
        d.dropoff_time,
        -- SQLite uses strftime to compute time differences in minutes
        (strftime('%s', d.dropoff_time) - strftime('%s', d.pickup_time)) / 60.0
            AS delivery_duration_minutes
    FROM orders o
    JOIN deliveries d ON o.order_id = d.order_id
    JOIN restaurants r ON o.restaurant_id = r.restaurant_id
    WHERE d.pickup_time IS NOT NULL
      AND d.dropoff_time IS NOT NULL
)
SELECT
    prep_time,
    AVG(delivery_duration_minutes) AS avg_delivery_duration
FROM delivery_stats
GROUP BY prep_time
ORDER BY prep_time;
```

Figure 2: SQL query for dish preparation time and total delivery duration

Interpretation

The results show that restaurants with longer average preparation times tend to have slightly higher delivery durations. Although the increase is not huge, there is a clear pattern: as preparation time goes up, the total delivery duration generally rises as well. This makes sense

because couriers must wait at the restaurant until the food is ready, so longer prep times push the whole delivery window later.

Even a few minutes of extra prep time can accumulate into noticeable differences, especially during peak hours or when couriers batch orders.

Output

prep_time	avg_delivery_duration
10	25.4917355371901
11	24.3662790697674
12	24.8337236533958
13	24.9844720496894
14	25.1316614420063
15	24.2564841498559
16	24.2797427652733
17	25.2164948453608
18	24.709219858156
19	24.9117647058824
20	25.2881944444444
21	25.1899224806202
22	25.2163120567376
23	25.1714285714286
24	24.7517006802721
25	24.3624678663239
26	25.4528795811518
27	24.5420875420875
28	25.4769874476987
29	25.6884210526316
30	24.8745762711864
31	25.0047169811321
32	25.4556113902848
33	25.3589743589744
34	25.3023255813953
35	25.4792899408284

Figure 3: Results for how preparation time affects average delivery duration

Why Does It Matter to Food Delivery?

Understanding the relationship between preparation time and delivery duration helps identify where delays originate. Even if drivers travel efficiently, long kitchen prep times can still lead to slow deliveries and unhappy customers. Food-delivery platforms can use this insight to:

- Adjust estimated delivery times for high-prep restaurants
- Recommend operational improvements for restaurants with slow prep times
- Improve courier assignment by reducing unnecessary waiting
- Identify which restaurants are causing bottlenecks in the system

This helps improve accuracy, customer satisfaction, and the overall efficiency of delivery operations.

5 STANDARD ANALYTICAL QUERIES

Question 1: Which cuisine types generate the highest average order value?

SQL Query and Explanation

```sql
-- Which cuisine types generate the highest average order value?
SELECT
    cuisine_type AS "Cuisine Type",
    AVG(order_total) AS "Average Order Value"
FROM (
    -- Calculate total value for each delivered order
    SELECT
        o.order_id,
        r.cuisine_type,
        SUM(oi.item_price_at_order * oi.quantity) AS order_total
    FROM orders o
    JOIN restaurants r
        ON r.restaurant_id = o.restaurant_id
    JOIN order_items oi
        ON o.order_id = oi.order_id
    WHERE o.status = 'delivered'
    GROUP BY o.order_id, r.cuisine_type
) AS order_totals
GROUP BY cuisine_type
ORDER BY "Average Order Value" DESC;
```

Figure 3: SQL query for cuisine types and their average order value

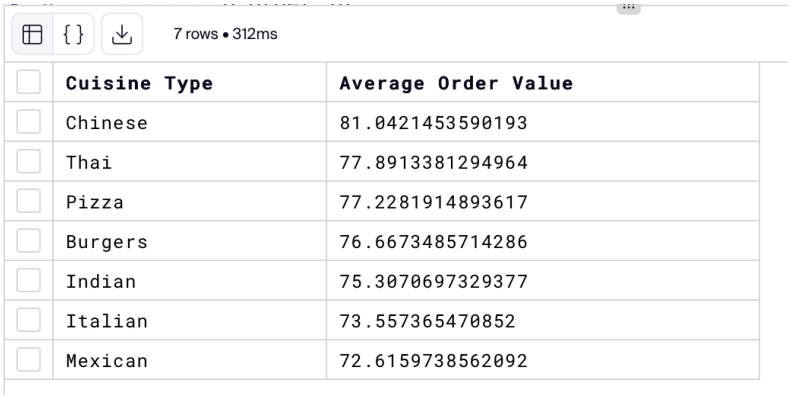
To answer this question, i used 3 tables:

- *restaurants* - identifies the **cuisine type**
- *orders* - contains **order-level attributes** and **order status**
- *order_items* - provides **item prices and quantities** needed to compute order totals

The query consists of two parts:

1. **Subquery:** calculates the total value of each delivered order by summing item_price_at_order × quantity.
2. **Outer query:** groups these order totals by cuisine type and computes the **average order value** for each cuisine

Output



7 rows • 312ms

Cuisine Type	Average Order Value
Chinese	81.0421453590193
Thai	77.8913381294964
Pizza	77.2281914893617
Burgers	76.6673485714286
Indian	75.3070697329377
Italian	73.557365470852
Mexican	72.6159738562092

Figure 4: Results for cuisine types and their average order value

Interpretation

Chinese cuisine has the highest average order value at ~$81.04, suggesting customers spend more on these orders. Thai, Pizza, and Burgers follow closely, all averaging between $76-$78. Indian, Italian, and Mexican cuisines have slightly lower averages, ranging from $72-$75.

Why Does It Matter to Food Delivery?

Identifying which cuisine types generate the highest average order value helps food delivery businesses optimize their pricing strategy. If certain cuisines consistently bring in larger orders, platforms and restaurants can adjust pricing, introduce premium bundles, or experiment with dynamic pricing to capture more value. For lower-value cuisines, this insight can guide discount structures, combo deals, or minimum order thresholds to increase profitability.

Overall, understanding cuisine-level spending patterns allows companies to price more effectively, maximize margins, and align menu strategies with customer willingness to pay.

Question 2: How often and how recently do customers place orders?

SQL Query and Explanation

```sql
--2. How often and how recently do customers place orders?
WITH customer_orders AS (
  SELECT customer_id,
         COUNT(order_id) AS total_orders,  -- Total number of orders placed by each customer
         MAX(order_datetime) AS last_order,  -- Most recent order date per customer
         MIN(order_datetime) AS first_order,  -- First order date per customer
         JULIANDAY(MAX(order_datetime)) - JULIANDAY(MIN(order_datetime)) AS active_days
  FROM orders
  GROUP BY customer_id
)
SELECT customer_id,
       total_orders,
       last_order,
       ROUND(active_days / total_orders, 2) AS avg_days_between_orders -- Average days between orders per customer
FROM customer_orders;
```

Figure 5: SQL query to calculate customer order frequencies

We used the order table to count how many orders customers have placed. We also find the dates of their first and most recent orders, and calculate how many days they've been using the service. Then, we calculate the average number of days between their orders by dividing the active period by the total number of orders.

Output

500 rows • 267ms

customer_id	total_orders	last_order	avg_days_between_orders
1	24	2025-11-13 13:03:03	10.74
2	13	2025-10-31 21:54:44	18.33
3	21	2025-11-09 19:13:46	11.67
4	17	2025-11-07 18:46:13	14
5	20	2025-11-13 17:15:09	12.66
6	23	2025-11-07 17:32:42	11.22
7	18	2025-11-15 17:46:17	13.77
8	14	2025-09-23 22:56:37	14.74
9	13	2025-11-08 22:30:43	13.31
10	13	2025-11-07 22:40:43	19.39
11	17	2025-11-07 21:23:30	14.12
12	17	2025-10-25 19:02:11	14.06
13	22	2025-10-06 19:07:36	10.36
14	23	2025-11-01 22:55:09	10.57
15	22	2025-11-12 20:09:25	10.72
16	24	2025-11-09 22:08:27	10.42
17	18	2025-11-08 20:58:07	14.45
18	12	2025-11-15 21:45:10	22.17
19	19	2025-11-15 18:03:45	12.26
20	16	2025-10-15 22:11:50	14.69

Figure 6: Results for customer order frequencies

Interpretation

Customers with shorter average days between orders are more engaged and are more valuable to the platform. Customers with bigger gaps between orders can require special marketing or promotions to improve ordering frequency and retention.

Why Does It Matter to Food Delivery?

It can segment customers to identify and retain their loyalty, but also identify risks for reactivating existing customers. In addition, it can analyze customers' purchase frequency as well as how long it has been since they last purchased to create campaigns for retaining customers.

Question 3: Which delivery partners are fastest and which are slowest?

SQL Query and Explanation

Two tables were used to solve this question: delivery_partners and deliveries. To find out which delivery partners are the fastest and which are the slowest, we calculated the average speed in kilometers per hour for each delivery partner. We used the JULIANDAY function to convert the pick up and drop off datetime values into Julian day numbers. This conversion allowed us to perform mathematical operations on the datetime values. Then, we used the ORDER BY clause to sort the results in descending order for the fastest, and ascending order for the slowest delivery partners.

```
SELECT dp.dp_id,
  dp.name,

---calculate average distance: kilometers per hour------------
  ROUND(AVG(d.distance_km / ((JULIANDAY(d.dropoff_time) -
  JULIANDAY(d.pickup_time)) * 24)), 3) AS avg_speed_km_per_hr,

  dp.vehicle_type
FROM deliveries d
JOIN delivery_partners dp ON d.dp_id = dp.dp_id
GROUP BY d.dp_id

----to sort the result view using ASC or DESC------
ORDER BY avg_speed_km_per_hr DESC;
```

Figure 7: SQL query for fastest and slowest delivery partners

Output

dp_id	name	avg_speed_km_per_hr	vehicle_type
78	Nathan Richards	13.197	bike
43	Patty Holt	13.14	scooter
24	James Thomas	13.005	car
76	Stephanie Armstrong	12.865	scooter
52	Beth James	12.833	car
72	Lauren Morse	12.742	car
34	Joanna Jones	12.607	car
35	Shelley Lucas	12.493	bike
18	Taylor Patton	12.445	scooter
50	Dawn Mack DDS	12.362	car

Figure 8: Top 10 fastest delivery partners based on average speed in kilometers per hour

dp_id	name	avg_speed_km_per_hr	vehicle_type
73	Alan Johnson	9.396	car
41	Rachel Hayes	9.468	bike
17	Jeanette Kaiser	9.544	bike
61	Matthew Foster	9.95	bike
80	Darrell Anderson	9.972	scooter
22	Carrie Chavez	10.031	scooter
85	Connie Gonzalez	10.042	car
15	Martha Torres	10.067	car
49	Christopher Perez	10.072	scooter
53	Denise Perkins	10.169	scooter

Figure 9: Top 10 slowest delivery partners based on average speed in kilometers per hour

Interpretation

While the slowest 10 delivery partner's average speed ranges from 9-10 km/h, the fastest 10 ranges from 12-13 km/h. Nathan Richards, who uses a bike for delivery, is the fastest delivery partner with an average speed of ~13.2 km/h. Alan Johnson, who uses a car, is the slowest with an average speed of ~9.4 km/h. There is a range of vehicle types represented in both results: bike, car, and scooter. And there are speed variations within the same vehicle categories, which means delivery speed depends on the individual delivery partner and

whatever conditions they face during their journey. We should investigate why certain partners are slower than others to find strategies for speed improvements.

Why Does It Matter to Food Delivery?

Speed is a critical factor of food delivery businesses because meals are highly time-sensitive. Faster deliveries ensure that customers receive their food at the optimal temperature and freshness. Also, customer satisfaction gained from faster deliveries will give our food delivery service a competitive edge in a crowded market. The importance of speed makes delivery partners the backbone of the food delivery system.

Fast and reliable partners increase operational efficiency by completing more deliveries per shift, while slower ones can cause delays and harm customer satisfaction. For example, a delivery at 10 km/h for a 5 km distance takes 30 minutes, while the same distance at 13 km/h takes ~23 minutes. The 7-minute difference is significant, especially when completing multiple orders during a day. This business question helps us determine which delivery partners are efficient and which ones need to find better strategies to improve their speed.

Question 4: During which hours or days are cancellations most common?

SQL Query and Explanation

```
SELECT
    STRFTIME('%H', order_datetime) AS hour,
    COUNT(*) AS cancel_count
FROM orders
WHERE status = 'canceled'
GROUP BY hour
ORDER BY cancel_count DESC;
```

Figure 10: SQL query to calculate order cancellations throughout different hours of the day

The query looks at cancelled orders and breaks them down by the hour they were placed. It pulls data from the orders table, filters for cancellations, and extracts the hour from each order's date time value. Then it counts how many cancellations happened in each hour and sorts the results to show which times of day tend to have the most cancellations.

Output

12 rows • 206ms

hour	cancel_count
18	151
21	147
19	147
20	131
17	124
22	110
11	32
12	30
14	29
16	27
13	27
15	22

Figure 11: Results showing order cancellations throughout different hours of the day

Interpretation

Most of the cancellations happen around dinner times, with 6 PM having the highest count at 151 cancellations. That value is followed by 147 cancellations at 9 PM and 7 PM , and 131 cancellations at 8 PM. In contrast, midday hours from 11 AM - 3 PM have far fewer cancellations, with counts ranging from 22 to 32.

Why Does It Matter to Food Delivery?

Understanding when cancellations are most common helps the platform and restaurants manage risk and resources more effectively. Lower cancellation rates lead to higher customer satisfaction, more stable revenue for restaurants and delivery partners, and better overall operational efficiency.

Question 5: How does tip percentage vary across different order amounts?

SQL Query and Explanation

This query calculates the total value of each order and includes the tip amount id there is one. Then, it groups orders into three price ranges: under $20, Between $20 and $50, and over

$50. It calculates the average tip percentage for each price range group, helping us see if people tip more or less depending on how much they spend.

```sql
WITH order_values AS (
    SELECT o.order_id,
           SUM(oi.quantity * oi.item_price_at_order) AS order_value, -- Total order amount
           d.tip_amount                                              -- Tip given for the order, can be NULL
    FROM orders o
    JOIN order_items oi ON o.order_id = oi.order_id -- Join to get all items in the order
    LEFT JOIN deliveries d ON o.order_id = d.order_id -- Left join to include tip info if delivery exists
    GROUP BY o.order_id, d.tip_amount -- Group by order and tip amount to aggregate total order value
)
SELECT CASE
           WHEN order_value < 20 THEN '< $20' -- Price category
           WHEN order_value BETWEEN 20 AND 50 THEN '$20-$50'
           ELSE '> $50'
       END AS order_range,
       AVG(COALESCE(tip_amount, 0) / order_value * 100) AS avg_tip_percent -- Avg tip % per category
FROM order_values
GROUP BY order_range  -- Group by defined order range buckets
ORDER BY order_range; -- Sort results by order range categories
```

Figure 12: SQL query to calculate tip percentage variations across different order amounts

Output

order_range	avg_tip_percent
$20-$50	10.7167928568881
< $20	11.1049488104589
> $50	10.6789204829648

Figure 12: Results for tip percentage variations across different order amounts

Interpretation

The minor decrease in tip percentages for larger orders indicates that perhaps consumers tend to tip individuals serving them slightly more for smaller orders. Maybe it is because either as an incentive for good service or perhaps where fixed tips seem more substantial.

Why Does It Matter to Food Delivery?

This matters because delivery drivers rely on tips as part of their income. If tips are smaller on bigger orders, drivers might feel it's unfair since bigger orders usually take more time and effort. Knowing this helps the food delivery platform suggest fair tips and keep drivers

happy. It also helps the business improve customer experience and keep delivery running smoothly.

2 WINDOW FUNCTION QUERIES

Question 6: What is the sequence number of each customer's orders over time?

SQL Query and Explanation

We used attributes from the order table and customer table to answer this question. We used ROW_NUMBER function to number each customer's orders based on when they were placed. The first order is 1, the second is 2, and so on. The numbering resets for each customer, and the orders are sorted by date to keep the sequence accurate. In the result view, each customer's orders appear together and in the sequence order.

```
SELECT o.customer_id,
  c.name,
  o.order_id,
  o.order_datetime,

  ---window function to generate sequence number for customer orders---
  ROW_NUMBER() OVER(
    PARTITION BY o.customer_id ---for each customer
    ORDER BY o.order_datetime
  ) AS order_sequence

FROM orders o
JOIN customers c ON o.customer_id = c.customer_id
ORDER BY o.customer_id, order_sequence;
```

Figure 13: SQL query for customer order sequencing

Interpretation

Paul Sullivan (Customer 1) placed their first order on February 28, 2025 and has ordered monthly throughout the data period. This consistency makes Paul a recurring customer, who places at least one order per month. They also placed two separate orders on March 30, April 16, and July 11. Paul's multiple orders suggest higher demand during certain months. We can look at the order sequencing for each customer to find loyal customers like Paul, or customers who do not place orders as frequently.

Why Does It Matter to Food Delivery?

Sequencing orders helps us with customer life cycle analysis. Sequence 1 is acquisition, sequence 2-3 are early repeats, and the subsequent sequences tell us about our loyal customers. Loyal customers provide a steady revenue to the business and should be retained. We can also identify patterns in customer behavior to forecast demand and identify promotional strategies.

Output

customer_id	name	order_id	order_datetime	order_sequence
1	Paul Sullivan	293	2025-02-28 21:01:49	1
1	Paul Sullivan	932	2025-03-17 12:44:18	2
1	Paul Sullivan	1001	2025-03-20 21:31:31	3
1	Paul Sullivan	1379	2025-03-30 18:40:55	4
1	Paul Sullivan	1364	2025-03-30 19:40:13	5
1	Paul Sullivan	1952	2025-04-16 18:57:49	6
1	Paul Sullivan	1943	2025-04-16 21:28:12	7
1	Paul Sullivan	2062	2025-04-19 20:23:09	8
1	Paul Sullivan	3115	2025-05-19 14:46:48	9
1	Paul Sullivan	3666	2025-06-06 19:55:38	10
1	Paul Sullivan	4061	2025-06-15 19:48:34	11
1	Paul Sullivan	4960	2025-07-11 18:04:00	12
1	Paul Sullivan	4968	2025-07-11 19:13:05	13
1	Paul Sullivan	5106	2025-07-13 18:32:05	14
1	Paul Sullivan	6232	2025-08-16 21:22:42	15
1	Paul Sullivan	6935	2025-09-06 18:17:26	16
1	Paul Sullivan	7164	2025-09-13 16:15:57	17
1	Paul Sullivan	7282	2025-09-16 17:48:12	18
1	Paul Sullivan	7319	2025-09-17 16:30:16	19
1	Paul Sullivan	8104	2025-10-10 18:03:28	20
1	Paul Sullivan	8204	2025-10-13 19:56:51	21
1	Paul Sullivan	8780	2025-10-31 22:18:43	22
1	Paul Sullivan	9062	2025-11-07 20:22:59	23
1	Paul Sullivan	9239	2025-11-13 13:03:03	24
2	Stacey Woods	524	2025-03-07 14:22:16	1
2	Stacey Woods	727	2025-03-12 22:15:29	2
2	Stacey Woods	2224	2025-04-24 15:16:48	3
2	Stacey Woods	2634	2025-05-05 13:57:57	4
2	Stacey Woods	3589	2025-06-03 21:36:45	5
2	Stacey Woods	4850	2025-07-07 20:14:41	6

Figure 14: Partial view of customer order sequence numbers based on order time

Question 7: What is the cumulative spend of each customer across all their orders?

SQL Query and Explanation

```sql
WITH order_totals AS (
    SELECT
        o.order_id,
        o.customer_id,
        o.order_datetime,
        SUM(oi.quantity * oi.item_price_at_order) + o.delivery_fee AS order_total
    FROM orders o
    JOIN order_items oi ON o.order_id = oi.order_id
    WHERE o.status = 'delivered'
    GROUP BY o.order_id, o.customer_id, o.order_datetime
)
SELECT
    customer_id,
    order_id,
    order_datetime,
    order_total,
    SUM(order_total) OVER (
        PARTITION BY customer_id
        ORDER BY order_datetime
    ) AS cumulative_spend
FROM order_totals
ORDER BY customer_id, order_datetime;
```

Figure 15: SQL query to find the cumulative spend of each customer across all their orders

Output

7945 rows • 1s

customer_id	order_id	order_datetime	order_total	cumulative_spend
1	4061	2025-06-15 19:48:34	84.12	788.19
1	4960	2025-07-11 18:04:00	122.57	910.76
1	4968	2025-07-11 19:13:05	64.69	975.45
1	6232	2025-08-16 21:22:42	30.77	1006.22
1	6935	2025-09-06 18:17:26	50.9	1057.12
1	7164	2025-09-13 16:15:57	22.97	1080.09
1	7282	2025-09-16 17:48:12	25.2	1105.29
1	7319	2025-09-17 16:30:16	149.63	1254.92
1	8104	2025-10-10 18:03:28	36.06	1290.98
1	8204	2025-10-13 19:56:51	144.66	1435.64
1	8780	2025-10-31 22:18:43	111.56	1547.2
1	9062	2025-11-07 20:22:59	20.63	1567.83
1	9239	2025-11-13 13:03:03	62.59	1630.42
2	524	2025-03-07 14:22:16	142.4	142.4
2	2634	2025-05-05 13:57:57	84.44	226.84
2	3589	2025-06-03 21:36:45	44.74	271.58
2	4850	2025-07-07 20:14:41	34.02	305.6
2	6441	2025-08-22 18:07:03	94.24	399.84
2	7482	2025-09-22 19:17:26	36.73	436.57
2	7512	2025-09-23 19:35:04	75.48	512.05
2	7523	2025-09-23 20:42:38	114.48	626.53
2	8040	2025-10-08 21:10:38	79.27	705.8

Figure 16: Results showing cumulative spend of each customer across all their orders

Interpretation

For every order a customer places, we calculate the order amount and then add it to all previous spending. This creates a running total that increases with each new purchase. By looking at cumulative spend, we can see which customers buy frequently, which ones make larger purchases, and who the most valuable customers are. This helps the business understand long-term customer value and identify loyal or high-spending users more effectively.

Explanation

order_totals calculates the total value of each delivered order.
SUM(order_total) OVER (...) creates a running (cumulative) total.
PARTITION BY customer_id means each customer gets their own running total.
ORDER BY order_datetime ensures the spending increases in correct time order.

Why Does It Matter to Food Delivery?

Understanding the cumulative spend of each customer helps a business see how much a customer contributes over time, not just from a single order. By adding each new purchase to all previous spending, we can identify customers who consistently place orders, spend more than average, or show long-term loyalty. These high-value customers often drive a large share of total revenue. Knowing who they are allows the business to design targeted promotions, personalized offers, and loyalty rewards that match their behavior.