

Northern University Of Bangladesh

Department: E- Cse

Semester: 4th

Group: A

Group Name: Syntax Squad

Id Numbers: 42240301875, 42240301876, 42240301877, 42240301850,
42230200893.

Project Name: FoodieSom, An Integrated Food Delivery
Management System using SQL and Web Interface.

Why FoodiSom:

Body Point:

The Problem: Traditional food ordering systems often face data redundancy and slow report generation.

The Solution: An automated Relational Database System (RDBMS) to handle large-scale food delivery operations.

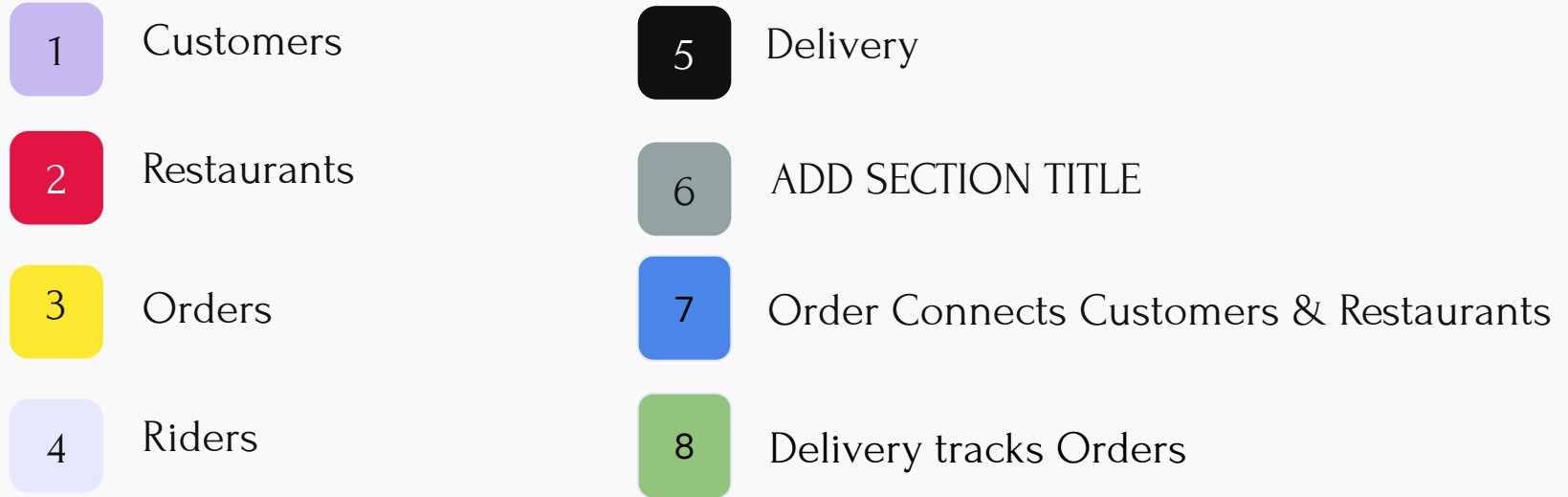
Key Scope:

Data Accuracy: Managing relationships between 5 core entities (Customers, Restaurants, Orders, Riders, and Deliveries).

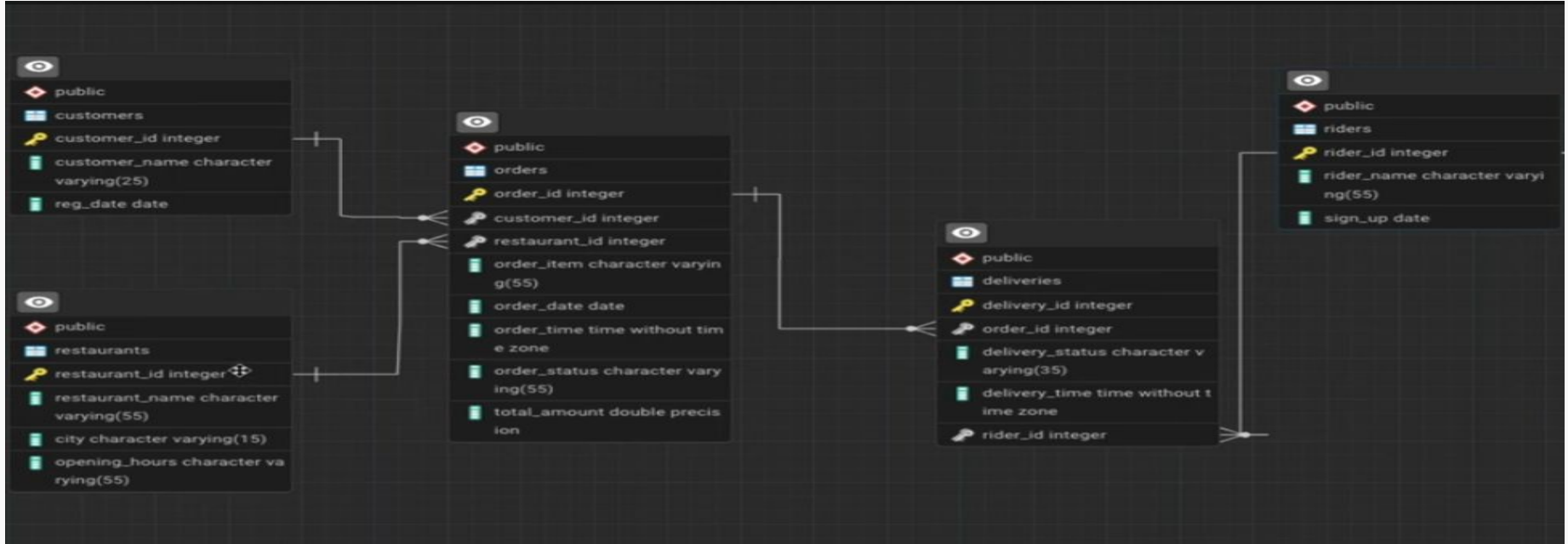
Real-time Analysis: Identifying peak business hours and high-value customers.

Operational Efficiency: Reducing undelivered orders through better tracking.

System Architecture (ER Diagram)

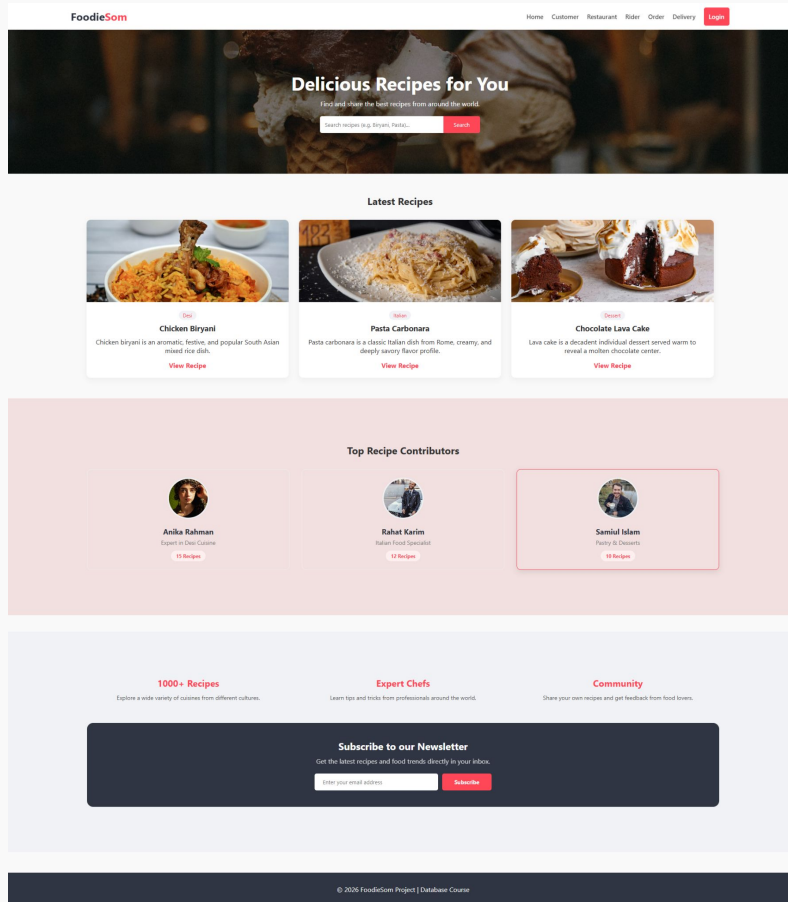


Visual ER Diagram



This diagram shows how Customers, Restaurants, Riders, and Orders are interconnected using Primary and Foreign Keys.

Frontend Implementation (The View)



The frontend interface of 'FoodieSom' is designed to act as a seamless bridge between the users and the underlying SQL database. Its primary purpose is to simplify complex database transactions, allowing users to interact with relational data through intuitive forms for Customer Registration, Rider Enrollment, and Order Placement. Each user action on the frontend triggers a structured data flow that is captured and stored within the SQL environment, ensuring high data integrity and consistency. Furthermore, the interface provides real-time visibility into the database's analytical outputs, such as live order status tracking (Pending/Delivered) and restaurant availability, making the power of SQL backend accessible to non-technical users.

FoodieSom

[Home](#) [Customer](#) [Restaurant](#) [Rider](#) [Order](#) [Delivery](#) [Login](#)

Add New Restaurant

Restaurant Name

City

Opening Hours

Add Restaurant

This interface acts as a dynamic gateway to the SQL backend, translating user inputs into structured relational data for seamless order management and real-time delivery tracking. It ensures high data integrity by directly communicating with the core database schemas for all registration and transaction processes.

Create New Order

Customer ID

Restaurant ID

Order Items

Total Amount (BDT)

Order Status

Pending

Confirm Order

This project utilizes advanced SQL techniques to transform raw transactional data into actionable business intelligence. By implementing complex joins, aggregations, and window functions like `DENSE_RANK()`, the system identifies critical trends such as peak ordering time slots and high-value customer segments who have spent over 100,000 BDT. Furthermore, the reporting logic monitors operational efficiency by tracking undelivered orders across different cities and restaurants, ensuring a data-driven approach to improving delivery performance and customer satisfaction.

Rider Enrollment

Rider Name

Sign Up Date



Register Rider

This project implements advanced SQL analytical logic to identify high-value customer segments, determine peak ordering hours, and monitor delivery efficiency across different cities. These data-driven reports provide critical business intelligence to optimize restaurant operations and improve overall service reliability

Add New Restaurant

Restaurant Name

City

Opening Hours

Add Restaurant

This section showcases the use of advanced SQL queries to derive business intelligence from the relational database. By implementing complex joins and window functions like `DENSE_RANK()`, the system identifies top-performing menu items and high-value customer segments with total expenditures exceeding 100,000 BDT. Additionally, the logic uses time-based aggregations to pinpoint peak order periods and operational filters to track undelivered orders by city and restaurant, providing a comprehensive data-driven overview of the delivery ecosystem.

Delivery Status Tracking

Order ID

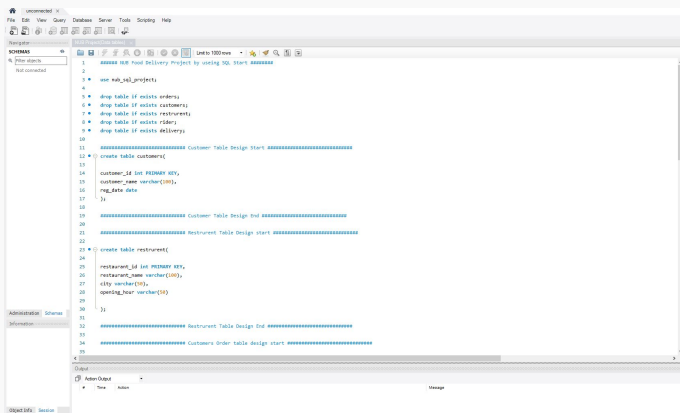
Rider ID

Delivery Status

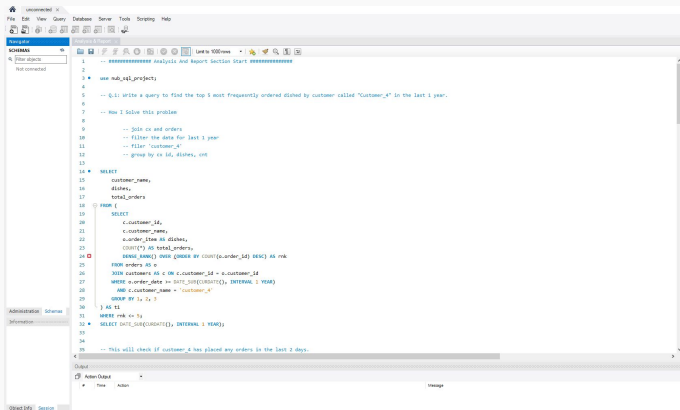
Delivery Time

Update Delivery

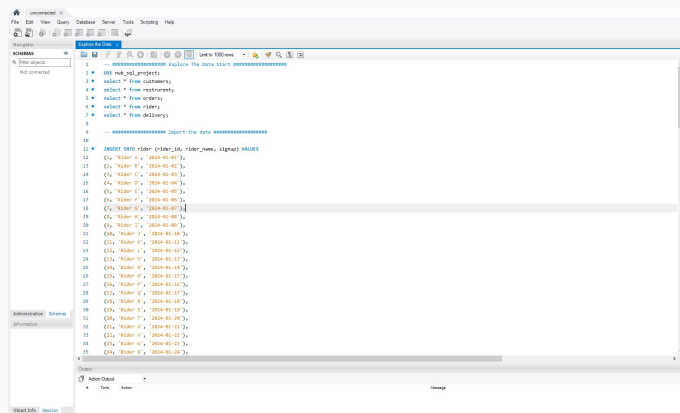
This project utilizes complex SQL joins and window functions to identify high-value customers, peak order times, and delivery efficiency across different cities. These queries transform raw data into actionable insights for optimized restaurant management.



```
1 --===== Not food delivery project by using SQL Start =====
2
3 use ndb_nlp_project;
4
5
6 -- drop table if exists orders;
7 -- drop table if exists customers;
8 -- drop table if exists restaurants;
9 -- drop table if exists riders;
10 -- drop table if exists delivery;
11
12 --===== Customer Table Design Start =====
13
14 create table customers(
15     customer_id int primary key,
16     customer_name varchar(255),
17     reg_date date
18 );
19
20 --===== Restaurant Table Design End =====
21
22 --===== Restaurant Table Design Start =====
23
24 create table restaurants(
25     restaurant_id int primary key,
26     restaurant_name varchar(255),
27     city varchar(255),
28     opening_hour varchar(255)
29 );
30
31 --===== Restaurant Table Design End =====
32
33 --===== Customer Order Table Design Start =====
34
35 create table orders(
36     order_id int primary key,
37     customer_id int,
38     restaurant_id int,
39     rider_id int,
40     order_date datetime,
41     delivery_status varchar(255)
42 );
```



```
1 --===== Insert Data Start =====
2
3 use ndb_nlp_project;
4
5
6 --===== Insert Data Start =====
7
8 --===== Insert Data Start =====
9
10 --===== Insert Data Start =====
11
12 --===== Insert Data Start =====
13
14 --===== Insert Data Start =====
15
16 --===== Insert Data Start =====
17
18 --===== Insert Data Start =====
19
20 --===== Insert Data Start =====
21
22 --===== Insert Data Start =====
23
24 --===== Insert Data Start =====
25
26 --===== Insert Data Start =====
27
28 --===== Insert Data Start =====
29
30 --===== Insert Data Start =====
31
32 --===== Insert Data Start =====
33
34 --===== Insert Data Start =====
35
36 --===== Insert Data Start =====
37
38 --===== Insert Data Start =====
39
40 --===== Insert Data Start =====
41
42 --===== Insert Data Start =====
43
44 --===== Insert Data Start =====
45
46 --===== Insert Data Start =====
47
48 --===== Insert Data Start =====
49
50 --===== Insert Data Start =====
51
52 --===== Insert Data Start =====
53
54 --===== Insert Data Start =====
55
56 --===== Insert Data Start =====
57
58 --===== Insert Data Start =====
59
60 --===== Insert Data Start =====
61
62 --===== Insert Data Start =====
63
64 --===== Insert Data Start =====
65
66 --===== Insert Data Start =====
67
68 --===== Insert Data Start =====
69
70 --===== Insert Data Start =====
71
72 --===== Insert Data Start =====
73
74 --===== Insert Data Start =====
75
76 --===== Insert Data Start =====
77
78 --===== Insert Data Start =====
79
80 --===== Insert Data Start =====
81
82 --===== Insert Data Start =====
83
84 --===== Insert Data Start =====
85
86 --===== Insert Data Start =====
87
88 --===== Insert Data Start =====
89
90 --===== Insert Data Start =====
91
92 --===== Insert Data Start =====
93
94 --===== Insert Data Start =====
95
96 --===== Insert Data Start =====
97
98 --===== Insert Data Start =====
99
100 --===== Insert Data Start =====
```



```
1 --===== Insert Data Start =====
2
3 use ndb_nlp_project;
4
5
6 --===== Insert Data Start =====
7
8 --===== Insert Data Start =====
9
10 --===== Insert Data Start =====
11
12 --===== Insert Data Start =====
13
14 --===== Insert Data Start =====
15
16 --===== Insert Data Start =====
17
18 --===== Insert Data Start =====
19
20 --===== Insert Data Start =====
21
22 --===== Insert Data Start =====
23
24 --===== Insert Data Start =====
25
26 --===== Insert Data Start =====
27
28 --===== Insert Data Start =====
29
30 --===== Insert Data Start =====
31
32 --===== Insert Data Start =====
33
34 --===== Insert Data Start =====
35
36 --===== Insert Data Start =====
37
38 --===== Insert Data Start =====
39
40 --===== Insert Data Start =====
41
42 --===== Insert Data Start =====
43
44 --===== Insert Data Start =====
45
46 --===== Insert Data Start =====
47
48 --===== Insert Data Start =====
49
50 --===== Insert Data Start =====
51
52 --===== Insert Data Start =====
53
54 --===== Insert Data Start =====
55
56 --===== Insert Data Start =====
57
58 --===== Insert Data Start =====
59
60 --===== Insert Data Start =====
61
62 --===== Insert Data Start =====
63
64 --===== Insert Data Start =====
65
66 --===== Insert Data Start =====
67
68 --===== Insert Data Start =====
69
70 --===== Insert Data Start =====
71
72 --===== Insert Data Start =====
73
74 --===== Insert Data Start =====
75
76 --===== Insert Data Start =====
77
78 --===== Insert Data Start =====
79
80 --===== Insert Data Start =====
81
82 --===== Insert Data Start =====
83
84 --===== Insert Data Start =====
85
86 --===== Insert Data Start =====
87
88 --===== Insert Data Start =====
89
90 --===== Insert Data Start =====
91
92 --===== Insert Data Start =====
93
94 --===== Insert Data Start =====
95
96 --===== Insert Data Start =====
97
98 --===== Insert Data Start =====
99
100 --===== Insert Data Start =====
```

This section showcases the operational modules of the 'FoodieSom' system, where user-driven actions are directly captured into the SQL backend. Through these interfaces, administrators can perform **Rider Enrollment** and **Restaurant Onboarding**, which automatically populates the respective relational tables with structured data. Furthermore, the **Delivery Status Tracking** module allows for real-time updates to the 'delivery' table, ensuring seamless synchronization between the order status and rider assignments within the database.

Future Enhancements

1. **City-based Revenue Ranking:** Rank restaurants by total annual revenue within each city.
2. **Localized Dish Popularity:** Identify the most ordered dishes per city for better inventory planning.
3. **Inactive Customer Analysis:** Re-engage customers who ordered in 2023 but have been inactive in 2024.
4. **Order Cancellation Trends:** Compare year-over-year cancellation rates to improve service reliability.
5. **Rider Efficiency Tracking:** Calculate average delivery times per rider to optimize performance.
6. **Dynamic Slot Management:** Refine peak-hour time slots based on seasonal ordering patterns.
7. **Operational Cost Monitoring:** Track undelivered order impacts on restaurant and city levels.



Conclusion: The Future of Data-Driven Delivery

In conclusion, the 'FoodieSom' project demonstrates how a robust SQL backend can effectively manage a complex food delivery ecosystem. By integrating real-time data entry through a user-friendly frontend with advanced analytical reporting, we have created a system that not only handles daily operations but also provides deep business insights. From tracking delivery performance to identifying high-value customers, this project highlights the power of structured data in making informed decisions and driving operational efficiency in the modern food industry.



Thank You

