Northeastern University

IE 6700 Data Management for Analytics

USECASE MILESTONE 1 & 2 & 3 & 4 & 5

FINAL REPORT

Integrated Hotel and Restaurant Management Database System

Vachan Srinivasagowda

Kuchibhotla Anirudha Raghava Sarma

srinivasagowda.v@northeastern.edu

kuchibhotla.an@northeastern.edu

Signature of Student 1: Vachan Srinivasagowda

Signature of Student 2: Kuchibhotla Anirudha Raghava Sarma

Submission Date: 12/06/2023

Problem Statement:

Over the past few years, there have been substantial changes in the hotel and restaurant management business, mostly due to changes in customer behavior and the effects of situations like the COVID-19 pandemic. Various types of establishments are adapting and looking for new tactics to be competitive and relevant in this industry that is changing. This adaptability includes looking into innovative approaches to meet various wants of customers. Our plan intends to alter the hotel and restaurant management industry by fusing independently owned hotels and the online restaurant model into a single, competitive service provider on a nationwide scale.

Theory:

Every Hotel and Restaurant needs an orderly and systematic way to keep their data given the incoming Customer and order details. The maintenance of All the entities in the databases must be handled by each Hotel and Restaurant's administration. These databases' records provide basic information including room details, order details, employee details, payment details, etc.

Our solution involves the development of an integrated system with multiple modules, including Customer, Employee, Delivery System, and Hotel/Restaurant Details. This system will automate and consolidate information, allowing data to be easily passed from one entity to another. Here's an example of how this system would work:

Customer will interact with the system for various purposes, such as making reservations, placing orders, or tracking their requests. They will have unique identifiers, such as a Cust ID, to access their information within the system.

Employees in hotels and restaurants will have access to the system to manage guest requests, orders, and reservations. When a guest inquires about the status of their order or reservation, the staff can easily retrieve this information by entering the customer's ID.

Our primary objective is data-driven approach, we aspire to offer actionable information that can guide the restaurant in marking strategies, and overall operations. Ultimately, our analysis will empower the restaurant to make informed decisions and enhance its competitiveness in an ever-evolving industry.

Data Requirements:

1. Menu Data

2. Customer Data

3. Employee Data

4. Payment Data

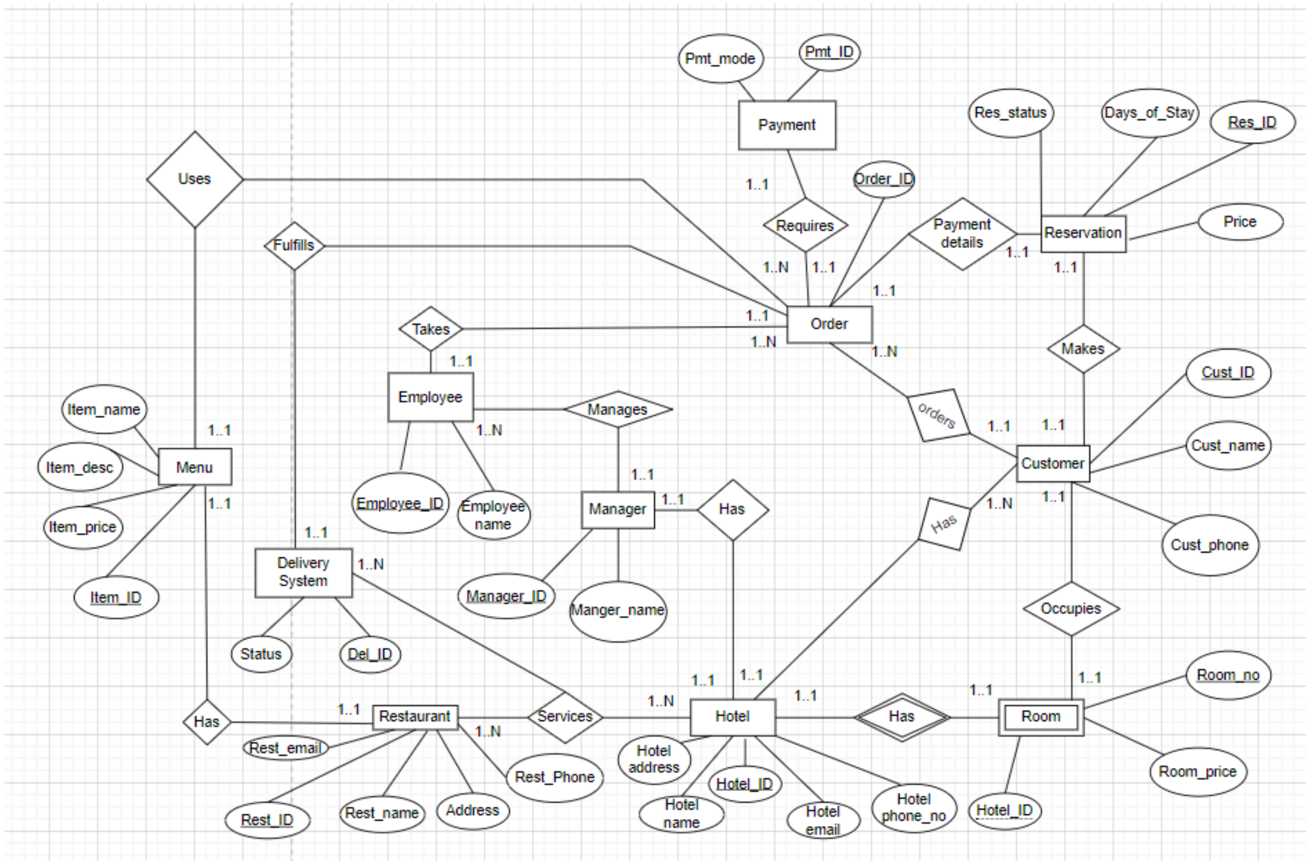
5.Delivery system Data

6. Order Data

Referential Data: Customer, Employee, Hotel and Restaurant data is used for the reference.

Transactional data: Delivery system, Order and Payment table is considered for the transactional data

Conceptual data model using the EER model:



Relationships between entities:

1. A Hotel has a manager, who manages Employees.

2. A Hotel has many customers, having that a customer can book a room by reservation and one can order items for the delivery system option.

3. Employee works on different levels in the restaurant like chef, Cashier, and Catering.

4. A delivery System can deliver one or many orders at a time.

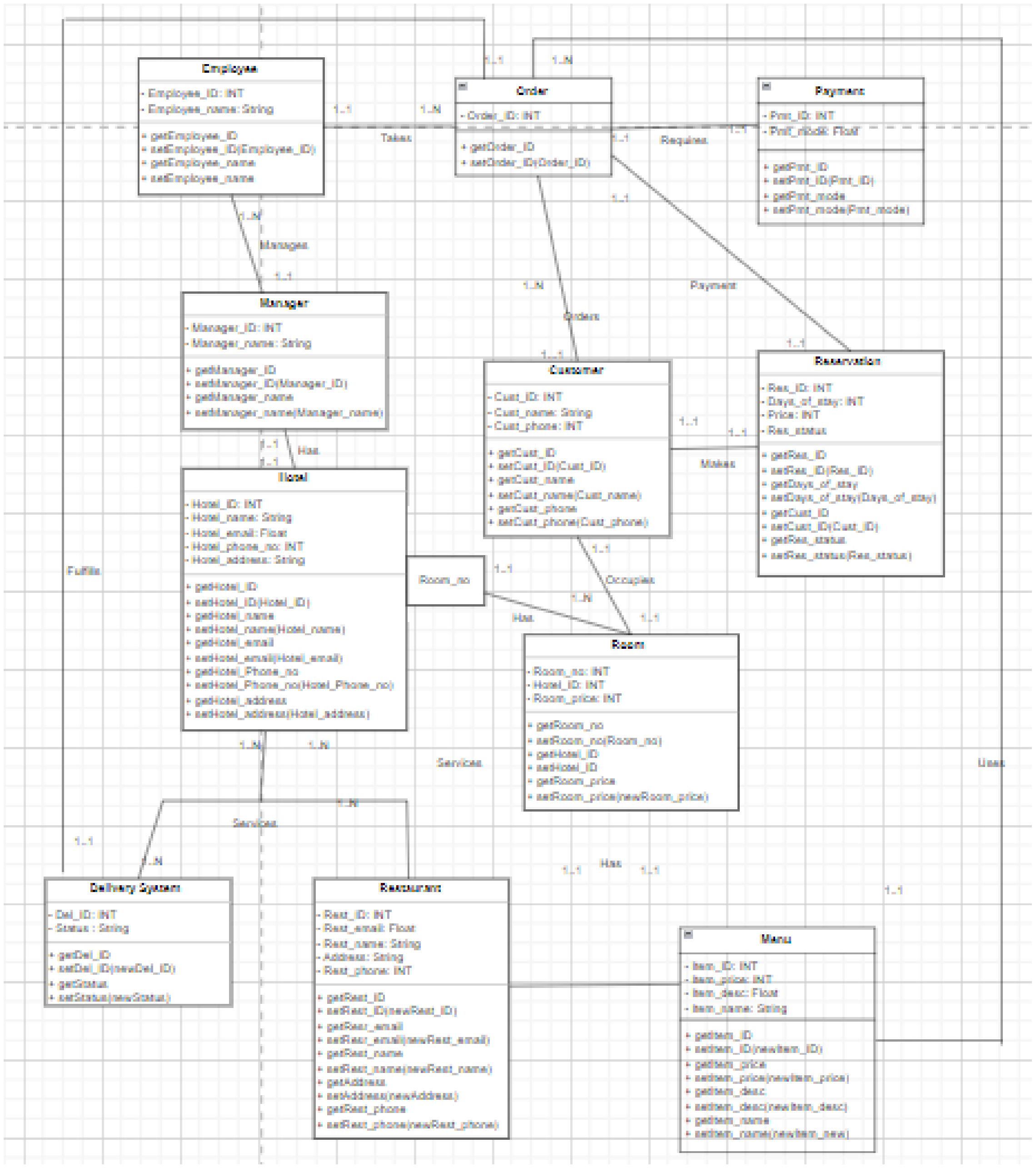
5. One customer can place many orders and can perform one payment for the orders.

6. The customer has to reserve a Room to use in services provided by the restaurant.

7. Order uses a menu for selecting items.

8. Restaurant and Delivery System serves the hotel.

UML



Employee
- Employee_ID: INT
- Employee_name: String
+ getEmployee_ID
+ setEmployee_ID(Employee_ID)
+ getEmployee_name
+ setEmployee_name
Order
- Order_ID: INT
+ getOrder_ID
+ setOrder_ID(Order_ID)
Payment
- Pmt_ID: INT
- Pmt_mode: Float
+ getPmt_ID
+ setPmt_ID(Pmt_ID)
+ getPmt_mode
+ setPmt_mode(Pmt_mode)
Manager
- Manager_ID: INT
- Manager_name: String
+ getManager_ID
+ setManager_ID(Manager_ID)
+ getManager_name
+ setManager_name(Manager_name)
Customer
- Cust_ID: INT
- Cust_name: String
- Cust_phone: INT
+ getCust_ID
+ setCust_ID(Cust_ID)
+ getCust_name
+ setCust_name(Cust_name)
+ getCust_phone
+ setCust_phone(Cust_phone)
Reservation
- Res_ID: INT
- Days_of_stay: INT
- Price: INT
- Res_status
+ getRes_ID
+ setRes_ID(Res_ID)
+ getDays_of_stay
+ setDays_of_stay(Days_of_stay)
+ getCust_ID
+ setCust_ID(Cust_ID)
+ getRes_status
+ setRes_status(Res_status)
Hotel
- Hotel_ID: INT
- Hotel_name: String
- Hotel_email: Float
- Hotel_phone_no: INT
- Hotel_address: String
+ getHotel_ID
+ setHotel_ID(Hotel_ID)
+ getHotel_name
+ setHotel_name(Hotel_name)
+ getHotel_email
+ setHotel_email(Hotel_email)
+ getHotel_Phone_no
+ setHotel_Phone_no(Hotel_Phone_no)
+ getHotel_address
+ setHotel_address(Hotel_address)
Room_no
Room
- Room_no: INT
- Hotel_ID: INT
- Room_price: INT
+ getRoom_no
+ setRoom_no(Room_no)
+ getHotel_ID
+ setHotel_ID
+ getRoom_price
+ setRoom_price(newRoom_price)
Delivery System
- Del_ID: INT
- Status : String
+ getDel_ID
+ setDel_ID(newDel_ID)
+ getStatus
+ setStatus(newStatus)
Restaurant
- Rest_ID: INT
- Rest_email: Float
- Rest_name: String
- Address: String
- Rest_phone: INT
+ getRest_ID
+ setRest_ID(newRest_ID)
+ getRest_email
+ setRest_email(newRest_email)
+ getRest_name
+ setRest_name(newRest_name)
+ getAddress
+ setAddress(newAddress)
+ getRest_phone
+ setRest_phone(newRest_phone)
Menu
- Item_ID: INT
- Item_price: INT
- Item_desc: Float
- Item_name: String
+ getItem_ID
+ setItem_ID(newItem_ID)
+ getItem_price
+ setItem_price(newItem_price)
+ getItem_desc
+ setItem_desc(newItem_desc)
+ getItem_name
+ setItem_name(newItem_new)
Takes
Requires
Manages
Orders
Payment
Has
Makes
Occupies
Fulfills
Services
Uses
1..1
1..N

Relational model mapped from the Conceptual model:

1.Customer(Cust_ID,Cust_name,Cust_phone,*Hotel_ID*)

- Hotel_ID is Foreign Key referencing Hotel_ID in Hotel – NOT NULL

2.Reservation(Res_ID,Days_of_stay,Res_status,Price,*Cust_ID)*

- Cust_ID is Foreign Key referencing Cust_ID in Customer – NOT NULL

3.Hotel(Hotel_ID,Hotel_email,Hotel_phone_no,Hotel_name,Hotel_address)

4.Room(Room_no,*Hotel_ID*,Room_price,*Cust_ID)*

- Hotel_ID is Foreign Key referencing Hotel_ID in Hotel – NOT NULL
- Cust_ID is Foreign Key referencing Cust_ID in Customer – NOT NULL

5.Menu (Item_ID,Item_name,Item_desc,Item_price)

6.Restaurant(Rest_ID,Rest_email,Rest_name,Address,Rest_phone,*Menu_ID*)

- Item_ID is Foreign Key referencing Item_ID from Menu – NOT NULL

7.Order(Order_ID,*Cust_ID,Item_ID,Employee_ID,Res_ID,Item_ID*)

- Cust_ID is Foreign Key referencing Cust_ID in Customer – NOT NULL
- Item_ID is Foreign_key referencing Item_ID from Menu – NOT NULL
- Res_ID is Foreign Key referencing Res_ID from Reservation – NOT NULL
- Item_ID is Foreign Key referencing Item_ID from Menu – NOT NULL
- Employee_ID is Foreign Key referencing Employee_ID from Employee – NOT NULL

8.Manager(Manager_ID,Manager_name,*Hotel_ID*)

- Hotel_ID is Foreign Key referencing Hotel_ID in Hotel – NOT NULL

9.Employee(Employee_ID,Employee_name,*Manager_ID*)

- Manager_ID is Foreign Key referencing Manager_ID in Manager – NOT NULL

10.Payment(Pmt_ID,Pmt_mode,*Order_ID*)

- Order_ID is Foreign Key referencing Order_ID from Order – NOT NULL

11.Delivery_system(Del_ID,Status, *Order_ID*)

- Order_ID is Foreign Key referencing Order_ID from Order – NOT NULL

12.Delivery_Supply_Chain(*Del_ID,Hotel_ID,Rest_ID*)

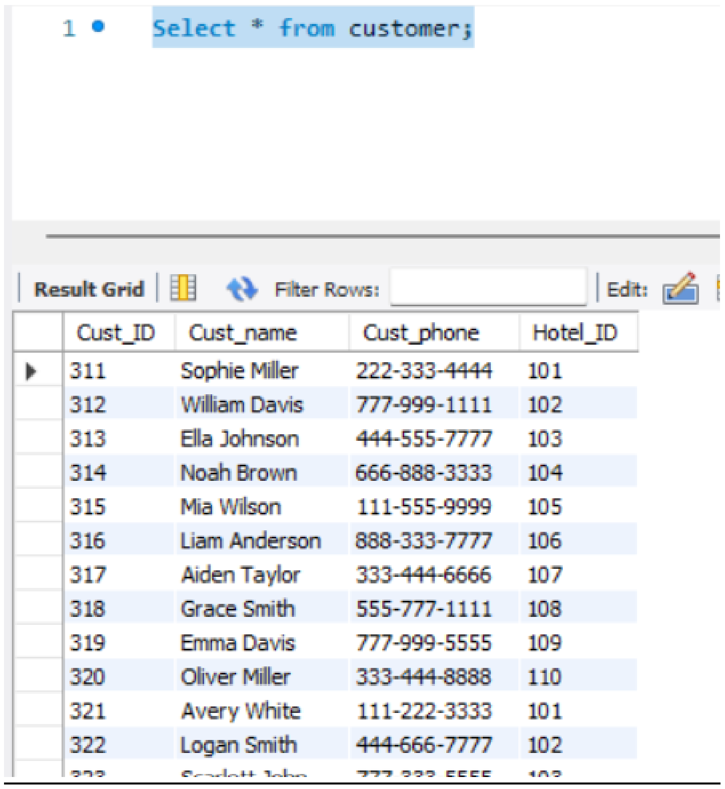
- Del_ID is Foreign Key referencing Del_ID from Delivery_system – NOT NULL
- Hotel_ID is Foreign Key referencing Hotel_ID from Hotel - NOT NULL
- Rest_ID is Foreign Key referencing Rest_ID from Restaurant – NOT NULL

Normalizing the above Relations:

- The above tables are already normalized as there is no transitive or trivial dependency in our relations.

Sql

Data for customer

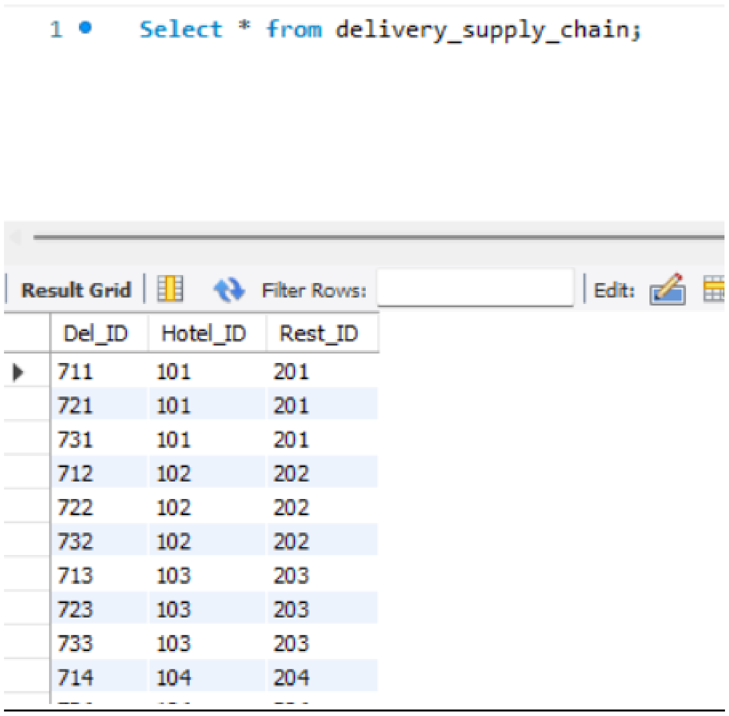


```
Select * from customer;
```

Result Grid | Filter Rows: | Edit:

Cust_ID	Cust_name	Cust_phone	Hotel_ID
311	Sophie Miller	222-333-4444	101
312	William Davis	777-999-1111	102
313	Ella Johnson	444-555-7777	103
314	Noah Brown	666-888-3333	104
315	Mia Wilson	111-555-9999	105
316	Liam Anderson	888-333-7777	106
317	Aiden Taylor	333-444-6666	107
318	Grace Smith	555-777-1111	108
319	Emma Davis	777-999-5555	109
320	Oliver Miller	333-444-8888	110
321	Avery White	111-222-3333	101
322	Logan Smith	444-666-7777	102

Data for delivery_supply_chain

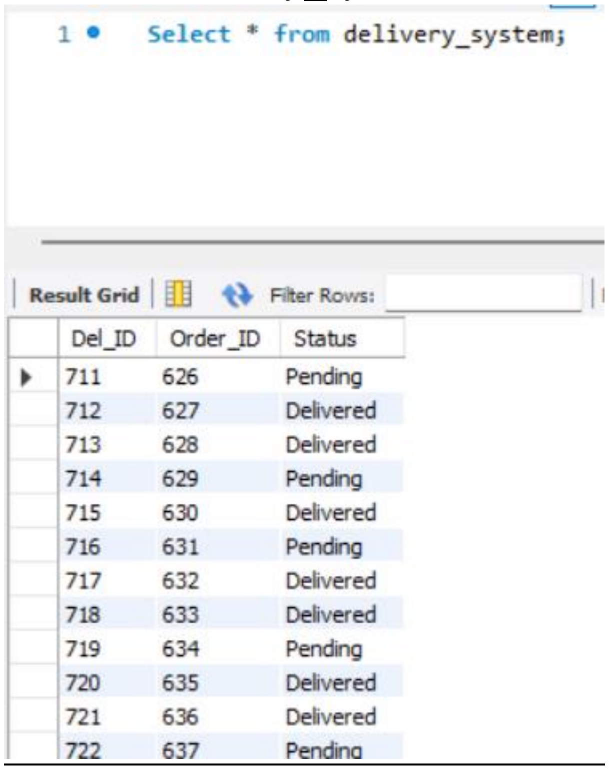


```
Select * from delivery_supply_chain;
```

Result Grid | Filter Rows: | Edit:

Del_ID	Hotel_ID	Rest_ID
711	101	201
721	101	201
731	101	201
712	102	202
722	102	202
732	102	202
713	103	203
723	103	203
733	103	203
714	104	204

Data for Delivery_system

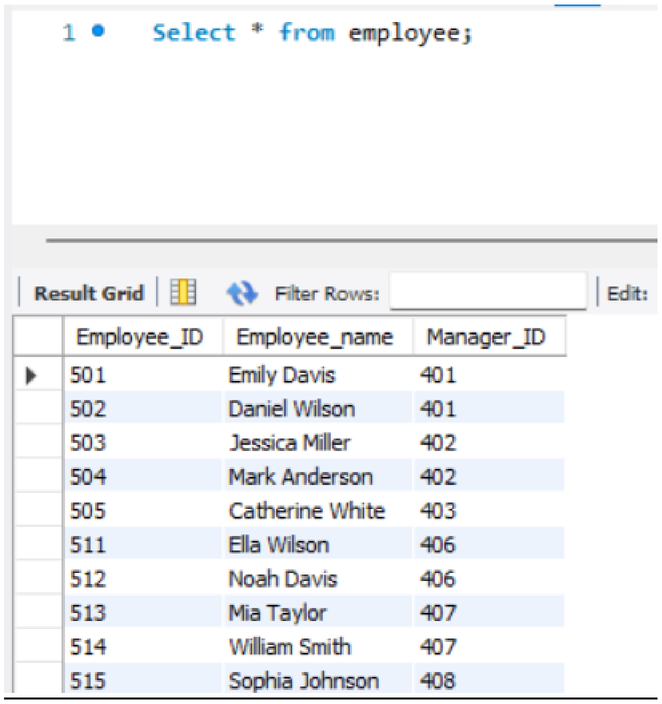


```
Select * from delivery_system;
```

Result Grid | Filter Rows:

Del_ID	Order_ID	Status
711	626	Pending
712	627	Delivered
713	628	Delivered
714	629	Pending
715	630	Delivered
716	631	Pending
717	632	Delivered
718	633	Delivered
719	634	Pending
720	635	Delivered
721	636	Delivered
722	637	Pending

Data for Employee

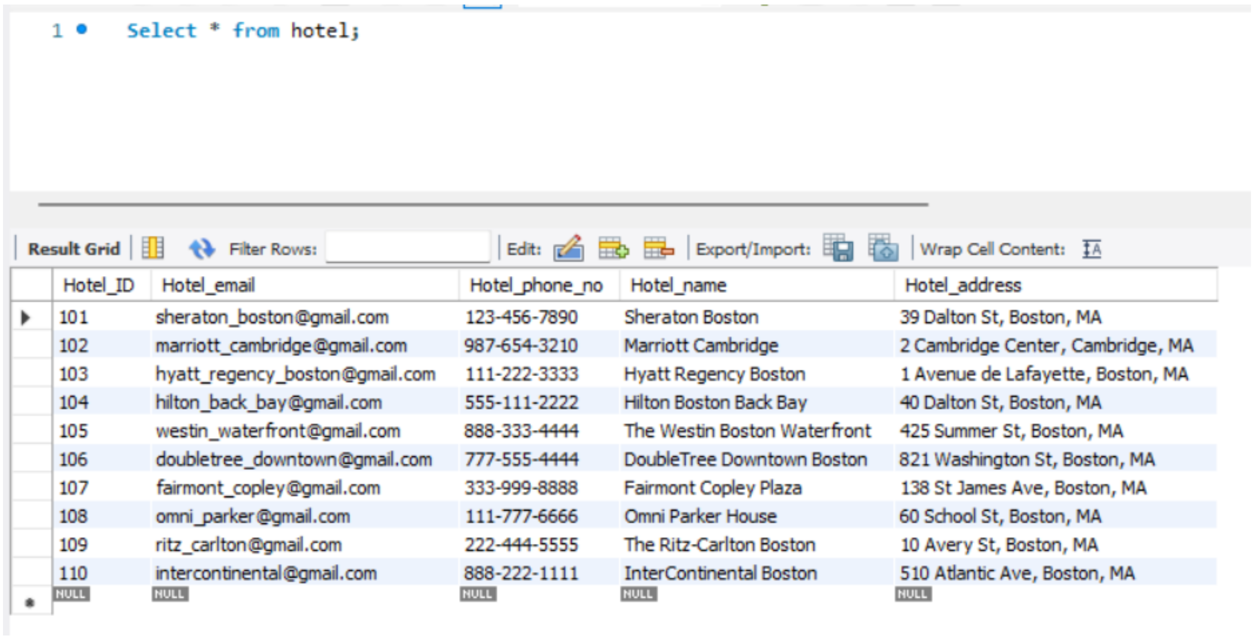


```
Select * from employee;
```

Result Grid | Filter Rows: | Edit:

Employee_ID	Employee_name	Manager_ID
501	Emily Davis	401
502	Daniel Wilson	401
503	Jessica Miller	402
504	Mark Anderson	402
505	Catherine White	403
511	Ella Wilson	406
512	Noah Davis	406
513	Mia Taylor	407
514	William Smith	407
515	Sophia Johnson	408

Data for hotel

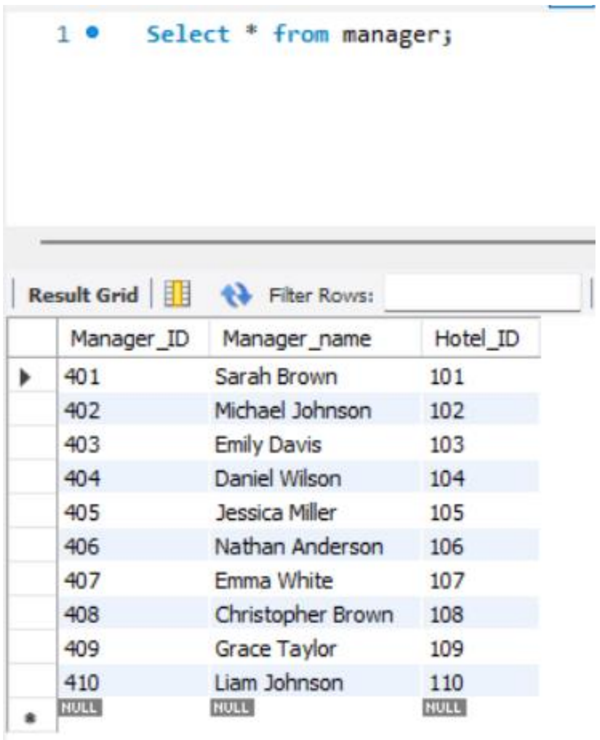


```
Select * from hotel;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Hotel_ID	Hotel_email	Hotel_phone_no	Hotel_name	Hotel_address
101	sheraton_boston@gmail.com	123-456-7890	Sheraton Boston	39 Dalton St, Boston, MA
102	marriott_cambridge@gmail.com	987-654-3210	Marriott Cambridge	2 Cambridge Center, Cambridge, MA
103	hyatt_regency_boston@gmail.com	111-222-3333	Hyatt Regency Boston	1 Avenue de Lafayette, Boston, MA
104	hilton_back_bay@gmail.com	555-111-2222	Hilton Boston Back Bay	40 Dalton St, Boston, MA
105	westin_waterfront@gmail.com	888-333-4444	The Westin Boston Waterfront	425 Summer St, Boston, MA
106	doubletree_downtown@gmail.com	777-555-4444	DoubleTree Downtown Boston	821 Washington St, Boston, MA
107	fairmont_copley@gmail.com	333-999-8888	Fairmont Copley Plaza	138 St James Ave, Boston, MA
108	omni_parker@gmail.com	111-777-6666	Omni Parker House	60 School St, Boston, MA
109	ritz_carlton@gmail.com	222-444-5555	The Ritz-Carlton Boston	10 Avery St, Boston, MA
110	intercontinental@gmail.com	888-222-1111	InterContinental Boston	510 Atlantic Ave, Boston, MA
NULL	NULL	NULL	NULL	NULL

Data for Manager

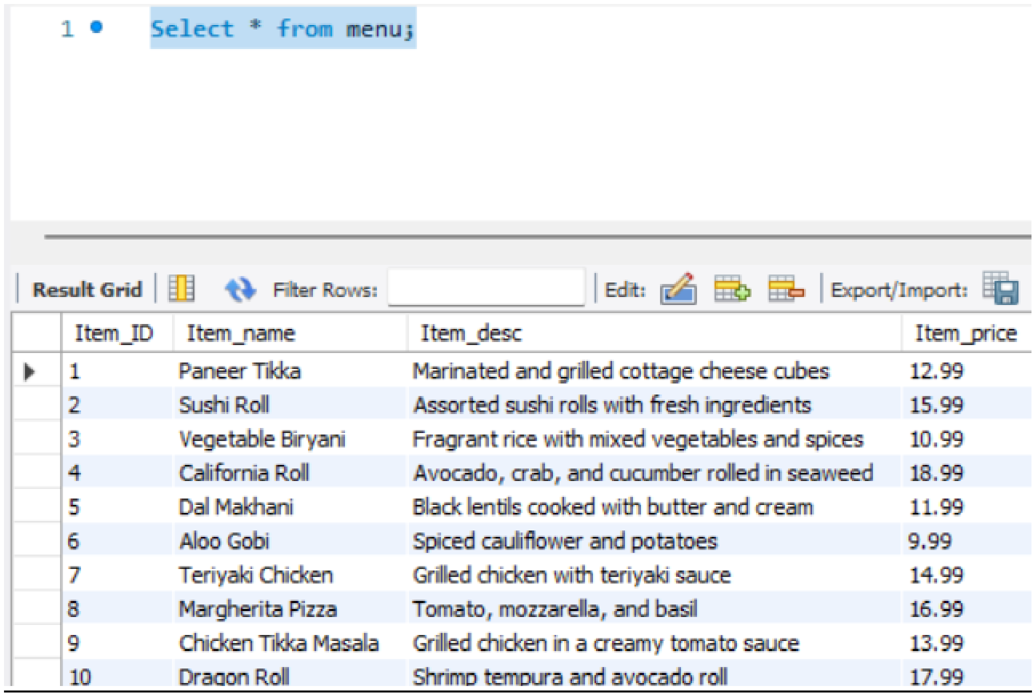


```
Select * from manager;
```

Result Grid | Filter Rows:

Manager_ID	Manager_name	Hotel_ID
401	Sarah Brown	101
402	Michael Johnson	102
403	Emily Davis	103
404	Daniel Wilson	104
405	Jessica Miller	105
406	Nathan Anderson	106
407	Emma White	107
408	Christopher Brown	108
409	Grace Taylor	109
410	Liam Johnson	110
NULL	NULL	NULL

Data for Menu

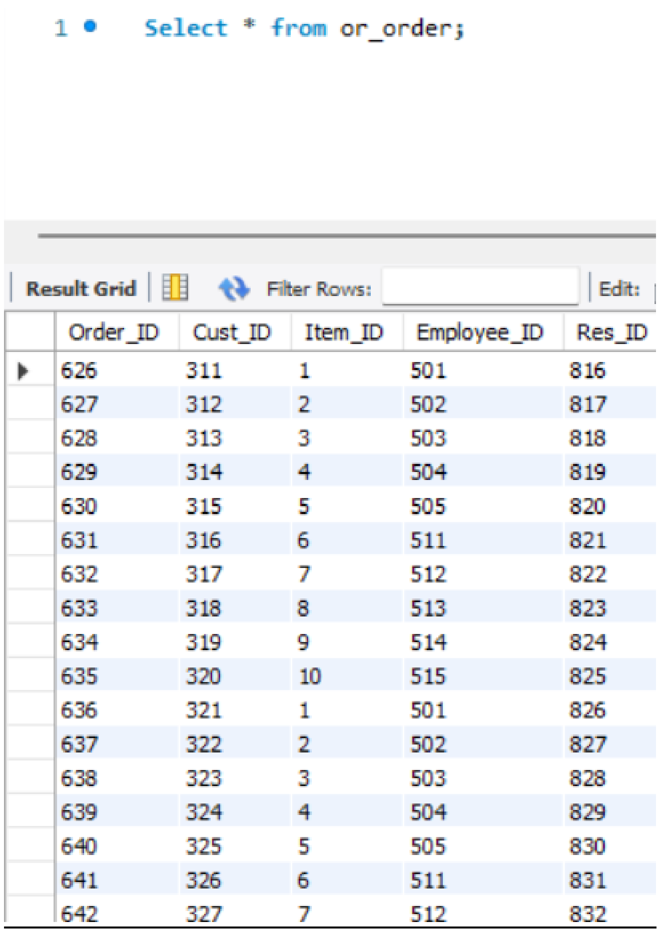


```
Select * from menu;
```

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

Item_ID	Item_name	Item_desc	Item_price
1	Paneer Tikka	Marinated and grilled cottage cheese cubes	12.99
2	Sushi Roll	Assorted sushi rolls with fresh ingredients	15.99
3	Vegetable Biryani	Fragrant rice with mixed vegetables and spices	10.99
4	California Roll	Avocado, crab, and cucumber rolled in seaweed	18.99
5	Dal Makhani	Black lentils cooked with butter and cream	11.99
6	Aloo Gobi	Spiced cauliflower and potatoes	9.99
7	Teriyaki Chicken	Grilled chicken with teriyaki sauce	14.99
8	Margherita Pizza	Tomato, mozzarella, and basil	16.99
9	Chicken Tikka Masala	Grilled chicken in a creamy tomato sauce	13.99
10	Dragon Roll	Shrimp tempura and avocado roll	17.99

Data for Orders

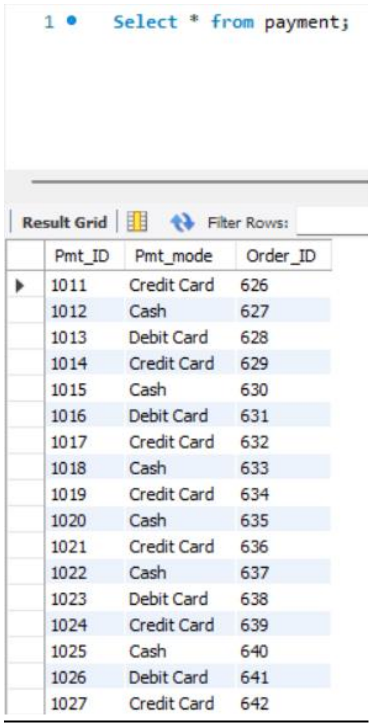


```
Select * from or_order;
```

Result Grid | Filter Rows: | Edit:

Order_ID	Cust_ID	Item_ID	Employee_ID	Res_ID
626	311	1	501	816
627	312	2	502	817
628	313	3	503	818
629	314	4	504	819
630	315	5	505	820
631	316	6	511	821
632	317	7	512	822
633	318	8	513	823
634	319	9	514	824
635	320	10	515	825
636	321	1	501	826
637	322	2	502	827
638	323	3	503	828
639	324	4	504	829
640	325	5	505	830
641	326	6	511	831
642	327	7	512	832

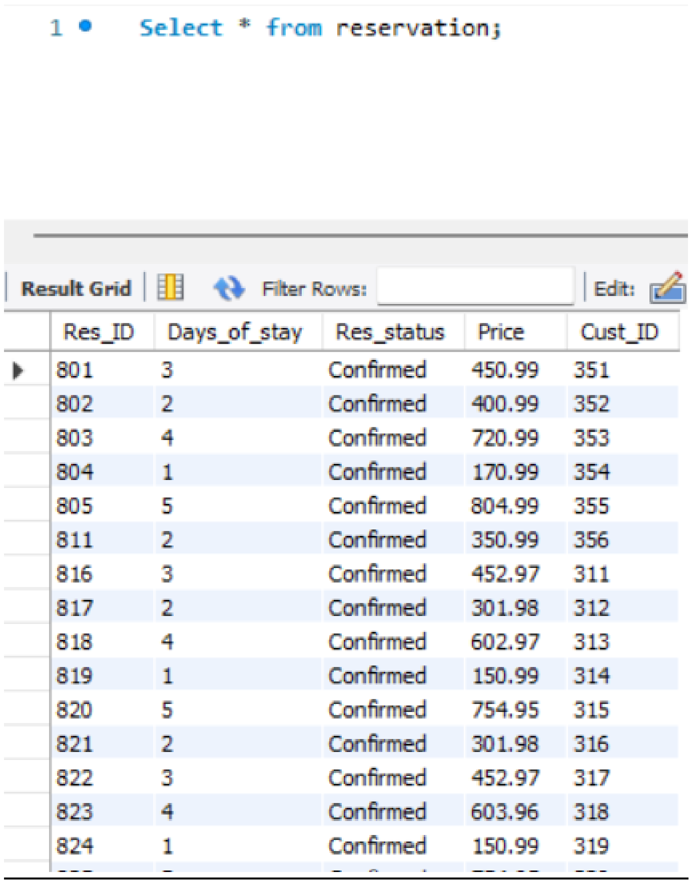
Data for Payments



```
Select * from payment;
```

Pmt_ID	Pmt_mode	Order_ID
1011	Credit Card	626
1012	Cash	627
1013	Debit Card	628
1014	Credit Card	629
1015	Cash	630
1016	Debit Card	631
1017	Credit Card	632
1018	Cash	633
1019	Credit Card	634
1020	Cash	635
1021	Credit Card	636
1022	Cash	637
1023	Debit Card	638
1024	Credit Card	639
1025	Cash	640
1026	Debit Card	641
1027	Credit Card	642

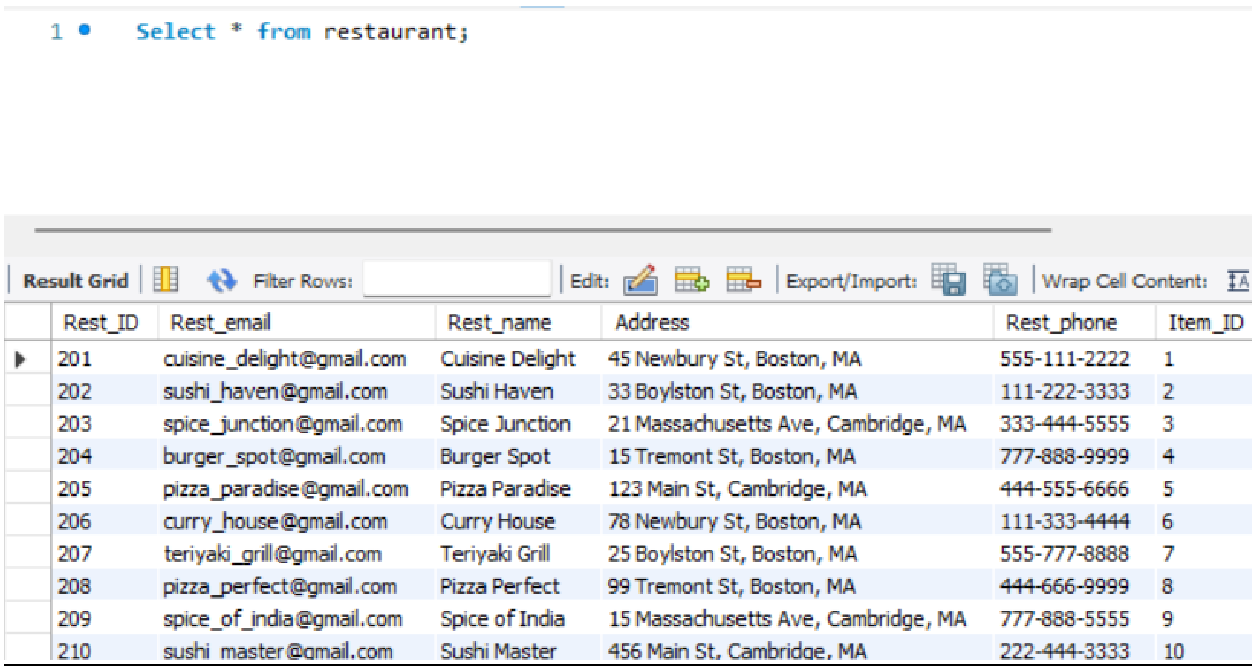
Data For reservation



```
Select * from reservation;
```

Res_ID	Days_of_stay	Res_status	Price	Cust_ID
801	3	Confirmed	450.99	351
802	2	Confirmed	400.99	352
803	4	Confirmed	720.99	353
804	1	Confirmed	170.99	354
805	5	Confirmed	804.99	355
811	2	Confirmed	350.99	356
816	3	Confirmed	452.97	311
817	2	Confirmed	301.98	312
818	4	Confirmed	602.97	313
819	1	Confirmed	150.99	314
820	5	Confirmed	754.95	315
821	2	Confirmed	301.98	316
822	3	Confirmed	452.97	317
823	4	Confirmed	603.96	318
824	1	Confirmed	150.99	319

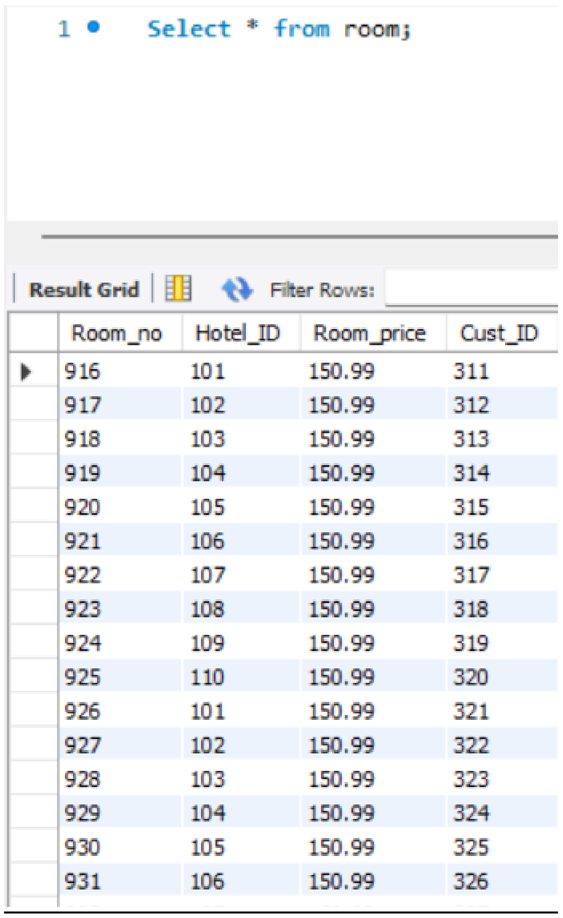
Data for Restaurant



```
Select * from restaurant;
```

Rest_ID	Rest_email	Rest_name	Address	Rest_phone	Item_ID
201	cuisine_delight@gmail.com	Cuisine Delight	45 Newbury St, Boston, MA	555-111-2222	1
202	sushi_haven@gmail.com	Sushi Haven	33 Boylston St, Boston, MA	111-222-3333	2
203	spice_junction@gmail.com	Spice Junction	21 Massachusetts Ave, Cambridge, MA	333-444-5555	3
204	burger_spot@gmail.com	Burger Spot	15 Tremont St, Boston, MA	777-888-9999	4
205	pizza_paradise@gmail.com	Pizza Paradise	123 Main St, Cambridge, MA	444-555-6666	5
206	curry_house@gmail.com	Curry House	78 Newbury St, Boston, MA	111-333-4444	6
207	teriyaki_grill@gmail.com	Teriyaki Grill	25 Boylston St, Boston, MA	555-777-8888	7
208	pizza_perfect@gmail.com	Pizza Perfect	99 Tremont St, Boston, MA	444-666-9999	8
209	spice_of_india@gmail.com	Spice of India	15 Massachusetts Ave, Cambridge, MA	777-888-5555	9
210	sushi_master@gmail.com	Sushi Master	456 Main St, Cambridge, MA	222-444-3333	10

Data for Room



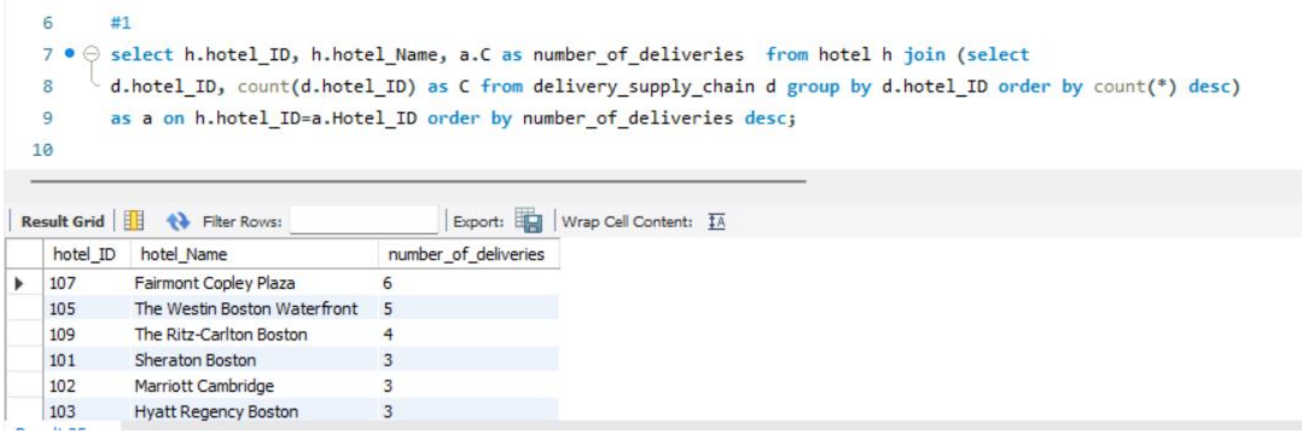
```
Select * from room;
```

Room_no	Hotel_ID	Room_price	Cust_ID
916	101	150.99	311
917	102	150.99	312
918	103	150.99	313
919	104	150.99	314
920	105	150.99	315
921	106	150.99	316
922	107	150.99	317
923	108	150.99	318
924	109	150.99	319
925	110	150.99	320
926	101	150.99	321
927	102	150.99	322
928	103	150.99	323
929	104	150.99	324
930	105	150.99	325
931	106	150.99	326

Analytical Queries

1. Hotel name and number of deliveries it got

Reasoning: The number of deliveries can be indicative of a hotel's popularity and customer demand for its services. High delivery counts suggest that a hotel is in demand for food delivery, possibly indicating a strong market presence and customer satisfaction.

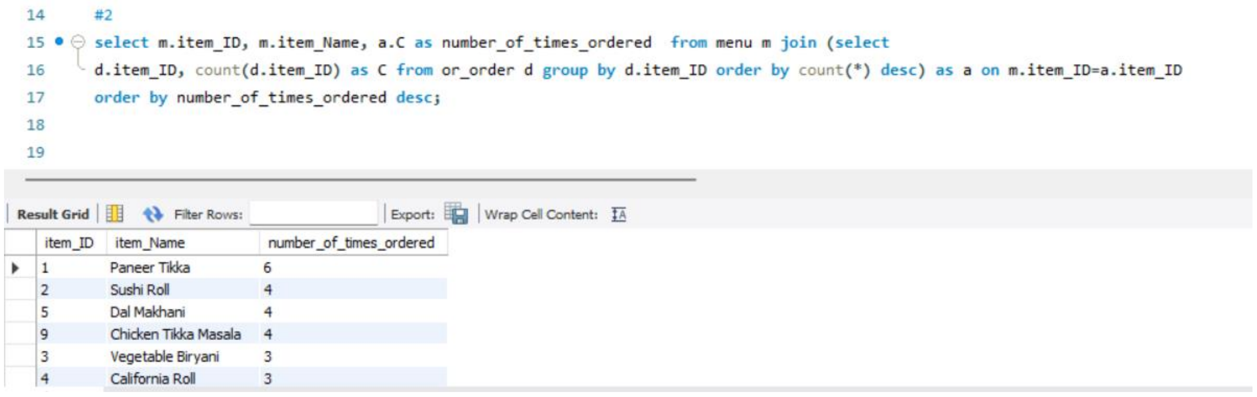


```
#1
select h.hotel_ID, h.hotel_Name, a.C as number_of_deliveries  from hotel h join (select
d.hotel_ID, count(d.hotel_ID) as C from delivery_supply_chain d group by d.hotel_ID order by count(*) desc)
as a on h.hotel_ID=a.Hotel_ID order by number_of_deliveries desc;
```

hotel_ID	hotel_Name	number_of_deliveries
107	Fairmont Copley Plaza	6
105	The Westin Boston Waterfront	5
109	The Ritz-Carlton Boston	4
101	Sheraton Boston	3
102	Marriott Cambridge	3
103	Hyatt Regency Boston	3

2. Number of times each item is ordered

Reasoning: Analysis of item popularity helps in understanding customer preferences. It enables businesses to tailor their offerings to align with customer tastes and potentially introduce new items based on popular trends.

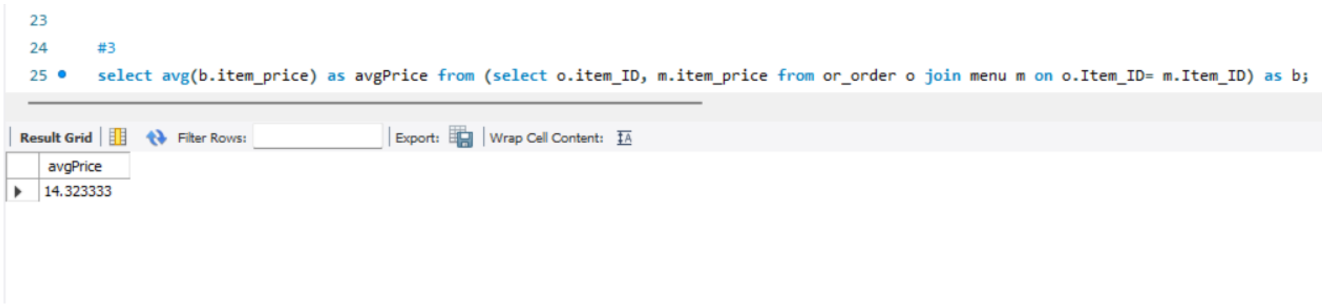


```
#2
select m.item_ID, m.item_Name, a.C as number_of_times_ordered  from menu m join (select
d.item_ID, count(d.item_ID) as C from or_order d group by d.item_ID order by count(*) desc) as a on m.item_ID=a.item_ID
order by number_of_times_ordered desc;
```

item_ID	item_Name	number_of_times_ordered
1	Paneer Tikka	6
2	Sushi Roll	4
5	Dal Makhani	4
9	Chicken Tikka Masala	4
3	Vegetable Biryani	3
4	California Roll	3

3. Average price of items

Reasoning: The average item price is indicative of the value perception customers have for the menu items. It helps businesses understand how customers perceive the pricing relative to the perceived value of the offerings.

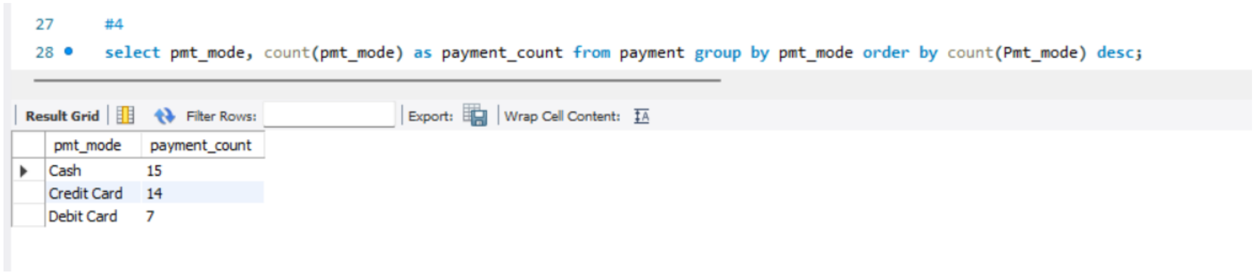


```sql
#3
select avg(b.item_price) as avgPrice from (select o.item_ID, m.item_price from or_order o join menu m on o.Item_ID= m.Item_ID) as b;
```

avgPrice
14.323333

4. Type of payment methods and how many times they have been used

Reasoning: The primary objective of the query is to understand the distribution of payment methods used by customers. This information is crucial for assessing the popularity and prevalence of different payment options.

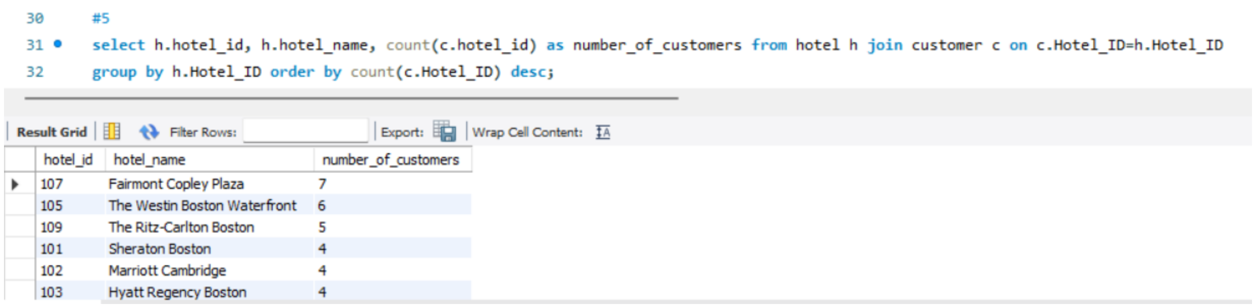


```sql
#4
select pmt_mode, count(pmt_mode) as payment_count from payment group by pmt_mode order by count(Pmt_mode) desc;
```

pmt_mode	payment_count
Cash	15
Credit Card	14
Debit Card	7

5. Hotel name and number of customers it has

Reasoning: Hotel managers can use the insights from the query to tailor marketing strategies. Hotels with a larger customer base may focus on retention strategies, while those with fewer customers may prioritize attracting new customers.

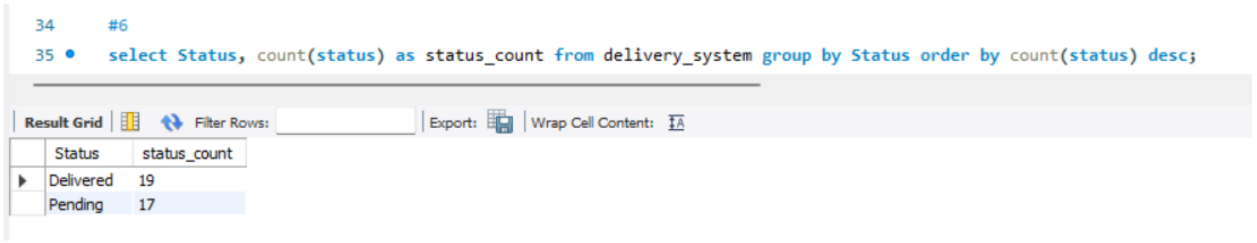


```sql
#5
select h.hotel_id, h.hotel_name, count(c.hotel_id) as number_of_customers from hotel h join customer c on c.Hotel_ID=h.Hotel_ID
group by h.Hotel_ID order by count(c.Hotel_ID) desc;
```

hotel_id	hotel_name	number_of_customers
107	Fairmont Copley Plaza	7
105	The Westin Boston Waterfront	6
109	The Ritz-Carlton Boston	5
101	Sheraton Boston	4
102	Marriott Cambridge	4
103	Hyatt Regency Boston	4

6. Delivery Status

Reasoning: The primary objective of the query is to evaluate the performance of the delivery system by understanding the distribution of delivery statuses.

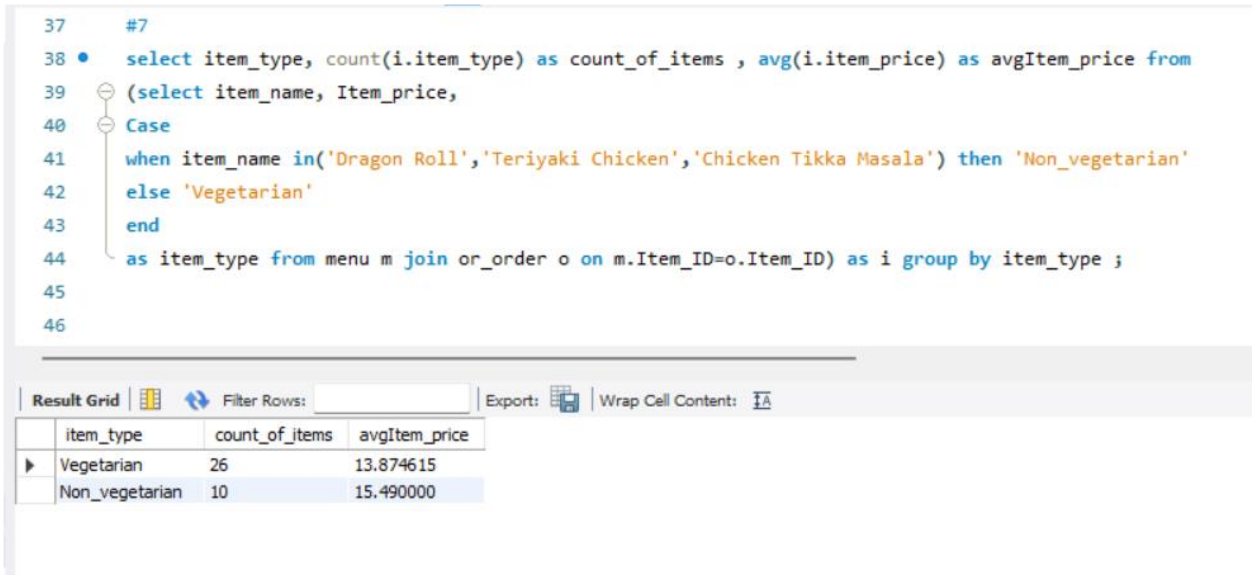


```
#6
select Status, count(status) as status_count from delivery_system group by Status order by count(status) desc;
```

Status	status_count
Delivered	19
Pending	17

7. Number of times vegetarian and non-vegetarian items were bought, along with their average price.

Reasoning: The query results provide insights into the popularity of vegetarian and non-vegetarian items on the menu. This information is crucial for menu optimization and understanding customer preferences.

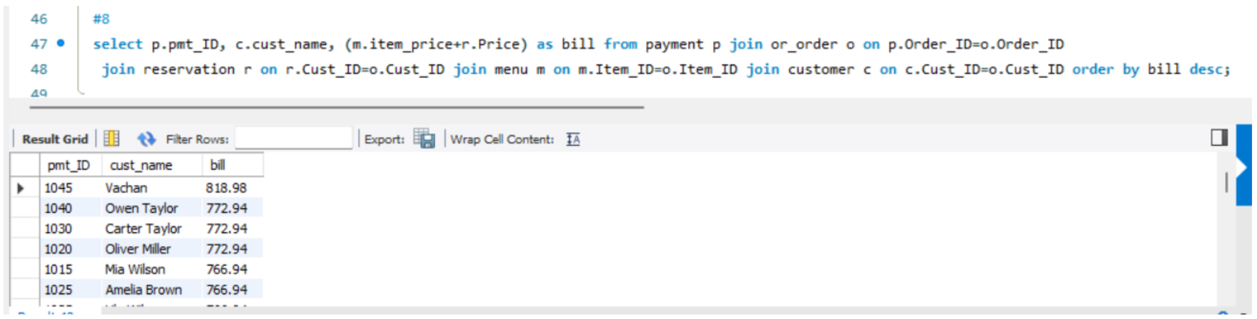


```
#7
select item_type, count(i.item_type) as count_of_items , avg(i.item_price) as avgItem_price from
(select item_name, Item_price,
Case
when item_name in('Dragon Roll','Teriyaki Chicken','Chicken Tikka Masala') then 'Non_vegetarian'
else 'Vegetarian'
end
as item_type from menu m join or_order o on m.Item_ID=o.Item_ID) as i group by item_type ;
```

item_type	count_of_items	avgItem_price
Vegetarian	26	13.874615
Non_vegetarian	10	15.490000

8. Bill per customer based on stay at hotel along with orders from restaurant.

Reasoning: The query enables customer segmentation based on spending behavior. Businesses can identify high-value customers who contribute significantly to overall revenue and tailor marketing efforts accordingly.



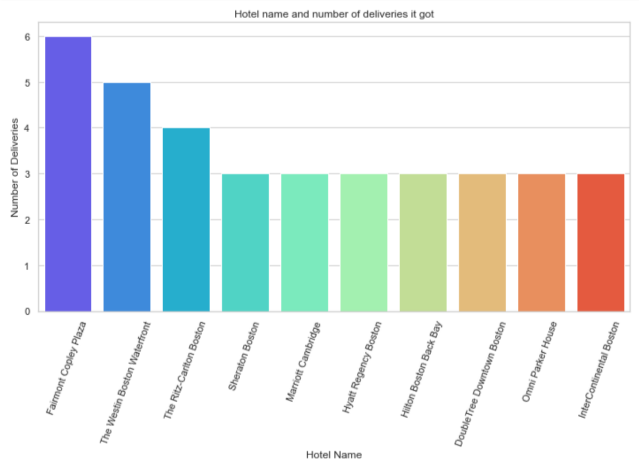
```
#8
select p.pmt_ID, c.cust_name, (m.item_price+r.Price) as bill from payment p join or_order o on p.Order_ID=o.Order_ID
 join reservation r on r.Cust_ID=o.Cust_ID join menu m on m.Item_ID=o.Item_ID join customer c on c.Cust_ID=o.Cust_ID order by bill desc;
```

pmt_ID	cust_name	bill
1045	Vachan	818.98
1040	Owen Taylor	772.94
1030	Carter Taylor	772.94
1020	Oliver Miller	772.94
1015	Mia Wilson	766.94
1025	Amelia Brown	766.94

Python

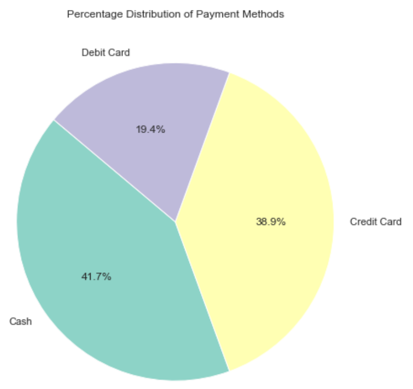
1. The number of deliveries can be indicative of a hotel's popularity and customer demand for its services. High delivery counts suggest that a hotel is in demand for food delivery, possibly indicating a strong market presence and customer satisfaction.

```
sns.set(style="whitegrid")
plt.figure(figsize=(12, 6))
sns.barplot(x="hotel_Name", y="number_of_deliveries", data=df, palette="rainbow")

plt.title("Hotel name and number of deliveries it got")
plt.xlabel("Hotel Name")
plt.ylabel("Number of Deliveries")
plt.xticks(rotation=70)
plt.show()
```

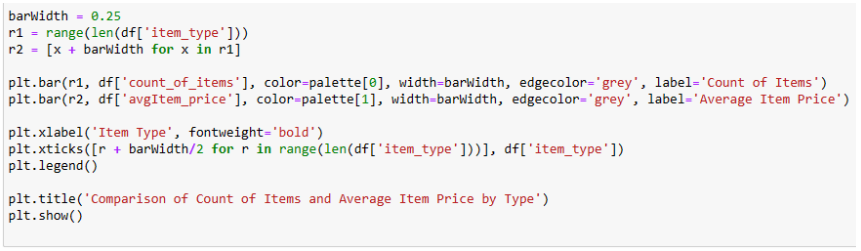


2. The primary objective of the query is to understand the distribution of payment methods used by customers. This information is crucial for assessing the popularity and prevalence of different payment options.

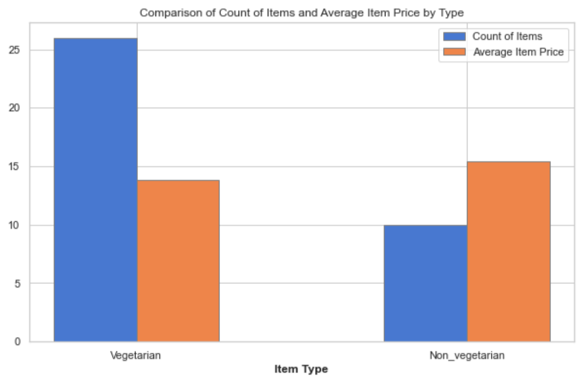
```
plt.pie(df['payment_count'], labels=df['pmt_mode'], autopct='%1.1f%%', startangle=140, colors=custom_palette)
plt.title("Percentage Distribution of Payment Methods")
plt.show()
```



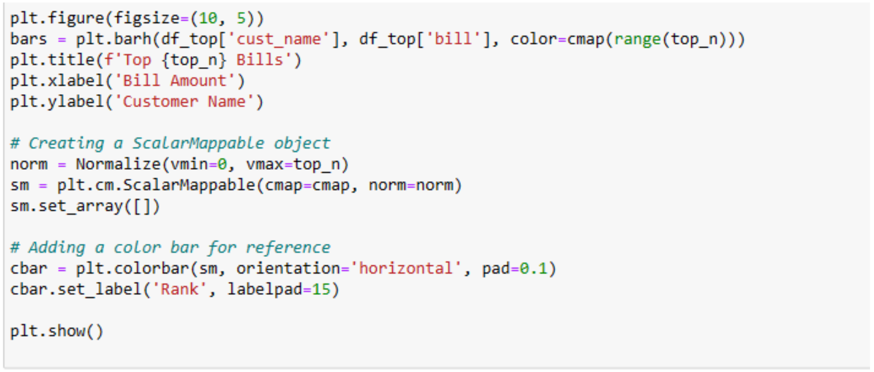
3. The query results provide insights into the popularity of vegetarian and non-vegetarian items on the menu. This information is crucial for menu optimization and understanding customer preferences.



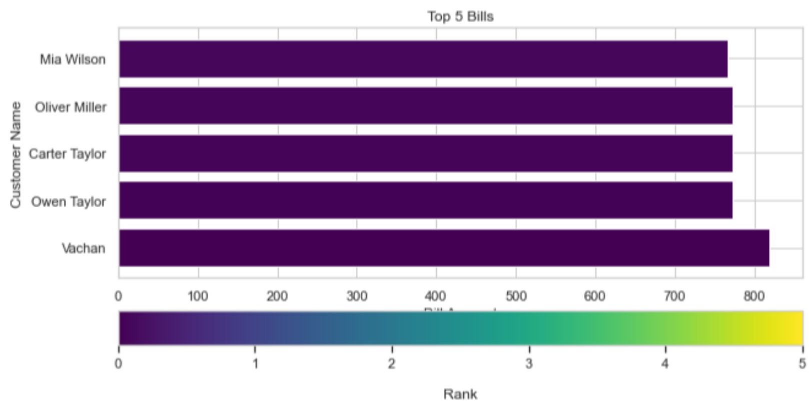
```
barWidth = 0.25
r1 = range(len(df['item_type']))
r2 = [x + barWidth for x in r1]

plt.bar(r1, df['count_of_items'], color=palette[0], width=barWidth, edgecolor='grey', label='Count of Items')
plt.bar(r2, df['avgItem_price'], color=palette[1], width=barWidth, edgecolor='grey', label='Average Item Price')

plt.xlabel('Item Type', fontweight='bold')
plt.xticks([r + barWidth/2 for r in range(len(df['item_type']))], df['item_type'])
plt.legend()

plt.title('Comparison of Count of Items and Average Item Price by Type')
plt.show()
```



4. The query enables customer segmentation based on spending behavior. Businesses can identify high-value customers who contribute significantly to overall revenue and tailor marketing efforts accordingly.

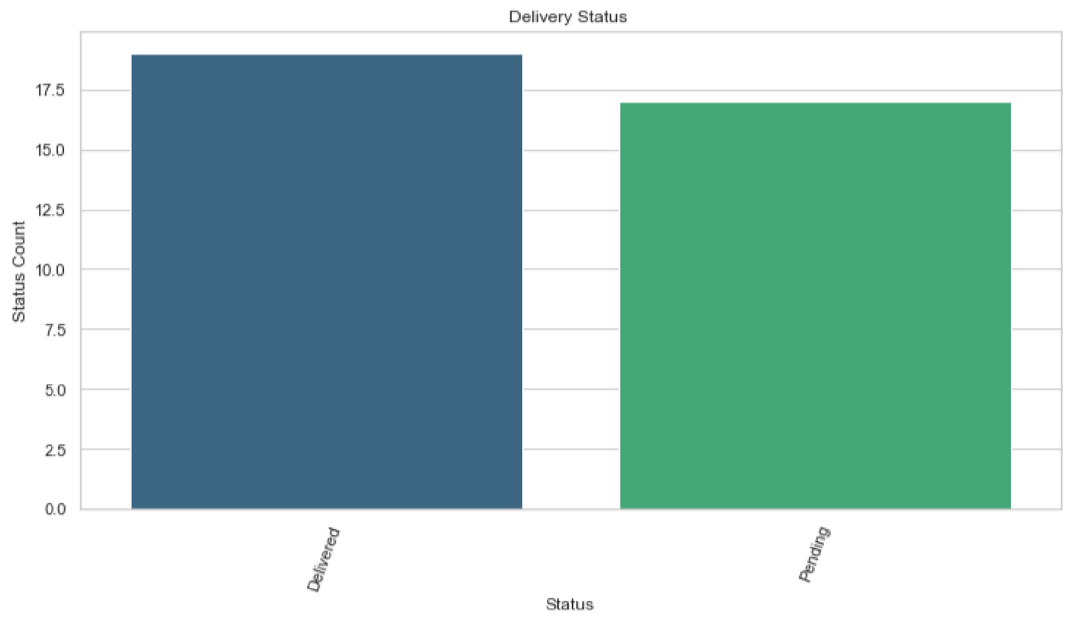


```
plt.figure(figsize=(10, 5))
bars = plt.barh(df_top['cust_name'], df_top['bill'], color=cmap(range(top_n)))
plt.title(f'Top {top_n} Bills')
plt.xlabel('Bill Amount')
plt.ylabel('Customer Name')

# Creating a ScalarMappable object
norm = Normalize(vmin=0, vmax=top_n)
sm = plt.cm.ScalarMappable(cmap=cmap, norm=norm)
sm.set_array([])

# Adding a color bar for reference
cbar = plt.colorbar(sm, orientation='horizontal', pad=0.1)
cbar.set_label('Rank', labelpad=15)

plt.show()
```



5. The primary objective of the query is to evaluate the performance of the delivery system by understanding the distribution of delivery statuses.

```
]: sns.set(style="whitegrid")
   plt.figure(figsize=(12, 6))
   sns.barplot(x="Status", y="status_count", data=df, palette="viridis")
   plt.title("Delivery Status ")
   plt.xlabel("Status")
   plt.ylabel("Status Count")
   plt.xticks(rotation=70)
   plt.show()
```

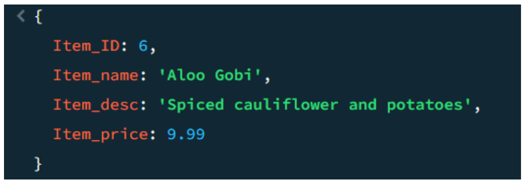


NOSQL

Least expensive item

To provide more Pocket friendly options Hotel's can use this query and promote dishes as budget friendly.

```
db.menu.aggregate([  {
   $sort: { Item_price: 1 }
 },
 {
   $limit: 1
 },
 {
   $project: {
     _id: 0,
     Item_ID: 1,
     Item_name: 1,
     Item_desc: 1,
     Item_price: 1
   }
 }
])
```

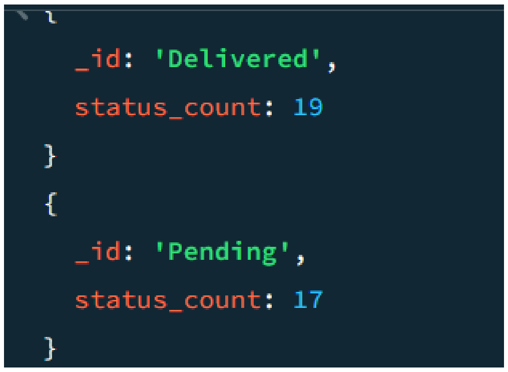


```
< {
    Item_ID: 6,
    Item_name: 'Aloo Gobi',
    Item_desc: 'Spiced cauliflower and potatoes',
    Item_price: 9.99
  }
```

Delivery Status

The primary objective of the query is to evaluate the performance of the delivery system by understanding the distribution of delivery statuses.

```
db.delivery_system.aggregate([
 {
  $group: {
   _id: "$Status",
   status_count: { $sum: 1 }
  }
 },
 {
  $sort: {
   status_count: -1
  }
 }
]);
```

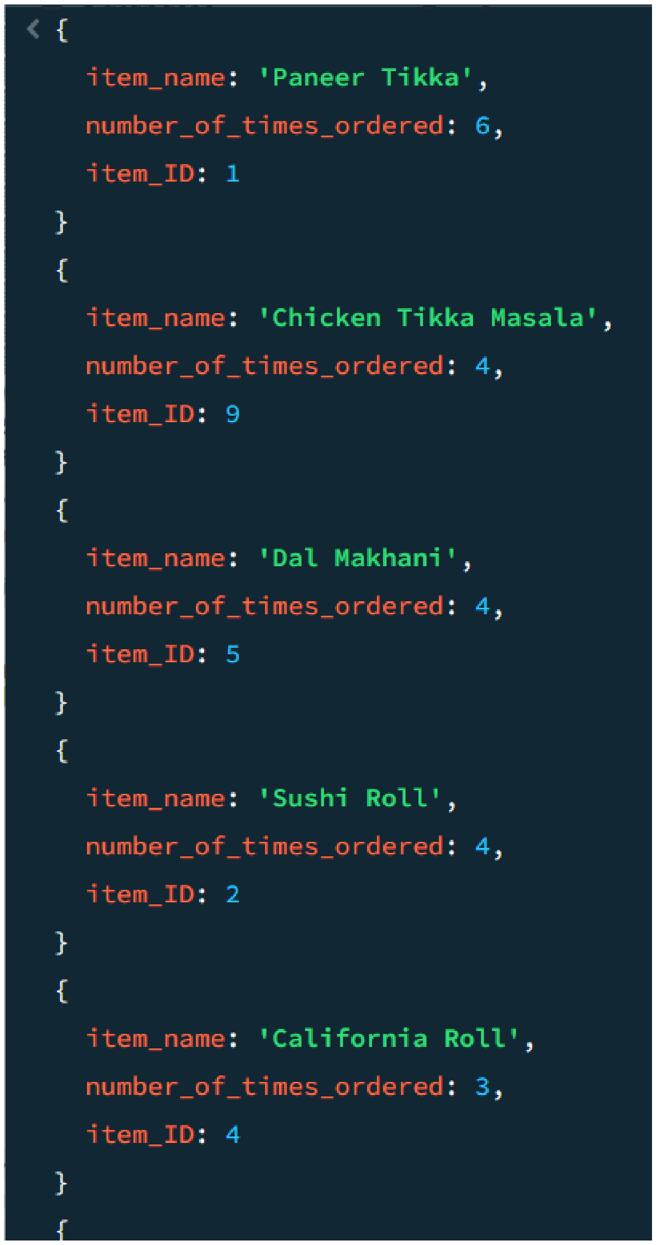


```
    _id: 'Delivered',
    status_count: 19
  }
  {
    _id: 'Pending',
    status_count: 17
  }
```

Number of times each item is ordered

Analysis of item popularity helps in understanding customer preferences. It enables businesses to tailor their offerings to align with customer tastes and potentially introduce new items based on popular trends.

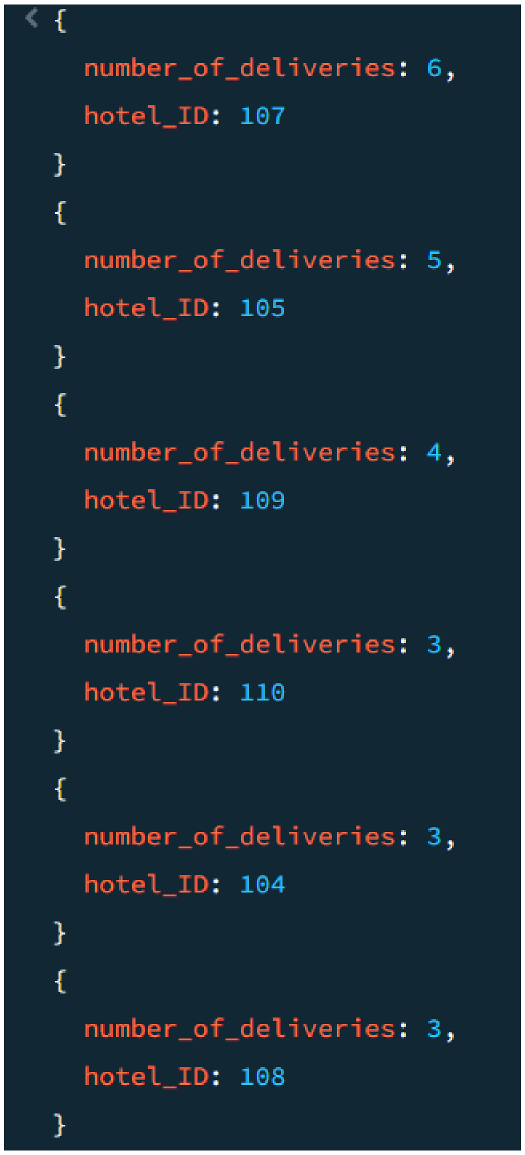
```
db.menu.aggregate([
 {
  $lookup: {
    from: "or_order",
    localField: "Item_ID",
    foreignField: "Item_ID",
    as: "order_details"
  }
 },
 {
  $unwind: "$order_details"
 },
 {
  $group: {
    _id: "$Item_ID",
    item_name: { $first: "$Item_name" },
    number_of_times_ordered: { $sum: 1 }
  }
 },
 {
  $sort: {
    number_of_times_ordered: -1
  }
 },
 {
  $project: {
    _id: 0,
    item_ID: "$_id",
    item_name: 1,
    number_of_times_ordered: 1
  }
 }
]);
```


```
< {
    item_name: 'Paneer Tikka',
    number_of_times_ordered: 6,
    item_ID: 1
  }
  {
    item_name: 'Chicken Tikka Masala',
    number_of_times_ordered: 4,
    item_ID: 9
  }
  {
    item_name: 'Dal Makhani',
    number_of_times_ordered: 4,
    item_ID: 5
  }
  {
    item_name: 'Sushi Roll',
    number_of_times_ordered: 4,
    item_ID: 2
  }
  {
    item_name: 'California Roll',
    number_of_times_ordered: 3,
    item_ID: 4
  }
  {
```

Hotel name and number of deliveries it got

Number of deliveries can be indicative of a hotel's popularity and customer demand for its services.

```
db.hotel.aggregate([
 {
  $lookup: {
   from: "delivery_supply_chain",
   localField: "Hotel_ID",
   foreignField: "Hotel_ID",
   as: "delivery_details"
  }
 },
 {
  $unwind: "$delivery_details"
 },
 {
  $group: {
   _id: {
    hotel_ID: "$Hotel_ID",
    hotel_Name: "$Hotel_Name"
   },
   number_of_deliveries: { $sum: 1 }
  }
 },
 {
  $sort: {
   number_of_deliveries: -1
  }
 },
 {
  $project: {
   _id: 0,
   hotel_ID: "$_id.hotel_ID",
   hotel_Name: "$_id.hotel_Name",
   number_of_deliveries: 1
  }
 }
]);
```

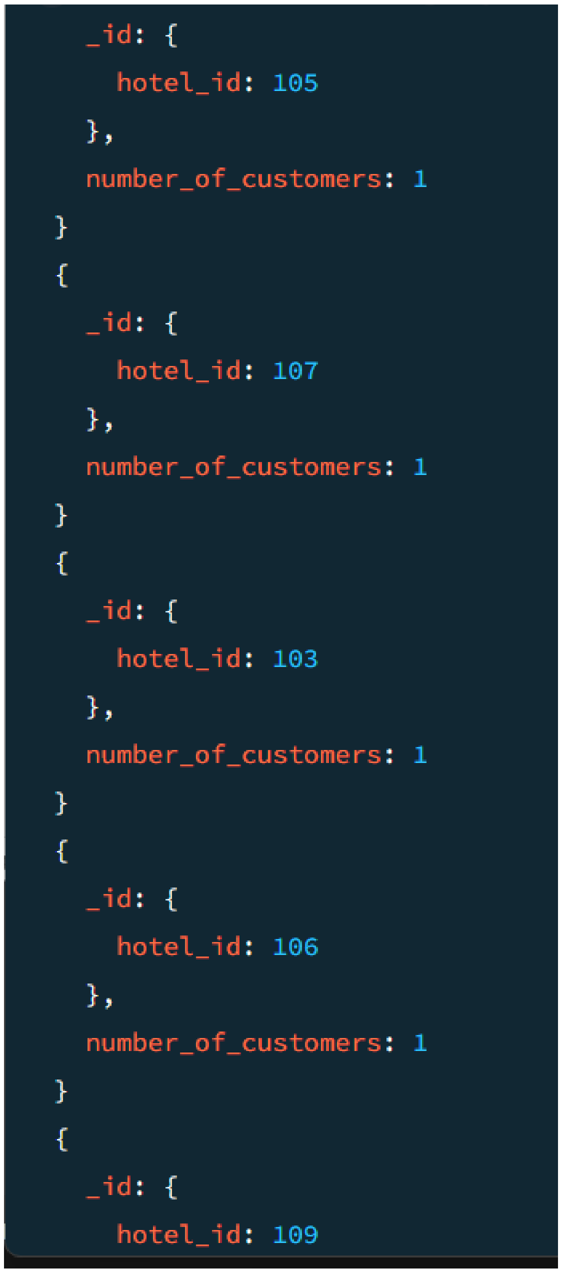


```
{
  number_of_deliveries: 6,
  hotel_ID: 107
}
{
  number_of_deliveries: 5,
  hotel_ID: 105
}
{
  number_of_deliveries: 4,
  hotel_ID: 109
}
{
  number_of_deliveries: 3,
  hotel_ID: 110
}
{
  number_of_deliveries: 3,
  hotel_ID: 104
}
{
  number_of_deliveries: 3,
  hotel_ID: 108
}
```

<u>Hotel name and number of customers it has</u>

Number of deliveries can be indicative of a hotel's popularity and customer demand for its services.

```
db.hotel.aggregate([
 {
  $lookup: {
   from: "customer",
   localField: "Hotel_ID",
   foreignField: "Hotel_ID",
   as: "customer_details"
  }
 },
 {
  $group: {
   _id: {
    hotel_id: "$Hotel_ID",
    hotel_name: "$hotel_name"
   },
   number_of_customers: { $sum: 1 }
  }
 },
 {
  $sort: {
   number_of_customers: -1
  }
 }
]);
```



```
    _id: {
      hotel_id: 105
    },
    number_of_customers: 1
  }
  {
    _id: {
      hotel_id: 107
    },
    number_of_customers: 1
  }
  {
    _id: {
      hotel_id: 103
    },
    number_of_customers: 1
  }
  {
    _id: {
      hotel_id: 106
    },
    number_of_customers: 1
  }
  {
    _id: {
      hotel_id: 109
```

Summary:

The successful implementation of our hotel and restaurant management database marks a pivotal achievement in streamlining operations and maintaining data accuracy. The structured schema ensures a user-friendly and error-free environment, promoting smooth restaurant functionality. Consistent updates guarantee up-to-date information, enhancing the overall efficiency of the system. The database lays a robust foundation for organized data storage and retrieval, crucial for informed decision-making.

Future Recommendations:

Future scope: 1. Strategies can be developed to either capitalize on the popularity of a meal at a specific restaurant at a specific hotel, or an effort can be made to promote other dishes in order to enhance income.
2. Building on the previous suggestion, a menu consisting solely of well-liked products unique to the hotel could make it easier for patrons to get the greatest meals, increasing the likelihood that they would use the services again.
3. In order to expand their services to hotels that interest them and boost earnings for both hotels and restaurants, restaurants can be provided with information about the money hotels receive from food orders.