INST 733 Final Report Team H - Restaurant Management System

Introduction

Many people enjoy eating out at restaurants and hotels more and more these days, and as a result, restaurants need to monitor their business even more closely to get an idea of how it is doing in a competitive market. We believe storing this data effectively can help various food-centric businesses like restaurants, cafes and hotels in tracking various performance metrics that can help them be more profitable. Our system will focus on 1 restaurant and its multiple branches. It stores information regarding customers, their orders and order status. It contains employee information as well as the menu, payment, bills, delivery partners etc. Building a restaurant management system like this enables businesses to keep track of orders, workforce performance, branch operations and revenue, total sales, and help identify a loyal customer base.

Database Design and Implementation

Logical Design

The logical design is designed using an Entity Relationship Diagram

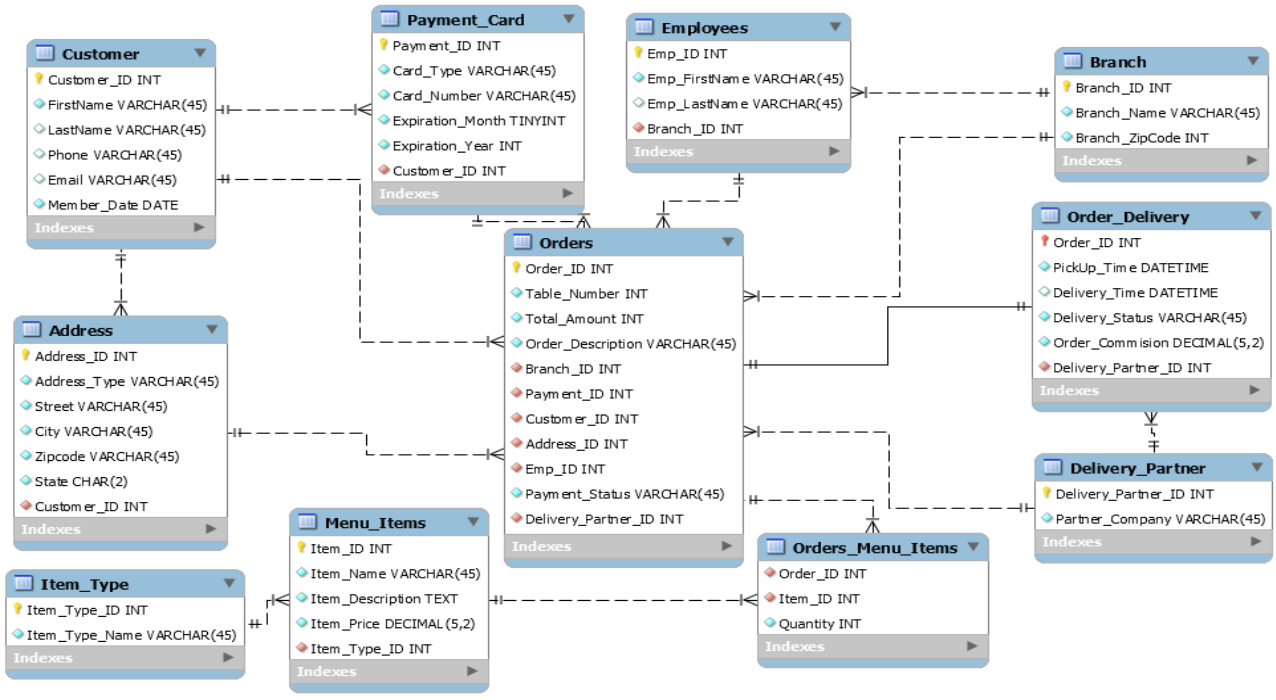

Table structure and contained data is as follows:

Address	
Fields	**Data Structure**
Address_ID	int(10) UN AI PK
Address_Type	varchar(45)
Street	varchar(45)
City	varchar(45)
Zipcode	varchar(45)
State	char(2)
Customer_ID	int(10) UN FK

Branch	
Fields	**Data Structure**
Branch_ID	int(10) UN AI PK
Branch_Name	varchar(45)
Branch_Zipcode	int(11)

Customer	
Fields	**Data Structure**
Customer_ID	int(10) UN AI PK
FirstName	varchar(45)
LastName	varchar(45)
Phone	varchar(45)
Email	varchar(45)
Member_Date	date

Delivery_Partner

Fields	Data Structure
Delivery_Partner_ID	int(10) UN PK
Partner_Company	varchar(45)

Employees	
Fields	**Data Structure**
Emp_ID	int(10) UN AI PK
Emp_FirstName	varchar(45)
Emp_LastName	varchar(45)
Branch_ID	int(10) UN FK

Item_Type	
Fields	**Data Structure**
Item_Type_ID	int(10) UN PK
Item_Type_Name	varchar(45)

Menu_items	
Fields	**Data Structure**
Item_ID	int(10) UN AI PK
Item_Name	varchar(45)
Item_Description	text
Item_Price	decimal(5,2)
Item_Type_ID	int(10) UN FK

Order_Delivery

Fields	Data Structure
Order_ID	int(10) UN AI PK
PickUp_Time	datetime
Delivery_Time	datetime
Delivery_Status	varchar(45)
Order_Commission	decimal(5,2)
Delivery_Partner_ID	int(10) UN FK

Orders	
Fields	**Data Structure**
Order_ID	int(10) UN PK
Table_Number	int(11)
Total_Amount	int(11)
Branch_ID	int(10) UN FK
Payment_ID	int(10) UN FK
Customer_ID	int(10) UN FK
Address_ID	int(10) UN FK
Emp_ID	int(10) UN FK
Paymenr_Status	varchar(45)
Delivery_Partner_ID	int(10) UN FK

Orders_menu_items	
Fields	**Data Structure**
Order_ID	int(10) UN FK
Item_ID	int(10) UN FK
Quantity	int(11)

Payment_card	
Fields	**Data Structure**
Payment_ID	int(10) UN AI PK
Card_Type	varchar(45)
Card_Number	varchar(45)
Expiration_Month	tinyint(4)
Expiration_Year	int(11)
Customer_ID	int(10) UN FK

Physical Database

We created the physical database using MySQL Workbench. The first step was to create sample data on separate Excel sheets. Using the import function of Workbench, we created the different tables. The entire scheme is then exported as a single, self – contained file. We created 20 records for most tables and around 32 records for the tables that would require more, for example, the orders table.

Sample Data

Address

Address_ID	Address_Type	Street	City	Zipcode	State	Customer_ID
101	Home	Tulane Dr	College Park	20740	MD	20
102	Work	14th Street NW	Washington DC	20015	DC	9
103	Home	16th Street NW	Washington DC	20018	DC	12
104	Home	18th Street NW	Washington DC	20021	DC	1
105	Home	7th Street NW	Washington DC	20740	DC	7

Branch

Branch_ID	Branch_Name	Branch_ZipCode
1	Tai Mumbai	20740
2	Tai Jaipur	20022
3	Tai Sikkim	20044
4	Tai Chennai	20008
5	Tai Kashmir	20078

Customer

Customer_ID	FirstName	LastName	Phone	Email	Member_Date
1	Krishna	Nambi	301-...	dea...	2018-02-21
2	Vi	Kamath	201-...	123...	2017-12-23
3	Abhishek	Shinde	301-...	alex...	2016-05-18
4	Alan	Walker	701-...	sim...	2019-04-11
5	Ariiit	Singh	201-...	mad...	2015-11-18

Delivery Partner

Delivery_Partner_ID	Partner_Company
1	Doordash
2	Postmates
3	Uber Eats
4	Grubhub
5	Yelp

Employee

Emp_ID	Emp_FirstName	Emp_LastName	Branch_ID
1	Rahul	Shah	2
2	Shvam	Verma	1
3	Murari	Chacha	2
4	Ram	Sharma	5
5	Govind	Dubbev	4

Item Type

Item_Type_ID	Item_Type_Name
1	Snack
2	Beverage
3	Main
4	Side
5	Dessert
6	Breakfast

Menu Items

Item_ID	Item_Name	Item_Description	Item_Price	Item_Type_ID
1	Nachos	Tortilla chips topped with cheese and jalapenos	3.99	1
2	Coffee	High-quality fresh-roasted coffee	2.99	2
3	Tea	A blend of select black tea and green teas. ias...	2.99	2
4	Onion Rings	Batter-fried onion rings, served with Zestable S...	3.99	4
5	Veggie Burger	A seasoned all-natural patty of brown rice, Pep...	6.99	3

Order Delivery

Order_ID	PickUp_Time	Delivery_Time	Delivery_Status	Order_Commision	Delivery_Partner_ID
1	2018-05-12 10:40:12	2018-05-12 11:15:16	On time	11.12	1
2	2018-05-14 17:35:12	2018-05-14 18:10:04	On time	9.36	2
3	2018-06-17 19:40:45	2018-06-17 20:55:07	Delaved	12.11	2
4	2018-06-21 21:45:14	2018-06-21 23:04:12	Delaved	8.45	3
5	2018-07-23 14:12:10	2018-07-23 15:32:11	Delaved	11.09	3

Orders

Order_ID	Table_Number	Total_Amount	Branch_ID	Payment_ID	Customer_ID	Address_ID	Emp_ID	Payment_Status	Delivery_Partner_ID
1	2	11.97	1	1	11	112	15	Paid	1
2	3	20.97	3	5	20	101	9	Paid	2
3	1	86.86	2	11	7	105	4	Paid	2
4	5	23.96	4	7	18	116	1	Paid	3
5	7	80.88	3	10	10	106	13	Paid	3

Order-Menu-Items Join

Order_ID	Item_ID	Quantity
1	10	1
1	3	1
1	20	1
2	8	2
2	17	1
3	14	1
3	12	5

Payment Card

Payment_ID	Card_Type	Card_Number	Expiration_Month	Expiration_Year	Customer_ID
1	AmericanExpress	345522118021408	4	2024	20
2	Visa	4556418843815500	7	2020	15
3	AmericanExpress	348371115672111	2	2021	11
4	Visa	4539078111739870	2	2025	8
5	Discover	6011460144020760	9	2021	14

Views/Queries

1. **We want to identify the top employees based on who is pulling the greatest number of orders**

```
drop view if exists team_h_restaurant_db.TopEmployees;
create view TopEmployees as
select e.Emp_ID, count(o.Order_ID) as count, e.Emp_FirstName, e.Emp_LastName
from orders o, employees e
where o.Emp_ID = e.Emp_ID
group by o.Emp_ID
order by count DESC
limit 5;
select * from TopEmployees;
```

Emp_ID	count	Emp_FirstName	Emp_LastName
10	3	Gangadhar	tilak
4	3	Ram	Sharma
7	3	Santosh	Mishra
8	2	Bhushan	Seth
2	2	Shyam	Verma

From this query we can identify the employees, their employee IDs and the number of orders they have served. This helps us in recognizing the employees who are working hard to serve tables in the restaurant and to see if they deserve raises/promotions.

2. **We want to see which branch is doing the best out of all 4 branches.**

```
drop view if exists team_h_restaurant_db.TopBranches;
create view TopBranches as
select b.Branch_ID, count(*) as count, b.Branch_Name, SUM(o.Total_Amount) as AmountTotal
from orders o, branch b
where o.Branch_ID = b.Branch_ID
group by o.Branch_ID
order by AmountTotal desc;
select * from TopBranches;
```

Branch_ID	count	Branch_Name	AmountTotal
2	10	Taj Jaipur	502.16
3	7	Taj Sikkim	369.45
1	8	Taj Mumbai	221.60
4	7	Taj Chennai	193.68

We can see that the Jaipur branch receives most orders resulting in a higher generated revenue. This will help the restaurant identify the profitable branches that are doing well and the branches like Taj Chennai that isn't doing as well and hence, would need changes in the way they operate.

3. **Next, we would like to see a quantity breakdown by zipcode to see what areas are buying bulk orders/buying the most in quantity**

```
drop view if exists zipcode_quantity;
create view zipcode_quantity as
select om.order_id, o.address_ID, a.Zipcode, sum(om.quantity) as sum
from orders_menu_items om join orders o on om.order_ID = o.order_ID join address a on o.address_ID = a.address_ID
group by a.Zipcode
order by sum desc;
select * from zipcode_quantity;
```

order_id	address_ID	Zipcode	sum
2	101	20740	44
12	103	20018	41
15	108	20016	33
13	120	20020	21
7	115	20078	17
1	112	20070	14
8	107	20022	14
17	110	20017	11
11	102	20015	5
20	113	20071	5
19	104	20021	3

It looks like Zip Code 20740 leads on buying the most, quantity wise. This could translate into restaurants recommending discounts and other incentives to those from zipcodes who aren't buying as much

4. **Another technique for restaurants to increase footfall is to capitalize on customer loyalty.**

```
drop view if exists CustomerLoyalty;
create view CustomerLoyalty as
select c.Customer_ID, count(*) as count, c.FirstName, c.LastName, c.Email
from orders o, customer c
where o.Customer_ID = c.Customer_ID
group by o.Customer_ID
order by count DESC;

select * from CustomerLoyalty;
```

Customer_ID	count	FirstName	LastName	Email
11	4	Ronald		ronald.g@gmail.com
3	4	Abhishek	Shinde	alex@outlook.com
10	3	Halev	D	mmk@icloud.com
2	3	Vi	Kamath	123@gmail.com
18	2	Simon		seeumd@umd.edu
6	2	James	Philip	ii@icloud.com
8	1	Chris	Martin	mart@icloud.com
15	1	Edward	Jones	jonese@umd.edu
4	1	Alan	Walker	sim@outlook.com
7	1	Alex	Dunphy	alex@outlook.com
14	1	Ted		daal@icloud.com

We can see the customers who have ordered more frequently from the restaurant. Accordingly, we can reward them with discounts and promotional offers to increase purchases and maybe even promote through word of mouth.

5. **We would like the restaurant to analyze the gross revenue breakdown by zipcode to identify the profitable areas.**

We identified the gross revenue generated across each zip code serviced by the chain of restaurants.

This excludes the order commission paid for each order to the delivery partner.

```
CREATE VIEW GROSS_REVENUE_PER_ZIPCODE AS
SELECT concat('$',sum(total_amount)-sum(order_commision)) as Revenue_Generated, zipcode as ZipCode
FROM orders a, address b, order_delivery c
WHERE a.address_id = b.address_id and a.order_id = c.order_id
GROUP BY b.zipcode;
SELECT * FROM GROSS_REVENUE_PER_ZIPCODE;
```

Revenue_Generated	ZipCode
$10.80	20015
$190.98	20016
$56.91	20017
$221.39	20018
$14.85	20019
$98.50	20020
$3.58	20021
$84.76	20022
$38.12	20070
$25.79	20071
$72.71	20078
$183.96	20740

As per the results obtained, we observe that the gross revenue generated is highest for zip code area 20740 whereas zip code area 20021 has generated the least amount of order and revenue.

6. **Another avenue of increasing revenue is by collaborating with credit card companies for promotional offers and discounts**

```
CREATE VIEW TopPaymentMethod AS
SELECT count(*) as 'No. of Transactions', Card_type
FROM payment_card
GROUP BY card_type
ORDER BY count(*) asc;
SELECT * FROM TopPaymentMethod;
```

No. of Transactions	Card_type
3	Discover
4	MasterCard
5	Visa
8	AmericanExpress

As seen from the above result, most of the orders paid through the portal are done via American Express card payment method. This analysis will help the restaurant build up offers with payment carrier for offers and discounts and help increase potential customers for the restaurant.

7. **In order for the restaurant to hold needed inventory, we would like to identify the popular food items**

```
CREATE VIEW TopFoodItem AS
SELECT sum(quantity) as Order_Count, item_name as 'Item Name', item_type_name as 'Item Type'
FROM orders_menu_items a, menu_items b, item_type c
WHERE a.item_id = b.item_id and b.item_type_id = c.item_type_id
GROUP BY a.item_id
ORDER BY Order_Count desc;

SELECT * FROM TopFoodItem;
```

Order_Count	Item Name	Item Type			
18	Finger Sandwich	Breakfast	10	Lemon Tart	Dessert
17	Cheesecake	Dessert	10	Spaghetti with M...	Main
17	Onion Rings	Side	8	Veggie Burger	Main
16	Nachos	Snack	8	Coffee	Beverage
15	Tea	Beverage	8	Chicken Burger	Main
14	Omlette	Breakfast	6	Hash Brown	Side
13	Apple Crumble Pie	Dessert	5	Soda	Beverage
13	Pancakes	Breakfast	4	Hummus Platter	Snack
11	Tater Tots	Side	4	Tacos	Snack
11	Pesto Pasta	Main	3	French Fries	Side

The above results obtained will help the restaurant to identify their top selling items and item types in the restaurant to decide their famous speciality. This will also help them to identify the least ordered dishes in order to improve the count or even change the menu items accordingly.

8. **Identifying the top delivery partners is essential to the restaurant to keep business booming when it comes to off-site food delivery.**

```
CREATE VIEW DeliveryPartner AS
SELECT count(delivery_partner_id) as 'No. of Orders', partner_company as 'Delivery Partner'
FROM order_delivery JOIN delivery_partner
USING(Delivery_Partner_ID)
GROUP BY Delivery_Partner_ID
ORDER BY count(delivery_partner_id) desc;
```

SELECT * FROM DeliveryPartner;

No. of Orders	Delivery Partner
9	Postmates
8	Doordash
8	Uber Eats
4	Grubhub
3	Yelp

The above results displayed identifies the top delivery partners associated with the restaurant based on the number of orders delivered and their availability to restaurant delivery calls. This will help the restaurant identify the top delivery platforms that customers usually use for ordering at their restaurants and will help them to work on offers and deals on food items with these partners.

9. **We would also like to know the average time it takes to deliver food to know how the delivery employees are performing**

```
CREATE VIEW AvgOrderTime AS
SELECT SEC_TO_TIME(avg(TIME_TO_SEC(TIMEDIFF(delivery_time, pickup_time)))) as 'Avg Time Difference', delivery_status as 'Delivery Status'
FROM order_delivery
GROUP BY delivery_status
ORDER BY delivery_status desc;
```

SELECT * FROM AvgOrderTime;

Avg Time Difference	Delivery Status
00:39:07.7142	On time
01:14:37.0000	Delaved

The average delivery time for orders delivered on time and for delayed delivers will help the management to identify ways to better improve their services offered to the customers. Delayed orders usually lead to losing customers and hence the average time taken for on time deliveries and delayed deliveries will help the management to identify areas and zip codes that they can serve based on the average time taken and avoid late deliveries.

10. **We would also like to analyze the performance of the delivery partner companies and their delivery performance w.r.t. delayed and on-time deliveries**

```
CREATE VIEW DeliveryPartnerRating AS
SELECT X.ON_TIME, X.DELAYED_ORDERS, Partner_Company, concat(Round(((X.ON_TIME)/(X.ON_TIME + X.DELAYED_ORDERS))*100,0),'%') as Rating
FROM
(SELECT A.on_time_count as ON_TIME, B.delayed_count as DELAYED_ORDERS, A.on_time_partner_id as PARTNER_ID
FROM
(
SELECT count(delivery_status) as on_time_count,delivery_partner_id as on_time_partner_id, delivery_status
from order_delivery
where delivery_status like 'On time'
group by delivery_partner_id,delivery_status
)A
INNER JOIN
(
SELECT count(delivery_status) as delayed_count,delivery_partner_id as delayed_partner_id, delivery_status
from order_delivery
where delivery_status like 'Delayed'
group by delivery_partner_id,delivery_status
)B
ON A.on_time_partner_id=B.delayed_partner_id)X
INNER JOIN delivery_partner
ON X.PARTNER_ID=delivery_partner.delivery_partner_id;
```

```
SELECT * FROM DeliveryPartnerRating;
```

ON_TIME	DELAYED_ORDERS	Partner_Company	Rating
4	4	Doordash	50%
7	2	Postmates	78%
5	3	Uber Eats	63%
3	1	Grubhub	75%
2	1	Yelp	67%

The above analysis will help track the performance of the delivery partners so the restaurant can decide and improve their partnerships with the delivery companies. The rating factor is calculated on the number of on time deliveries made over the total number of deliveries made.

The queries above satisfy all requirements as follows:

Query	**Requirement A**	**Requirement B**	**Requirement C**	**Requirement D**	**Requirement E**
Query 1	X	X	X		
Query 2	X	X	X		
Query 3	X	X		X	

Query 4	X	X	X		
Query 5	X	X	X		
Query 6	X		X		
Query 7	X	X	X		
Query 8	X		X	X	
Query 9	X		X		
Query 10	X	X	X	X	X

Changes from original design

- Our original design had other table elements describing the workflow of a restaurant management such as suppliers, reservation system and billing system.
- Some of these entities were merged with other entities in the ERD such as payment methods and payment status in order table. Entities such as reservation system and suppliers were dropped to reduce the complexity of relationships between the entities.
- New entities such delivery partners, menu items, item types, payment card and address were introduced in the system.
- Various relationships such as one-to-many and many-to-many relationships were created after the addition of new entities.

Issues during development

- **Issues Encountered and Solutions**
    - We first encountered certain issues while implementing the forward engineering process which involved various integrity and constraints issues. We had to identify the duplicate foreign key value names and differentiate them.
    - There were also some issues related to the relationship established with certain tables and had to be changed from one-to-many to one-to-one cardinality, for example, the relationship between order delivery and orders tables
    - We also faced issues due to the atomicity of the entities such as customer and address where we use the field values as name for customer name instead of differentiating it as first name and last name and for the address table where we had to break the address down to street, city, zip code and state.
    - Issues were also encountered with respect to the consistency of the database while adding the required data. Foreign key columns in orders and order delivery table

were not consistent with each other which were then solved using the change data structure module in workbench.

- Many values were replicated which were not supposed to be replicated which led to inconsistent query results. This was identified and rectified using the change data structure module in workbench.
- Certain queries had insufficient data to process and analyze the result which was rectified by adding more data to the orders table. For example, while populating the orders table we realized we would have to repeat certain customers although with different order IDs but same payment method. This led to us populating other connected tables accordingly and we went from 20 orders to 32 orders.

Lessons Learned

- We realized that we had to plan our data properly before proceeding with the update to the schema. When we encountered issues while adding data to Workbench, we decided to pause, finalize the different table data in Excel first and then proceed.
- We miscalculated the data required, inconsistent data addition and problems with adding data to the child table involved working with all the connected parent tables again.

Potential Future Scope

- ### Extensions within current technology and design

  We would definitely like to add some more functionality in the system. For example, allow users to store multiple addresses (like Home, Work, etc.) and multiple payment cards. We could also add table reservation systems, suppliers' entities and events entities to enhance the system and create a central tracking system for analysis and decisions. We could also modify employee information to show specific titles that we can later analyze, for example, if we wanted to identify the managers across branches who are doing well, or the wait staff that across branches that serve tables efficiently.

- ### Feasibility of alternative implementations

  We can implement a NoSQL data structure in order to reduce the complexity of multiple data structures and also add customer reviews to the food items in the restaurant.