**INST 733 Final Report Team H - Restaurant Management System**

Krishnakumar Nambi

Vyjayanthi Kamath

**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

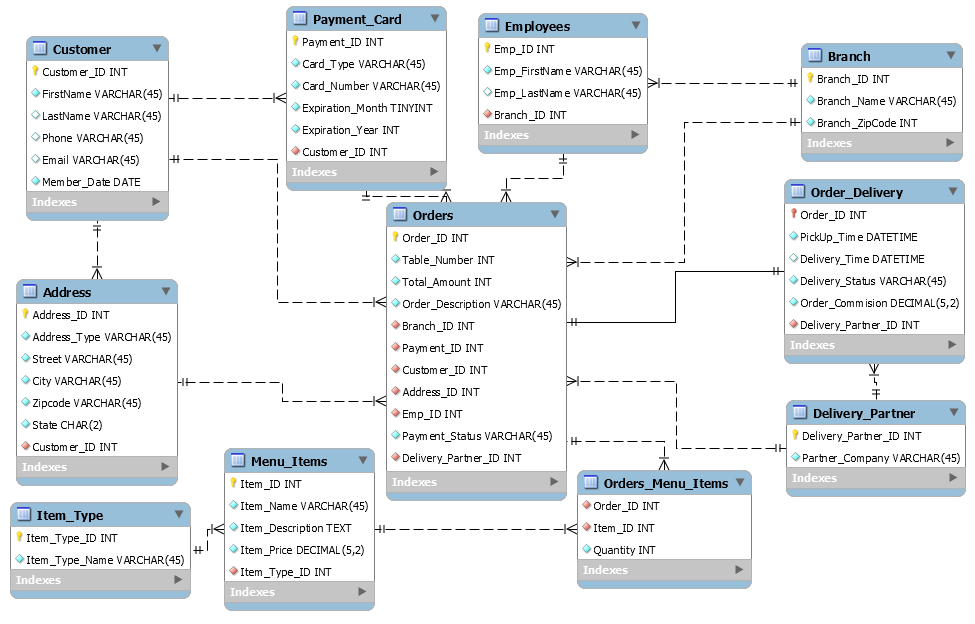


Table structure and contained data is as follows:

|  |  |
| --- | --- |
| **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) |

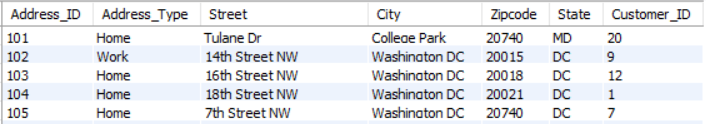
|  |  |
| --- | --- |
| **Payment\_card** | |
| **Fields** | **Data Structure** |
| Payment\_ID | int(10) UN AI PK |
| Card\_Type | varchar(45) |
| Card\_Number | varchar(45) |
| Expiration\_Month | tinyint(4) |
| Expiration\_Year | int(11) |
| Customer\_ID | int(10) UN FK |

**Physical Database**

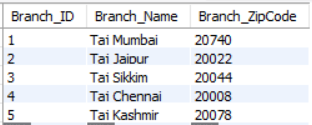
We created the physical database using MySQL Workbench. The first step was to create sample data on separate Excel sheets. Using the import function of Workbench, we created the different tables. The entire scheme is then exported as a single, self – contained file. We created 20 records for most tables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

**Sample Data**

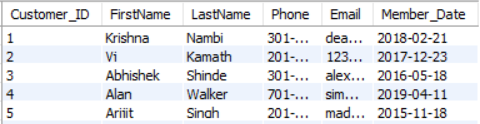
Address



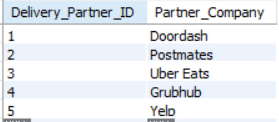
Branch



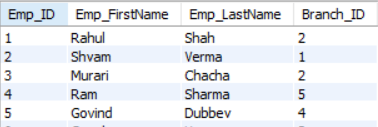
Customer



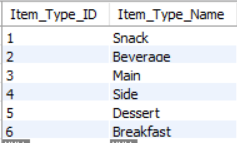
Delivery Partner



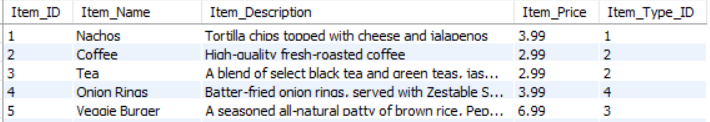
Employee



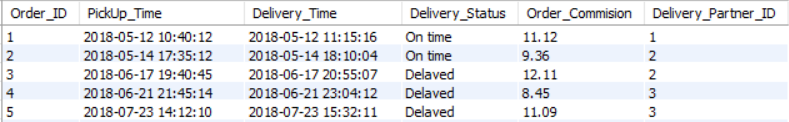
Item Type



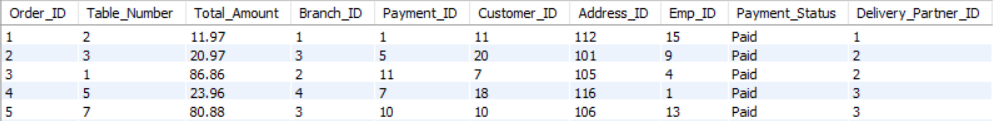
Menu Items



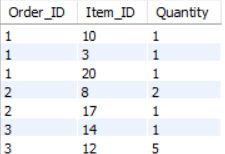
Order Delivery



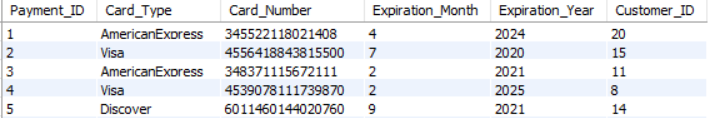
Orders



Order-Menu-Items Join



Payment Card



**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

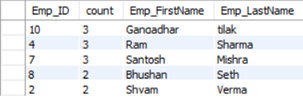
where o.Emp\_ID = e.Emp\_ID

group by o.Emp\_ID

order by count DESC

limit 5;

select \* from TopEmployees;



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.

1. **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

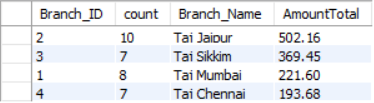
from orders o, branch b

where o.Branch\_ID = b.Branch\_ID

group by o.Branch\_ID

order by AmountTotal desc;

select \* from TopBranches;



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.

1. **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

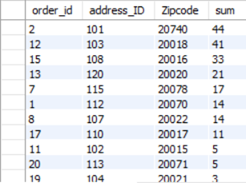
select om.order\_id, o.address\_ID, a.Zipcode, sum(om.quantity) as sum

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;



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

1. **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

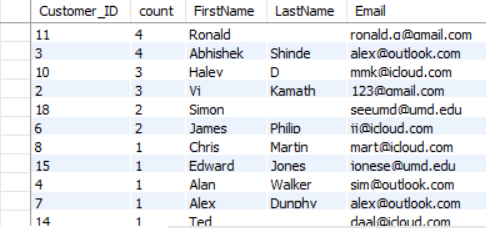
from orders o, customer c

where o.Customer\_ID = c.Customer\_ID

group by o.Customer\_ID

order by count DESC;

select \* from CustomerLoyalty;



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.

1. **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

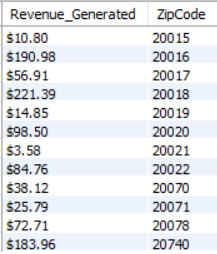
SELECT concat('$',sum(total\_amount)-sum(order\_commision)) as Revenue\_Generated, zipcode as ZipCode

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;



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.

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

CREATE VIEW TopPaymentMethod AS

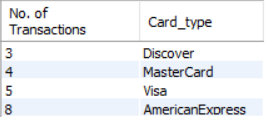
SELECT count(\*) as 'No. of Transactions', Card\_type

FROM payment\_card

GROUP BY card\_type

ORDER BY count(\*) asc;

SELECT \* FROM TopPaymentMethod;



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.

1. **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'

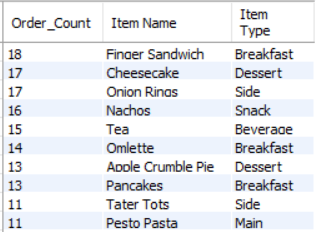
FROM orders\_menu\_items a, menu\_items b, item\_type c

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;



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.

1. **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'

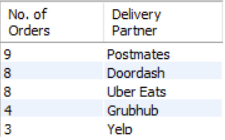
FROM order\_delivery JOIN delivery\_partner

USING(Delivery\_Partner\_ID)

GROUP BY Delivery\_Partner\_ID

ORDER BY count(delivery\_partner\_id) desc;

SELECT \* FROM DeliveryPartner;



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.

1. **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

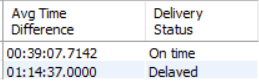
SELECT SEC\_TO\_TIME(avg(TIME\_TO\_SEC(TIMEDIFF(delivery\_time, pickup\_time)))) as 'Avg Time Difference', delivery\_status as 'Delivery Status'

FROM order\_delivery

GROUP BY delivery\_status

ORDER BY delivery\_status desc;

SELECT \* FROM AvgOrderTime;



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.

1. **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

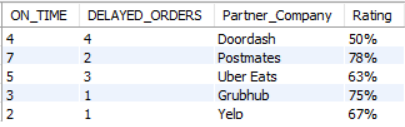
)B

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;



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.