# **Restaurant Management System**



## INTRODUCTION

Database management has been a critical and common practice employed across industries for many years. At its core, data management encompasses all disciplines related to managing data as a strategic and valuable resource, including collecting, processing, governing, sharing, and analyzing data and doing it all in a cost-efficient, cost-effective and reliable manner. We have decided on a project which focuses on creating, storing, and retrieving the data.

We as a team of five from different backgrounds came together to work on an exciting project using real time data where we’ll be managing the huge data of a restaurant. We provide managers with all the information they require to operate a restaurant in our project, which is a restaurant management system, from making food orders to creating bills and collecting payments. To view employee and client information, a manager must log in and have permission to access the project. He/she can also modify, include, or eliminate a consumer.

## PROJECT DESCRIPTION

A restaurant ordering system that manages users, table reservations, menus, inventory, orders and payments. It provides a food order database design for managing restaurant food orders.

The ordering system is implemented to automate order processing and efficiently handle peak orders. This increases customer satisfaction with less effort. This would be a win-win situation for hospitality businesses.

The design consists of a user table to store user information. The same spreadsheet can be used to manage different types of users such as manager, chefs, and customers. This can be used to associate users with menus, items, table reservations and orders. Users can track their tables and orders.

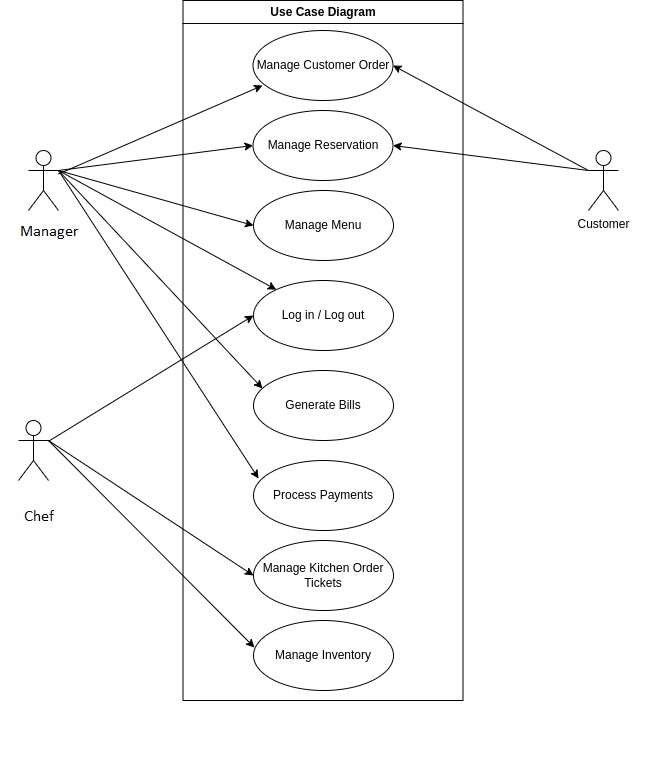
The item table is a table which consists of all the inventory related to the kitchen. They are mapped to identify suppliers who can supply uncooked items to replenish inventory. In a more advanced scenario, to support multiple suppliers for the same item, there may be another table that stores the relationship between items and suppliers. The same table can also be used to store materials and items for simulation.

There is a table for status for the user to check the status of his table. A table's status can be Free, Reserved, or Active. Booking Table allows you to reserve a restaurant table online or onsite. Logged-in or existing users can also be linked to bookings. It is also assumed that only tables with status Free can be reserved. Once your reservation is confirmed, you can change the status of your table to Reserved. Additionally, as soon as a guest occupies the table, the status of the table can be set to active.

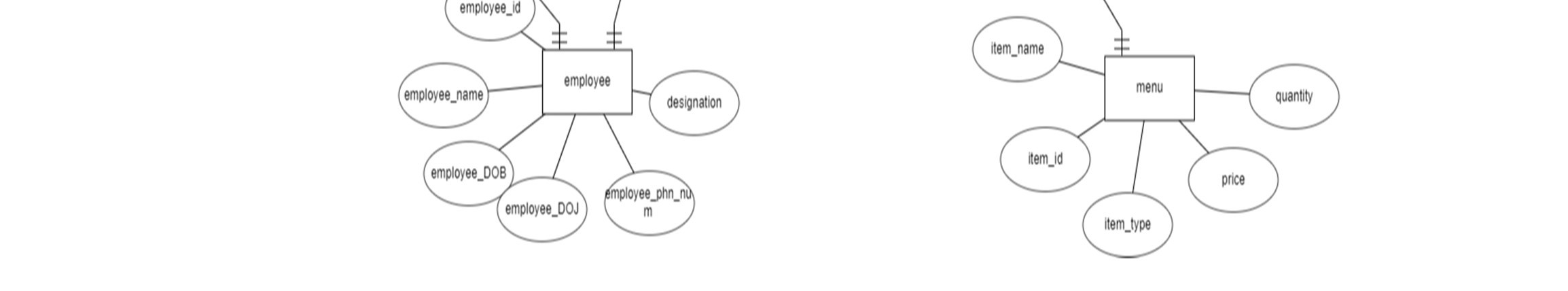
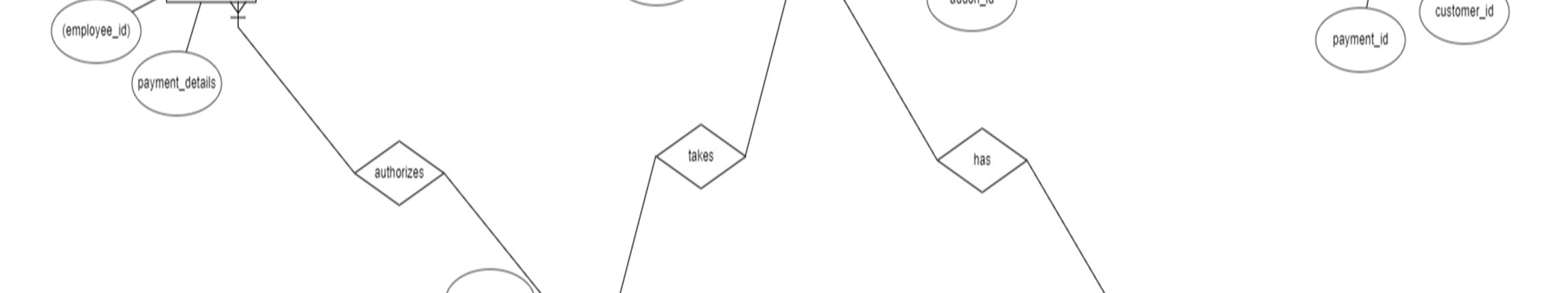
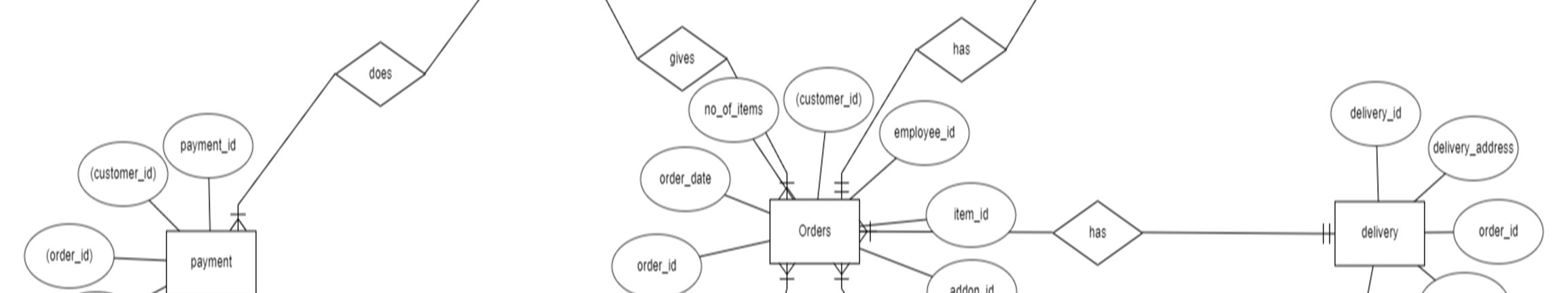
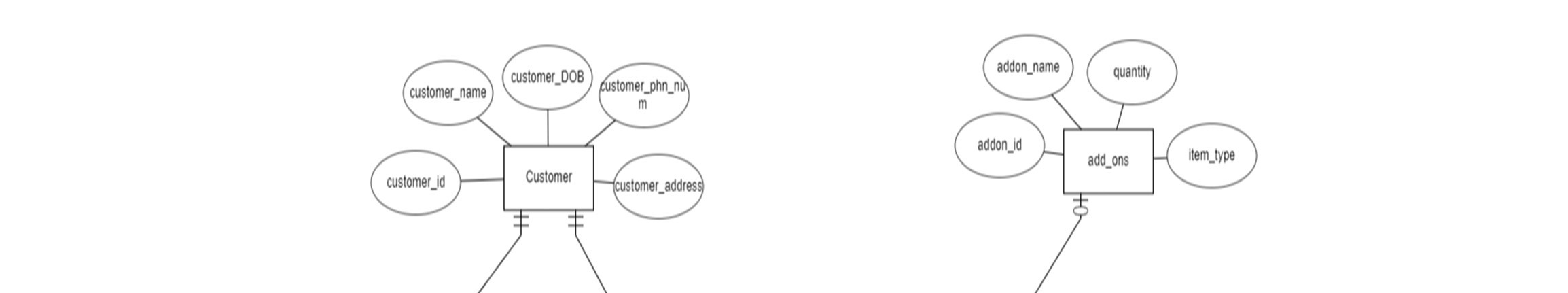
Order management tables is used to assign logged-in users to orders. The orders table can also be used to store completed reservations and supplier orders. A supplier's order status can be set to New while the order is being placed and to Completed after the item is received from the supplier. Also, you must manually enter the product price after receiving the product from the vendor.

A transaction table to track payments for orders made by guests to the restaurant and orders made by the restaurant to suppliers for accounting. You can also use the same spreadsheet to record credit and debit transactions.

## USE-CASE DIAGRAM



## Restaurant Management Database - Entity Relationship Diagram



Business Rules

### Customer

* A customer can place one or more orders
* A customer can do one or more payments

### Payment

* One or more payments can be done by one customer
* One or more payments can be recorded by one employee
* One payment can be done for one or more orders

### Employee

* One employee can record one or more payments
* One employee can record one or more orders

### Orders

* One or more orders can be placed by a customer
* One or more orders can be paid through one payment
* One or more orders can be recorded by an employee
* One order can have zero or many addons
* One or more orders can by delivered through a delivery
* One or more orders can have an item

Addons

* zero or more addons can be placed in an order

Delivery

* A delivery can have one or more orders

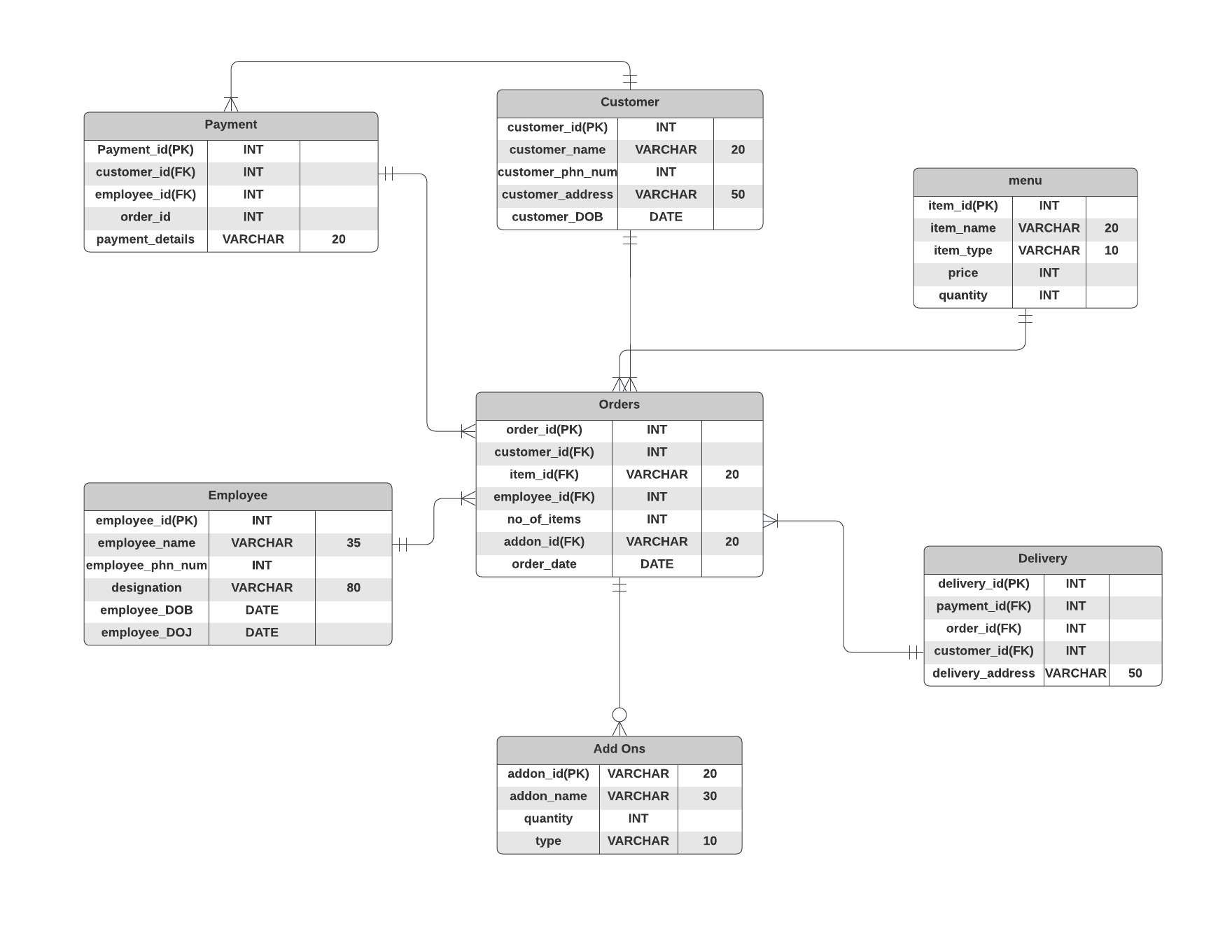
### Item

* One item can be placed in one or more orders
* One or more items can be in one Menu

Menu

* One menu can have one or more items

## Database Schema



Physical Design

#Creating Employee Table

CREATE TABLE Employee

(

employee\_id INT NOT NULL, employee\_name VARCHAR(95), employee\_DOB DATE, employee\_DOJ DATE, employee\_phn\_num INT , designation VARCHAR(80),

PRIMARY KEY (employee\_id)

);

#Creating Add ons Table

CREATE TABLE Add\_ons

(

addon\_id VARCHAR(20), addon\_name VARCHAR(30),

quantity INT NOT NULL, item\_type VARCHAR(10),

PRIMARY KEY (addon\_id)

);

#Creating Menu Table

CREATE TABLE Menu

(

item\_id INT NOT NULL, item\_name VARCHAR(20), item\_type VARCHAR(10),

price INT , quantity INT,

PRIMARY KEY (item\_id)

);

#creating customer table

CREATE TABLE Customer

(

customer\_id INT NOT NULL, customer\_name VARCHAR(20), customer\_DOB DATE, customer\_phn\_num INT , customer\_address VARCHAR(50),

PRIMARY KEY (customer\_id)

);

#Creating Orders Table

CREATE TABLE Orders

(

order\_id INT NOT NULL, order\_date DATE, no\_of\_items INT, customer\_id INT NOT NULL, employee\_id INT NOT NULL, item\_id INT NOT NULL, addon\_id VARCHAR(20),

PRIMARY KEY (order\_id),

FOREIGN KEY (customer\_id) REFERENCES Customer(customer\_id),

FOREIGN KEY (employee\_id) REFERENCES Employee(employee\_id),

FOREIGN KEY (item\_id) REFERENCES Menu(item\_id),

FOREIGN KEY (addon\_id) REFERENCES Add\_ons(addon\_id)

);

#Creating Payment Table

CREATE TABLE Payment

(

payment\_id INT NOT NULL, payment\_details VARCHAR(20), customer\_id INT NOT NULL, employee\_id INT NOT NULL, order\_id INT NOT NULL,

PRIMARY KEY (payment\_id),

FOREIGN KEY(customer\_id) REFERENCES Customer(customer\_id),

FOREIGN KEY (order\_id) REFERENCES Orders(order\_id),

FOREIGN KEY (employee\_id) REFERENCES Employee(employee\_id) );

#Creating Delivery Table

CREATE TABLE Delivery

(

delivery\_id INT NOT NULL, delivery\_address VARCHAR(50), order\_id INT NOT NULL, customer\_id INT NOT NULL, payment\_id INT NOT NULL,

PRIMARY KEY (delivery\_id),

FOREIGN KEY (customer\_id) REFERENCES Customer(customer\_id),

FOREIGN KEY (payment\_id) REFERENCES Payment(payment\_id),

FOREIGN KEY (order\_id) REFERENCES Orders(order\_id)

);

#insert employee

INSERT INTO Employee Values (001,"Lalith","1990-08-23","2022-03-01","256718321","Chef");

INSERT INTO Employee Values (002,"Jyoshna","1947-08-13","2022-01-

01","486634278","Manager");

INSERT INTO Employee Values (003,"Monisha","1996-02-09","2022-04-23","252144141","Chef");

INSERT INTO Employee Values (004,"Hruthika","1994-04-28","2022-05-

11","244416321","Supervisor");

INSERT INTO Employee Values (005,"Sanjay","1997-12-16","2022-05-20","313433546","Manager");

INSERT INTO Employee Values (006,"Sai","1991-06-12","2022-06-19","827362515","Assistant");

INSERT INTO Employee Values (007,"Raju","1987-05-03","2022-07-11","351414441","Delivery Agent");

INSERT INTO Employee Values (008,"Mahendiran","1987-02-09","2022-0712","826142363","Delivery Agent");

INSERT INTO Employee Values (009,"Kancharla","1994-07-06","2022-0711","282351534","Server");

INSERT INTO Employee Values (010,"Tarun","1996-12-09","2022-06-14","721735141","Server"); #insert Add\_ons

INSERT INTO Add\_ons Values("ADD001","Sambar",1,"Sides");

INSERT INTO Add\_ons Values("ADD002","Filter Coffee",1,"Drink");

INSERT INTO Add\_ons Values("ADD003","Chutney",1,"Sides");

INSERT INTO Add\_ons Values("ADD004","Bonda",3,"Snack");

INSERT INTO Add\_ons Values("ADD005","Kesari",1,"Dessert");

INSERT INTO Add\_ons Values("ADD006","Payasam",1,"Dessert");

INSERT INTO Add\_ons Values("ADD007","Badam Milk",2,"Drink");

INSERT INTO Add\_ons Values("ADD008","Bajji",5,"Snack");

INSERT INTO Add\_ons Values("ADD009","Tea",1,"Drink");

INSERT INTO Add\_ons Values("ADD010","Coke",1,"Drink");

#insert Menu

INSERT INTO Menu Values(001,"Idly","Tiffin",15,2);

INSERT INTO Menu Values(002,"Dosa","Tiffin",30,1);

INSERT INTO Menu Values(003,"Poori","Tiffin",40,2);

INSERT INTO Menu Values(004,"Chapathi","Tiffin",50,2);

INSERT INTO Menu Values(005,"Upma","Tiffin",60,1);

INSERT INTO Menu Values(006,"Rasam Rice","Lunch",50,1);

INSERT INTO Menu Values(007,"Curd Rice","Lunch",70,1);

INSERT INTO Menu Values(008,"Soup","Snack",150,1);

INSERT INTO Menu Values(009,"Chat","Snack",90,1);

INSERT INTO Menu Values(010,"Parota","Dinner",80,1);

#insert Customer

INSERT INTO Customer Values(01, "Aravind","1999-09-09", 23456789, "4121 Bailey Ave");

INSERT INTO Customer Values(02, "Varun", "1997-03-21",98765432, "4132 Bailey Ave");

INSERT INTO Customer Values(03, "Roshan","2003-05-30" ,567894321, "53 Tyler");

INSERT INTO Customer Values(04, "Neha", "1995-04-17",234567890, "14 Merimac");

INSERT INTO Customer Values(05, "Bindu", "1992-06-05",7890643, "63 Tyler");

INSERT INTO Customer Values(06, "Samantha","2000-03-19" ,34567892, "4252 Bailey Ave");

INSERT INTO Customer Values(07, "Smruthi","1998-04-15" ,45678322, "5172 Bailey Ave");

INSERT INTO Customer Values(08, "Rashmika","1995-09-25" ,56789432, "66 Tyler");

INSERT INTO Customer Values(09, "Prabhas", "1999-03-27",78965432, "78 Tyler");

INSERT INTO Customer Values(10, "Yashwanth","2001-02-02", 89076543, "15 Merimac");

#insert Order

INSERT INTO Orders Values(001,'2021-03-11',2,01,002,001,"ADD001");

INSERT INTO Orders Values(002,"2022-03-25",1,03,002,002,"ADD002");

INSERT INTO Orders Values(003,"2021-03-18",3,03,004,003,"ADD003");

INSERT INTO Orders Values(004,"2022-07-11",1,04,004,004,"ADD004");

INSERT INTO Orders Values(005,"2022-07-08",2,05,005,005,"ADD005");

INSERT INTO Orders Values(006,"2021-11-20",2,02,006,006,"ADD006");

INSERT INTO Orders Values(007,"2021-09-08",1,07,006,007,"ADD007");

INSERT INTO Orders Values(008,"2021-11-14",1,08,006,009,"ADD008");

INSERT INTO Orders Values(009,"2021-04-27",3,09,006,008,"ADD010");

INSERT INTO Orders Values(010,"2022-02-25",3,10,006,010,"ADD009");

INSERT INTO Orders Values(011,"2022-05-24",2,01,002,001,"ADD001");

INSERT INTO Orders Values(012,"2022-08-11",1,03,002,002,"ADD002");

INSERT INTO Orders Values(013,"2021-12-12",3,03,004,003,"ADD003");

INSERT INTO Orders Values(014,"2022-01-09",1,04,004,004,"ADD004");

INSERT INTO Orders Values(015,"2022-05-12",2,05,005,005,"ADD005");

INSERT INTO Orders Values(016,"2021-04-28",2,02,006,006,"ADD006");

INSERT INTO Orders Values(017,"2021-04-25",2,05,005,005,"ADD005");

INSERT INTO Orders Values(018,"2021-07-05",2,02,006,006,"ADD006");

INSERT INTO Orders Values(019,"2022-05-22",1,07,006,007,"ADD007");

INSERT INTO Orders Values(020,"2021-08-29",1,08,006,009,"ADD008");

INSERT INTO Orders Values(021,null,2,02,003,006,"ADD003");

INSERT INTO Orders Values(022,null,3,04,002,005,"ADD005"); INSERT INTO Orders Values(023,null,1,05,004,002,"ADD008");

#insert payment

INSERT INTO Payment Values (1,'Gpay',1,2,1);

INSERT INTO Payment Values (2,'Cash',3,2,2);

INSERT INTO Payment Values (3,'Phonepe',3,4,3);

INSERT INTO Payment Values (4,'Apple pay',4,4,4);

INSERT INTO Payment Values (5,'Paytm',5,5,5);

INSERT INTO Payment Values (6,'Cash',2,6,6);

INSERT INTO Payment Values (7,'Gpay',7,6,7);

INSERT INTO Payment Values (8,'Applepay',8,6,8);

INSERT INTO Payment Values (9,'Applepay',9,6,9);

INSERT INTO Payment Values (10,'Phonepe',10,6,10);

INSERT INTO Payment Values (11,'Gpay',1,2,11);

INSERT INTO Payment Values (12,'Cash',3,2,12);

INSERT INTO Payment Values (13,'Phonepe',3,4,13);

INSERT INTO Payment Values (14,'Apple pay',4,4,14);

INSERT INTO Payment Values (15,'Paytm',5,5,15);

INSERT INTO Payment Values (16,'Cash',2,6,16);

INSERT INTO Payment Values (17,'Gpay',5,5,17);

INSERT INTO Payment Values (18,'Applepay',2,6,18);

INSERT INTO Payment Values (19,'Applepay',7,6,19);

INSERT INTO Payment Values (20,'Phonepe',8,6,20);

#insert Delivery

INSERT INTO Delivery Values(1,"4121 Bailey Avenue",001,01,15);

INSERT INTO Delivery Values(2,"4252 Bailey Avenue",006,06,10);

INSERT INTO Delivery Values(3,"5172 Bailey Avenue",007,07,11);

INSERT INTO Delivery Values(4,"4132 Bailey Avenue",002,02,16);

INSERT INTO Delivery Values(5,"15 Merimac",010,10,19);

INSERT INTO Delivery Values(6,"14 Merimac",004,04,12);

INSERT INTO Delivery Values(7,"78 Tyler",009,09,18);

INSERT INTO Delivery Values(8,"66 Tyler",008,08,14);

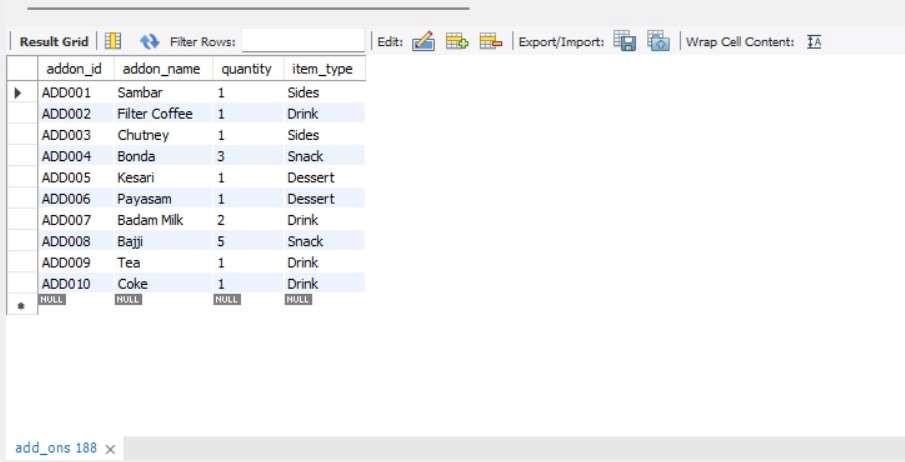
INSERT INTO Delivery Values(9,"53 Tyler",003,03,13);

INSERT INTO Delivery Values(10,"63 Tyler",005,05,17);

Display Table Queries:

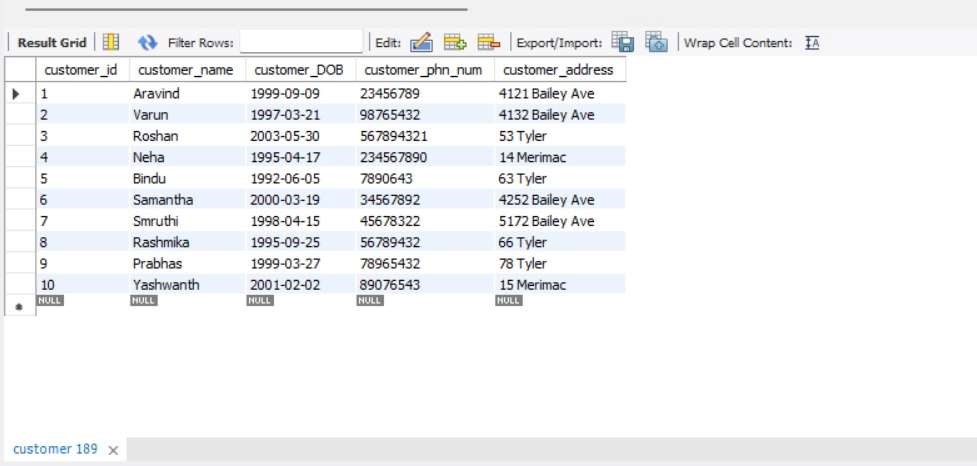
# Add ons Table

Select \* from add\_ons;



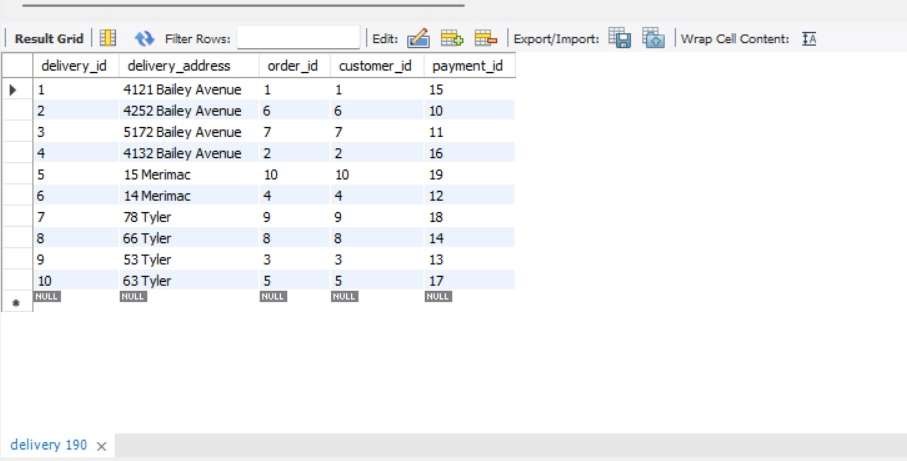
# Customer Table

Select \* from customer;



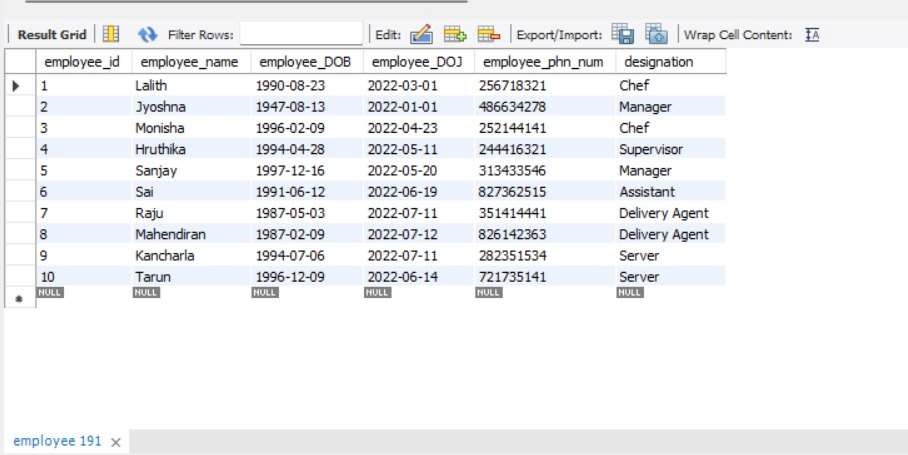
# Delivery Table

Select \* from delivery;



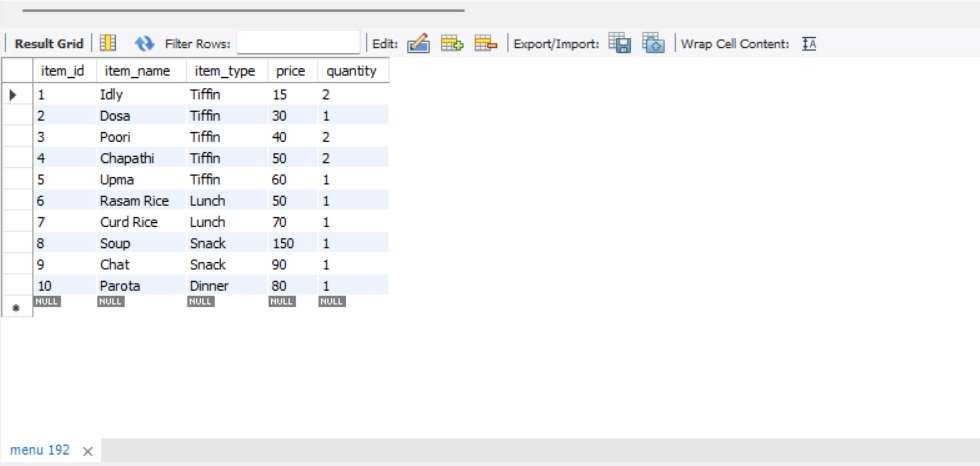
# Employee Table

Select \* from employee;



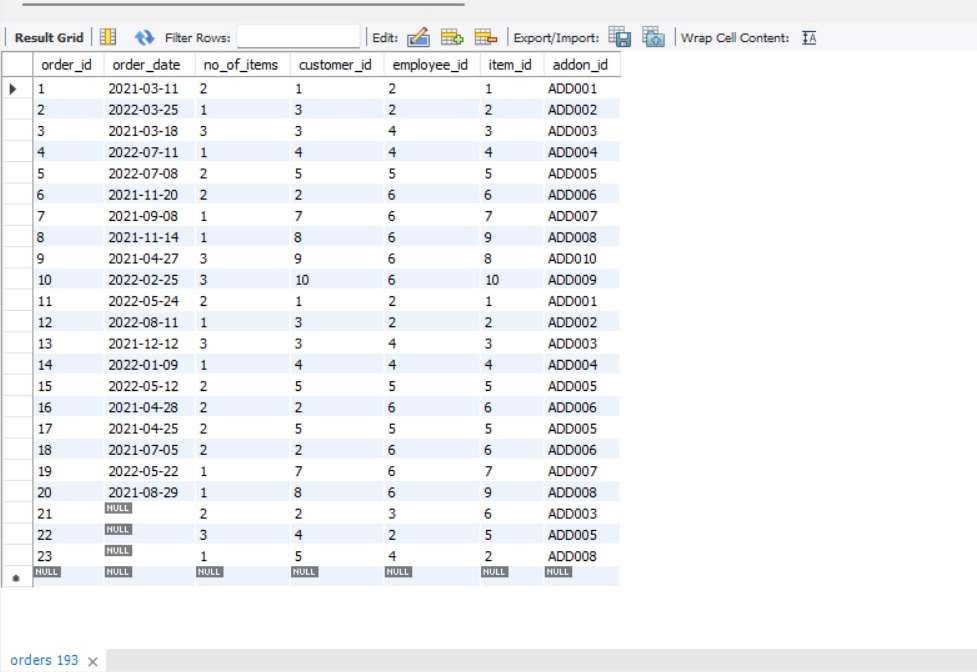
# Menu Table

Select \* from menu;



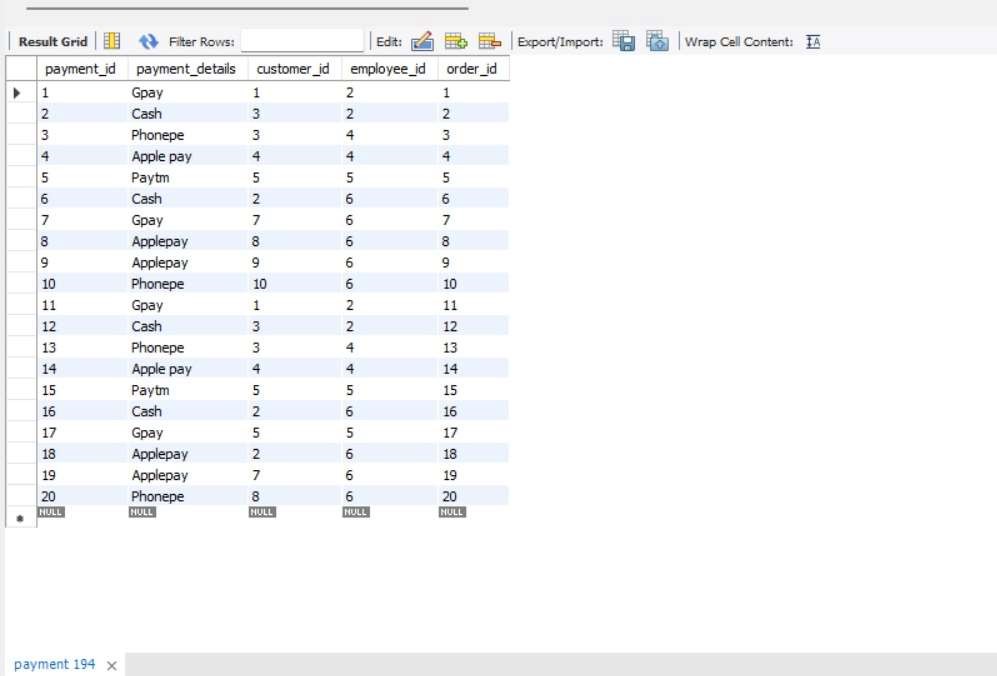
#Order Table

Select \* from orders;



# Payment Table

Select \* from payment;



## Populated data and query output

Query 1: Employees whose name starts with S who billed more than two orders

Select E.employee\_id as Employee\_ID,

E.employee\_name as Employee\_Name,

count(P.payment\_id) as No\_of\_transactions\_handled

From Employee as E

JOIN Orders as O on E.employee\_id = O.employee\_id

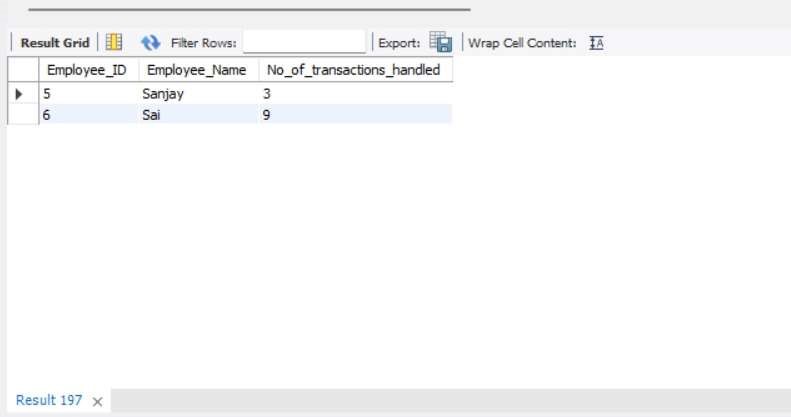
JOIN Payment as P on O.order\_id = P.payment\_id

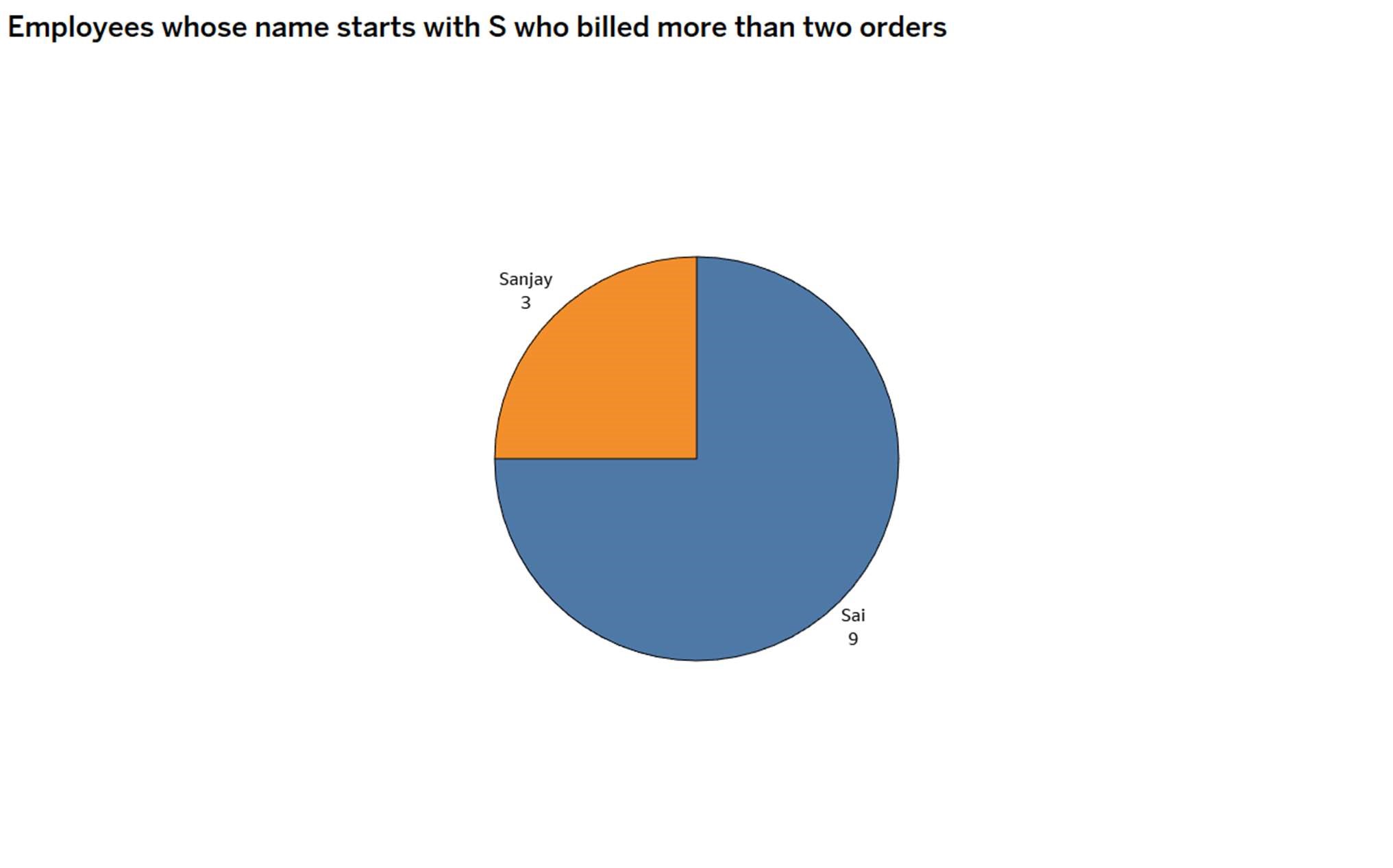
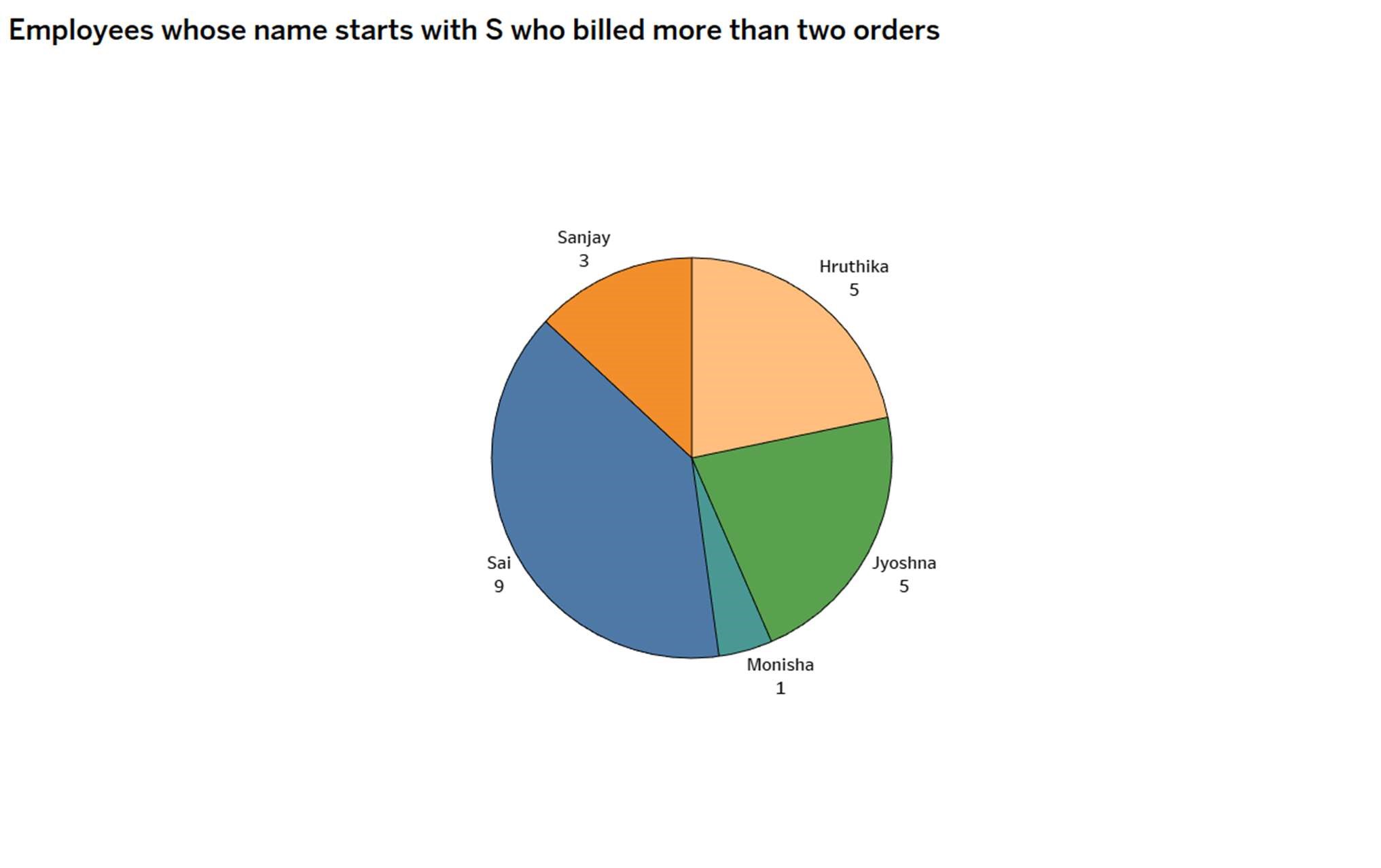
Where E.employee\_name Like "S%"

GROUP BY E.employee\_id

Having Count(P.payment\_id)>2;

OUTPUT:





Query 2: Display summary of customers who paid using cash or apple pay and the type of item is Tiffin using string functions.

Select concat(upper(c.customer\_name)," customer ID is ",c.customer\_id," and order ID is

",o.order\_id," bought ",m.item\_name," , item code being

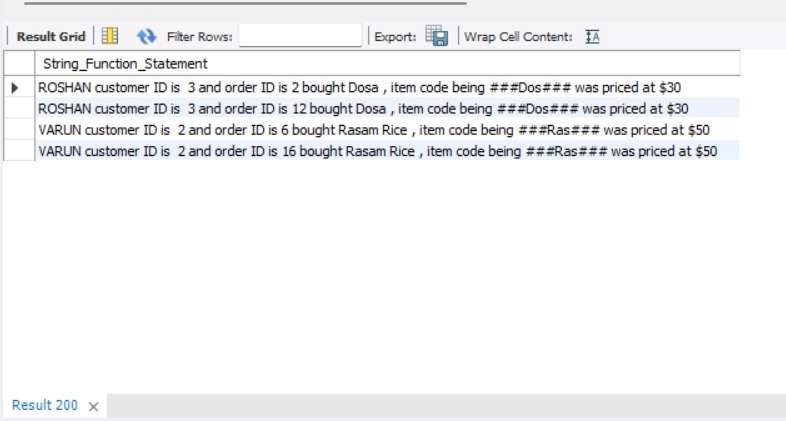
",LPAD(RPAD(substr(m.item\_name,1,3),6,'#'),9,'#'), " was priced at $",m.price) as String\_Function\_Statement

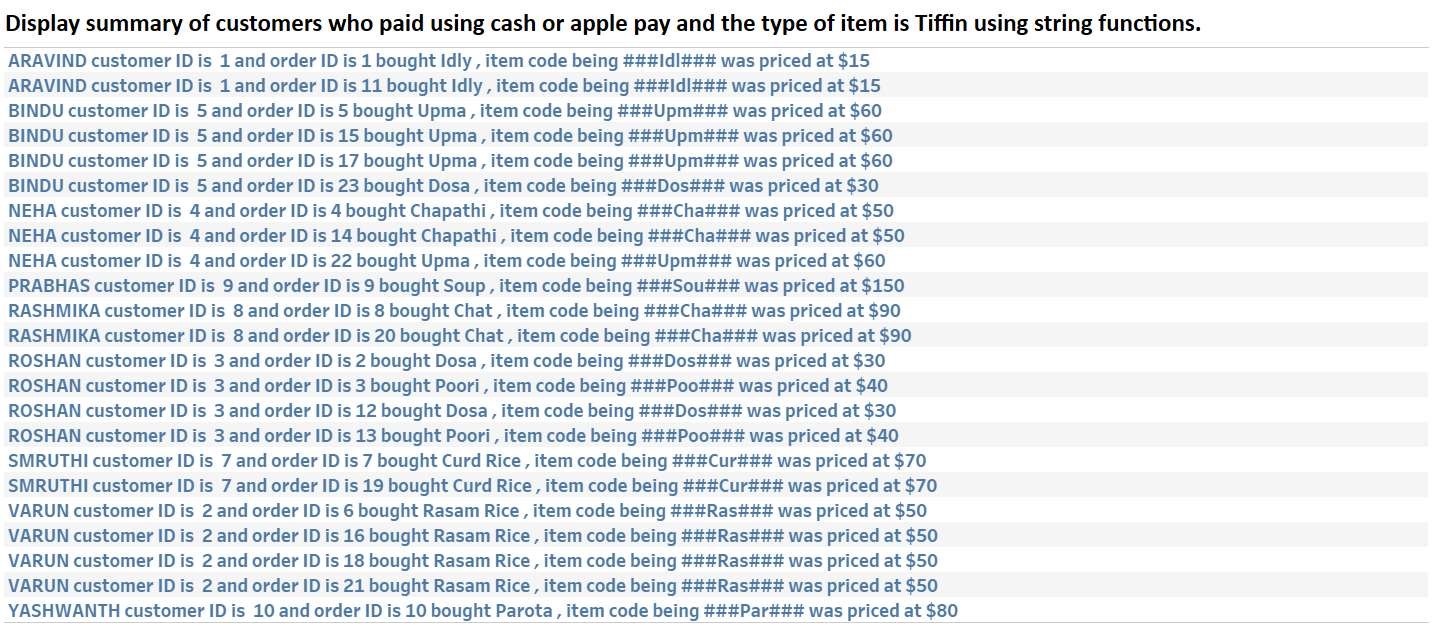
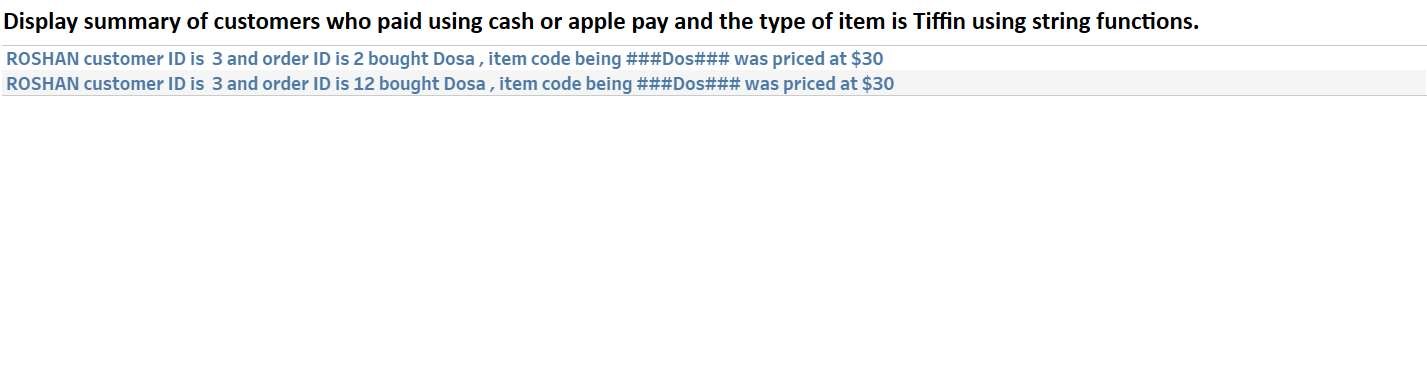
From customer c

JOIN orders o on c.customer\_id=o.customer\_id

JOIN menu m on o.item\_id=m.item\_id JOIN payment p on o.order\_id=p.order\_id where p.payment\_details="Cash" or p.payment\_details="Applepay" and m.item\_type in ("Tiffin") order by c.customer\_name;

OUTPUT:





Query 3: Display top 3 customer details who placed highest number of orders

Select c.customer\_name as Customer\_Name,

c.customer\_id as Customer\_ID,

o.order\_id as Order\_ID,

a.quantity as Quantity,

a.addon\_id as Addon\_ID,

Rank() Over(Order by a.quantity desc) as Rank\_based\_on\_quantity

From customer c

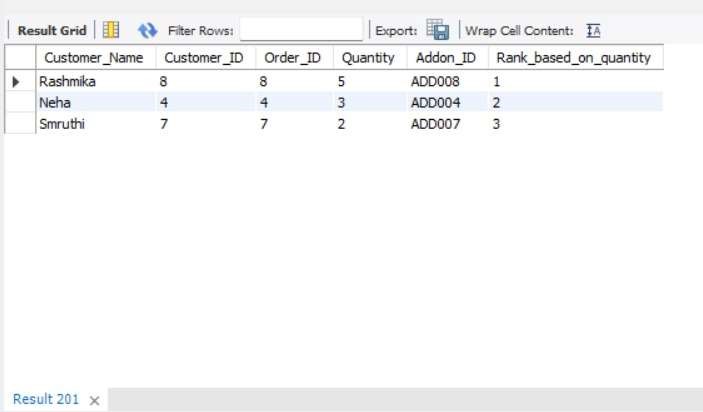
JOIN orders o on c.customer\_id=o.customer\_id

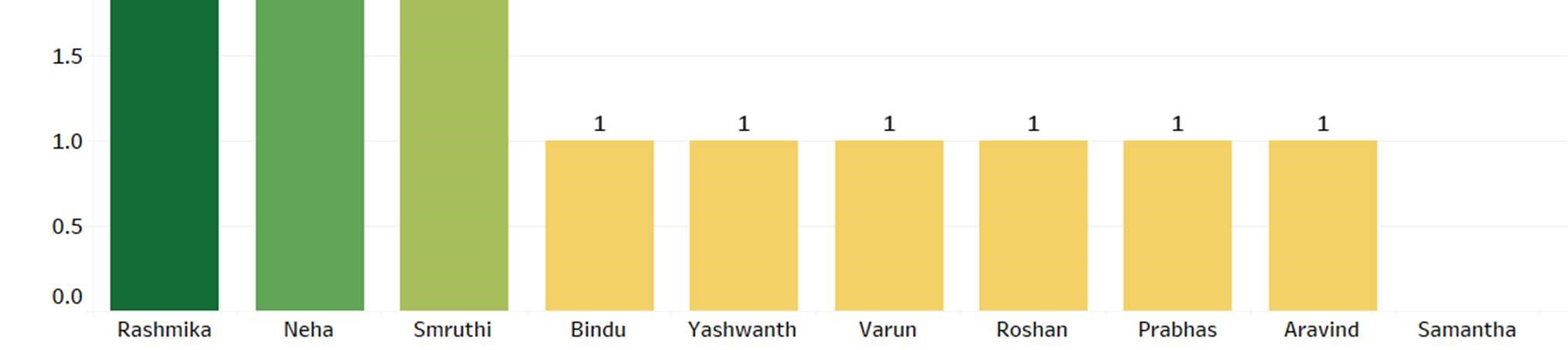
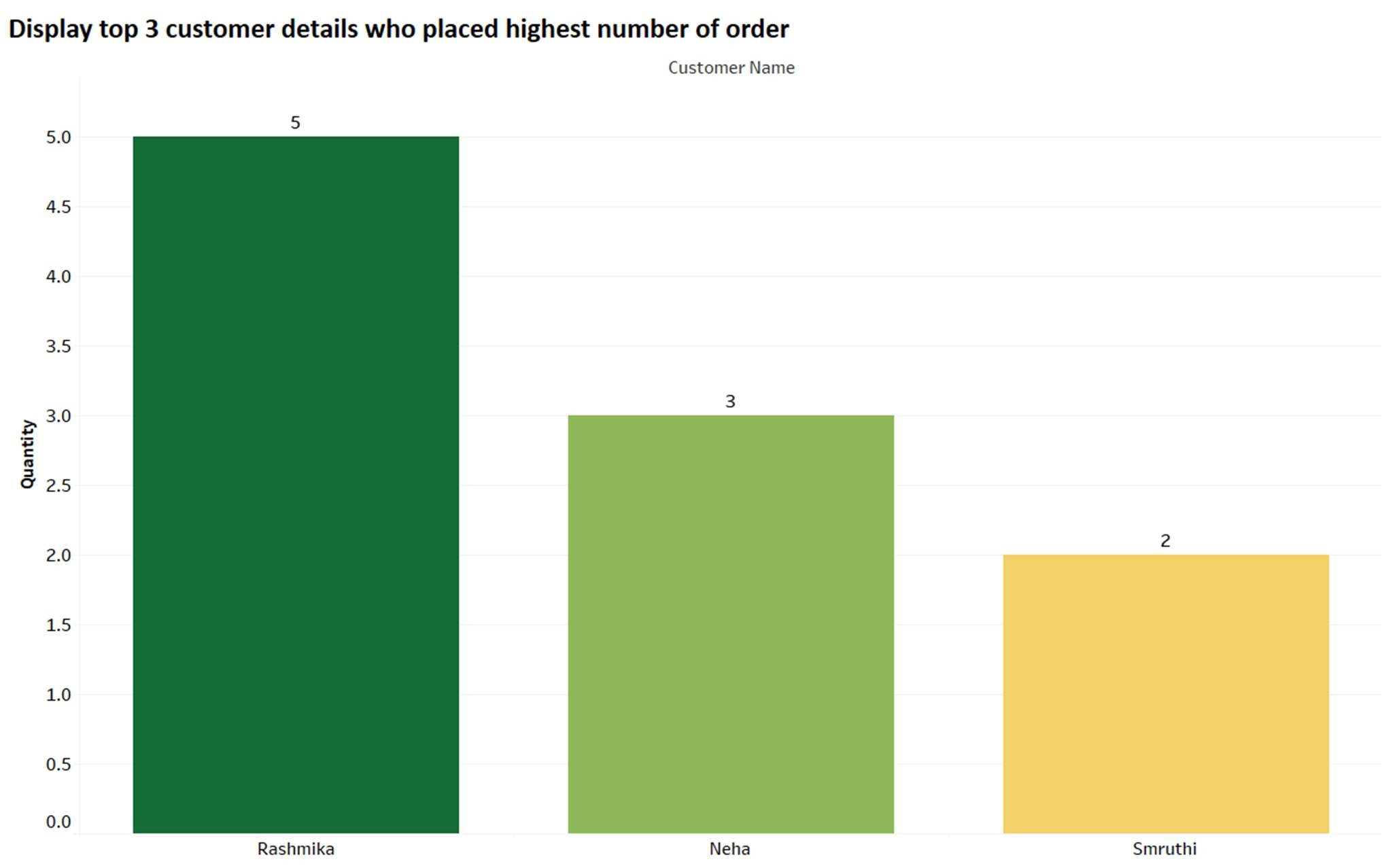
JOIN add\_ons a on a.addon\_id=o.addon\_id

GROUP BY c.customer\_name

Limit 3;

OUTPUT:





Query 4: Display the customer details who have paid highest amount for the orders on an average.

Select c.customer\_id as Customer\_ID,

c.customer\_name as Customer\_Name,

TRUNCATE(AVG(m.price),2) as Average\_Amount, COUNT(o.order\_id) as No\_of\_orders,

o.no\_of\_items as No\_of\_items from customer c

JOIN orders o on c.customer\_id = o.customer\_id JOIN menu m on o.item\_id=m.item\_id where c.customer\_id = (Select c.customer\_id from customer c

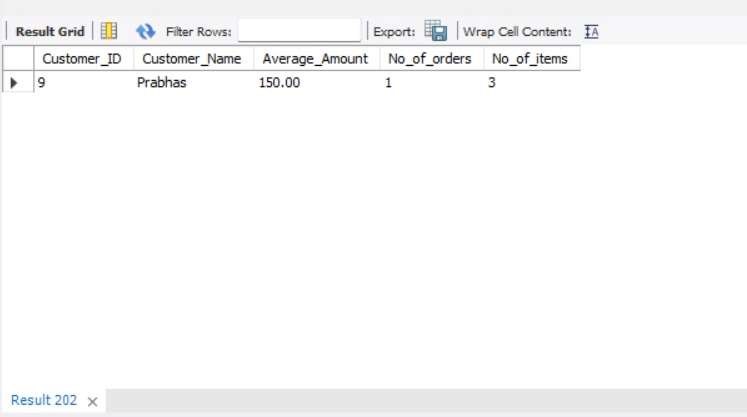
JOIN orders o on c.customer\_id = o.customer\_id

JOIN menu m on o.item\_id=m.item\_id GROUP BY c.customer\_id

order by AVG(m.price) desc

limit 1);

OUTPUT:





Query 5: Display purchase history and the number of employees they have interacted with.

Select c.customer\_id as Customer\_ID,

c.customer\_name as Customer\_Name,

SUM(m.price) as Total\_Amount,

COUNT (DISTINCT o.order\_id) as No\_of\_orders,

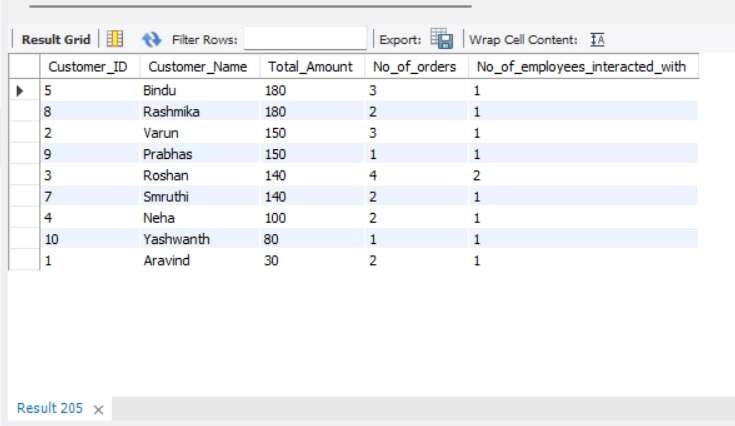
COUNT (DISTINCT p.employee\_id) as No\_of\_employees\_interacted\_with from customer c

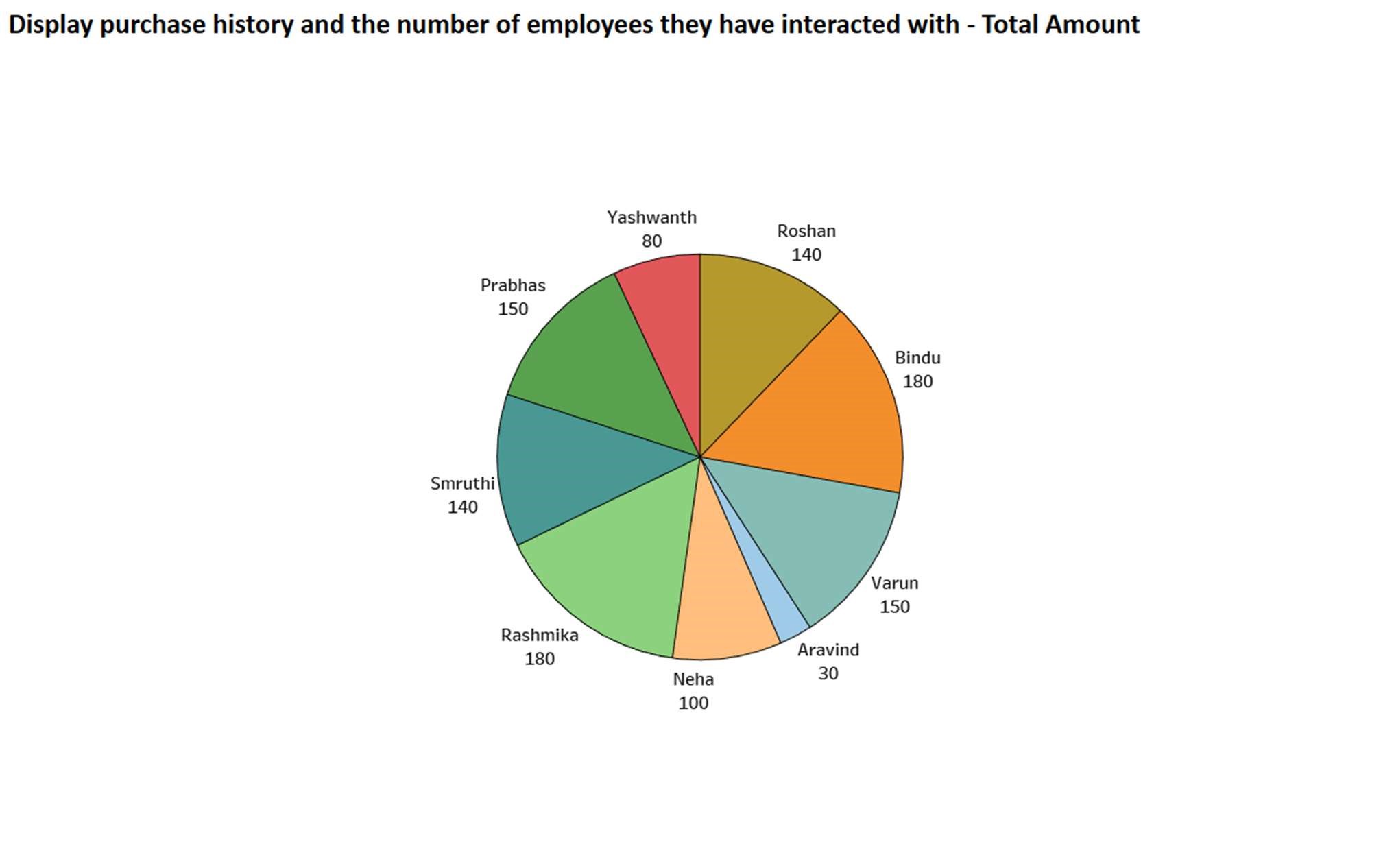
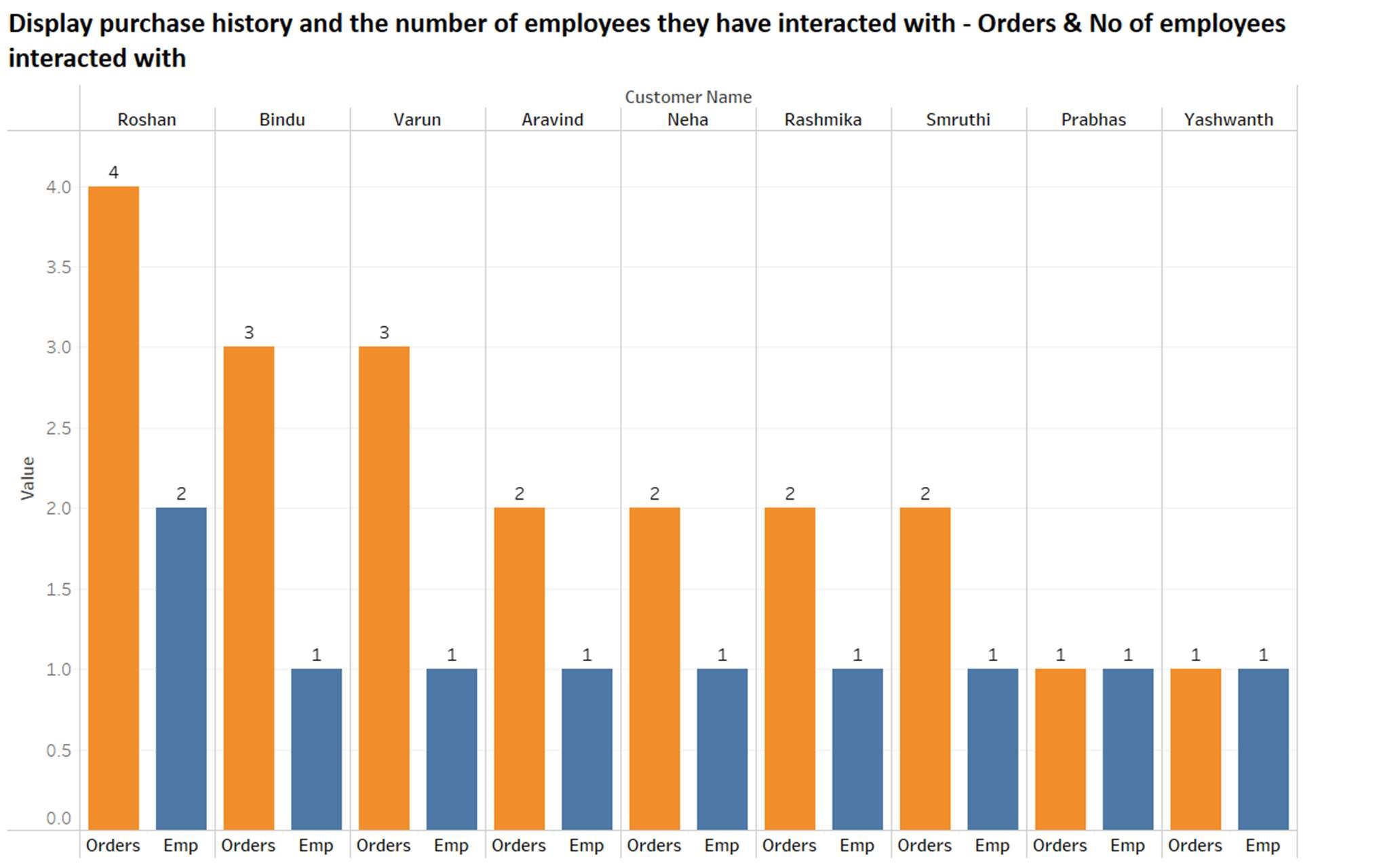
JOIN orders o on c.customer\_id = o.customer\_id

JOIN menu m on o.item\_id=m.item\_id

JOIN payment p on o.order\_id = p.order\_id GROUP BY c.customer\_id order by sum(m.price) desc;

OUTPUT:





Query 6: Display employee's employment details i.e, no of months of employment, no of transactions handled and rank them based on the no of transactions. Displaying the total amount handled along with the rank.

Select e.employee\_name as Employee\_Name,

e.employee\_id as Employee\_ID,

e.designation as Designation, employee\_DOJ as Date\_of\_JOINing,

(TIMESTAMPDIFF( month,employee\_DOJ,sysdate())) as No\_of\_months\_of\_employement, COUNT(p.payment\_id) as No\_of\_transactions\_handled ,

dense\_rank() over (partition by e.designation order by count(p.payment\_id) desc ) as Rank\_based\_on\_no\_of\_payments\_handled,

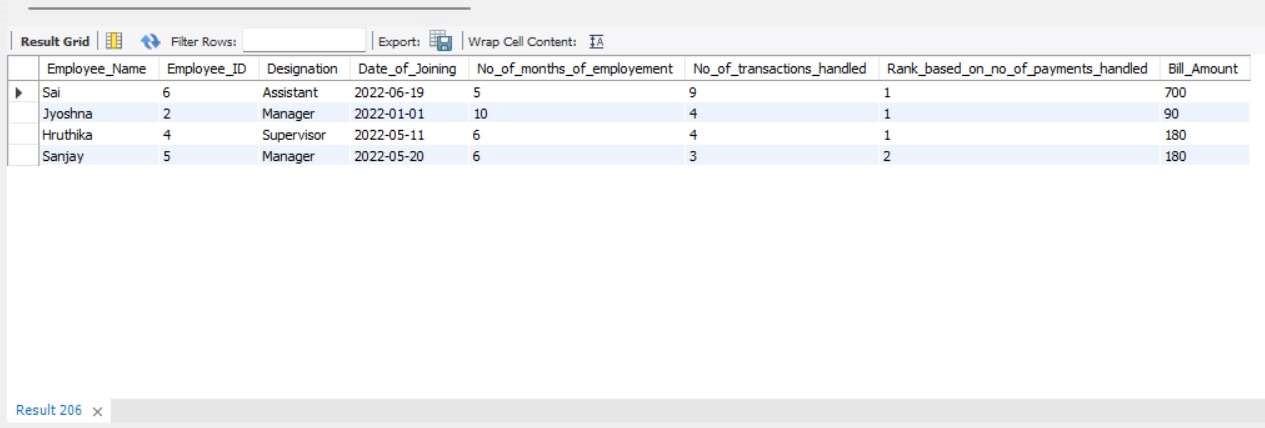
SUM(m.price) as Bill\_Amount

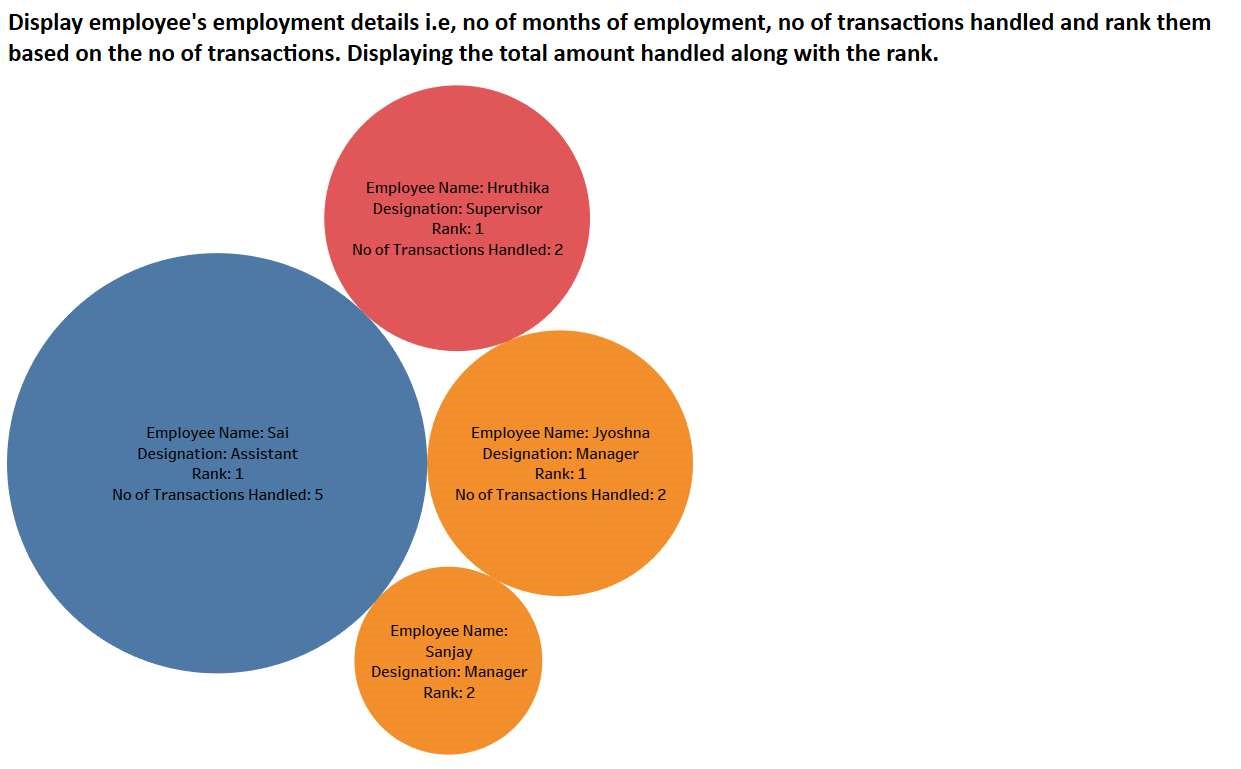
FROM employee e

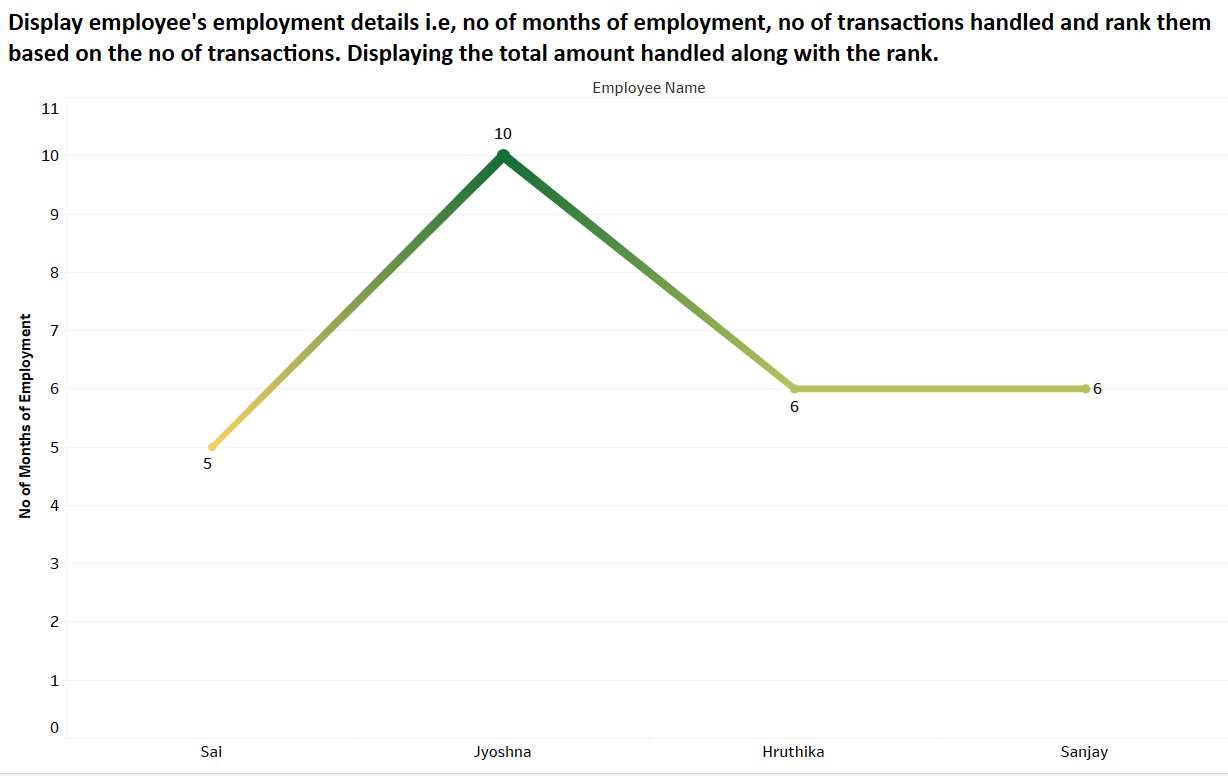
JOIN orders o on e.employee\_id = o.employee\_id

JOIN payment p on o.order\_id = p.order\_id

JOIN menu m on m.item\_id = o.item\_id GROUP BY e.employee\_name order by Rank\_based\_on\_no\_of\_payments\_handled; OUTPUT:







Query 7: Displaying all the orders which don't have an order date and by using Coalesce function replacing null values with 'Not Recorded' text. Displaying the maximum and minimum price of each order whose order date is not recorded.

SELECT c.customer\_name as Customer\_Name,

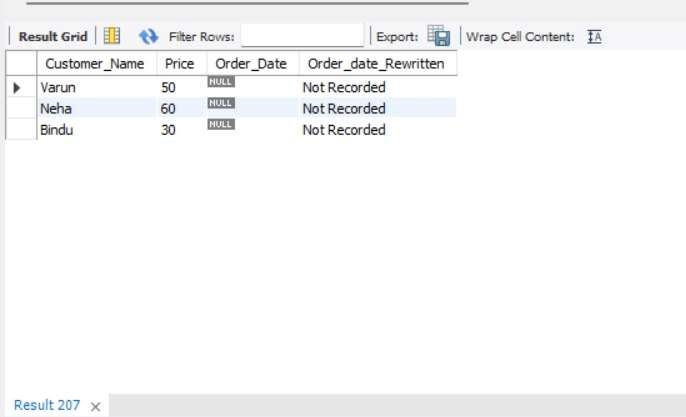
m.price as Price,

o.order\_date as Order\_Date,

COALESCE(o.order\_date,"Not Recorded") as Order\_date\_Rewritten

FROM customer c

Right OUTER JOIN orders o on c.customer\_id=o.customer\_id JOIN menu m on m.item\_id=o.item\_id where o.order\_date is NULL; OUTPUT:



SELECT c.customer\_name as Customer\_Name,

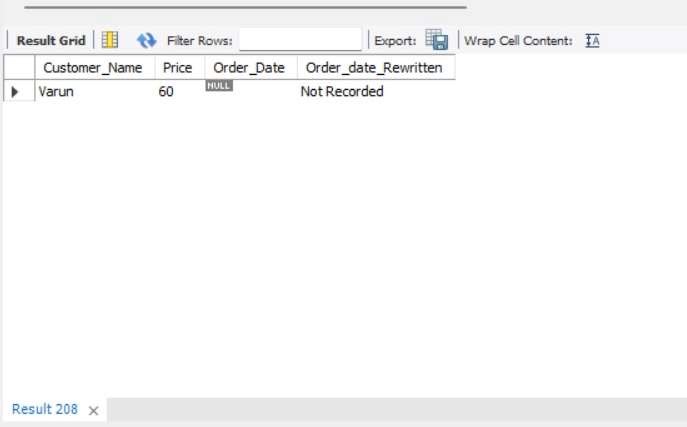
max(m.price) as Price,

o.order\_date as Order\_Date,

COALESCE (o.order\_date,"Not Recorded") as Order\_date\_Rewritten

FROM customer c

Right OUTER JOIN orders o on c.customer\_id=o.customer\_id JOIN menu m on m.item\_id=o.item\_id where o.order\_date is NULL; OUTPUT:



SELECT c.customer\_name as Customer\_Name,

MIN(m.price) as Price,

o.order\_date as Order\_Date,

COALESCE (o.order\_date,"Not Recorded") as Order\_date\_Rewritten

FROM customer c

Right OUTER JOIN orders o on c.customer\_id=o.customer\_id JOIN menu m on m.item\_id=o.item\_id where o.order\_date is NULL; OUTPUT:

