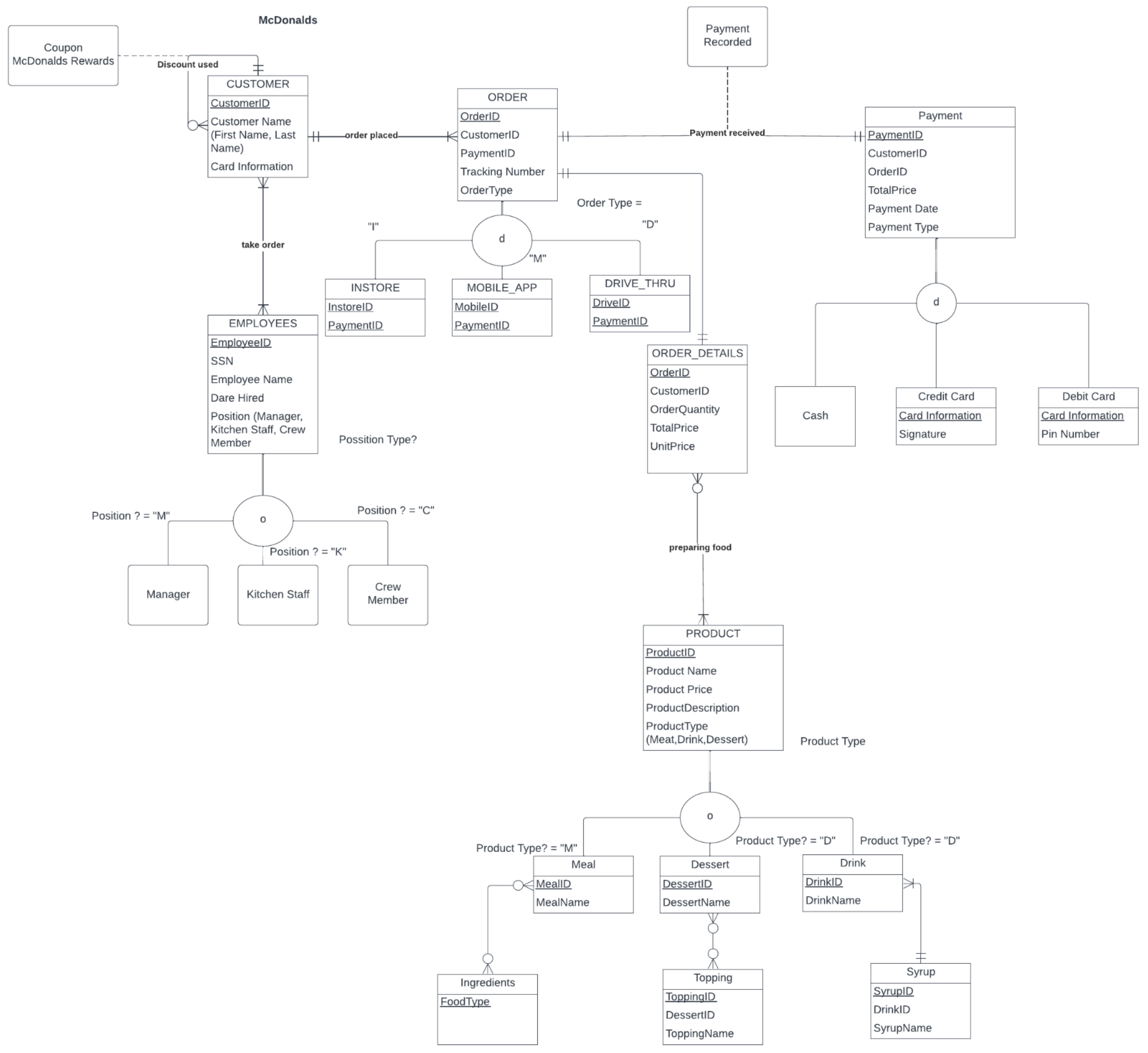
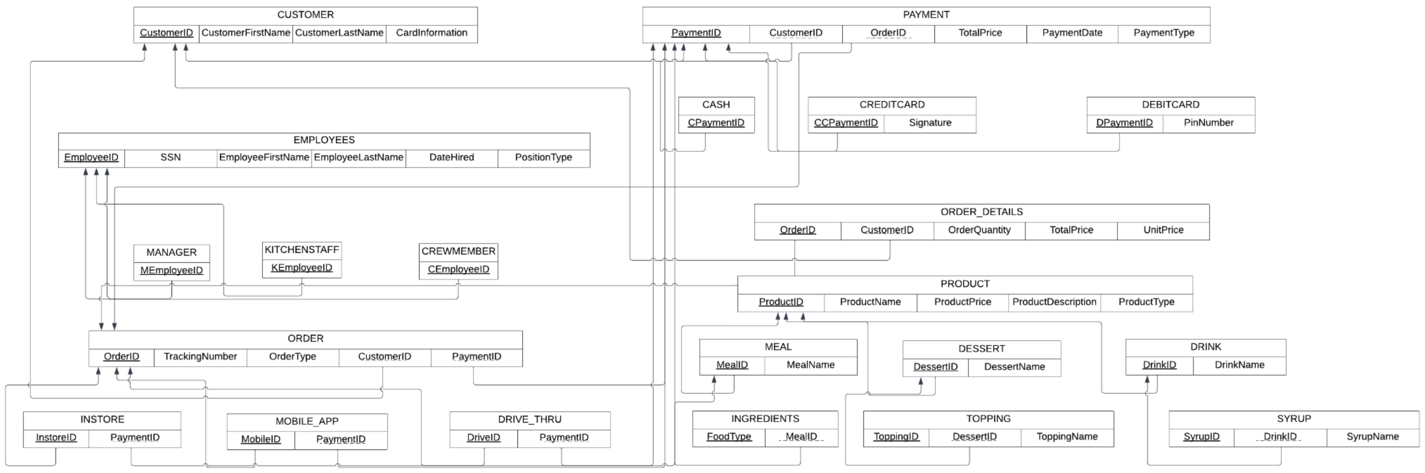
User Requirements:

1. McDonalds provides fairly fresh food daily to customers. In the US alone, McDonalds has more than 13,000 restaurants around the country. However, each location has individual operations run by their employees.
2. Employees who work at the restaurant, McDonald's, are separated into Managers, Crew Members, and Kitchen staff. Each employee can be identified by an Employee ID, Social Security Number, Employee Name (First and Last Name), Date hired and Position(s). Each employee can either be a manager, employee, or a shift lead. Employees can switch roles or exist as both a manager, kitchen staff, or a crewmember. However, there can only be one manager at a time.
3. Payment has Identifier Payment ID and has attributes Customer ID, Order ID, Total Price, Payment Date and Payment Type (Cash, Credit Card, Debit Card). Payments are made by one and only one customer and one customer can make one or more payments.
4. Customer has the identifier Customer ID, Customer Name (First Name and Last Name) and Card Information. One Customer can purchase many orders, but one and only one order can only be with one Customer.
5. Every order contains products. A product has the identifier Product ID, and has attributes Product Name, Product Price, Product Description. One order contains at least one or many products. One product can be ordered by none or many.
6. Order has Order ID as an identifier, OrderDate, Order Quantity, Total Price, and Tracking Number. Each customer can have one or many orders and one Order can only belong to one Customer.
7. Each order must include order detail, it has the identifier Order ID and attributes Customer ID, Order Quantity, Total Price and Unit Price.

Conceptual Data Modeling (E-R Diagram):

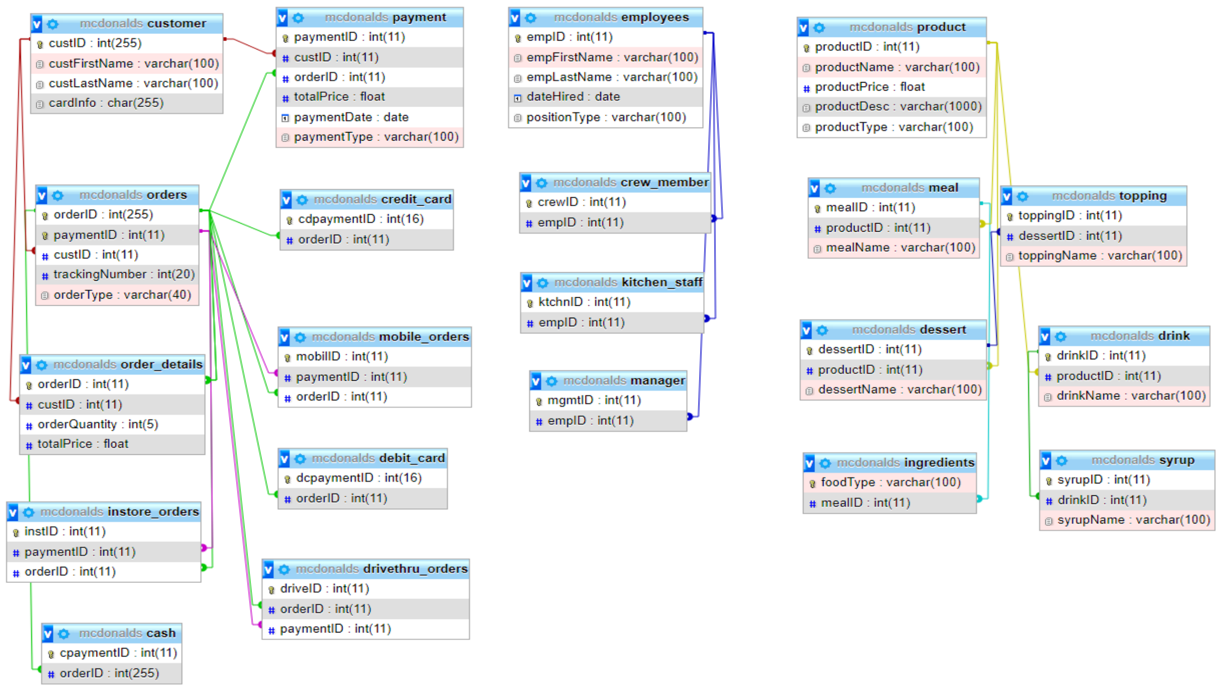


Logical Database Design (Relational Table):


CUSTOMER
CustomerID | CustomerFirstName | CustomerLastName | CardInformation
PAYMENT
PaymentID | CustomerID | OrderID | TotalPrice | PaymentDate | PaymentType
CASH
CPaymentID
CREDITCARD
CCPaymentID | Signature
DEBITCARD
DPaymentID | PinNumber
EMPLOYEES
EmployeeID | SSN | EmployeeFirstName | EmployeeLastName | DateHired | PositionType
MANAGER
MEmployeeID
KITCHENSTAFF
KEmployeeID
CREWMEMBER
CEmployeeID
ORDER_DETAILS
OrderID | CustomerID | OrderQuantity | TotalPrice | UnitPrice
PRODUCT
ProductID | ProductName | ProductPrice | ProductDescription | ProductType
ORDER
OrderID | TrackingNumber | OrderType | CustomerID | PaymentID
MEAL
MealID | MealName
DESSERT
DessertID | DessertName
DRINK
DrinkID | DrinkName
INSTORE
InstoreID | PaymentID
MOBILE_APP
MobileID | PaymentID
DRIVE_THRU
DriveID | PaymentID
INGREDIENTS
FoodType | MealID
TOPPING
ToppingID | DessertID | ToppingName
SYRUP
SyrupID | DrinkID | SyrupName

Implementation in MySQL:

McDonald’s Designer



McDonald's Database

Query 1. Adding Table

Our Data has 21 tables in total range from employees, orders, payment, and product, ect. All the tables indicate what Mcdonald's need to run in each individual store.

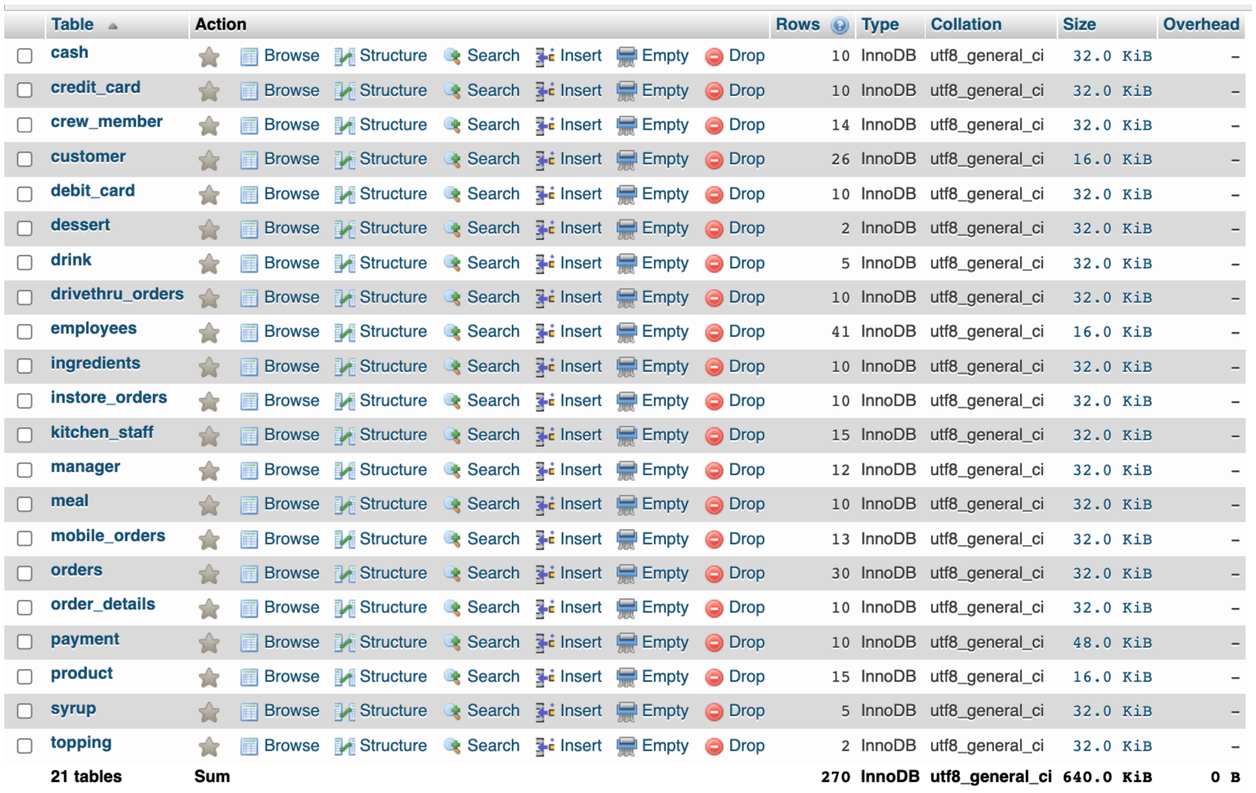
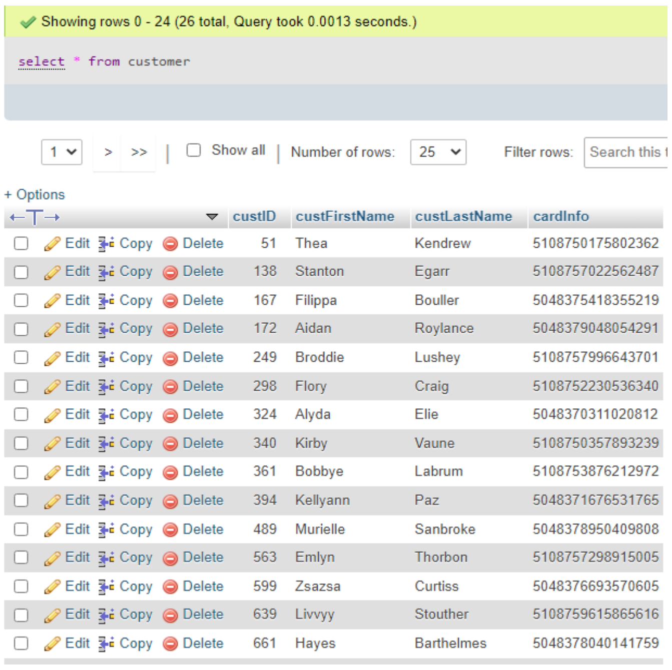


Table	Action							Rows	Type	Collation	Size	Overhead
cash		Browse	Structure	Search	Insert	Empty	Drop	10	InnoDB	utf8_general_ci	32.0 KiB	–
credit_card		Browse	Structure	Search	Insert	Empty	Drop	10	InnoDB	utf8_general_ci	32.0 KiB	–
crew_member		Browse	Structure	Search	Insert	Empty	Drop	14	InnoDB	utf8_general_ci	32.0 KiB	–
customer		Browse	Structure	Search	Insert	Empty	Drop	26	InnoDB	utf8_general_ci	16.0 KiB	–
debit_card		Browse	Structure	Search	Insert	Empty	Drop	10	InnoDB	utf8_general_ci	32.0 KiB	–
dessert		Browse	Structure	Search	Insert	Empty	Drop	2	InnoDB	utf8_general_ci	32.0 KiB	–
drink		Browse	Structure	Search	Insert	Empty	Drop	5	InnoDB	utf8_general_ci	32.0 KiB	–
drivethru_orders		Browse	Structure	Search	Insert	Empty	Drop	10	InnoDB	utf8_general_ci	32.0 KiB	–
employees		Browse	Structure	Search	Insert	Empty	Drop	41	InnoDB	utf8_general_ci	16.0 KiB	–
ingredients		Browse	Structure	Search	Insert	Empty	Drop	10	InnoDB	utf8_general_ci	32.0 KiB	–
instore_orders		Browse	Structure	Search	Insert	Empty	Drop	10	InnoDB	utf8_general_ci	32.0 KiB	–
kitchen_staff		Browse	Structure	Search	Insert	Empty	Drop	15	InnoDB	utf8_general_ci	32.0 KiB	–
manager		Browse	Structure	Search	Insert	Empty	Drop	12	InnoDB	utf8_general_ci	32.0 KiB	–
meal		Browse	Structure	Search	Insert	Empty	Drop	10	InnoDB	utf8_general_ci	32.0 KiB	–
mobile_orders		Browse	Structure	Search	Insert	Empty	Drop	13	InnoDB	utf8_general_ci	32.0 KiB	–
orders		Browse	Structure	Search	Insert	Empty	Drop	30	InnoDB	utf8_general_ci	32.0 KiB	–
order_details		Browse	Structure	Search	Insert	Empty	Drop	10	InnoDB	utf8_general_ci	32.0 KiB	–
payment		Browse	Structure	Search	Insert	Empty	Drop	10	InnoDB	utf8_general_ci	48.0 KiB	–
product		Browse	Structure	Search	Insert	Empty	Drop	15	InnoDB	utf8_general_ci	16.0 KiB	–
syrup		Browse	Structure	Search	Insert	Empty	Drop	5	InnoDB	utf8_general_ci	32.0 KiB	–
topping		Browse	Structure	Search	Insert	Empty	Drop	2	InnoDB	utf8_general_ci	32.0 KiB	–
21 tables	Sum							270	InnoDB	utf8_general_ci	640.0 KiB	0 B

Query 2.

`SELECT * FROM customer`

List of customer information from Customer data. For this query, we are looking for all the information that customer made an order. In each customers have their own unique CustID, custFirstName, CustLastName, and cardInfo.



Showing rows 0 - 24 (26 total, Query took 0.0013 seconds.)

```
select * from customer
```

1 > >> | Show all | Number of rows: 25 Filter rows: Search this t

+ Options

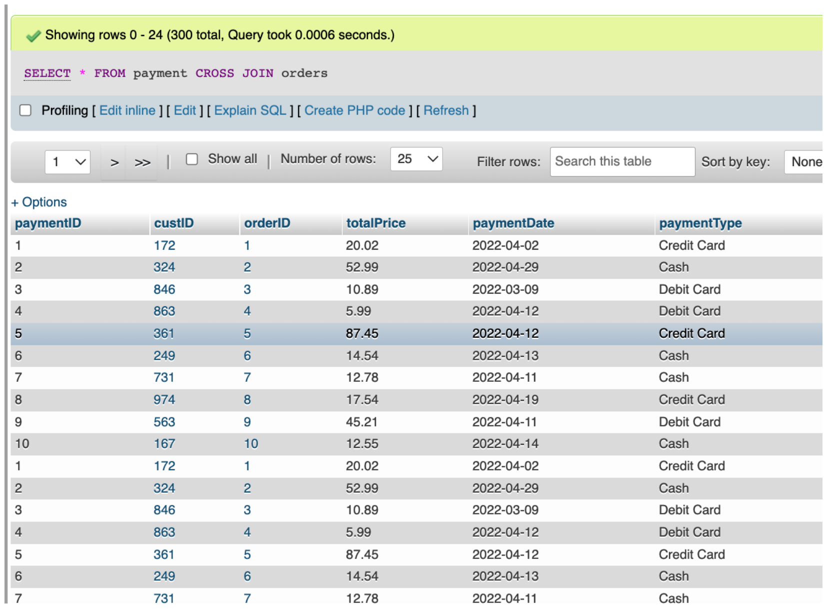
	custID	custFirstName	custLastName	cardInfo
Edit Copy Delete	51	Thea	Kendrew	5108750175802362
Edit Copy Delete	138	Stanton	Egarr	5108757022562487
Edit Copy Delete	167	Filippa	Bouller	5048375418355219
Edit Copy Delete	172	Aidan	Roylance	5048379048054291
Edit Copy Delete	249	Broddie	Lushey	5108757996643701
Edit Copy Delete	298	Flory	Craig	5108752230536340
Edit Copy Delete	324	Alyda	Elie	5048370311020812
Edit Copy Delete	340	Kirby	Vaune	5108750357893239
Edit Copy Delete	361	Bobbye	Labrum	5108753876212972
Edit Copy Delete	394	Kellyann	Paz	5048371676531765
Edit Copy Delete	489	Murielle	Sanbroke	5048378950409808
Edit Copy Delete	563	Emlyn	Thorbon	5108757298915005
Edit Copy Delete	599	Zsazsa	Curtiss	5048376693570605
Edit Copy Delete	639	Livvyy	Stouther	5108759615865616
Edit Copy Delete	661	Hayes	Barthelmes	5048378040141759

Query 3 (Join 1)

CROSS JOIN

To get both information from both payment and orders table because CROSS JOIN will eliminated the duplicate data.

```
SELECT * FROM payment CROSS JOIN orders
```



Showing rows 0 - 24 (300 total, Query took 0.0006 seconds.)

```
SELECT * FROM payment CROSS JOIN orders
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
1 > >> Show all Number of rows: 25 Filter rows: Search this table Sort by key: None
+ Options

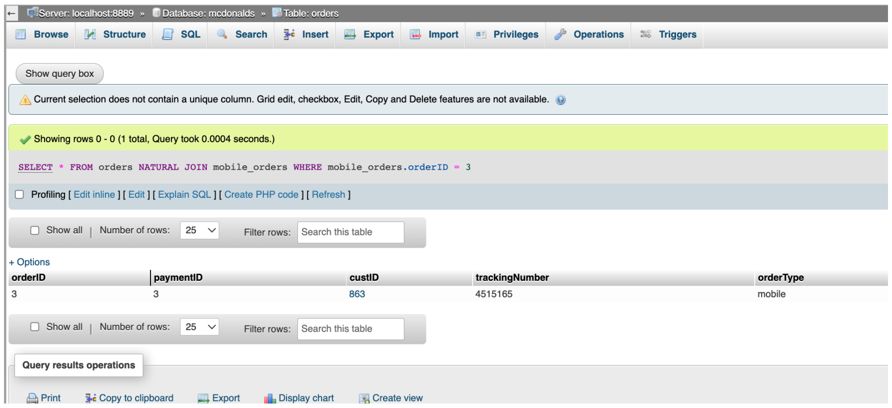
paymentID	custID	orderID	totalPrice	paymentDate	paymentType
1	172	1	20.02	2022-04-02	Credit Card
2	324	2	52.99	2022-04-29	Cash
3	846	3	10.89	2022-03-09	Debit Card
4	863	4	5.99	2022-04-12	Debit Card
5	361	5	87.45	2022-04-12	Credit Card
6	249	6	14.54	2022-04-13	Cash
7	731	7	12.78	2022-04-11	Cash
8	974	8	17.54	2022-04-19	Credit Card
9	563	9	45.21	2022-04-11	Debit Card
10	167	10	12.55	2022-04-14	Cash
1	172	1	20.02	2022-04-02	Credit Card
2	324	2	52.99	2022-04-29	Cash
3	846	3	10.89	2022-03-09	Debit Card
4	863	4	5.99	2022-04-12	Debit Card
5	361	5	87.45	2022-04-12	Credit Card
6	249	6	14.54	2022-04-13	Cash
7	731	7	12.78	2022-04-11	Cash

Query 4 (Join 2)

NATURAL JOIN

In this query, we are looking for the information of mobile_orders.orderID = 3. With NATURAL JOIN from orders and mobile_orders, the query gives the information of the trackingnumber, paymentID, and orderType.

```
SELECT * FROM orders NATURAL JOIN mobile_orders
WHERE mobile_orders.orderID = 3
```



Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 0 (1 total, Query took 0.0004 seconds.)

```
SELECT * FROM orders NATURAL JOIN mobile_orders WHERE mobile_orders.orderID = 3
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

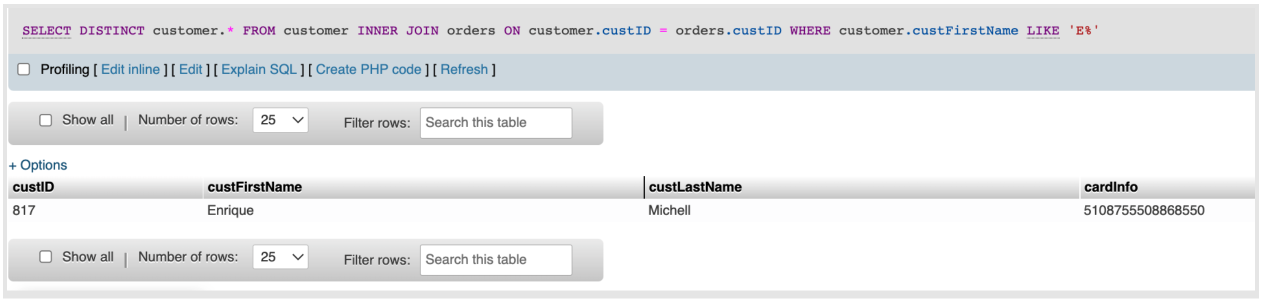
orderID	paymentID	custID	trackingNumber	orderType
3	3	863	4515165	mobile

Query 5 (Join 3)

DISTINCT and INNER JOIN

To look for the information from customer who has made an order whose name start from ‘E’.

```
SELECT DISTINCT customer.*
FROM customer INNER JOIN orders
ON customer.custID = orders.custID
WHERE customer.custFirstName LIKE 'E%';
```



```
SELECT DISTINCT customer.* FROM customer INNER JOIN orders ON customer.custID = orders.custID WHERE customer.custFirstName LIKE 'E%'
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

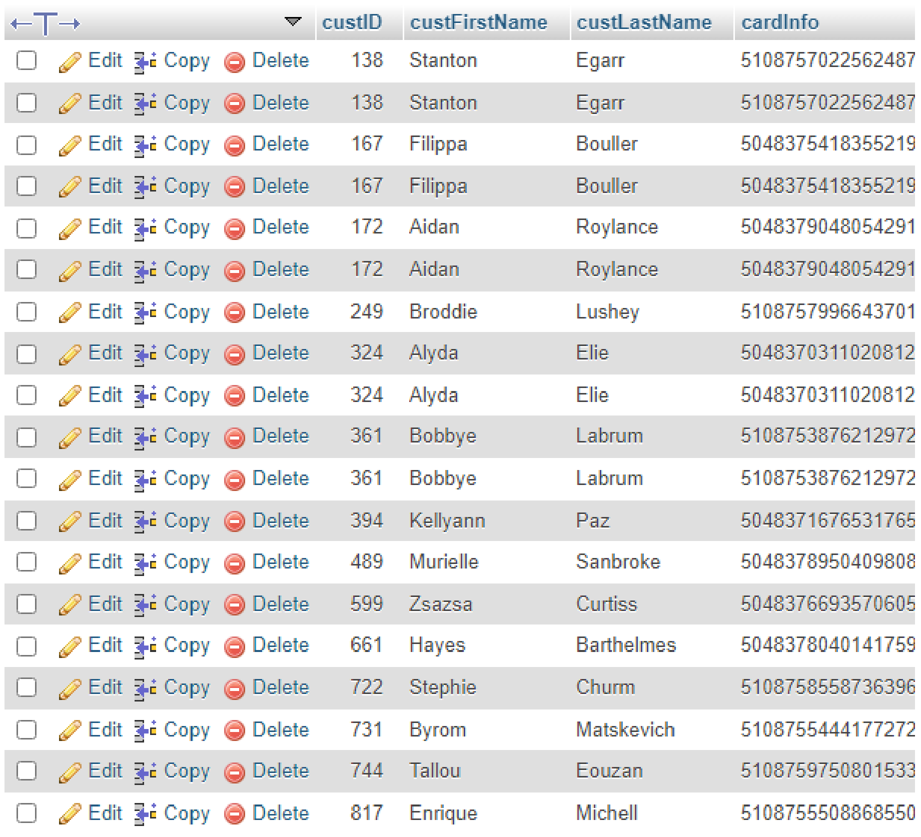
custID	custFirstName	custLastName	cardInfo
817	Enrique	Michell	5108755508868550

View 1

Customer Places Order

This view allows us to easily bring up data for customers that put in orders. That way if we are looking for a specific order that was placed, we wouldn’t have to wade through all the customers that haven’t placed an order, reducing search time by approximately half.

```
CREATE VIEW v_order_customer AS
(SELECT customer.* FROM customer, orders
WHERE customer.custID = orders.custID)
```



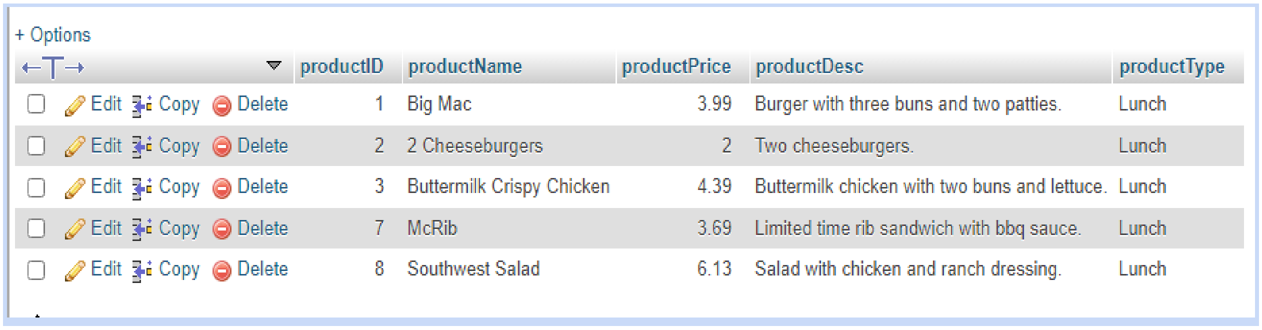
custID	custFirstName	custLastName	cardInfo
138	Stanton	Egarr	5108757022562487
138	Stanton	Egarr	5108757022562487
167	Filippa	Bouller	5048375418355219
167	Filippa	Bouller	5048375418355219
172	Aidan	Roylance	5048379048054291
172	Aidan	Roylance	5048379048054291
249	Broddie	Lushey	5108757996643701
324	Alyda	Elie	5048370311020812
324	Alyda	Elie	5048370311020812
361	Bobbye	Labrum	5108753876212972
361	Bobbye	Labrum	5108753876212972
394	Kellyann	Paz	5048371676531765
489	Murielle	Sanbroke	5048378950409808
599	Zsazsa	Curtiss	5048376693570605
661	Hayes	Barthelmes	5048378040141759
722	Stephie	Churm	5108758558736396
731	Byrom	Matskevich	5108755444177272
744	Tallou	Eouzan	5108759750801533
817	Enrique	Michell	5108755508868550

View 2

Lunch Product

This view brings up the products table but filters it out so the product type is only equal to lunch items, eliminating the dessert and breakfast items.

```
CREATE VIEW v_product_lunch AS (select product.* FROM product WHERE
product.productType = 'Lunch')
```



+ Options

	productID	productName	productPrice	productDesc	productType
Edit Copy Delete	1	Big Mac	3.99	Burger with three buns and two patties.	Lunch
Edit Copy Delete	2	2 Cheeseburgers	2	Two cheeseburgers.	Lunch
Edit Copy Delete	3	Buttermilk Crispy Chicken	4.39	Buttermilk chicken with two buns and lettuce.	Lunch
Edit Copy Delete	7	McRib	3.69	Limited time rib sandwich with bbq sauce.	Lunch
Edit Copy Delete	8	Southwest Salad	6.13	Salad with chicken and ranch dressing.	Lunch

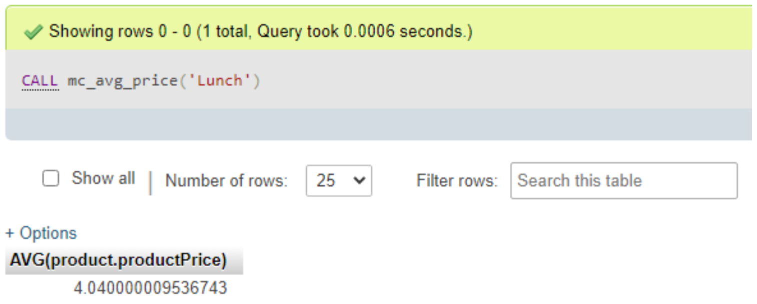
Stored Procedure 1

Average Price of Items based on input. Output is based on whatever menu item type the user inputs, gives average price of breakfast, lunch, or dessert items.

```
DELIMITER //
CREATE PROCEDURE mc_avg_price (IN productType_x Varchar(50))
BEGIN
SELECT AVG(product.productPrice)
FROM product
WHERE product.productType = productType_x;
END //
Delimiter ;
```

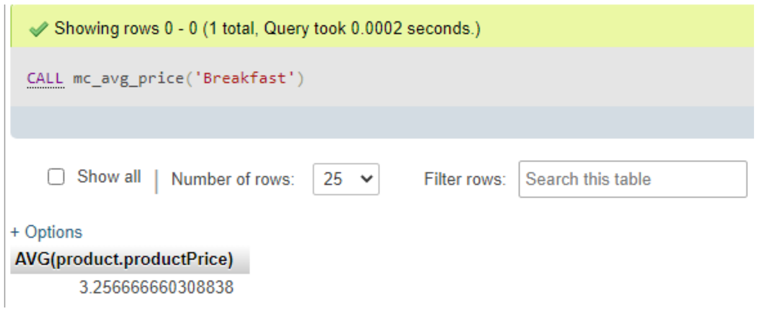
```
Invoke:
```

```
CALL mc_avg_price('Lunch')
```



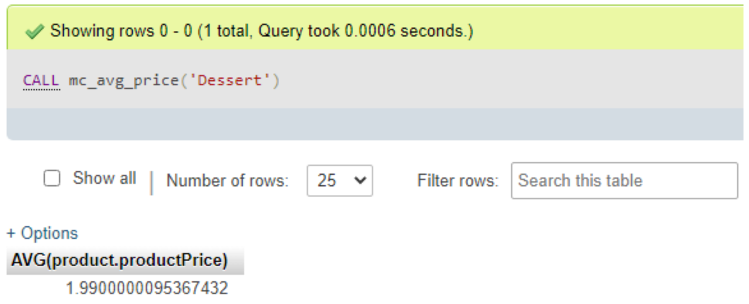
```
Showing rows 0 - 0 (1 total, Query took 0.0006 seconds.)
CALL mc_avg_price('Lunch')

AVG(product.productPrice)
        4.040000009536743
```



```
Showing rows 0 - 0 (1 total, Query took 0.0002 seconds.)
CALL mc_avg_price('Breakfast')

AVG(product.productPrice)
        3.256666660308838
```



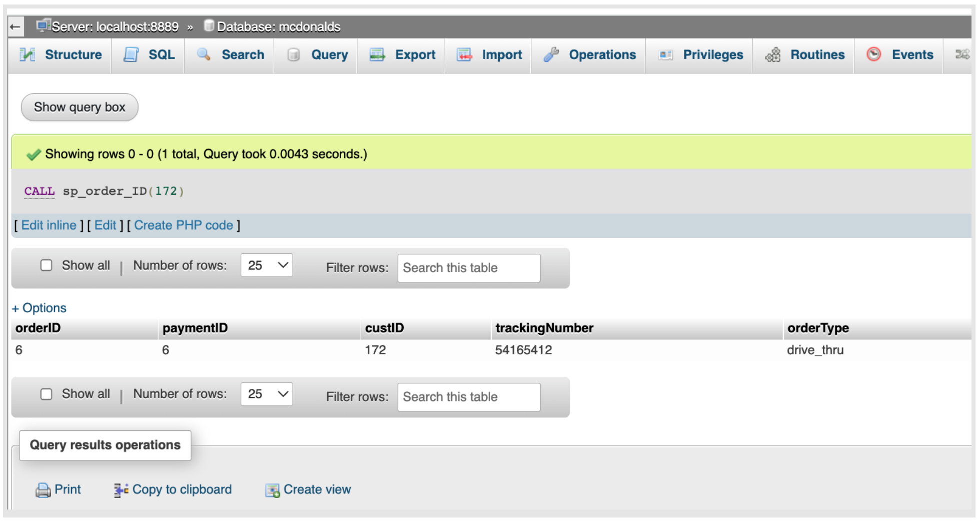
```
Showing rows 0 - 0 (1 total, Query took 0.0006 seconds.)
CALL mc_avg_price('Dessert')

AVG(product.productPrice)
        1.9900000095367432
```

Stored Procedure 2

Searching for order methods by entering order ID, allows us to see the order type of whatever the order id is inputed.

```
DELIMITER //
CREATE PROCEDURE sp_order_ID (IN x INT(50))
BEGIN
SELECT * FROM orders INNER JOIN drivethru_orders INNER JOIN mobile_orders INNER
JOIN instore_orders ON orders.OrderID = drivethru_orders.orderID or
orders.orderID = mobile_orders or orders.orderID = instore_orders and
orders.custID = x ;
END //
DELIMITER ;
Invoke
CALL sp_order_ID(172);
```


Server: localhost:8889 » Database: mcdonalds
Structure
SQL
Search
Query
Export
Import
Operations
Privileges
Routines
Events
Show query box

Showing rows 0 - 0 (1 total, Query took 0.0043 seconds.)

```
CALL sp_order_ID(172)
```

[Edit inline] [Edit] [Create PHP code]
Show all | Number of rows: 25 Filter rows: Search this table
+ Options

orderID	paymentID	custID	trackingNumber	orderType
6	6	172	54165412	drive_thru

Show all | Number of rows: 25 Filter rows: Search this table
Query results operations
Print
Copy to clipboard
Create view