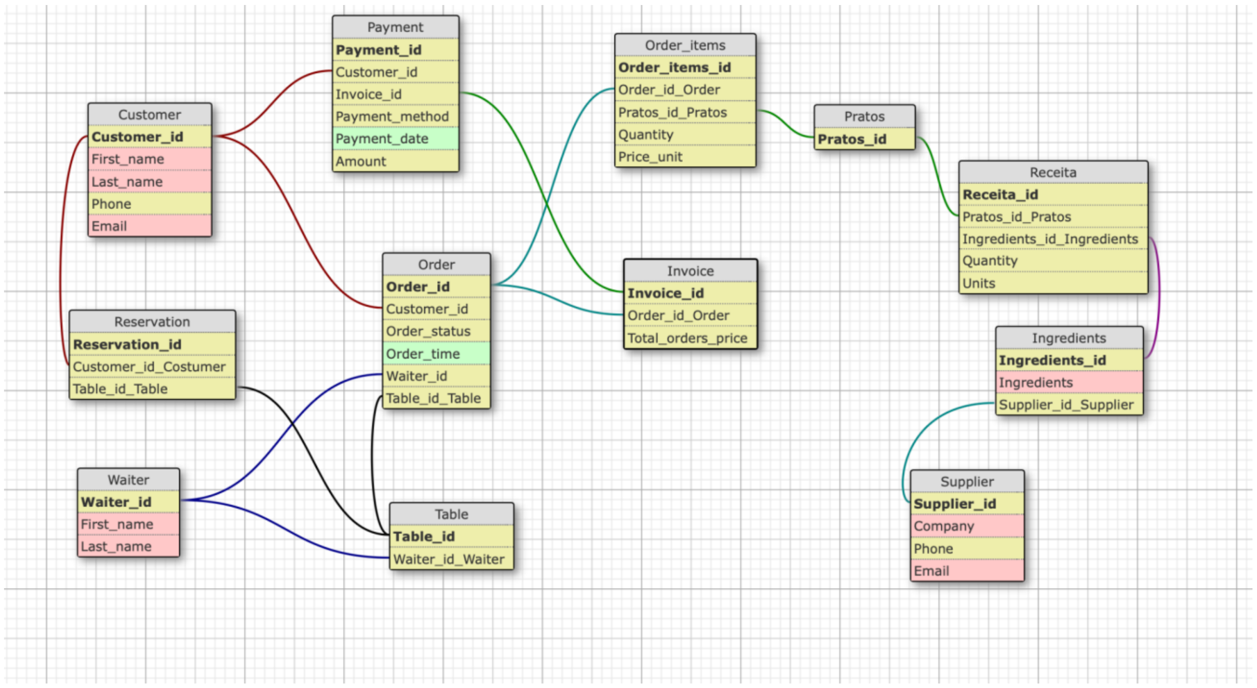
Database Implementation Report

* please find all the files included inside the RAR archive.
* we had issues with the \copy command when working together, so we switched to hard-coded insert instead.

1. Database Schema

The following schema was implemented based on the provided ERD.



2. Table Creation Scripts

Customer Table

```
create table Customer (
    Customer_id   SERIAL primary key,
    First_name    VARCHAR(50) not null,
    Last_name     VARCHAR(50) not null,
    Phone         VARCHAR(15), -- ig 15 is enough for a number?
    Email         VARCHAR(100)
);
```

-Reservation Table

```
create table Reservation (
    Reservation_id          SERIAL primary key,
    Customer_id_Customer    INT references Customer(Customer_id),
    Table_id_Table          INT references Table_rest(Table_id)
);
```

Table Table tableee

```
create table Table_rest (
    Table_id            SERIAL primary key,
    Waiter_id_Waiter    INT references Waiter(Waiter_id)
);
```

Waiter Table

```
create table Waiter (
    Waiter_id     SERIAL primary key,
    First_name    VARCHAR(50) not null,
    Last_name     VARCHAR(50) not null
);
```

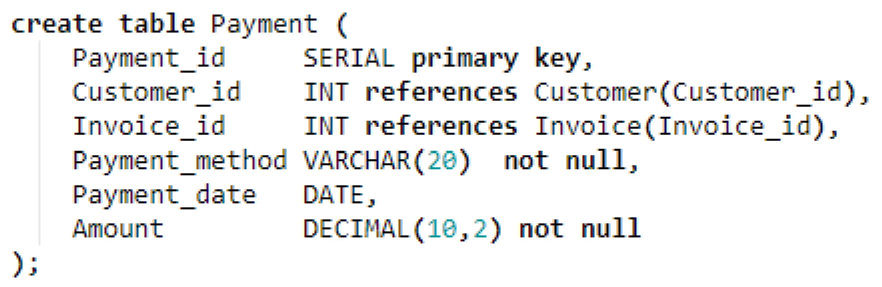
Oder Rest Table

```
create table Order_rest (
    Order_id          SERIAL primary key,
    Customer_id       INT references Customer(Customer_id),
    Reservation_id    INT references Reservation(Reservation_id),
    Order_status      VARCHAR(20)  default 'Pending',
    Order_time        TIMESTAMP    default current_timestamp
);
```

Invoice Table

```
create table Invoice (
    Invoice_id            SERIAL primary key,
    Order_id              INT unique references Order_rest(Order_id),
    Total_orders_price    DECIMAL(10,2) not null
);
```

Payment Table



```
create table Payment (
    Payment_id     SERIAL primary key,
    Customer_id    INT references Customer(Customer_id),
    Invoice_id     INT references Invoice(Invoice_id),
    Payment_method VARCHAR(20)  not null,
    Payment_date   DATE,
    Amount         DECIMAL(10,2) not null
);
```

3. database

We chose the data from Mackaroo and inserted all sample rows via INSERT. We could have used the csv file instead but that way it was kind of hard for us to work together. (see scripts below).

Total row count per table from Mackaroo (find CSV files attached as well):
Customer = 12, Waiter = 3, Table_rest = 6, Reservation = 6, Order_rest = 12, Invoice = 12, Payment = 12 (total = 63).

We first used used \copy for this but changed to insert after. (explanation at the bottom of Part C, in the “Issues We Had” part.)

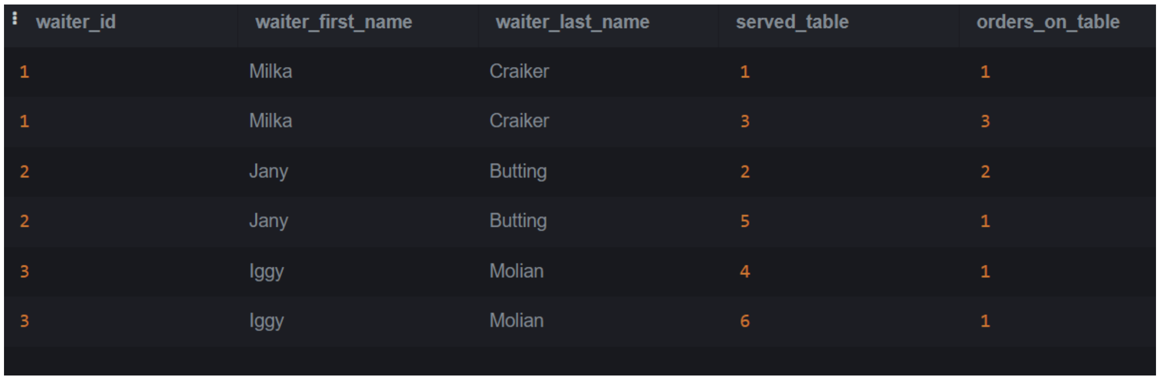
```sql
insert into Customer (First_name, Last_name, Phone, Email) values
  ('Arther',   'Gross',    '620-538-1719', 'agross0@tinyurl.com'),
  ('Seamus',   'Chin',     '855-908-9332', 'schin1@ucoz.com'),
  ('Brietta',  'Ebdin',    '368-113-2297', 'bebdin2@slate.com'),
  ('Pace',     'Yule',     '624-415-8707', 'pyule3@dmoz.org'),
  ('Natalya',  'Helm',     '620-600-2299', 'nhelm4@e-recht24.de'),
  ('Pearla',   'Winson',   '477-541-3729', 'pwinson5@guardian.co.uk'),
  ('Barrett',  'Stocking', '280-507-2632', 'bstocking6@soundcloud.com'),
  ('Corrina',  'Jeves',    '455-127-0495', 'cjeves7@people.com.cn'),
  ('Natal',    'Jeffers',  '658-394-4232', 'njeffers8@nba.com'),
  ('Elene',    'Doogue',   '938-137-0116', 'edoogue9@ftc.gov'),
  ('Rab',      'Brok',     '591-976-5217', 'rbroka@dell.com'),
  ('Lu',       'Albery',   '926-441-1608', 'lalberyb@about.com');

insert into Waiter (First_name, Last_name) values
  ('Milka', 'Craiker'),
  ('Jany',   'Butting'),
  ('Iggy',   'Molian');

insert into Table_rest (Waiter_id_Waiter) values
  (1), -- table 1 waiter 1
  (2), -- table 2 waiter 2
  (1), -- table 3 waiter 1
  (3), -- table 4 waiter 3
  (2), -- table 5 waiter 2
  (3); -- table 6 waiter 3

insert into Reservation (Customer_id_Customer, Table_id_Table) values
  (8,   3),    -- 1st reservation customer 8 table 3
  (9,   2),    -- 2nd reservation customer 9 table 2
  (3,   5),    -- 3rd reservation customer 3 table 5
  (7,   4),    -- 4th reservation customer 7 table 4
  (2,   1),    -- 5th reservation customer 2 table 1
  (4,   6);    -- 6th reservation customer 4 table 6

insert into Order_rest (Order_id, Customer_id, Reservation_id, Order_status, Order_time) values
  (1, 4, 6, 'delivered', '2024-06-12 00:20:18'),
  (2, 6, NULL, 'processed', '2025-05-06 19:34:16'),
  (3, 10, 1, 'delivered', '2025-04-25 15:00:29'),
  (4, 5, 1, 'delivered', '2024-10-31 20:57:56'),
  (5, 11, 4, 'processed', '2025-02-12 05:39:53'),
  (6, 3, NULL, 'processed', '2025-03-11 17:23:07'),
  (7, 7, 2, 'delivered', '2024-06-04 23:07:45'),
  (8, 12, 5, 'delivered', '2024-07-20 16:58:06'),
  (9, 5, 3, 'delivered', '2025-02-03 10:55:41'),
  (10, 11, 1, 'processed', '2025-02-13 22:06:51'),
  (11, 8, NULL, 'delivered', '2024-06-22 22:01:47'),
  (12, 10, 2, 'delivered', '2024-09-03 19:03:31');
```

```
insert into Invoice (Invoice_id, Order_id, Total_orders_price) values
  (1, 1, 2.59),
  (2, 2, 2.99),
  (3, 3, 8.99),
  (4, 4, 6.49),
  (5, 5, 3.79),
  (6, 6, 34.99),
  (7, 7, 3.99),
  (8, 8, 99.99),
  (9, 9, 299.99),
  (10, 10, 3.99),
  (11, 11, 34.99),
  (12, 12, 19.99);
```

4. Queries

1. Query the customer and their orders

```
select
  c.Customer_id,
  First_name,
  Last_name,
  Phone,
  Email
from Customer c
left join Order_rest o
  on c.Customer_id = o.Customer_id
order by c.Customer_id;
```

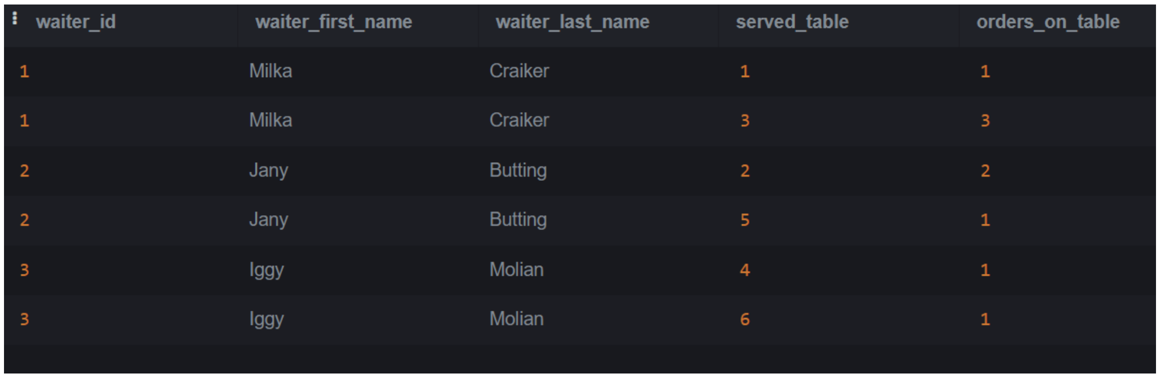


waiter_id	waiter_first_name	waiter_last_name	served_table	orders_on_table
1	Milka	Craiker	1	1
1	Milka	Craiker	3	3
2	Jany	Butting	2	2
2	Jany	Butting	5	1
3	Iggy	Molian	4	1
3	Iggy	Molian	6	1

This returns every customer and any orders they have placed. It can help the management see which customers are active and which have never ordered.

2. Count the number of orders placed by the waiters

```
select
  w.Waiter_id,
  w.First_name as Waiter_first_name, -- so that it doesn't conflict with customers' First_name
  w.Last_name as Waiter_last_name, -- so that it doesn't conflict with the customers' Last_name
  tr.Table_id as Served_Table,
  COUNT(o.Order_id) as Orders_on_Table
from Waiter w
left join Table_rest tr
  on w.Waiter_id = tr.Waiter_id_Waiter
left join Reservation r
  on tr.Table_id = r.Table_id_Table
left join Order_rest o
  on r.Reservation_id = o.Reservation_id
group by
  w.Waiter_id,
  w.First_name,
  w.Last_name,
  tr.Table_id
order by w.Waiter_id;
```

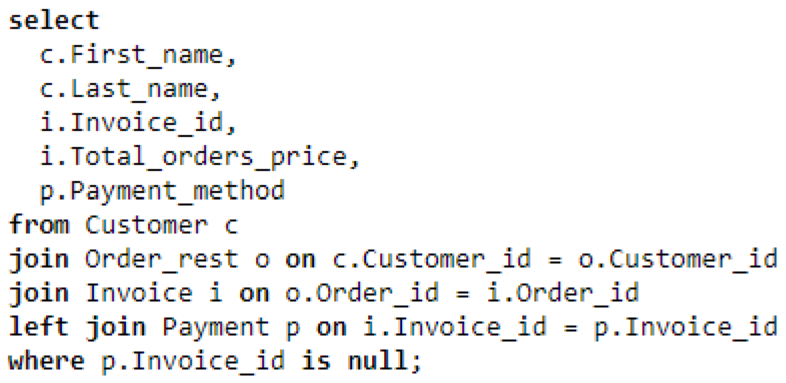


waiter_id	waiter_first_name	waiter_last_name	served_table	orders_on_table
1	Milka	Craiker	1	1
1	Milka	Craiker	3	3
2	Jany	Butting	2	2
2	Jany	Butting	5	1
3	Iggy	Molian	4	1
3	Iggy	Molian	6	1

This query groups by each waiter and table, than counts how many orders that are at that table under that waiters service.

It's useful to see which waiters are handling the busiest tables and it helps distribute the orders evenly among other waiters.

3. Search for unpaid orders



```
select
  c.First_name,
  c.Last_name,
  i.Invoice_id,
  i.Total_orders_price,
  p.Payment_method
from Customer c
join Order_rest o on c.Customer_id = o.Customer_id
join Invoice i on o.Order_id = i.Order_id
left join Payment p on i.Invoice_id = p.Invoice_id
where p.Invoice_id is null;
```

(empty table because there are no unpaid bills.)

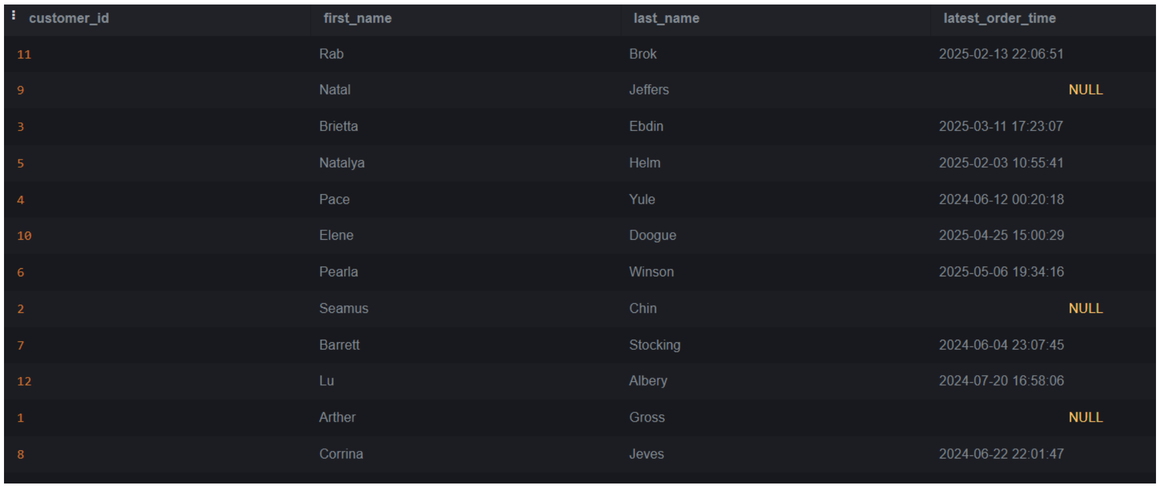
note: the sql query in this image is old, we fixed it. It would only find invoices where the invoice's Order_id value exactly equald to the customer's Customer_id. So it only worked if Order_id = Customer_id. now it works. -- i actually fixed it so np. Mehmet Emin Cicek

This query returns every invoice whose corresponding Payment_date is still NULL, along with the customer's name and the invoice total.

It allows the finance team to know which customers haven't payed the bill yet.

4. Query the latest order time

```
select
  c.Customer_id,
  c.First_name,
  c.Last_name,
  MAX(o.Order_time) as Latest_order_time
from Customer c
left join Order_rest o
  on c.Customer_id = o.Customer_id
group by
  c.Customer_id,
  c.First_name,
  c.Last_name;
```

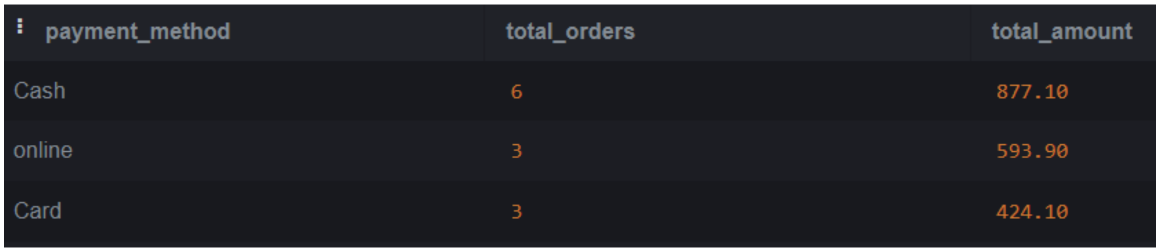


customer_id	first_name	last_name	latest_order_time
11	Rab	Brok	2025-02-13 22:06:51
9	Natal	Jeffers	NULL
3	Brietta	Ebdin	2025-03-11 17:23:07
5	Natalya	Helm	2025-02-03 10:55:41
4	Pace	Yule	2024-06-12 00:20:18
10	Elene	Doogue	2025-04-25 15:00:29
6	Pearla	Winson	2025-05-06 19:34:16
2	Seamus	Chin	NULL
7	Barrett	Stocking	2024-06-04 23:07:45
12	Lu	Albery	2024-07-20 16:58:06
1	Arther	Gross	NULL
8	Corrina	Jeves	2024-06-22 22:01:47

This query calculates the timestamp of their latest order for each customer (or NULL if they've never ordered).
It helps the restaurant understand how often they come to eat. For example, customers who haven't ordered in a while can be called and advertised.

5. Calculate the total amount and the number of orders for all payment methods

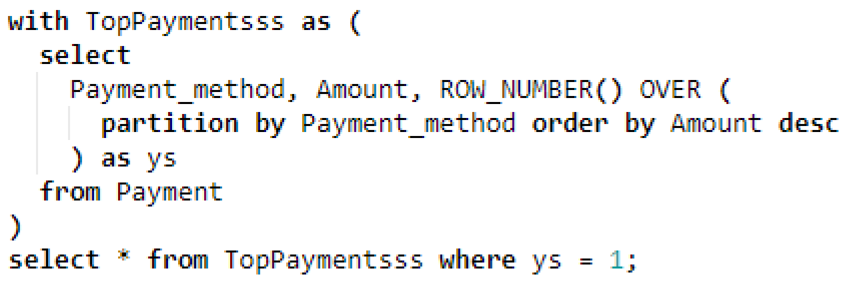
```
select
  Payment_method,
  count(Payment_id)        as Total_Orders,
  sum(Amount)              as Total_Amount
from Payment
group by Payment_method
having sum(Amount) > 100
order by Total_Amount desc;
```



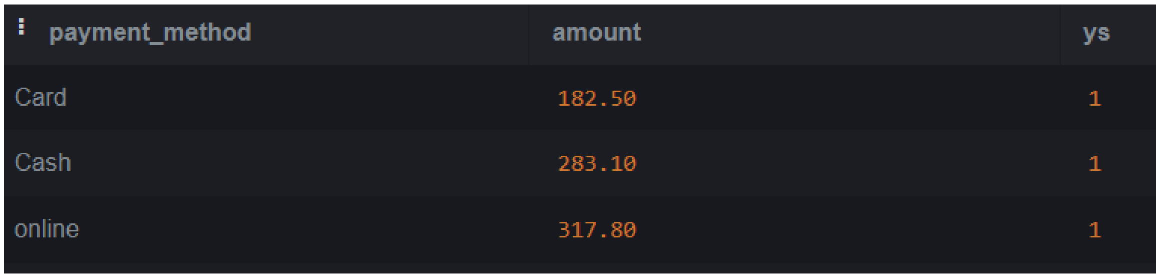
payment_method	total_orders	total_amount
Cash	6	877.10
online	3	593.90
Card	3	424.10

This query shows only those payment methods whose combined payment amounts is more than 100 dollas. So maybe the restaurant can focus on highest spending customers or something

6. ezxtra credit 6th querty it is THE common table expresso (cte called toppaymentsss):



```
with TopPaymentsss as (
  select
    Payment_method, Amount, ROW_NUMBER() OVER (
      partition by Payment_method order by Amount desc
    ) as ys
  from Payment
)
select * from TopPaymentsss where ys = 1;
```



payment_method	amount	ys
Card	182.50	1
Cash	283.10	1
online	317.80	1

This shows the largest single payment for each payment method.

I dont know how it would be useful to the restraunt. Maybe they could offer a red carpet for the fattest customer or something to boost engagement idk...

the window function is "SELECT...FROM Payment". it gives each payment's amount and method, and ranks that payment in its payment_method, rank 1 = highest amount.

ISSUES WE HAD

-The foreign key references matching the correct tables (adjusted Query 3’s JOIN so that Order_rest.Order_id joined Invoice.Order_id instead of Customer.Customer_id).

-We first tried working by hosting a server and uploading the csv mock database there, but we had issues so we used INSERT instead in order to have a hard-coded database which we could share easily.

-We had syntax errors in the INSERT statements (some rows incorrectly listed five values for four columns).

PART B:

(we have changed this a little bit, but it’s still valid.)

Entities and Attributes

1. Customer (Regular Entity)
- CustomerID (Key Attribute): Unique identifier for each customer
- Name: Customer's name
- Contact: Customer's contact information

2. Reservation (Weak Entity)
- ReservationNumber (Partial Key): Identifier unique only within the context of a specific customer
- Date: Date of reservation
- Number of People: Number of people in the party
- Special Requests: Any special accommodations needed

3. Menu Item (Regular Entity)
- Item Number (Key Attribute): Unique identifier for each menu item
- Name: Name of the dish
- Price: Cost of the item
- Availability: Whether the item is currently available
- Allergy Information (Multivalued Attribute): List of potential allergens
- Diet Info (Multivalued Attribute): List of dietary categories (vegan, gluten-free, etc.)
- Preparation Time (Derived Attribute): Estimated time to prepare the dish

4. Menu Category (Regular Entity)
- Category ID (Key Attribute): Unique identifier for each category
- DisplayOrder: Sequence number for display ordering
- Description: Description of the category

- Name: Name of the category (main dish, dessert, etc.)

5. Order (Regular Entity)

- Order Number (Key Attribute): Unique identifier for each order
- (order) Date: Date the order was placed
- Status: Current status (pending, in progress, completed, etc.)
- (order) Special Requests: Any special instructions for the entire order
- TotalAmount (Derived Attribute): Calculated sum of all order items

6. Order Detail (Weak Entity)

- (order) Special Requests: Any special instructions for this specific item
- Quantity: Number of this item ordered
- Status: Current status of this item (pending, in preparation, served, etc.)
- Subtotal (Derived Attribute): Calculated price * quantity

7. Table (Regular Entity)

- Table Number (Key Attribute): Unique identifier for each table
- Capacity: Maximum number of people who can be seated
- Status: Current status (available, occupied, reserved, etc.)

8. Ingredient (Regular Entity)

- Ingredient ID (Key Attribute): Unique identifier for each ingredient
- Name: Name of the ingredient
- Description: Description of the ingredient
- Number of in Stock: Current inventory level
- Restock Date: Last time the ingredient was restocked
- Expiry Date (Multivalued Attribute): List of expiration dates for different batches

9. Supplier (Regular Entity)

- Supplier ID (Key Attribute): Unique identifier for each supplier
- Contact Information (Composite Attribute):
- Phone: Supplier's phone number
- Email: Supplier's email address
- Person: Name of primary contact
- Address (Composite Attribute):
- Street: Street address
- City: City name
- Zip Code: Postal code
- Supplied Items (Multivalued Attribute): List of items provided by this supplier

10. Payment (Weak Entity)

- Payment Number (Partial Key): Identifier unique only within the context of a specific order
- Amount: Payment amount
- (payment) Date: Date of payment
- Method: Method used (cash, credit card, etc.)
- Status: Current status (pending, completed, refunded, etc.)

Relationships

1. Makes (Customer to Reservation)
- Cardinality: 1:N
- Description: A customer can make multiple reservations, but each reservation is made by exactly one customer.
- Participation: Total participation of Reservation in Makes (identifying relationship) as a reservation cannot exist without a customer. Partial participation of Customer in Makes as not all customers make reservations.
2. Place (Customer to Order)
- Cardinality: 1:N
- Description: A customer can place multiple orders, but each order is placed by exactly one customer.
- Participation: Partial participation of Customer in Places as not all customers place orders. Total participation of Order in Places as every order must be placed by a customer.
4. Assigned To (Table to Reservation)
- Cardinality: 1:N
- Description: A table can be assigned to multiple reservations (at different times), but each reservation is assigned to exactly one table.
- Participation: Total participation of Reservation in Assigned To as every reservation must be assigned to a table. Partial participation of Table in Assigned To as not all tables are reserved at all times.
5. Contains (Order to Order Detail)
- Cardinality: 1:N
- Description: An order contains multiple order details, but each order detail belongs to exactly one order.
- Participation: Total participation of Order Detail in Contains (identifying relationship) as an order detail cannot exist without an order. Total participation of Order in Contains as every order must contain at least one order detail.
6. Process (Order to Payment)
- Cardinality: 1:1
- Description: An order has exactly one payment, and a payment is for exactly one order.
- Participation: Total participation of Payment in Processes (identifying relationship) as a payment cannot exist without an order. Total participation of Order in Processes as every order must have a payment.
7. Contains (Order Detail to Menu Item)
- Cardinality: N:1
- Description: Multiple order details can reference the same menu item, but each order detail references exactly one menu item.

- Participation: Total participation of Order Detail in References as every order detail must reference a menu item. Partial participation of Menu Item in References as not all menu items are ordered.

8. Belongs To (Menu Item to Menu Category)

- Cardinality: N:1
- Description: Multiple menu items can belong to the same category, but each menu item belongs to exactly one category.
- Participation: Total participation of Menu Item in Belongs To as every menu item must belong to a category. Total participation of Menu Category in Belongs To as every category must contain at least one menu item.

9. Needs Ingredient (Menu Item to Ingredient)

- Cardinality: M:N
- Description: A menu item requires multiple ingredients, and an ingredient can be used in multiple menu items.
- Attributes: QuantityRequired - amount of ingredient needed for the menu item
- Participation: Total participation of Menu Item in Needs Ingredient as every menu item must require at least one ingredient. Partial participation of Ingredient in Needs Ingredient as some ingredients might be in stock but not currently used in any menu item.

10. Supplies (Supplier to Ingredient)

- Cardinality: 1:N
- Description: A supplier supplies multiple ingredients, but each ingredient is supplied by exactly one supplier.
- Participation: Partial participation of Ingredient in Supplies as some ingredients might be produced in-house. Total participation of Supplier in Supplies as every supplier must supply at least one ingredient.

*Part A is in the first page.

Mehmet Emin Cicek #96769995
mehmet.cicek@ue-germany.de

Aloisio Marques De Oliveira Junio #48215888
aloisio.marques@ue-germany.de

Xiaoshan Zhou #31740706
xiaoshan.zhou@ue-germany.de

Sneha Jain #58648566
sneha.jain@ue-germany.de