DATABASE SPECIFICATIONS

Next-Gen Restaurant

avi5117@psu.edu Afolabi Isiaka

School of Graduate Professional Studies
Information Science Department
IN SC 521 - Introduction to Database Concepts

Spring 2024

1. DOCUMENT CONTROL

Work carried out by:

Name	Email Address	Other
Afolabi Isiaka	**avi5117@psu.edu**	

Revision Sheet

Release No.	Date	Revision Description
1.0	**1/20/2024**	Defined Milestone 1– Data Requirements–review pages 1-2 for context on this release change for theData Requirements section.
1.1	**1/4/2024**	Defined Milestone 2– Conceptual Design–review pages 3-4 for context on this release change for the Conceptual Design section.
1.2	**2/18/2024**	Defined Milestone 3– Logical Design–review pages 4-5 for context on this release change for the Logical Design section.
1.3	**2/28/2024**	Defined Milestone 4– Normalization–review pages 6-7 for context on this release change for Normalization section.
1.4	**3/18/2024**	Defined Milestone 5 → Added Oracle SQl queries for creation and matching data types. Review pages 8-9 Updated Milestone 1 →Added Data Requirements Updated Milestone 3 →Added captions to Logical Design Updated Milestone 4→Naming conventions used are in snake case
1.5	**4/11/2024**	Defined Milestone 6– SQl Queries for data insights–review pages 27-33 f Updated Milestone 5 → Updated Database FKs to better match Conceptual Design. Generated report for Logical ERD

DATABASE SPECIFICATIONS

TABLE OF CONTENTS

2. MILESTONE 1: DATA REQUIREMENTS

System Name

Next-Gen Restaurant Application

Purpose

Increase customer service through the efficient automation of consumer and staff interactions while collecting trend data that will offer stakeholders analytics to maximize operating efficiencies and reduce expenses.

Outcomes

The primary goal of this task is to identify and document the specific data types essential for the Next-Gen Restaurant Application's storage and management. These include customer bookings, table structures, order processing, staff rostering, and sales transactions. The process involves transforming the SRS's described functions into concrete data elements, ensuring the inclusion of all necessary information to support these functions. The objective is to create a correct and thematic blueprint of the required data structure, which is vital for the next phases of database and application development.

Core requirements

No.	Requirement	Referenced page in SRS	Referenced Section in SRS	Referenced Paragraph in Section
1	The system should store digital map renderings of the restaurant's table layout to allow administrators to customize and update the seating arrangements as needed.	10	3.5.2	3.5.2.1-3.5.2.10
2	The system should store customer information for both walk-in customers and those with reservations.	10	3.5.3	3.5.3.1-3.5.3.8
3	The system should store workforce management information for staff scheduling, including staff schedules, roles, hours worked, and shift patterns.	3	3.5.1	3.5.1.1-3.5.1.12
4	The system should store order details including menu items ordered, quantities, prices, and special instructions.	9	3.5.1	3.5.1.1-3.5.1.12
5	The system should store payment information including method, amount, and transaction details.	9	3.5.1	3.5.1.9-3.5.1.12
6	The system should store transaction records including transaction ID, order ID, payment method, transaction amount, date, and time.	9	3.5.1	3.5.1.9-3.5.1.12

7	The system should store staff authentication details including usernames and passwords or other authentication methods.	13	3.5.1	5.1.1-5.1.2
8	The system should store wait queue information including customer ID, name, party size, wait time, and contact information.	10	3.5.3	3.5.3.1-3.5.3.8
9	The system should store information on gratuities given, including amount, associated order, and staff member.			
10	The system should store bar tab details including tab ID, customer ID, legal drinking age, open and close times, and total amount.	8	3.1	3.1.1-3.1.2
11	The system should store ingredients information including ingredient ID, name, quantity in stock, and supplier information.			
12	The system should store comprehensive accounting information including all financial transactions, sales tax rate, payments, receipts, gratuities, and bar tabs.	10	3.5.2	3.5.2.1-3.5.2.10

3. MILESTONE 2: CONCEPTUAL DESIGN

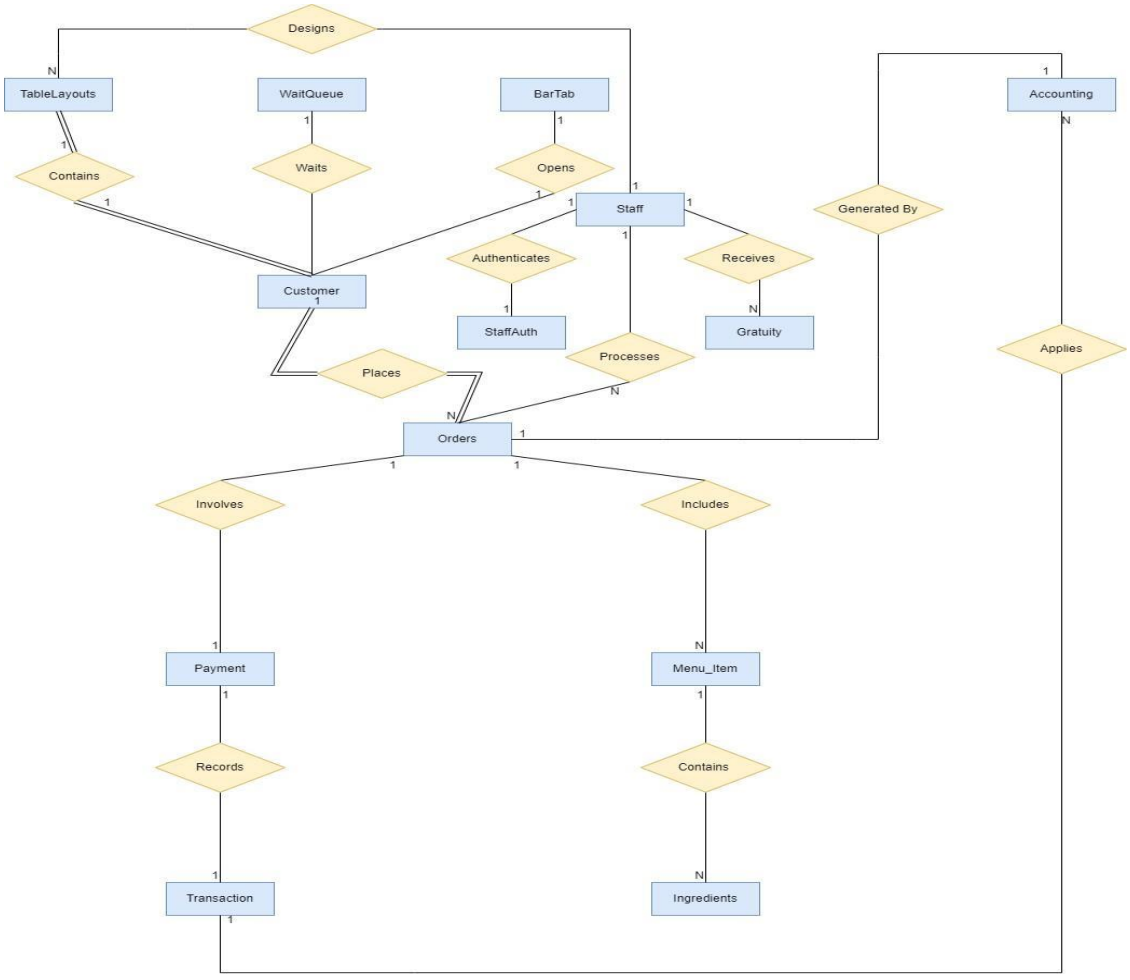
Outcomes

An Entity-Relationship Diagram in Chen notation is generated from the core requirements that were gathered in previous milestones.

Diagram

Chen Notation-Conceptual Entity Relationship Diagram

- 1:N One-to-Many
- 1:1: One-to-One
- M:N Many-to-Many
- N:1 Many-to-1



Assumptions and Constraints

Assumptions

1. **Unique Identifiers**: Each table in the database has a unique identifier (primary key) to distinguish each record.
2. **Customer Identification**: Customers can be identified uniquely by a combination of their name, phone number, and email address.
3. **Staff Roles**: Each staff member has a unique role and can have multiple schedules but is associated with one set of authentication credentials.
4. **Order Complexity**: Orders can contain multiple menu items, and each menu item can appear in multiple orders (many-to-many relationship).
5. **Payment Methods**: Payments are associated with orders, and each order has exactly one payment method but can result in multiple transactions (e.g., split bills).
6. **Bar Tabs and Gratuity**: Bar tabs are considered separate from regular orders and payments. Gratuity is optionally associated with each payment.

Constraints

1. **Referential Integrity**: Foreign keys are used to maintain referential integrity between entities like orders to customers or menu items to orders.
2. **Non-null Constraints**: Essential fields like customer ID, staff ID, order ID, and payment ID cannot be null.
3. **Unique Constraints**: Email addresses for customers, usernames for staff, and identifiers like order ID, payment ID must be unique.
4. **Participation Constraints**:

a. Orders concerning Customers and TableLayouts have total participation, as every order must be linked to a customer and a table layout. This is depicted with a double line.

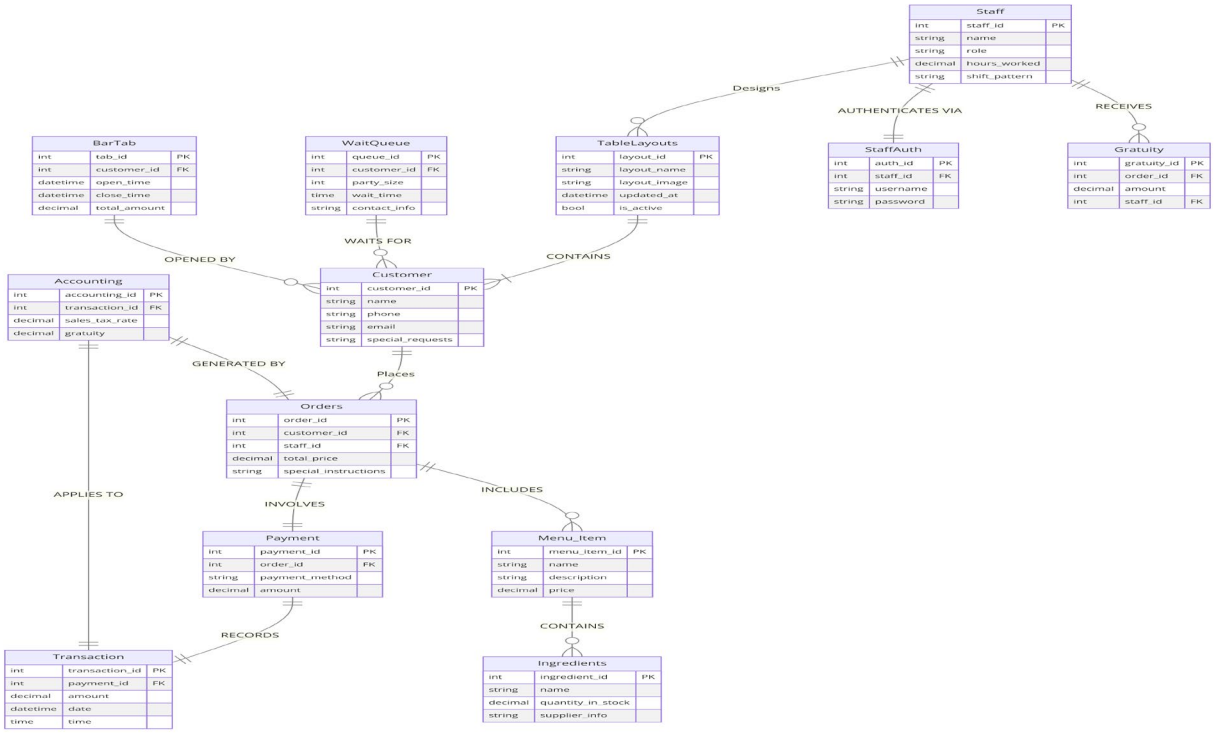
b. Staff concerning Orders might be partial, indicating that not all staff members are involved in orders. This is represented by a single line.

5. **Business Rules**:

- A legal drinking age is enforced on bar tab transactions.
- The sales tax rate applies to all transactions uniformly.
- Staff schedules must not overlap for the same staff member.

4. MILESTONE 3: LOGICAL DESIGN

Entity Relationship Diagram; Crow Foot Notation



Entity name: TableLayouts

Attributes:

Example: layout_id, layout_name, layout_image, updated_at, is_active

Functional dependencies:

Example: layout_id → layout_name, layout_image, updated_at, is_active

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	Layout_id		layout_name, layout_image, updated_at, is_active

Attribute closures (if any):
(layout_id)+ = { layout_id, layout_name, layout_image, updated_at, is_active}

Unique keys: the key for this table is/are
layout_id

Entity name: WaitQueue

Attributes:

Example: queue_id, customer_id, party_size, wait_time, contact_info

Functional dependencies:

Example: queue_id → customer_id, party_size, wait_time, contact_info

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	Queue_id		customer_id, party_size, wait_time, contact_info

Attribute closures (if any):
{queue_id} = {queue_id, customer_id, party_size, wait_time, contact_info}

Unique keys: the key for this table is/are queue_id

Entity name: BarTab

Attributes:

Example: tab_id, customer_id, open_time, close_time, total_amount

Functional dependencies:

Example: tab_id →customer_id, open_time, close_time, total_amount

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	Tab_id		customer_id, open_time, close_time, total_amount

Attribute closures (if any):
{tab_id} = {tab_id, customer_id, open_time, close_time, total_amount}

Unique keys: the key for this table is/are tab_id

Entity name: Customer

Attributes:

Example: customer_id, name, phone, email, special_requests

Functional dependencies:

Example: customer_id → name, phone, email, special_requests

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	customer_id		name, phone, email, special_requests

Attribute closures (if any):
{customer_id} = {customer_id, name, phone, email, special_requests}

Unique keys: the key for this table is/are customer_id

Entity name: Staff

Attributes:

Example: staff_id, name, role, hours_worked, shift_pattern

Functional dependencies:

Example: staff_id →name, role, hours_worked, shift_pattern

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	staff_id		name, role, hours_worked, shift_pattern

Attribute closures (if any):
{staff_id} = {staff_id, name, role, hours_worked, shift_pattern}

Unique keys: the key for this table is/are "staff_id".

Entity name: StaffAuth

Attributes:

Example: auth_id, staff_id, username, password

Functional dependencies:

Example: order_id →customer_id, staff_id, total_price, special_instructions

customer_id →(none without order_id)

staff_id →(none without order_id)

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	auth_id		staff_id, username, password

Attribute closures (if any):
{tab_id} = { auth_id, staff_id, username, password}

Unique keys: the key for this table is/are tab_id

Entity name: Orders

Attributes:

Example: order_id, customer_id, staff_id, total_price, special_instructions

Functional dependencies:

Example: auth_id →staff_id, username, password

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	order_id		customer_id, staff_id, total_price, special_instructions

Attribute closures (if any):
{order_id} = {customer_id, staff_id, total_price, special_instructions}

Unique keys: the key for this table is/are order_id

Entity name: Payment

Attributes:

Example: payment_id, order_id, payment_method, amount

Functional dependencies:

Example: payment_id →order_id, payment_method, amount

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side

	payment_id		order_id, payment_method, amount

Attribute closures (if any):
{payment_id} = {payment_id, order_id, payment_method, amount}

Unique keys: the key for this table is/are payment_id

Entity name: Transaction

Attributes:

Example: transaction_id, payment_id, amount, date, time

Functional dependencies:

Example: transaction_id → payment_id, amount, date, time

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	transaction_id		payment_id, amount, date, time

Attribute closures (if any):
{transaction_id} = {payment_id, amount, date, time}

Unique keys: the key for this table is/are transaction_id

Entity name: Menu_Item

Attributes:

Example: menu_item_id, name, description, price

Functional dependencies:

Example: menu_item_id →name, description, price

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	menu_item_id		name, description, price

Attribute closures (if any):
{menu_item_id} = {menu_item_id, name, description, price}

Unique keys: the key for this table is/are menu_item_id

Entity name: Ingredients

Attributes:

Example: ingredient_id, name, quantity_in_stock, supplier_info

Functional dependencies:

Example: ingredient_id → name, quantity_in_stock, supplier_info

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	ingredient_id		name, quantity_in_stock, supplier_info

Attribute closures (if any):
{ingredient_id} = {ingredient_id, name, quantity_in_stock, supplier_info}

Unique keys: the key for this table is/are ingredient_id

Entity name: Accounting

Attributes:

Example: accounting_id, transaction_id, sales_tax_rate, gratuity

Functional dependencies:

Example: accounting_id →transaction_id, sales_tax_rate, gratuity

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	accounting_id		transaction_id, sales_tax_rate, gratuity

Attribute closures (if any):
{accounting_id} = {accounting_id, transaction_id, sales_tax_rate, gratuity}

Unique keys: the key for this table is/are accounting_id

Entity name: Gratuity

Attributes:

Example: gratuity_id, order_id, amount, staff_id

Functional dependencies:

Example: gratuity_id →order_id, amount, staff_id

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side

	gratuity_id		order_id, amount, staff_id

Attribute closures (if any):
{gratuity_id} = {order_id, amount, staff_id}

Unique keys: the key for this table is/are gratuity_id

Assumptions and Constraints

For the entities related to a restaurant management system, here are some assumptions and constraints:

1. Assumptions:

1. Each entity's primary key uniquely identifies a record within its table, ensuring no duplicate entries.
2. Foreign keys in entities like Orders, Payments, and Transactions correctly reference primary keys from other tables, maintaining referential integrity.
3. The database supports transactions to ensure data consistency, especially for operations that span multiple tables (e.g., creating an order and updating inventory).
4. Data entered into the system reflects real-world operations and interactions within the restaurant (e.g., orders placed are for menu items that exist).
5. The system's time is accurately synchronized with the real world to correctly timestamp transactions, orders, and updates.

2. Constraints:

1. String attributes such as names, descriptions, and contact information are limited to specific lengths to ensure data uniformity.
2. Numerical values like price, quantity, and amount have minimum and maximum values to prevent unrealistic data entry.
3. Certain fields are required (cannot be null) to ensure complete records; for example, an Order must have an associated Customer.
4. Relationships between entities are enforced through foreign keys, where deleting a record in a primary table (like a Customer) might restrict deletion if related records exist in dependent tables (like Orders).
5. Data types and formats (e.g., datetime for timestamps, and decimal for monetary values) are strictly enforced to ensure data integrity and accurate calculations/reporting.

5. MILESTONE 4 & 5: NORMALIZATION AND PHYSICAL DESIGN

Assumptions and Constraints

3. Assumptions

1. **Unique Identification**: Each entity has a unique identifier, a primary key, serving as a distinctive beacon to differentiate each entity within its table. This unique identity is paramount for the clear identification and retrieval of data.

2. **Data Integrity**: The sanctity of relationships between tables is upheld through the meticulous application of foreign keys. These keys serve as the linchpins of referential integrity, ensuring that connections between tables remain pristine and unbroken.

3. **Atomicity**: The principle of atomicity dictates that attributes must be indivisible, ensuring that each attribute retains its fundamental essence without subdivision. This granularity ensures that the data remains coherent and meaningful at the most elemental level.

4. **Business Rules**: The database architecture is a reflection of the organizational ethos and operational paradigms. For instance, the distribution of gratuity among staff based on serviced orders is a manifestation of such business rules encoded within the database design.

4. Constraints

1. **Primary Key Constraint**: This constraint acts as the guardian of uniqueness, mandating that each record within a table can be unequivocally identified by its primary key.

2. **Foreign Key Constraint**: A cornerstone of relational integrity, this constraint ensures that the value of a foreign key corresponds to an existing value within the referenced table, thus maintaining the integrity of the relational links.

3. **Not Null Constraint**: This constraint enforces the imperative that certain fields cannot remain unfilled, ensuring that every record is complete and devoid of nullities for essential attributes.

4. **Unique Constraint**: Beyond the primary key, this constraint ensures the exclusivity of values within a column, critical for attributes such as username in the StaffAuth table, where uniqueness is paramount.

5. **Check Constraint**: This safeguard imposes conditions on the values within a column, ensuring adherence to defined rules, such as the positivity of numerical amounts.

Naming Conventions

Discuss the naming standards and conventions that you have used for table creation.

Tables

	Name of the table	***TableLayouts***			
	Description	A Table Layout is the restaurant design for seating arrangements.			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	Layout_id	Id of layout	int	1-99999	NOT NULL
	Layout_name	Name of layout	string	John	
	Layout_image	Layout image	string		
	Updated_at	Layout update	datetime	1/1/2021	
	is_active	Is table Active	bool	“Active”, “Inactive”	Notations can be used for space
	Functional Dependencies and Keys				
	Functional dependencies	layout_id → layout_name, layout_image, updated_at, is_active			
	Candidate keys	**Layout_id**			
	Normalization				
	1NF	**Yes**	There are no repeating groups		
	2NF	**Yes**	all non-key attributes are fully functionally dependent on the primary key		
	3NF	**Yes**	In 2NF and all non-key attributes are fully functionally dependent on the primary key		
	BCNF	**Yes**	In 3NF and the left-hand side is a superkey.		
	Physical Design				
	Primary Key	Layout_id			
	Foreign Keys	**-**			
	SQL Code	CREATE TABLE tablelayouts(layout_id INT PRIMARY KEY, layout_name VARCHAR(20) NOT NULL, updated_at TIMESTAMP NOT NULL, is_active NUMBER(1));			
	Count of records in the table	**20**			

	Name of the table	***Customer***			
	Description	The Customer is the individual our staff will attend to at the table.			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	customer_id	Id of customer	int	1-99999	
	name	The customer's name	string	"John Doe", "Jane Smith"	Not Null
	phone	customer's phone number	string	"555-1234", "555-5678"	Can be unique; format validation may be applied.
	email	customer's email address	string	jane.smith@example.com	Must be unique; validation for email format
	special_requests	Any special requests made by the customer.	string	Allergic to peanuts	Can be null
	Functional Dependencies and Keys				
	Functional dependencies	customer_id → name customer_id →phone customer_id →email customer_id →special_requests			
	Candidate keys	**customer_id**			
	Normalization				
	1NF	**Yes**	There are no repeating groups		
	2NF	**Yes**	table is already in 1NF and all non-key attributes are fully dependent on the primary key		
	3NF	**Yes**	All attributes are directly dependent on the primary key		
	BCNF	**Yes**	table is in BCNF because each determinant is a candidate key		

	Physical Design	
	Primary Key	customer_id
	Foreign Keys	-
	SQL Code	CREATE TABLE customer(customer_id INT PRIMARY KEY, name VARCHAR(20) NOT NULL, phone VARCHAR(50) NOT NULL, email VARCHAR(50), special_requests VARCHAR(50));

Count of records in the table	255

	Name of the table	***WaitQueue***			
	Description	A Wait Queue is the line in which we will hold customers until a table is ready.			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	queue_id	Id of queue	int	1-99999	Not Null
	customer_id	Id of customer	int	1-99999	FK, Not Null
	party_size	size of the party	int	1-100	
	wait_time	Expected wait time	time	'00:15:00' 15 minutes	'HH:MM:SS' format.
	contact_info	Contact information for the customer in queue	string	'555-1234', 'contact@example.com'	
	Functional Dependencies and Keys				
	Functional dependencies	queue_id → customer_id queue_id →party_size queue_id →wait_time queue_id →contact_info			
	Candidate keys	**queue_id**			
	Normalization				
	1NF	**Yes**	There are no repeating groups		
	2NF	**Yes**	table is already in 1NF and all non-key attributes are fully dependent on the primary key		
	3NF	**Yes**	All attributes are directly dependent on the primary key		
	BCNF	**Yes**	table is in BCNF because each determinant is a candidate key		

	Physical Design	
	Primary Key	**queue_id**
	Foreign Keys	customer_id
	SQL Code	CREATE TABLE waitqueue(queue_id INT NOT NULL PRIMARY KEY, customer_id INT, party_size INT NOT NULL, wait_time TIMESTAMP NOT NULL, email VARCHAR(50), contact_info VARCHAR(225),

	CONSTRAINT fk_customer FOREIGN KEY (customer_id) REFERENCES customer(customer_id)
Count of records in the table	255

Name of the table	***Staff***			
Description	The staff are your restaurant's employees.			
Attribute	**Description**	**Type**	**Examples of values**	**Notes**
staff_id	Id of staff	int	1-99999	
name	The staff's name	string	"Alex Johnson", "Maria Garcia"	Not Null
role	customer's phone number	string	"Waiter", "Chef"	Defines the staff member's
hours_worked	customer's email address	total hours worked	40.5, 38.75	used for payroll calculations; can vary week by week.
shift_pattern	The shift pattern of the staff member	string	"Morning", "Evening", "Night"	Indicates the usual shifts

Functional Dependencies and Keys		
Functional dependencies	staff_id → name staff_id →role staff_id →hours_worked staff_id →shift_pattern	
Candidate keys	**staff_id**	
Normalization		
1NF	**Yes**	There are no repeating groups
2NF	**Yes**	table is already in 1NF and all non-key attributes are fully dependent on the primary key
3NF	**Yes**	All attributes are directly dependent on the primary key with no transitive dependencies.
BCNF	**Yes**	table is in BCNF because each determinant is a candidate key

Physical Design	
Primary Key	staff_id

Foreign Keys	-
SQL Code	CREATE TABLE staff (staff_id INT, name VARCHAR(50), role VARCHAR(50), hours_worked DECIMAL(5, 2), shift_pattern VARCHAR(255), PRIMARY KEY (staff_id));
Count of records in the table	255.

Name of the table	***StaffAuth***			
Description	The staff login credentials.			
Attribute	**Description**	**Type**	**Examples of values**	**Notes**
auth_id	Id of authentication staff	int	1-99999	
staff_id	Id of staff	int	1-99999	FK
username	username used for authentication	string	"alexj", "mariag"	Must be unique
password	password used for authentication	string	"Password123", "SecurePass!@#"	Stored securely; possibly hashed for security.
Functional Dependencies and Keys				
Functional dependencies	auth_id → staff_id auth_id →username auth_id →password			
Candidate keys	**auth_id**			
Normalization				
1NF	**Yes**	There are no repeating groups		
2NF	**Yes**	table is already in 1NF and all non-key attributes are fully dependent on the primary key		
3NF	**Yes**	All attributes are directly dependent on the primary key with no transitive dependencies.		
BCNF	**Yes**	table is in BCNF because each determinant is a candidate key		

Physical Design	
Primary Key	auth_id

Foreign Keys	staff_id
SQL Code	CREATE TABLE staffauth (auth_id INT, staff_id INT, username VARCHAR(255) NOT NULL, password VARCHAR(255) NOT NULL, PRIMARY KEY (auth_id), CONSTRAINT fk_staff FOREIGN KEY (staff_id) REFERENCES staff(staff_id));
Count of records in the table	255

	Name of the table	***Orders***			
	Description	Customer requests for meals are processed as orders.			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	order_id	Id of Orders	int	1-99999	
	staff_id	Id of staff	int	1-99999	FK
	total_price	total price of the order	decimal	23.50, 45.00, 8.75	Must be unique
	special_instructions	customer's preferences	total hours worked	"Extra sauce", "No onions", "Allergy to peanuts"	can be left empty
	Functional Dependencies and Keys				
	Functional dependencies	order_id → customer_id order_id →staff_id order_id →total_price order_id →special_instructions			
	Candidate keys	**order_id**			
	Normalization				
	1NF	**Yes**	There are no repeating groups		
	2NF	**Yes**	table is already in 1NF and all non-key attributes are fully dependent on the primary key		
	3NF	**Yes**	All attributes are directly dependent on the primary key with no transitive dependencies.		
	BCNF	**Yes**	table is in BCNF because each determinant is a candidate key		

	Physical Design	
	Primary Key	order_id
	Foreign Keys	staff_id
	SQL Code	CREATE TABLE orders (order_id INT, staff_id INT, total_price DECIMAL(10, 2), -- Specified precision and scale special_instructions VARCHAR(255), PRIMARY KEY (order_id), CONSTRAINT fk_staff_1 FOREIGN KEY (staff_id) REFERENCES staff(staff_id));
	Count of records in the table	255

	Name of the table	***Payment***			
	Description	Customer payment for order			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	payment_id	Id of authentication staff	int	1-99999	
	order_id	Id of staff, Foreign Key	int	1-99999	FK
	payment_method	The method of payment	string	"Cash", "Credit Card", "PayPal"	
	amount	the amount paid	decimal	20.00, 45.50, 100.75	
	Functional Dependencies and Keys				
	Functional dependencies	payment_id → order_id payment_id →payment_method payment_id →amount			
	Candidate keys	**payment_id**			
	Normalization				
	1NF	**Yes**	There are no repeating groups		
	2NF	**Yes**	table is already in 1NF and all non-key attributes are fully dependent on the primary key		
	3NF	**Yes**	All attributes are directly dependent on the primary key with no transitive dependencies.		
	BCNF	**Yes**	table is in BCNF because each determinant is a candidate key		

	Physical Design	
	Primary Key	payment_id
	Foreign Keys	order_id
	SQL Code	CREATE TABLE payment (payment_id INT, order_id INT, payment_method VARCHAR(25), amount DECIMAL(10, 2), -- Specified precision and scale PRIMARY KEY (payment_id), CONSTRAINT fk_order FOREIGN KEY (order_id) REFERENCES orders(order_id));
	Count of records in the table	255

	Name of the table	***Transaction***			
	Description	payment specifications for order			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	transaction_id	Id of transaction	int	1-99999	
	payment_id	Id of payment, Foreign Key	int	1-99999	FK
	amount	The amount of the transaction	decimal	50.00, 75.25, 99.99	
	date	The date of the transaction.	datetime	'2024-01-01', '2024-01-02'	
	time	The time of the transaction	time	'13:00:00', '14:30:00'	
	Functional Dependencies and Keys				
	Functional dependencies	transaction_id → payment_id transaction_id →amount transaction_id →date transaction_id →time			
	Candidate keys	**transaction_id**			
	Normalization				
	1NF	**Yes**	There are no repeating groups		
	2NF	**Yes**	table is already in 1NF and all non-key attributes are fully dependent on the primary key		
	3NF	**Yes**	All attributes are directly dependent on the primary key with no transitive dependencies.		

BCNF	**Yes**	table is in BCNF because each determinant is a candidate key

Physical Design	
Primary Key	transaction_id
Foreign Keys	payment_id
SQL Code	Create Table Transactions (Transaction_Id Int, Payment_Id Int, Amount Decimal(10, 2) Not Null, -- Specified precision and scale Transaction_Date Date, -- Renamed to avoid reserved word conflict Transaction_Time Timestamp, -- Renamed for clarity and to avoid reserved word conflict Primary Key (Transaction_Id), Constraint Fk_Payment Foreign Key (Payment_Id) References Payment(Payment_Id));
Count of records in the table	50

Name of the table	***Menu_Item***			
Description	Items in the menu for the restaurant			
Attribute	**Description**	**Type**	**Examples of values**	**Notes**
menu_item_id	Id of menu items	int	1-99999	
name	The name of the menu item	string	"Margherita Pizza", "Caesar Salad"	FK
description	the amount of the transaction	string	"Classic Italian pizza with fresh mozzarella and basil", "Romaine lettuce with Caesar dressing"	
price	The price of the menu item	decimal	8.99, 12.50,	
Functional Dependencies and Keys				
Functional dependencies	menu_item_id → name menu_item_id →description menu_item_id →price			
Candidate keys	**menu_item_id**			
Normalization				

	1NF	**Yes**	There are no repeating groups
	2NF	**Yes**	table is already in 1NF and all non-key attributes are fully dependent on the primary key
	3NF	**Yes**	All attributes are directly dependent on the primary key with no transitive dependencies.
	BCNF	**Yes**	table is in BCNF because each determinant is a candidate key

	Physical Design	
	Primary Key	menu_item_id
	Foreign Keys	
	SQL Code	CREATE TABLE Menu_Item(menu_Item_id int, name varchar(50) not NULL, description varchar(255) not NULL, price decimal not null, PRIMARY KEY (menu_Item_id));
	Count of records in the table	50

	Name of the table	***Ingredients***			
	Description	Foods or substances that are combined to make a meal			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	ingredient_id	Id of ingredient	int	1-99999	
	name	The name of the ingredient	string	"Tomatoes", "Flour", "Mozzarella Cheese"	
	quantity_in_stock	the amount of the transaction	decimal	50.0 (kilograms), 100.0 (liters)	
	supplier_info	The price of the menu item	string	"Supplier A - Phone: 555-1234"	crucial for reordering and supplier relations.
	Functional Dependencies and Keys				
	Functional dependencies	ingredient_id → name ingredient_id →quantity_in_stock ingredient_id →supplier_info			

	Candidate keys	**ingredient_id**	
	Normalization		
	1NF	**Yes**	There are no repeating groups
	2NF	**Yes**	table is already in 1NF and all non-key attributes are fully dependent on the primary key
	3NF	**Yes**	All attributes are directly dependent on the primary key with no transitive dependencies.
	BCNF	**Yes**	table is in BCNF because each determinant is a candidate key

	Physical Design	
	Primary Key	ingredient_id
	Foreign Keys	**-**
	SQL Code	CREATE TABLE Ingredients(ingredient_id int, name varchar(50) not null, quantity_in_stock decimal not null, supplier_info varchar(255) not null, PRIMARY KEY (ingredient_id));
	Count of records in the table	50.

	Name of the table	***Accounting***			
	Description	Table containing details of transactions for bookkeeping.			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	accounting_id	accounting	int	1-99999	
	transaction_id	foreign key linking to the Transaction entity	int	1-99999	
	sales_tax_rate	sales tax rate applied to the transaction.	decimal	0.07 (7%), 0.08 (8%),	
	gratuity	gratuity amount associated with the transaction	decimal	5.00	
	Functional Dependencies and Keys				
	Functional dependencies	accounting_id → transaction_id accounting_id →sales_tax_rate accounting_id →gratuity			

	Candidate keys	**accounting_id**	
	Normalization		
	1NF	**Yes**	There are no repeating groups
	2NF	**Yes**	table is already in 1NF and all non-key attributes are fully dependent on the primary key
	3NF	**Yes**	All attributes are directly dependent on the primary key with no transitive dependencies.
	BCNF	**Yes**	table is in BCNF because each determinant is a candidate key

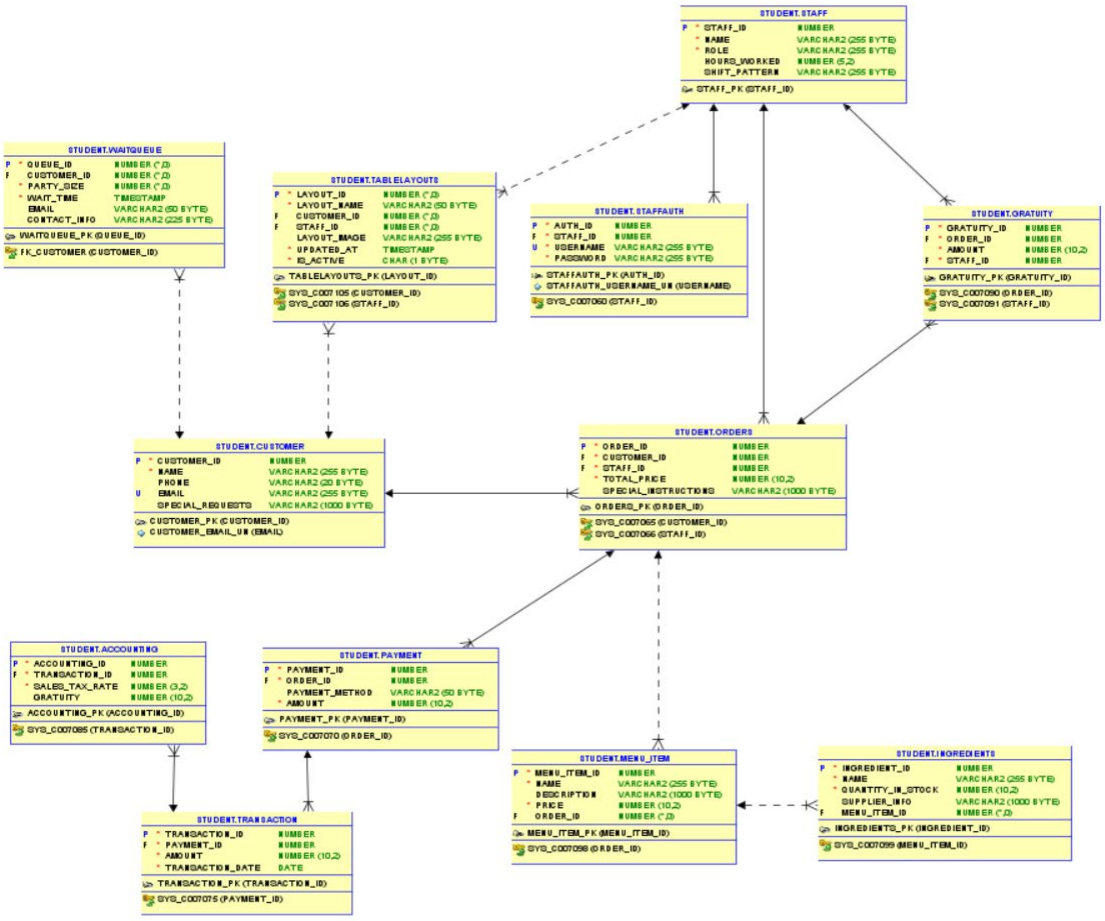
	Physical Design	
	Primary Key	accounting_id
	Foreign Keys	transaction_id
	SQL Code	CREATE TABLE Accounting(accounting_id int, transaction_id int not null, sales_tax_rate decimal not null, gratuity decimal not null, PRIMARY KEY (accounting_id), CONSTRAINT fk_Transaction FOREIGN KEY (transaction_id) REFERENCES Payment(Payment_id));
	Count of records in the table	50

	Name of the table	***Gratuity***			
	Description	Table containing details of transaction for bookkeeping.			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	gratuity_id	id of gratuity	int	1-99999	
	order_id	foreign key linking to the order entity	int	1-99999	
	staff_id	the foreign key linking to the staff entity	int	1-99999	
	amount	sales tax rate applied to the transaction.	decimal	0.07 (7%), 0.08 (8%),	
	Functional Dependencies and Keys				

	Functional dependencies	gratuity_id → order_id gratuity_id →amount gratuity_id →staff_id	
	Candidate keys	**gratuity_id**	
	Normalization		
	1NF	**Yes**	There are no repeating groups
	2NF	**Yes**	table is already in 1NF and all non-key attributes are fully dependent on the primary key
	3NF	**Yes**	All attributes are directly dependent on the primary key with no transitive dependencies.
	BCNF	**Yes**	table is in BCNF because each determinant is a candidate key

	Physical Design	
	Primary Key	gratuity_id
	Foreign Keys	order_id, staff_id
	SQL Code	CREATE TABLE gratuity(gratuity_id int, order_id int, staff_id int, PRIMARY KEY (gratuity_id), CONSTRAINT fk_Orders_1 FOREIGN KEY (order_id) REFERENCES Orders(order_id), CONSTRAINT fk_Staff_2 FOREIGN KEY (staff_id) REFERENCES Staff(staff_id));
	Count of records in the table	**Note**: Please make sure you add 2 records in each table.

Diagram of Physical Design



6. MILESTONE 6: SQL QUERIES

Note: Please make sure you add/have 25 records in each table, on average.

Query 1	
English version	Retrieve the email addresses and special requests of customers who have 'John Doe' as their name.
Source for the query need in the SRS document	SRS document, page 12, section 3.5.4

SQL sentence	`SELECT email, special_requests`
	`FROM Customer`
	`WHERE name = 'John Doe';`

Example of returned rows (cropped screen caption)	14 rows selected (36.563 seconds)

Query 2	
English version	Display order details including customer name and total price for orders above $50.
Source for the query need in the SRS document	SRS document, page 12, section 3.5.4

SQL sentence	`-- Query 2: Display order details for orders above $50`
	`SELECT C.name, O.total_price`
	`FROM Orders O`
	`JOIN Customer C ON O.customer_id = C.customer_id`
	`WHERE O.total_price > 50;`

Example of returned rows (cropped screen caption)	7 rows selected (40.05 seconds)

Query 3	
English version	Find staff who have never taken an order
Source for the query need in the SRS document	SRS document, page 3, section 3.5.2

SQL sentence	-- Query 3: Find staff who have never taken an order SELECT name FROM Staff WHERE staff_id NOT IN (SELECT staff_id FROM Orders);
Example of returned rows (cropped screen caption)	6 rows selected (21.016 seconds)

Query 4	
English version	List menu items sold more than 10 times
Source for the query need in the SRS document	SRS document, page 9, section 3.5.1
SQL sentence	-- Query 4: List menu items sold more than 10 times SELECT M.name, COUNT(*) AS total_sold FROM Orders O JOIN Menu_Item M ON O.order_id = M.menu_item_id GROUP BY M.name HAVING COUNT(*) > 10;
Example of returned rows (cropped screen caption)	17 rows selected (15.769 seconds)

Query 5	
English version	List all customer and staff names
Source for the query need in the SRS document	SRS document, page 10, section 3.5.3
SQL sentence	SELECT name FROM Customer UNION SELECT name FROM Staff;
Example of returned rows (cropped screen caption)	10 rows selected (57.734 seconds)

Query 6	
English version	Show all staff login attempts with roles
Source for the query need in the SRS document	SRS document, page 13, section 3.5.1

SQL sentence	`-- Query 6: Show all staff login attempts with roles`
	`SELECT S.name, S.role, A.username`
	`FROM Staff S`
	`JOIN StaffAuth A ON S.staff_id = A.staff_id;`

Example of returned rows (cropped screen caption)	14 rows selected (21.21 seconds)

Query 7	
English version	Find all orders with payment details where payment was by credit card
Source for the query need in the SRS document	SRS document, page 9, section 3.5.1

SQL sentence	`SELECT O.order_id, P.payment_method, P.amount`
	`FROM Orders O`
	`JOIN Payment P ON O.order_id = P.order_id`
	`WHERE P.payment_method = 'Credit Card';`

Example of returned rows (cropped screen caption)	16 rows selected (27.08 seconds)

Query 8	
English version	Calculate total revenue per staff member from orders
Source for the query need in the SRS document	SRS document, page 9, section 3.5.1

SQL sentence	

	`-- Query 8: Calculate total revenue per staff member from orders`
	`SELECT S.name, SUM(P.amount) AS total_revenue`
	`FROM Staff S`
	`JOIN Orders O ON S.staff_id = O.staff_id`
	`JOIN Payment P ON O.order_id = P.order_id`
	`GROUP BY S.name;`

Example of returned rows (cropped screen caption)	4 rows selected (32.623 seconds)

Query 9	
English version	Report transactions for the first quarter of 2024
Source for the query need in the SRS document	SRS document, page 9, section 3.5.1

SQL sentence	`-- Query 9: Report transactions for the first quarter of 2024`
	`SELECT *`
	`FROM Transaction`
	`WHERE transaction_date BETWEEN '2024-01-01' AND '2024-03-31';`

Example of returned rows (cropped screen caption)	16 rows selected (47.402 seconds)

Query 10	
English version	List all ingredients below a certain inventory level
Source for the query need in the SRS document	(Not Referenced)

SQL sentence	-- Query 10: List all ingredients below a certain inventory level SELECT name, quantity_in_stock FROM Ingredients WHERE quantity_in_stock < 10;
Example of returned rows (cropped screen caption)	8 rows selected (22.567 seconds)

Query 11	
English version	Show detailed accounting information for each transaction
Source for the query need in the SRS document	SRS document, page 10, section 3.5.2
SQL sentence	-- Query 11: Show detailed accounting information for each transaction SELECT A.transaction_id, T.payment_id, A.sales_tax_rate, A.gratuity FROM Accounting A JOIN Transaction T ON A.transaction_id = T.transaction_id;
Example of returned rows (cropped screen caption)	17 rows selected (26.667 seconds)

Query 12	
English version	List all gratuities including the order and staff involved
Source for the query need in the SRS document	SRS document, page 10, section 3.5.2
SQL sentence	-- Query 12: List all gratuities including the order and staff involved SELECT G.gratuity_id, G.amount, O.order_id, S.name FROM Gratuity G JOIN Orders O ON G.order_id = O.order_id JOIN Staff S ON G.staff_id = S.staff_id;

Example of returned rows (cropped screen caption)	29 rows selected (18.271 seconds)

Query 13	
English version	Find all orders that included 'Margherita Pizza' served by staff with over 20 hours worked
Source for the query need in the SRS document	SRS document, page 9, section 3.5.1

SQL sentence	`SELECT O.order_id, S.name, M.name AS menu_item`
	`FROM Orders O`
	`JOIN Staff S ON O.staff_id = S.staff_id`
	`JOIN Menu_Item M ON O.order_id = M.menu_item_id`
	`WHERE S.hours_worked > 20`
	`AND M.name = 'Margherita Pizza';`

Example of returned rows (cropped screen caption)	18 rows selected (5.302 seconds)

Query 14	
English version	Calculate the average amount of transactions per day
Source for the query need in the SRS document	

SQL sentence	`-- Query 14: Calculate the average amount of transactions per day`
	`SELECT transaction_date, AVG(amount) AS average_amount`
	`FROM Transaction`
	`GROUP BY transaction_date;`

Example of returned rows (cropped screen caption)	14 rows selected (43.613 seconds)

Query 15	
English version	Identify customers who have placed orders but have no registered email

Source for the query need in the SRS document	SRS document, page 11, section 3.5.5

SQL sentence	`-- Query 15: Identify customers who have placed orders but have no registered email`
	`SELECT name`
	`FROM Customer`
	`WHERE customer_id IN (SELECT customer_id FROM Orders)`
	`AND email IS NULL;`

Example of returned rows (cropped screen caption)	19 rows selected (39.934 seconds)