

**DATABASE**

**SPECIFICATIONS**

*Next-Gen Restaurant*

[*avi5117@psu.edu*](mailto: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**

***Document Control i***

**Work carried out by: i**

**Revision Sheet i**

***Milestone 1: Data Requirements 1***

**System Name or Title 1**

**Core requirements 1**

***Milestone 2: Conceptual Design 3***

**Diagram 3**

**Assumptions and Constraints 3**

***Milestone 3: Logical Design 4***

**Entity Relationship Diagram 4**

**Assumptions and Constraints 4**

***Milestone 4: Normalization and 5***

***Milestone 5: Physical Design 5***

**Assumptions and Constraints 5**

**Naming Conventions 5**

**Tables 5**

**Examples of values** 5

**Notes** 5

***Milestone 6: SQL queries and 7***

1. **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 |

1. **MILESTONE 2: CONCEPTUAL DESIGN**

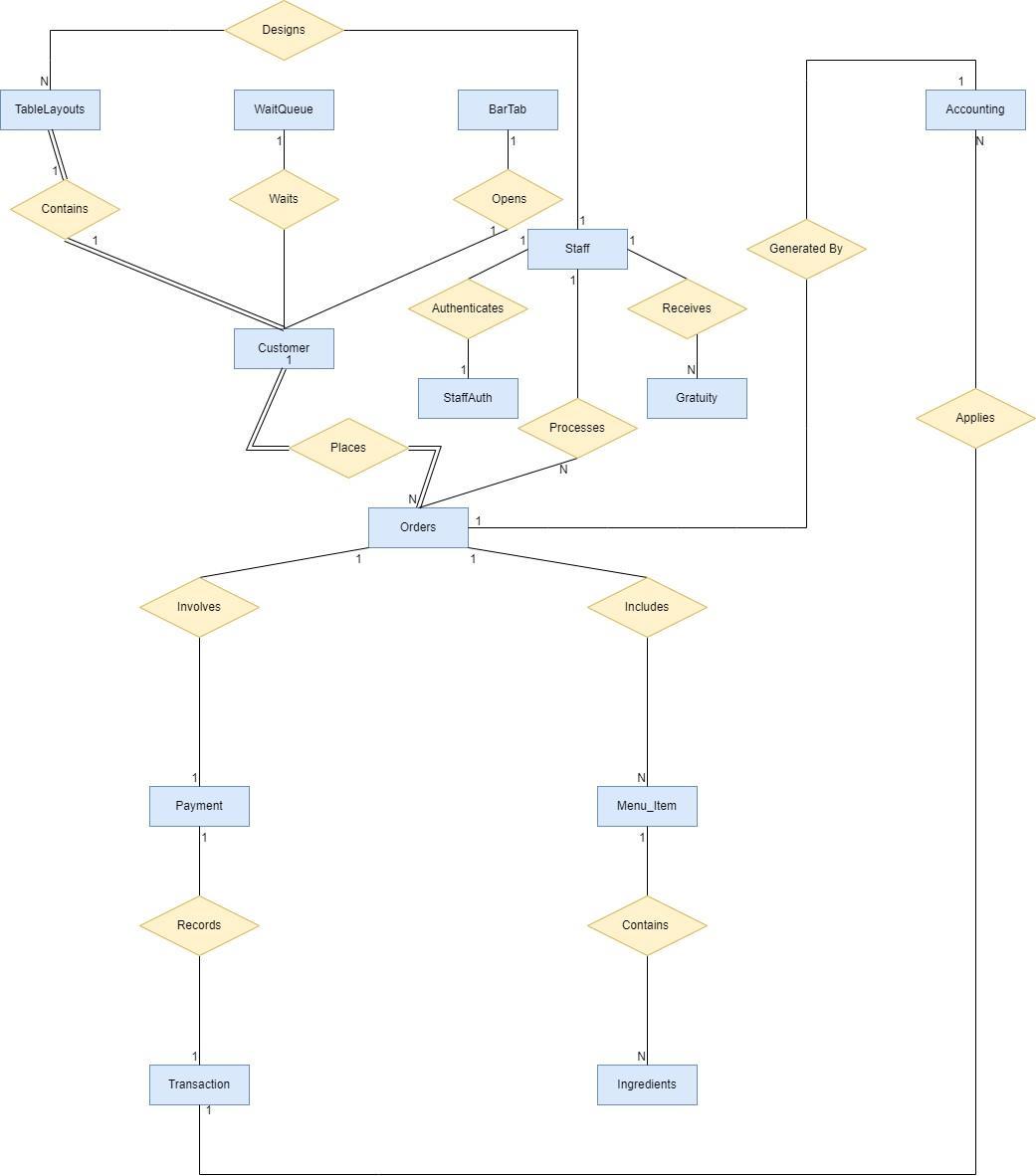
## Outcomes

## 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**

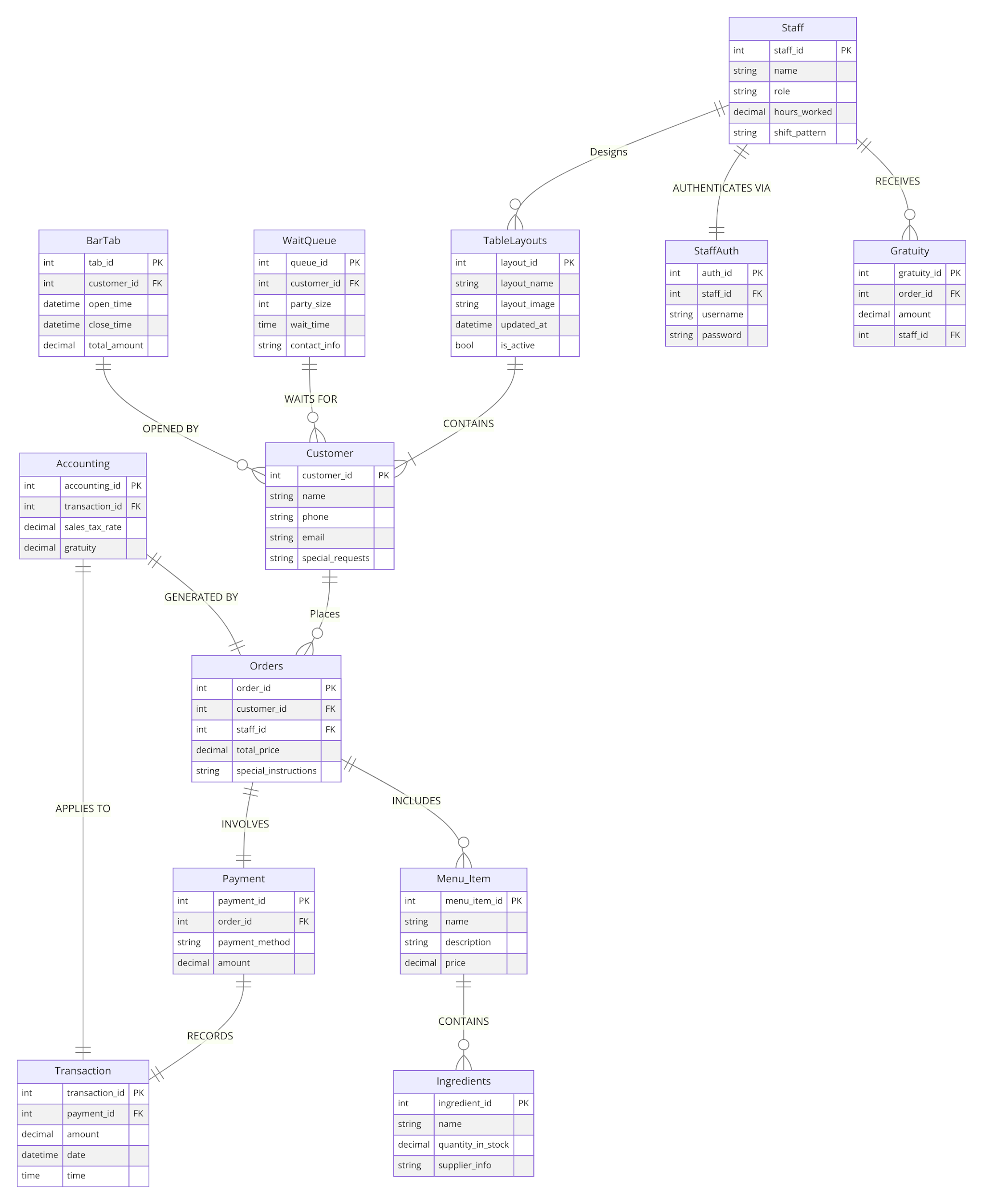
1. **Unique Identifiers**: Each table in the database has a unique identifier (primary key) to distinguish each record.
2. **Customer Identification**: Customers can be identified uniquely by a combination of their name, phone number, and email address.
3. **Staff Roles**: Each staff member has a unique 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**:
   * 1. 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.
     2. 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.

1. **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:

### 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.

### 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.

1. **MILESTONE 4 & 5: NORMALIZATION AND PHYSICAL DESIGN**

## Assumptions and Constraints

### 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.

### 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. |

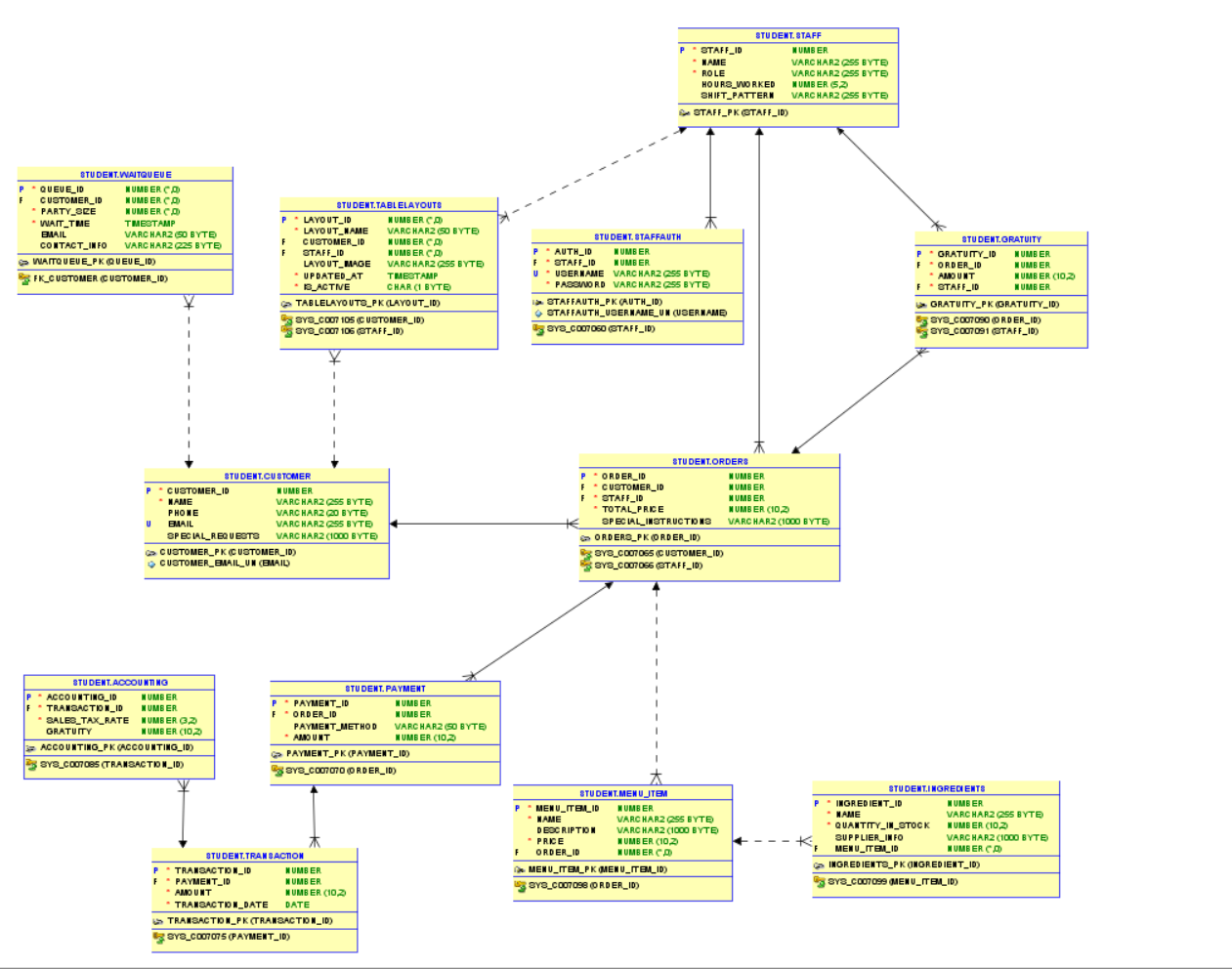
|  | ***Name of the table*** | ***Accounting*** | | | | |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Description** | Table containing details of transactions for bookkeeping. | | | | |
|  | **Attribute** | **Description** | | **Type** | **Examples of values** | **Notes** |
|  | accounting\_id | 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



1. **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) |