

**DATABASE**

**SPECIFICATIONS**

*Next-Gen Restaurant Application Database System (NRADS)*

*Nicholas Forleo: nvf5178@psu.edu*

**School of Graduate Professional Studies**

Information Science Department

INSC 521 - Introduction to Database Concepts

Apr. 2024

# Document Control

## Work carried out by:

|  |  |  |
| --- | --- | --- |
| **Name** | **Email Address** | **Other** |
| **Nicholas Forleo** | **nvf5178@psu.edu** |  |
|  |  |  |
|  |  |  |

## Revision Sheet

Changes in **Bold** are new since the last submission.

|  |  |  |
| --- | --- | --- |
| **Release No.** | **Date** | **Revision Description** |
| 1.0 | 01/21/2024 | Milestone 1 Creation |
| 1.1 | 01/26/2024 | - Added username and password to Requirement No. 9  - Changed “shall” to “will”  - Added Purpose and Outcome section to Milestone 1  - Modified DR regarding logs to note location of stored logs  - Separated menu and tab into respective DR  - Modified attributes for all Core Requirements to be more specific and their referenced page numbers |
| 2.0 | 01/29/2024 | - Milestone 2 Creation |
| 2.1 | 02/07/2024 | - Updated diagram to show Staff is-a User relationship  - Deleted and re-inserted diagram to verify it will persist on download (The web version of MS word was giving me problems, will save via MS Teams and submit as PDF from this version on) |
| 3.0 | 02/13/2024 | - Milestone 3 Creation  - Modified DR1 to remove attribute *table\_layout* and replace with directory location where layouts will be saved to  - Modified DR3 to include the last 3 attributes for more accurate data records  - Removed *gratuity* from DR7  - Updated ERD to show Customer pays Transaction relationship |
| 3.1 | 02/25/2024 | - Expanded all many-to-many type relationships in Logical Model (StaffLocations (Works\_At), Locations Layouts (Location\_Contains\_Layout), Layout Tables (Layout\_Contains\_Table), Tab Menu (Tab\_Has\_Menu\_Item))  - Changed store\_id to location\_id in Locations table  - Added Legend to Logical diagram  - Added Function Dependency Descriptor for new tables mentioned above  - Modified Month on Title page |
| 4.0 | 02/26/2024 | - Milestone 4 creation  - Added Address table to Logical model and its Functional Dependencies (FD)  - Added quantity attribute to Tab\_Has\_Menu\_Item table in FD and Logical Model |
| 5.0 | 03/29/2024 | - Milestone 5 creation  - Added system generated ERD |
| **5.1** | **03/31/2024** | **- Removed position attribute from Staff table, was created in error**  **- Added location\_id to Reservations. Updated accompanying tables.**  **- Updated accurate row counts, didn’t copy from working draft correctly.**  **- Removed NOT NULL constraint from times in Reservation attributes**  **- Removed attributes (and FK) to customer and staff from Transactions table. Duplicated data** |
| **6.0** | **04/07/2024** | **- Milestone 6 creation** |

**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 2

Diagram 2

Assumptions and Constraints 2

Milestone 3: Logical Design 3

Entity Relationship Diagram 3

Assumptions and Constraints 3

Milestone 4: Normalization and 4

Milestone 5: Physical Design 4

Assumptions and Constraints 4

Naming Conventions 4

Tables 4

**Examples of values** 4

**Notes** 4

Milestone 6: SQL queries and 6

# Milestone 1: Data Requirements

## System Name or Title

Next-Gen Restaurant Application Database System (NRADS)

**Purpose**

This section will describe the data requirements needed to facilitate the successful creation of the Next-Gen Restaurant Application (NRA) as defined in the Software Requirements Specification (SRS).

**Outcome**

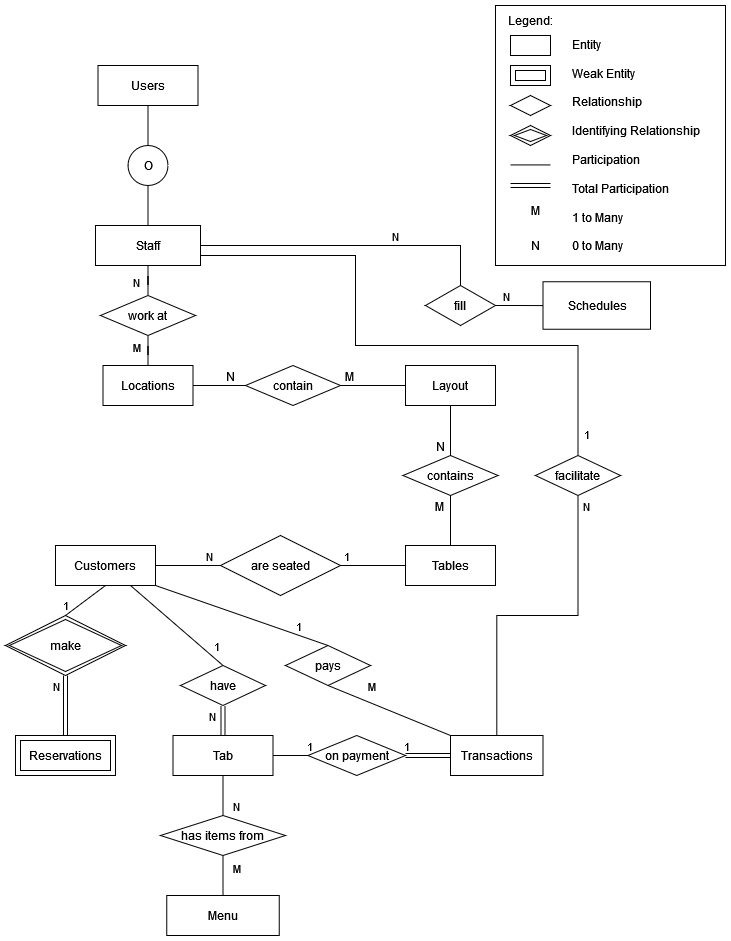
This section will define the data entities and their attributes needed to support the NRA.

## Core requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Requirement | Referenced page in SRS | Referenced Section in SRS | Referenced Paragraph in Section |
| 1 | The system will store general restaurant information including store\_id, address, table layout file location (/home/user/layouts/), table\_occupancy, table\_availability, table\_type, table\_id, layout\_id | 3  5  10  11 | 1.2  2.2  3.5.2  3.5.3 | 1.2.1  2.2.1  2.2.9  all  3.5.3.6 |
| 2 | The system will store customer information including customer\_id, phone\_number, first\_name, last\_name, date\_of\_birth. | 3  5  10  11  13 | 1.2  2.2  3.5.3  3.5.3  5.3 | 1.2.2  2.2.2  3.5.3.2  3.5.3.3  3.5.3.4  5.3.1 |
| 3 | The system will store walk-in and reservation information including reservation\_id, customer\_id, party\_size, wait\_time, is\_walkin, expected\_start\_time, and actual\_start\_time.. | 3  5  5  10  11  12 | 1.2  2.1  2.2  3.5.3  3.5.3  3.5.6 | 1.2.2  3  2.2.2  2.2.7  2.2.9  3.5.3.1  3.5.3.7  3.5.3.8  all |
| 4 | The system will store staff (host/hostess, Servers, Kitchen Staff, Restaurant Management) information including staff\_id, date\_of\_birth, phone\_number, address, and availability. | 3  5  5  11 | 1.2  2.1  2.2  3.5.3 | 1.2.3  1.2.7  3  2.2.2  3.5.3.5 |
| 5 | The system will store transaction information including transaction\_id, staff\_id, customer\_id, tab\_id, total, tip, datetime, currency, payment\_method, and location. | 3  5  5 | 1.2  2.1  2.2 | 1.2.4  3  2.2.4 |
| 6 | The system will store menu information including menu\_item\_id, menu\_item\_description, menu\_item\_name, menu\_item\_ingredients, minimum\_age\_to\_order, price | 3  5  5  11 | 1.2  2.1  2.2  3.5.4 | 1.2.5  3  2.2.3  2.2.5  2.2.8  all |
| 7 | The system will store tab information including sales\_tax, menu\_item\_id, customer\_id, staff\_id, tab\_total, tab\_id, datetime, customer\_id, and tab\_limit\_amount | 5  9 | 2.2  3.5 | 2.2.5  all |
| 8 | The system will store schedule structure information including required\_days, time, and required\_staff\_levels, staff\_assignment. | 3  5  5 | 1.2  2.1  2.2 | 1.2.7  3  2.2.6 |
| 9 | The system will handle user authentication including username, password, account\_type. | 13 | 5 | 5.1  5.2 |
| 10 | The system will store log information on file system in /var/log/ | 11 | 3.5.5 | all |

# Milestone 2: Conceptual Design

## Diagram



## Assumptions and Constraints

1. Staff ‘is-a’ user.
2. Staff must work at at least one location.
3. A location can have any number of staff assigned to work.
4. A location can contain different layouts.
5. A layout must belong to a location.
6. Staff can fill any number of schedule slots (or none).
7. A schedule can have any number of staff assigned to work.
8. A layout must contain at least one table.
9. A table can belong to any number of layouts.
10. A customer can only be seated at one table.
11. A table can have any number of customers seated at it.
12. A customer can make a reservation but does not have to make a reservation.
13. A reservation must belong to a customer.
14. A tab must belong to a customer.
15. A tab must be compiled of at least one menu item.
16. A menu item can be associated with any number of tabs.
17. A customer can have any number of tabs open.
18. A customer pays to create a transaction.
19. A transaction must close out a tab.
20. A tab can only have one transaction.
21. A staff member must facilitate a transaction.
22. Staff can facilitate any number of transactions.

# Milestone 3: Logical Design

## Functional Dependencies

**Entity name**: Users

**Attributes**:

user\_id, name, position, access\_level

**Functional dependencies**:

user\_id → name, position, access\_level

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes not in FD** | **Attributes on the left** | **Attributes on both sides** | **Attributes on the right side** |
|  | user\_id |  | name, postion, access\_level |

**Attribute closures**:

user\_id+ = user\_id, name, position, access\_level

**Unique keys**: user\_id

**Entity name**: Staff

**Attributes**:

staff\_id, user\_id, date\_of\_birth, phone\_number, address1, address2, city, state, availability, is\_active, location\_id

**Functional dependencies**:

staff\_id → user\_id, date\_of\_birth, phone\_number, address1, address2, city, state, availability, name, is\_active, location\_id

user\_id → staff\_id, date\_of\_birth, phone\_number, address1, address2, city, state, availability, name, is\_active, location\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes not in FD** | **Attributes on the left** | **Attributes on both sides** | **Attributes on the right side** |
|  |  | staff\_id, user\_id | date\_of\_birth, phone\_number, address1, address2, city, state, availability, name, is\_active, location\_id |

**Attribute closures** (if any):

user\_id+ = staff\_id, user\_id, date\_of\_birth, phone\_number, address1, address2, city, state, availability, name, is\_active, location\_id

staff\_id+ = staff\_id, user\_id, date\_of\_birth, phone\_number, address1, address2, city, state, availability, name, is\_active, location\_id

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

staff\_id\*

user\_id

Both can be used, but because a user can exist without being staff, we will just use staff\_id to be explicit as possible.

**Entity name**: Scheduled

**Attributes**:

schedule\_id, staff\_id

**Functional dependencies**:

None

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes not in FD** | **Attributes on the left** | **Attributes on both sides** | **Attributes on the right side** |
| schedule\_id, staff\_id |  |  |  |

**Attribute closures** (if any):

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

schedule\_id, staff\_id (super key)

**Entity name**: Works\_At

**Attributes**:

staff\_id, location\_id

**Functional dependencies**:

none

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes not in FD** | **Attributes on the left** | **Attributes on both sides** | **Attributes on the right side** |
| staff\_id, location\_id |  |  |  |

**Attribute closures** (if any):

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

staff\_id, location\_id (super key)

**Entity name**: Locations

**Attributes**:

store\_id, address1, address2, city, state, layout\_id

**Functional dependencies**:

store\_id → address1, address2, city, state, layout\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes not in FD** | **Attributes on the left** | **Attributes on both sides** | **Attributes on the right side** |
|  | store\_id |  | address1, address2, city, state, layout\_id |

**Attribute closures** (if any):

store\_id+ = store\_id, address1, address2, city, state, layout\_id

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

store\_id

**Entity name**: Schedules

**Attributes**:

schedule\_id, required\_start\_time, required\_end\_time, position, filled\_by\_staff\_id

**Functional dependencies**:

schedule\_id → required\_start\_time, required\_end\_time, position, filled\_by\_staff\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes not in FD** | **Attributes on the left** | **Attributes on both sides** | **Attributes on the right side** |
|  | schedule\_id |  | required\_start\_time, required\_end\_time, position, filled\_by\_staff\_id |

**Attribute closures** (if any):

schedule\_id+ = required\_start\_time, required\_end\_time, position, filled\_by\_staff\_id

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

schedule\_id

**Entity name**: Location\_Contains\_Layouts

**Attributes**:

location\_id, layout\_id, active

**Functional dependencies**:

location\_id, layout\_id → active

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes not in FD** | **Attributes on the left** | **Attributes on both sides** | **Attributes on the right side** |
|  | location\_id, layout\_id |  | active |

**Attribute closures** (if any):

location\_id, layout\_id+ = active

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

location\_id, layout\_id (super key)

**Entity name**: Layouts

**Attributes**:

layout\_id, layout\_description, layout\_filename, layout\_file\_type, table\_id

**Functional dependencies**:

layout\_id, table\_id → layout\_description, layout\_filename, layout\_file\_type

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes not in FD** | **Attributes on the left** | **Attributes on both sides** | **Attributes on the right side** |
|  | layout\_id, table\_id |  | layout\_description, layout\_filename, layout\_file\_type |

**Attribute closures** (if any):

layout\_id, table\_id+ = layout\_id, table\_id, layout\_description, layout\_filename, layout\_file\_type

(layout\_id, table\_id) is a superkey

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

layout\_id, table\_id

**Entity name**: Layout\_Contains\_Tables

**Attributes**:

layout\_id, table\_id

**Functional dependencies**:

none

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes not in FD** | **Attributes on the left** | **Attributes on both sides** | **Attributes on the right side** |
| layout\_id, table\_id |  |  |  |

**Attribute closures** (if any):

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

layout\_id, table\_id (super key)

**Entity name**: Tables

**Attributes**:

table\_id, table\_occupancy, table\_type, table\_notes

**Functional dependencies**:

table\_id → table\_occupancy, table\_type, table\_notes

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes not in FD** | **Attributes on the left** | **Attributes on both sides** | **Attributes on the right side** |
|  | table\_id |  | table\_occupancy, table\_type, table\_notes |

**Attribute closures** (if any):

table\_id+ = table\_id, table\_occupancy, table\_type, table\_notes

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

table\_id

**Entity name**: Customers

**Attributes**:

customer\_id, date\_of\_birth, phone\_number, first\_name, last\_name, address1, address2, city, state, table\_id

**Functional dependencies**:

customer\_id → date\_of\_birth, phone\_number, first\_name, last\_name, address1, address2, city, state, table\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes not in FD** | **Attributes on the left** | **Attributes on both sides** | **Attributes on the right side** |
|  | customer\_id |  | date\_of\_birth, phone\_number, first\_name, last\_name, address1, address2, city, state, table\_id |

**Attribute closures** (if any):

customer\_id+ = customer\_id, date\_of\_birth, phone\_number, first\_name, last\_name, address1, address2, city, state, table\_id

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

customer\_id

**Entity name**: Reservations

**Attributes**:

reservation\_id, customer\_id, party\_size, wait\_time, is\_walkin, expected\_start\_time, actual\_start\_time, arrival\_time, location\_id

**Functional dependencies**:

reservation\_id → customer\_id, party\_size, wait\_time, is\_walkin, expected\_start\_time, actual\_start\_time, arrival\_time

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes not in FD** | **Attributes on the left** | **Attributes on both sides** | **Attributes on the right side** |
|  | reservation\_id |  | customer\_id, party\_size, wait\_time, is\_walkin, expected\_start\_time, actual\_start\_time, arrival\_time , location\_id |

**Attribute closures** (if any):

reservation\_id+ = reservation\_id, customer\_id, party\_size, wait\_time, is\_walkin, expected\_start\_time, actual\_start\_time, arrival\_time, location\_id

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

reservation\_id

**Entity name**: Tabs

**Attributes**:

tab\_id, sales\_tax, menu\_item\_id, customer\_id, staff\_id, datetime, tab\_limit\_amount

**Functional dependencies**:

tab\_id, menu\_item\_id → sales\_tax, customer\_id, staff\_id, datetime, tab\_limit\_amount

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes not in FD** | **Attributes on the left** | **Attributes on both sides** | **Attributes on the right side** |
|  | tab\_id, menu\_item\_id |  | sales\_tax, customer\_id, staff\_id, datetime, tab\_limit\_amount |

**Attribute closures** (if any):

tab\_id, menu\_item\_id+ = tab\_id, sales\_tax, menu\_item\_id, customer\_id, staff\_id, datetime, tab\_limit\_amount

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

tab\_id, menu\_item\_id

**Entity name**: Tab\_Has\_Menu\_Item

**Attributes**:

tab\_id, menu\_item\_id, quantity

**Functional dependencies**:

tab\_id, menu\_item\_id → quantity

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes not in FD** | **Attributes on the left** | **Attributes on both sides** | **Attributes on the right side** |
|  | tab\_id, menu\_item\_id |  | quantity |

**Attribute closures** (if any):

tab\_id, menu\_item\_id += tab\_id, menu\_item\_id, quantity

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

tab\_id, menu\_item\_id (super key)

**Entity name**: Menu

**Attributes**:

menu\_item\_id, menu\_item\_description, menu\_item\_name, menu\_item\_ingredients, minimum\_age\_to\_order, price

**Functional dependencies**:

menu\_item\_id → menu\_item\_description, menu\_item\_name, menu\_item\_ingredients, minimum\_age\_to\_order, price

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes not in FD** | **Attributes on the left** | **Attributes on both sides** | **Attributes on the right side** |
|  | menu\_item\_id |  | menu\_item\_description, menu\_item\_name, menu\_item\_ingredients, minimum\_age\_to\_order, price |

**Attribute closures** (if any):

menu\_item\_id+ = menu\_item\_description, menu\_item\_name, menu\_item\_ingredients, minimum\_age\_to\_order, price

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

menu\_item\_id

**Entity name**: Transactions

**Attributes**:

transaction\_id, tab\_id, tip, datetime, currency, payment\_method, location

**Functional dependencies**:

transaction\_id → staff\_id, customer\_id, tab\_id, total, tip, datetime, currency, payment\_method, location

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes not in FD** | **Attributes on the left** | **Attributes on both sides** | **Attributes on the right side** |
|  | transaction\_id |  | tab\_id, tip, datetime, currency, total, payment\_method, location |

**Attribute closures** (if any):

transaction\_id+ = transaction\_id, tab\_id, tip, datetime, currency, payment\_method, location, total

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

transaction\_id

**Entity name**: Address

**Attributes**:

address\_type\_id, type\_name, address1, address2, city, state, zip

**Functional dependencies**:

address\_type\_id → type\_name, address1, address2, city, state, zip

zip → city, state

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes not in FD** | **Attributes on the left** | **Attributes on both sides** | **Attributes on the right side** |
|  | address\_type\_id | zip | type\_name, address1, address2, city, state, zip |

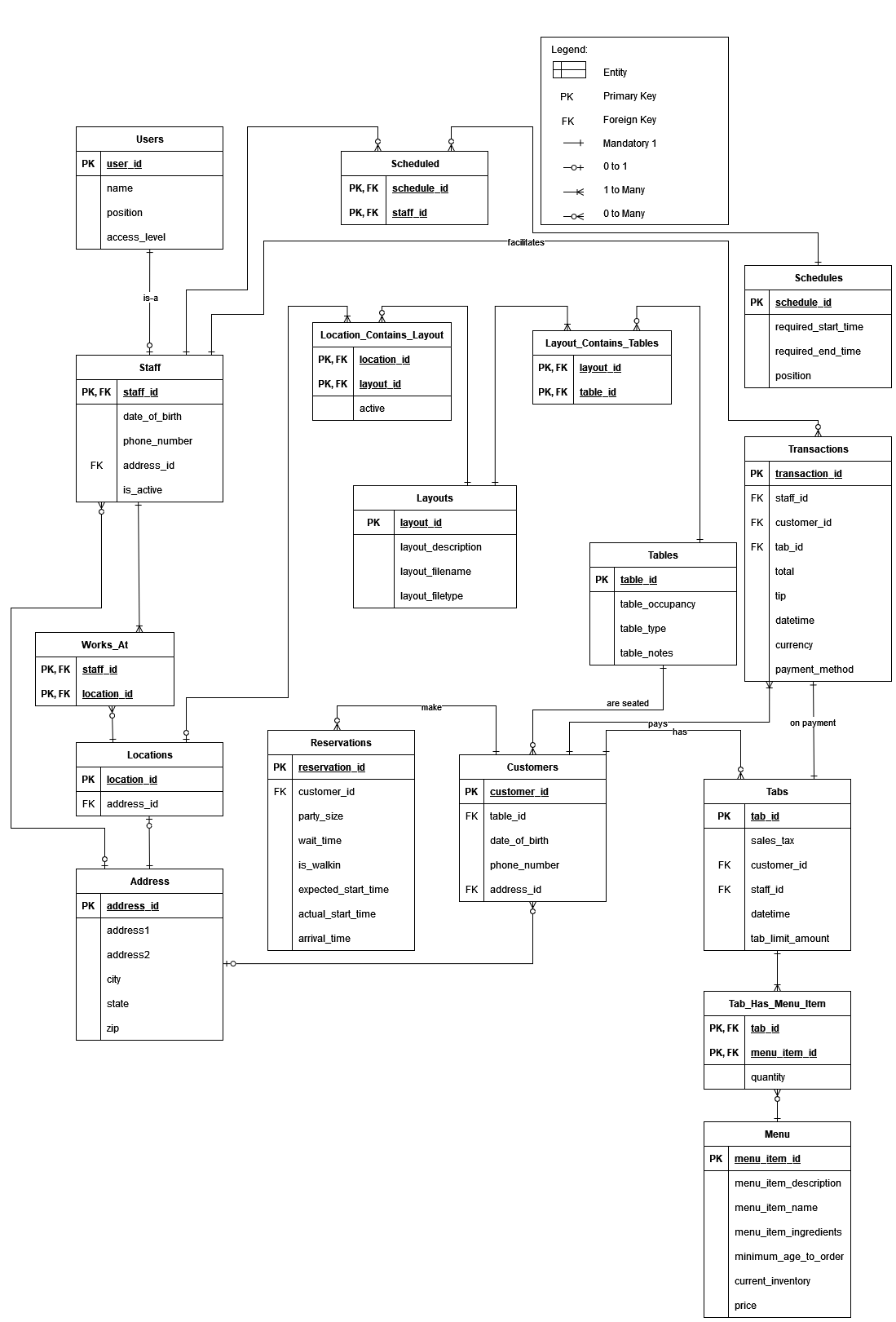
**Attribute closures** (if any):

address\_type\_id → address\_type\_id, type\_name, address1, address2, city, state, zip

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

address\_type\_id

## Diagram



## Assumptions and Constraints

1. Unless otherwise noted, functional dependencies that only return themselves were not written to keep the document concise.
2. Table availability can be computed based on which customers are seated at a given table.
3. Customer age can be calculated on the fly.
4. Tab total can be calculated on the fly.
5. The following tables were extrapolated from the relationship types as defined in the ERD
   1. Address
   2. Works\_At
   3. Location\_Contains\_Layout
   4. Layout\_Contains\_Tables
   5. Tab\_Has\_Menu\_Item

# 

# Milestone 4: Normalization

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ***Name of the table*** | ***Users*** | | | | |
|  | **Description** | A user is a person who has access to the system. This may include staff as a type of user who work in the restaurant, or others like system admins or auditors | | | | |
|  | **Attribute** | **Description** | | **Type** | **Examples of values** | **Notes** |
|  | user\_id | Unique identifier | | string | “9A9999” |  |
|  | first\_name | First name of user | | string | “John” | Cannot be Null |
|  | last\_name | Last name of user | | string | “Doe” | Cannot be Null |
|  | email | Email of user | | string | “john.doe@me.com” |  |
|  | position | Title of user in system | | “Kitchen Manager” | Can be null |  |
|  | access\_level | Level of access the user has to the system | | string | “admin” |  |
|  | **Functional Dependencies and Keys** | | | | | |
|  | **Functional dependencies** | user\_id → name, position, access\_level | | | | |
|  | **Candidate keys** | user\_id | | | | |
|  | **Normalization** | | | | | |
|  | **1NF** | **Yes** | Each cell contains an atomic value | | | |
|  | **2NF** | **Yes** | All non-key attributes are determined by the entire PK | | | |
|  | **3NF** | **Yes** | Table is in BCNF | | | |
|  | **BCNF** | **Yes** | There is no transitive dependency between non-key attributes (Attributes are not dependent on anything but user\_id) | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ***Name of the table*** | ***Staff*** | | | | |
|  | **Description** | Staff is a user of the system who regularly use system for its intended purpose | | | | |
|  | **Attribute** | **Description** | | **Type** | **Examples of values** | **Notes** |
|  | staff\_id | Unique identifier | | string | “9A9999” | FK to User |
|  | date\_of\_birth | DOB of staff member | | string | “2024-02-29” |  |
|  | phone\_number | Phone number of staff member | | string | “555-555-5555” |  |
|  | address\_id | FK to Address | | integer | 123456 |  |
|  | is\_active | Is the staff member currently an employee | | boolean |  |  |
|  | **Functional Dependencies and Keys** | | | | | |
|  | **Functional dependencies** | staff\_id→ date\_of\_birth, phone\_number, address\_id, is\_active | | | | |
|  | **Candidate keys** | staff\_id | | | | |
|  | **Normalization** | | | | | |
|  | **1NF** | **Yes** | Each cell contains an atomic value | | | |
|  | **2NF** | **Yes** | All non-key attributes are determined by the entire PK | | | |
|  | **3NF** | **Yes** | Table is in BCNF | | | |
|  | **BCNF** | **Yes** | There is no transitive dependency between non-key attributes (Attributes are not dependent on anything but staff\_id) | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ***Name of the table*** | ***Works\_At*** | | | | |
|  | **Description** | Maintains the location of which an employee is employed at | | | | |
|  | **Attribute** | **Description** | | **Type** | **Examples of values** | **Notes** |
|  | staff\_id | FK to User | | string | “9A9999” |  |
|  | location\_id | FK to Locations | | integer | 123456 |  |
|  | **Functional Dependencies and Keys** | | | | | |
|  | **Functional dependencies** | none | | | | |
|  | **Candidate keys** | staff\_id, location\_id | | | | |
|  | **Normalization** | | | | | |
|  | **1NF** | **Yes** | Each cell contains an atomic value | | | |
|  | **2NF** | **Yes** | All non-key attributes are determined by the entire PK (There are no non-key attributes) | | | |
|  | **3NF** | **Yes** | Table is in BCNF | | | |
|  | **BCNF** | **Yes** | There is no transitive dependency between non-key attributes (There are no non-key attributes) | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ***Name of the table*** | ***Locations*** | | | | |
|  | **Description** | Maintains the location of active stores | | | | |
|  | **Attribute** | **Description** | | **Type** | **Examples of values** | **Notes** |
|  | location\_id | Unique identifier | | integer | 123456 |  |
|  | address\_id | FK to Locations | | integer | 123456 |  |
|  | **Functional Dependencies and Keys** | | | | | |
|  | **Functional dependencies** | none | | | | |
|  | **Candidate keys** | location\_id, address\_id | | | | |
|  | **Normalization** | | | | | |
|  | **1NF** | **Yes** | Each cell contains an atomic value | | | |
|  | **2NF** | **Yes** | All non-key attributes are determined by the entire PK (There are no non-key attributes) | | | |
|  | **3NF** | **Yes** | Table is in BCNF | | | |
|  | **BCNF** | **Yes** | There is no transitive dependency between non-key attributes (There are no non-key attributes) | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ***Name of the table*** | ***Address*** | | | | |
|  | **Description** | Maintains a lookup of addresses | | | | |
|  | **Attribute** | **Description** | | **Type** | **Examples of values** | **Notes** |
|  | address\_id | Unique identifier | | integer | 123456 |  |
|  | address1 | First address line | | string | 42 Wallaby Way |  |
|  | address2 | Second address line | | string | APT 1 |  |
|  | city | City name | | string | Boston |  |
|  | state | State abbreviation | | string | MA |  |
|  | zip | Zip code digits | | string | “123456” | Uses string because some internal code use letters |
|  | **Functional Dependencies and Keys** | | | | | |
|  | **Functional dependencies** | address\_id→ address1, address2, city, state, zip  zip → city, state | | | | |
|  | **Candidate keys** | address\_id, zip | | | | |
|  | **Normalization** | | | | | |
|  | **1NF** | **Yes** | Each cell contains an atomic value | | | |
|  | **2NF** | **No** | The city and state attributes can be derived from zip, which is not a PK | | | |
|  | **3NF** | **No** | Not in BCNF | | | |
|  | **BCNF** | **No** | Not in 2NF | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ***Name of the table*** | ***Scheduled*** | | | | |
|  | **Description** | Links the staff to a schedule | | | | |
|  | **Attribute** | **Description** | | **Type** | **Examples of values** | **Notes** |
|  | schedule\_id | FK to Schedules | | integer | 123456 |  |
|  | staff\_id | FK to Staff | | integer | 123456 |  |
|  | **Functional Dependencies and Keys** | | | | | |
|  | **Functional dependencies** | none | | | | |
|  | **Candidate keys** | schedule\_id, staff\_id | | | | |
|  | **Normalization** | | | | | |
|  | **1NF** | **Yes** | Each cell contains an atomic value | | | |
|  | **2NF** | **Yes** | All non-key attributes are determined by the entire PK (There are no non-key attributes) | | | |
|  | **3NF** | **Yes** | Table is in BCNF | | | |
|  | **BCNF** | **Yes** | There is no transitive dependency between non-key attributes (There are no non-key attributes) | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ***Name of the table*** | ***Schedules*** | | | | |
|  | **Description** | Maintains a collection of schedule positions to be filled | | | | |
|  | **Attribute** | **Description** | | **Type** | **Examples of values** | **Notes** |
|  | schedule\_id | Unique identifier | | integer | 123456 |  |
|  | required\_start\_time | Day and time of shift start | | string | '2024-02-27T20:03:56.002Z' |  |
|  | required\_end\_time | Day and time of shift end | | string | '2024-02-27T20:04:56.002Z' |  |
|  | position | Position to be filled | | string | ‘host’ |  |
|  | **Functional Dependencies and Keys** | | | | | |
|  | **Functional dependencies** | schedule\_id → required\_start\_time, required\_end\_time, position | | | | |
|  | **Candidate keys** | schedule\_id | | | | |
|  | **Normalization** | | | | | |
|  | **1NF** | **Yes** | Each cell contains an atomic value | | | |
|  | **2NF** | **Yes** | All non-key attributes are determined by the entire PK | | | |
|  | **3NF** | **Yes** | Table is in BCNF | | | |
|  | **BCNF** | **Yes** | There is no transitive dependency between non-key attributes (Attributes are not dependent on anything but schedule\_id) | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ***Name of the table*** | ***Location\_Contains\_Layout*** | | | | |
|  | **Description** | Maintains the relationship between a Location and its Layouts. | | | | |
|  | **Attribute** | **Description** | | **Type** | **Examples of values** | **Notes** |
|  | location\_id | FK to Locations | | integer | 123456 |  |
|  | layout\_id | FK to Layouts | | integer | 123456 |  |
|  | active | Notes if the layout is currently active | | boolean | true |  |
|  | **Functional Dependencies and Keys** | | | | | |
|  | **Functional dependencies** | location\_id, layout\_id → active | | | | |
|  | **Candidate keys** | location\_id, layout\_id | | | | |
|  | **Normalization** | | | | | |
|  | **1NF** | **Yes** | Each cell contains an atomic value | | | |
|  | **2NF** | **Yes** | All non-key attributes are determined by the entire PK | | | |
|  | **3NF** | **Yes** | Table is in BCNF | | | |
|  | **BCNF** | **Yes** | There is no transitive dependency between non-key attributes (Attributes are not dependent on anything but [location\_id, layout\_id]) | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ***Name of the table*** | ***Layout\_Contains\_Tables*** | | | | |
|  | **Description** | Maintains the relationship between a Layout and its Tables. | | | | |
|  | **Attribute** | **Description** | | **Type** | **Examples of values** | **Notes** |
|  | table\_id | FK to Tables | | integer | 123456 |  |
|  | layout\_id | FK to Layouts | | integer | 123456 |  |
|  | **Functional Dependencies and Keys** | | | | | |
|  | **Functional dependencies** | none | | | | |
|  | **Candidate keys** | table\_id, layout\_id | | | | |
|  | **Normalization** | | | | | |
|  | **1NF** | **Yes** | Each cell contains an atomic value | | | |
|  | **2NF** | **Yes** | All non-key attributes are determined by the entire PK (There are no non-key attributes) | | | |
|  | **3NF** | **Yes** | Table is in BCNF | | | |
|  | **BCNF** | **Yes** | There is no transitive dependency between non-key attributes (No non-key attributes) | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ***Name of the table*** | ***Layouts*** | | | | |
|  | **Description** | Maintains a collection of Layout information | | | | |
|  | **Attribute** | **Description** | | **Type** | **Examples of values** | **Notes** |
|  | layout\_id | Unique identifier | | integer | 123456 |  |
|  | layout\_description | Descriptor of the layout | | string | 'Booth and High Table setup' |  |
|  | layout\_filename | Name of the layout design file | | string | ‘six\_booth\_8\_high\_table’ |  |
|  | layout\_filetype | File extension of the file | | string | ‘.xml’ |  |
|  | **Functional Dependencies and Keys** | | | | | |
|  | **Functional dependencies** | layout\_id → layout\_description, layout\_filename, layout\_filetype | | | | |
|  | **Candidate keys** | layout\_id | | | | |
|  | **Normalization** | | | | | |
|  | **1NF** | **Yes** | Each cell contains an atomic value | | | |
|  | **2NF** | **Yes** | All non-key attributes are determined by the entire PK | | | |
|  | **3NF** | **Yes** | Table is in BCNF | | | |
|  | **BCNF** | **Yes** | There is no transitive dependency between non-key attributes (Attributes are not dependent on anything but layout\_id) | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ***Name of the table*** | ***Tables*** | | | | |
|  | **Description** | Maintains a collection of the different types of Tables | | | | |
|  | **Attribute** | **Description** | | **Type** | **Examples of values** | **Notes** |
|  | table\_id | Unique identifier | | integer | 123456 |  |
|  | table\_occupancy | Number of seats at the table | | integer | 6 |  |
|  | table\_type | Type of Table | | string | ‘Booth’ |  |
|  | table\_notes | Additional information on the table | | string | ‘Broken table leg' |  |
|  | **Functional Dependencies and Keys** | | | | | |
|  | **Functional dependencies** | table\_id → table\_occupancy, table\_type, table\_notes | | | | |
|  | **Candidate keys** | table\_id | | | | |
|  | **Normalization** | | | | | |
|  | **1NF** | **Yes** | Each cell contains an atomic value | | | |
|  | **2NF** | **Yes** | All non-key attributes are determined by the entire PK | | | |
|  | **3NF** | **Yes** | Table is in BCNF | | | |
|  | **BCNF** | **Yes** | There is no transitive dependency between non-key attributes (Attributes are not dependent on anything but table\_id) | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ***Name of the table*** | ***Customers*** | | | | |
|  | **Description** | Maintains a collection of the customer information | | | | |
|  | **Attribute** | **Description** | | **Type** | **Examples of values** | **Notes** |
|  | customer\_id | Unique identifier | | integer | 123456 |  |
|  | table\_id | Table the customer is sitting at | | integer | 123456 | FK to Tables |
|  | location\_id | Which location are they visiting | | integer | 123456 | FK to Locations |
|  | date\_of\_birth | DOB of customer | | string | ‘2024-02-27’ |  |
|  | phone\_number | Customer phone number | | string | ‘555-555-5555' |  |
|  | address\_id | Address of customer | | integer | 123456 | FK to Addresses |
|  | **Functional Dependencies and Keys** | | | | | |
|  | **Functional dependencies** | customer\_id → table\_id, date\_of\_birth, phone\_number, address\_id | | | | |
|  | **Candidate keys** | customer\_id | | | | |
|  | **Normalization** | | | | | |
|  | **1NF** | **Yes** | Each cell contains an atomic value | | | |
|  | **2NF** | **Yes** | All non-key attributes are determined by the entire PK | | | |
|  | **3NF** | **Yes** | Table is in BCNF | | | |
|  | **BCNF** | **Yes** | There is no transitive dependency between non-key attributes (Attributes are not dependent on anything but customer\_id) | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ***Name of the table*** | ***Reservations*** | | | | |
|  | **Description** | Maintains a collection of customer reservations (as well as walk-ins, basically a reservation that starts immediately) | | | | |
|  | **Attribute** | **Description** | | **Type** | **Examples of values** | **Notes** |
|  | reservation\_id | Unique identifier | | integer | 123456 |  |
|  | customer\_id | FK to Customers | | integer | 123456 |  |
|  | party\_size | Number of people in party | | integer | 5 |  |
|  | wait\_time | Expected wait time for customer party (in minutes) | | integer | 45 |  |
|  | is\_walkin | Is this a future reservation | | boolean | false |  |
|  | expected\_start\_time | Time reservation should start | | string | '2024-02-27T20:04:56.002Z’ |  |
|  | actual\_start\_time | Time reservation actually starts | | string | '2024-02-27T20:04:56.002Z' |  |
|  | arrival\_time | Time customer shows up | | string | '2024-02-27T20:04:56.002Z' |  |
|  | **Functional Dependencies and Keys** | | | | | |
|  | **Functional dependencies** | reservation\_id → customer\_id, party\_size, wait\_time, is\_walkin, expected\_start\_time, actual\_start\_time, arrival\_time | | | | |
|  | **Candidate keys** | reservation\_id | | | | |
|  | **Normalization** | | | | | |
|  | **1NF** | **Yes** | Each cell contains an atomic value | | | |
|  | **2NF** | **Yes** | All non-key attributes are determined by the entire PK | | | |
|  | **3NF** | **Yes** | Table is in BCNF | | | |
|  | **BCNF** | **Yes** | There is no transitive dependency between non-key attributes (Attributes are not dependent on anything but reservation\_id) | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ***Name of the table*** | ***Transactions*** | | | | |
|  | **Description** | Maintains a collection of transactional information | | | | |
|  | **Attribute** | **Description** | | **Type** | **Examples of values** | **Notes** |
|  | transaction\_id | Unique identifier | | integer | 123456 |  |
|  | staff\_id | Staff member who processed the transaction | | integer | 123456 | FK to Staff |
|  | customer\_id | Customer to whom the transaction belongs too | | integer | 123456 | FK to Customers |
|  | tab\_id | Tab of which the transaction is paying for | | integer | 123456 | FK to Tabs |
|  | total | Amount the transaction covers. Can be partial or total for a tab. Before tip | | float | 1.45 |  |
|  | tip | Additional amount for tip | | float | 1.45 |  |
|  | datetime | Date and time transaction was processed | | string | '2024-02-27T20:04:56.002Z' |  |
|  | currency | Currency code of payment | | string | ‘USD’ |  |
|  | payment\_method | Payment method | | string | ‘Cash’ |  |
|  | **Functional Dependencies and Keys** | | | | | |
|  | **Functional dependencies** | transaction\_id → staff\_id, customer\_id, tab\_id, total, tip, datetime, currency, datetime, payment\_method | | | | |
|  | **Candidate keys** | transaction\_id | | | | |
|  | **Normalization** | | | | | |
|  | **1NF** | **Yes** | Each cell contains an atomic value | | | |
|  | **2NF** | **Yes** | All non-key attributes are determined by the entire PK | | | |
|  | **3NF** | **Yes** | Table is in BCNF | | | |
|  | **BCNF** | **Yes** | There is no transitive dependency between non-key attributes (Attributes are not dependent on anything but transaction\_id) | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ***Name of the table*** | ***Tabs*** | | | | |
|  | **Description** | Maintains a collection information related to customer tab information | | | | |
|  | **Attribute** | **Description** | | **Type** | **Examples of values** | **Notes** |
|  | tab\_id | Unique Identifier | | integer | 123456 |  |
|  | sales\_tax | Sales tax applied to tab | | float | 2.65 |  |
|  | customer\_id | Customer who owns tab | | integer | 123456 | FK to Customers |
|  | staff\_id | Staff who manages tab | | integer | 123456 | FK to Staff |
|  | datetime | Day and time when tab was started | | string | '2024-02-27T20:04:56.002Z' |  |
|  | tab\_limit\_amount | Dollar amount of when to notify customer of tab value | | float | 2.65 |  |
|  | **Functional Dependencies and Keys** | | | | | |
|  | **Functional dependencies** | tab\_id → sales\_tax, customer\_id, staff\_id, datetime, tab\_limit\_amount | | | | |
|  | **Candidate keys** | tab\_id | | | | |
|  | **Normalization** | | | | | |
|  | **1NF** | **Yes** | Each cell contains an atomic value | | | |
|  | **2NF** | **Yes** | All non-key attributes are determined by the entire PK | | | |
|  | **3NF** | **Yes** | Table is in BCNF | | | |
|  | **BCNF** | **Yes** | There is no transitive dependency between non-key attributes (Attributes are not dependent on anything but tab\_id) | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ***Name of the table*** | ***Tab\_Has\_Menu\_Item*** | | | | |
|  | **Description** | Maintains the relationship between a Tab and the menu items on the tab | | | | |
|  | **Attribute** | **Description** | | **Type** | **Examples of values** | **Notes** |
|  | tab\_id | FK to Tabs | | integer | 123456 |  |
|  | menu\_item\_id | FK to Menu | | integer | 123456 |  |
|  | quantity | Notes amount of menu item purhcased | | integer | 2 |  |
|  | **Functional Dependencies and Keys** | | | | | |
|  | **Functional dependencies** | tab\_id, menu\_item\_id → quantity | | | | |
|  | **Candidate keys** | tab\_id, menu\_item\_id | | | | |
|  | **Normalization** | | | | | |
|  | **1NF** | **Yes** | Each cell contains an atomic value | | | |
|  | **2NF** | **Yes** | All non-key attributes are determined by the entire PK | | | |
|  | **3NF** | **Yes** | Table is in BCNF | | | |
|  | **BCNF** | **Yes** | There is no transitive dependency between non-key attributes (Attributes are not dependent on anything but [tab\_id, menu\_item\_id]) | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ***Name of the table*** | ***Menu*** | | | | |
|  | **Description** | Maintains a collection information of related to the items on the menu | | | | |
|  | **Attribute** | **Description** | | **Type** | **Examples of values** | **Notes** |
|  | menu\_item\_id | Unique Identifier | | integer | 123456 |  |
|  | menu\_item\_description | Human readable item description | | string | ‘Cheeseburger’ |  |
|  | menu\_item\_ingredients | All ingredient in item, separated by a comma | | string | ‘gluten, ground beef, butter’ |  |
|  | minimum\_age\_to\_order | Age restriction on ordering | | integer | 21 |  |
|  | price | Cost of menu item | | float | 1.66 |  |
|  | **Functional Dependencies and Keys** | | | | | |
|  | **Functional dependencies** | menu\_item\_id → menu\_item\_description, menu\_item\_ingredients, minimum\_age\_to\_order, current\_inventory, price | | | | |
|  | **Candidate keys** | menu\_item\_id | | | | |
|  | **Normalization** | | | | | |
|  | **1NF** | **Yes** | Each cell contains an atomic value | | | |
|  | **2NF** | **Yes** | All non-key attributes are determined by the entire PK | | | |
|  | **3NF** | **Yes** | Table is in BCNF | | | |
|  | **BCNF** | **Yes** | There is no transitive dependency between non-key attributes (Attributes are not dependent on anything but menu\_item\_id) | | | |

# Milestone 5: Physical Design

## Assumptions and Constraints

## The attribute “name” in the USERS table was implemented as first\_name and last\_name since the full name can be computed.

All ids must be positive so that they are more meaningful as identification.

The tables in the database will be created with the same relationships as depicted in the Logical Model shown earlier in the document. Any changes to relationships will be noted.

## Naming Conventions

The tables were created under the Schema provided in the docker image. The table names use underscores (“\_”) to denote different words, or snake case as it is commonly referred as.

## Tables

|  |  |
| --- | --- |
| Physical Design | USERS |
| Primary Key | user\_id |
| Foreign Keys | - |
| SQL Code | CREATE TABLE users (  user\_id INTEGER CONSTRAINT user\_id CHECK (user\_id > 0),  first\_name VARCHAR2(30) NOT NULL,  last\_name VARCHAR2(30) NOT NULL,  position VARCHAR(30) NOT NULL,  access\_level VARCHAR(15),  email VARCHAR(30),  CONSTRAINT user\_email UNIQUE (email),  PRIMARY KEY (user\_id)  ) |
| Count of records in the table | 28 |

|  |  |
| --- | --- |
| Physical Design | STAFF |
| Primary Key | staff\_id |
| Foreign Keys | staff\_id --> USERS.user\_id  address\_id --> ADDRESS.address\_id |
| SQL Code | CREATE TABLE staff (  staff\_id INTEGER CONSTRAINT staff\_id CHECK (staff\_id > 0),  date\_of\_birth VARCHAR(10) NOT NULL CONSTRAINT s\_dob CHECK (length(date\_of\_birth) = 10),  phone\_number VARCHAR(12) CONSTRAINT s\_pn CHECK (length(phone\_number) = 12),  address\_id INTEGER,  is\_active INTEGER,  PRIMARY KEY (staff\_id),  FOREIGN KEY (staff\_id) REFERENCES users(user\_id),  FOREIGN KEY (address\_id) REFERENCES address(address\_id)  ) |
| Count of records in the table | 21 |

|  |  |
| --- | --- |
| Physical Design | SCHEDULES |
| Primary Key | schedule\_id |
| Foreign Keys |  |
| SQL Code | CREATE TABLE schedules (  schedule\_id INTEGER CONSTRAINT schedule\_id CHECK (schedule\_id > 0),  required\_start\_time VARCHAR(24) NOT NULL CONSTRAINT s\_start\_time CHECK (length(required\_start\_time) = 24),  required\_end\_time VARCHAR(24) NOT NULL CONSTRAINT s\_end\_time CHECK (length(required\_end\_time) = 24),  position VARCHAR(30) NOT NULL,  PRIMARY KEY (schedule\_id)  ) |
| Count of records in the table | 25 |

|  |  |
| --- | --- |
| Physical Design | SCHEDULED |
| Primary Key | schedule\_id, staff\_id |
| Foreign Keys | schedule\_id --> SCHEDULES.schedule\_id  staff\_id --> STAFF.staff\_id |
| SQL Code | CREATE TABLE scheduled (  schedule\_id INTEGER CONSTRAINT scheduled\_id CHECK (schedule\_id > 0),  staff\_id INTEGER CONSTRAINT scheduled\_staff\_id CHECK (staff\_id > 0),  PRIMARY KEY (schedule\_id, staff\_id),  FOREIGN KEY(schedule\_id) REFERENCES schedules(schedule\_id),  FOREIGN KEY(staff\_id) REFERENCES staff(staff\_id)  ) |
| Count of records in the table | 24 |

|  |  |
| --- | --- |
| Physical Design | ADDRESS |
| Primary Key | address\_id |
| Foreign Keys |  |
| SQL Code | CREATE TABLE address (  address\_id INTEGER CONSTRAINT address\_id CHECK (address\_id > 0),  address1 VARCHAR(30),  address2 VARCHAR(30),  city VARCHAR(30),  state VARCHAR(30),  zip VARCHAR(15),  PRIMARY KEY (address\_id)  ) |
| Count of records in the table | 44 |

|  |  |
| --- | --- |
| Physical Design | LOCATIONS |
| Primary Key | location\_id |
| Foreign Keys | Address\_id --> ADDRESS.address\_id |
| SQL Code | CREATE TABLE locations (  location\_id INTEGER CONSTRAINT location\_id CHECK (location\_id > 0),  address\_id INTEGER CONSTRAINT locations\_address\_id CHECK (address\_id > 0),  PRIMARY KEY (location\_id),  FOREIGN KEY (address\_id) REFERENCES address(address\_id)  ) |
| Count of records in the table | 3 |

|  |  |
| --- | --- |
| Physical Design | LAYOUTS |
| Primary Key | layout\_id |
| Foreign Keys |  |
| SQL Code | CREATE TABLE layouts (  layout\_id INTEGER CONSTRAINT l\_layout\_id CHECK (layout\_id > 0),  layout\_description VARCHAR(50),  layout\_filename VARCHAR(30),  layout\_filetype VARCHAR(10),  PRIMARY KEY (layout\_id)  ) |
| Count of records in the table | 25 |

|  |  |
| --- | --- |
| Physical Design | WORKS\_AT |
| Primary Key | staff\_id, location\_id |
| Foreign Keys | staff\_id --> STAFF.staff\_id  location\_id --> LOCATIONS.location\_id |
| SQL Code | CREATE TABLE layouts (  layout\_id INTEGER CONSTRAINT l\_layout\_id CHECK (layout\_id > 0),  layout\_description VARCHAR(50),  layout\_filename VARCHAR(30),  layout\_filetype VARCHAR(10),  PRIMARY KEY (layout\_id)  ) |
| Count of records in the table | 20 |

|  |  |
| --- | --- |
| Physical Design | LOCATIONS\_CONTAINS\_LAYOUTS |
| Primary Key | layout\_id, location\_id |
| Foreign Keys | layout\_id --> LAYOUTS.layout\_id  location\_id --> LOCATIONS.location\_id |
| SQL Code | CREATE TABLE locations\_contains\_layouts (  location\_id INTEGER CONSTRAINT lcl\_location\_id CHECK (location\_id > 0),  layout\_id INTEGER CONSTRAINT lcl\_layout\_id CHECK (layout\_id > 0),  active INTEGER,  PRIMARY KEY (location\_id, layout\_id),  FOREIGN KEY (layout\_id) REFERENCES layouts(layout\_id),  FOREIGN KEY (location\_id) REFERENCES locations(location\_id)  ) |
| Count of records in the table | 25 |

|  |  |
| --- | --- |
| Physical Design | TABLES |
| Primary Key | table\_id |
| Foreign Keys |  |
| SQL Code | CREATE TABLE tables (  table\_id INTEGER CONSTRAINT t\_table\_id CHECK (table\_id > 0),  table\_occupancy INTEGER,  table\_type STRING,  table\_notes VARCHAR(50),  PRIMARY KEY (table\_id)  ) |
| Count of records in the table | 25 |

|  |  |
| --- | --- |
| Physical Design | LAYOUT\_CONTAINS\_TABLES |
| Primary Key | table\_id, layout\_id |
| Foreign Keys | table\_id --> TABLES.table\_id  layout\_id --> LAYOUTS.layout\_id |
| SQL Code | CREATE TABLE layout\_contains\_tables (  table\_id INTEGER CONSTRAINT lct\_table\_id CHECK (table\_id > 0),  layout\_id INTEGER CONSTRAINT lct\_layout\_id CHECK (layout\_id > 0),  PRIMARY KEY (table\_id, layout\_id),  FOREIGN KEY (table\_id) REFERENCES tables(table\_id),  FOREIGN KEY (layout\_id) REFERENCES layouts(layout\_id) |
| Count of records in the table | 27 |

|  |  |
| --- | --- |
| Physical Design | CUSTOMERS |
| Primary Key | customer\_id, |
| Foreign Keys | table\_id --> TABLES.table\_id  address\_id --> ADDRESS.layout\_id |
| SQL Code | CREATE TABLE customers (  customer\_id INTEGER CONSTRAINT c\_customer\_id CHECK (customer\_id > 0),  table\_id INTEGER CONSTRAINT c\_table\_id CHECK (table\_id > 0),  address\_id INTEGER CONSTRAINT c\_address\_id CHECK (address\_id > 0),  Location\_id INTEGER CONSTRAINT c\_location\_id CHECK (location\_id > 0),  date\_of\_birth VARCHAR(10) NOT NULL CONSTRAINT c\_dob CHECK (length(date\_of\_birth) = 10),  phone\_number VARCHAR(12) CONSTRAINT c\_pn CHECK (length(phone\_number) = 12),  PRIMARY KEY (customer\_id),  FOREIGN KEY (table\_id) REFERENCES tables(table\_id),  FOREIGN KEY (address\_id) REFERENCES address(address\_id)  FOREIGN KEY (location\_id) REFERENCES locations(location\_id)  ) |
| Count of records in the table | 25 |

|  |  |
| --- | --- |
| Physical Design | RESERVATIONS |
| Primary Key | reservation\_id, |
| Foreign Keys | customer\_id --> CUSTOMERS.customer\_id |
| SQL Code | CREATE TABLE reservations (  reservation\_id INTEGER CONSTRAINT r\_reservation\_id CHECK (reservation\_id > 0),  customer\_id INTEGER CONSTRAINT r\_customer\_id CHECK (customer\_id > 0),  location\_id INTEGER CONSTRAINT r\_location\_id CHECK (location\_id > 0),  party\_size INTEGER,  wait\_time INTEGER,  is\_walkin INTEGER,  expected\_start\_time VARCHAR(24) CONSTRAINT r\_expected\_start\_time CHECK (length(expected\_start\_time) = 24),  actual\_start\_time VARCHAR(24) CONSTRAINT r\_actual\_start\_time CHECK (length(actual\_start\_time) = 24),  arrival\_time VARCHAR(24) CONSTRAINT r\_arrival\_time CHECK (length(arrival\_time) = 24),  PRIMARY KEY (reservation\_id),  FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)  FOREIGN KEY (location\_id) REFERENCES locations(location\_id)  ) |
| Count of records in the table | 26 |

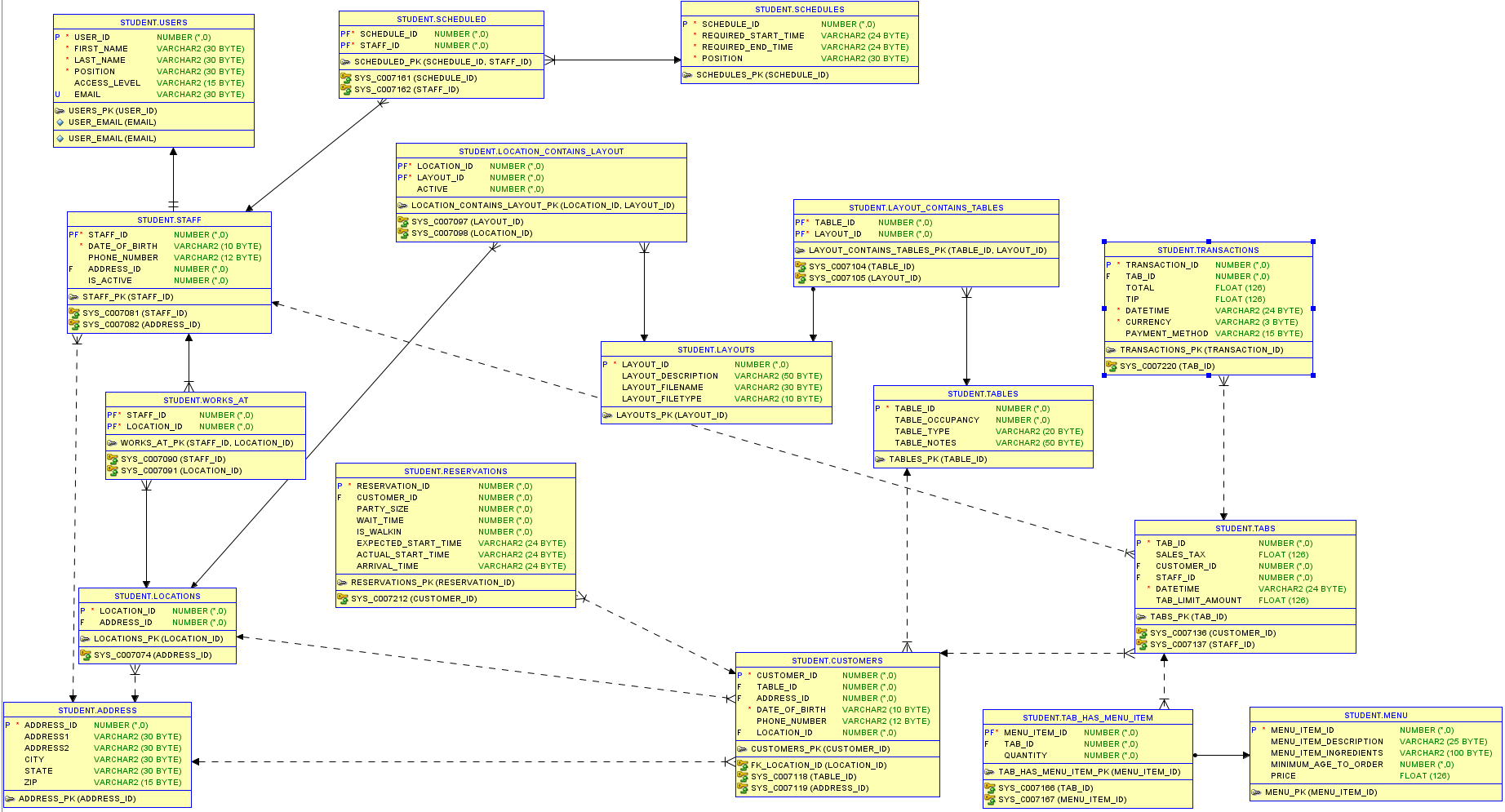
|  |  |
| --- | --- |
| Physical Design | TABS |
| Primary Key | tab\_id, |
| Foreign Keys | customer\_id --> CUSTOMERS.customer\_id  staff\_id --> STAFF.staff\_id |
| SQL Code | CREATE TABLE tabs (  tab\_id INTEGER CONSTRAINT t\_tab\_id CHECK (tab\_id > 0),  sales\_tax FLOAT,  customer\_id INTEGER CONSTRAINT t\_customer\_id CHECK (customer\_id > 0),  staff\_id INTEGER CONSTRAINT t\_staff\_id CHECK (staff\_id > 0),  datetime VARCHAR(24) NOT NULL CONSTRAINT t\_datetime CHECK (length(datetime) = 24),  tab\_limit\_amount FLOAT,  PRIMARY KEY (tab\_id),  FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id) ,  FOREIGN KEY (staff\_id) REFERENCES staff(staff\_id)  ) |
| Count of records in the table | 20 |

|  |  |
| --- | --- |
| Physical Design | TRANSACTIONS |
| Primary Key | transaction\_id, |
| Foreign Keys | tab\_id --> TABS.tab\_id |
| SQL Code | CREATE TABLE transactions (  transaction\_id INTEGER CONSTRAINT tr\_transaction\_id CHECK (transaction\_id > 0),  tab\_id INTEGER CONSTRAINT tr\_tab\_id CHECK (tab\_id > 0),  tip FLOAT,  total FLOAT,  datetime VARCHAR(24) NOT NULL CONSTRAINT tr\_datetime CHECK (length(datetime) = 24),  currency VARCHAR(3) NOT NULL CONSTRAINT tr\_currency CHECK (length(currency) = 3),  payment\_method VARCHAR(15),  PRIMARY KEY (transaction\_id),  FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id) ,  FOREIGN KEY (staff\_id) REFERENCES staff(staff\_id),  FOREIGN KEY (tab\_id) REFERENCES tabs(tab\_id)  ) |
| Count of records in the table | 20 |

|  |  |
| --- | --- |
| Physical Design | MENU |
| Primary Key | menu\_item\_id, |
| Foreign Keys |  |
| SQL Code | CREATE TABLE menu (  menu\_item\_id INTEGER CONSTRAINT m\_menu\_item\_id CHECK (menu\_item\_id > 0),  menu\_item\_description VARCHAR(25),  menu\_item\_ingredients VARCHAR(100),  minimum\_age\_to\_order INTEGER,  price FLOAT,  PRIMARY KEY (menu\_item\_id)  ) |
| Count of records in the table | 25 |

|  |  |
| --- | --- |
| Physical Design | TAB\_HAD\_MENU\_ITEM |
| Primary Key | menu\_item\_id, tab\_id |
| Foreign Keys | menu\_item\_id --> MENU.menu\_item\_id  tab\_id --> TABS.tab\_id |
| SQL Code | CREATE TABLE tab\_has\_menu\_items (  menu\_item\_id INTEGER CONSTRAINT thmi\_menu\_item\_id CHECK (menu\_item\_id > 0),  tab\_id INTEGER CONSTRAINT thmi\_tab\_id CHECK (tab\_id > 0),  quantity INTEGER,  PRIMARY KEY (menu\_item\_id),  FOREIGN KEY (tab\_id) REFERENCES tabs(tab\_id),  FOREIGN KEY (menu\_item\_id) REFERENCES menu(menu\_item\_id)  ) |
| Count of records in the table | 25 |

**SYSTEM GENERATED ERD**



# Milestone 6: SQL queries and

|  |  |
| --- | --- |
| **Query 1** | |
| **English version** | Return the data needed to display on a receipt including item quantity, price, sales tax, name, and total for each item ordered for a specific tab |
| **Source for the query need in the SRS document** | SRS document, page 5, section 2.2, fourth bullet |
| **SQL sentence** | SELECT tmi.tab\_id, m.menu\_item\_description, tmi.quantity, m.price, t.sales\_tax, tmi.quantity \* m.price \* (1 + t.sales\_tax / 100) AS total  FROM tab\_has\_menu\_item tmi  INNER JOIN menu m  ON m.menu\_item\_id = tmi.menu\_item\_id  INNER JOIN tabs t  on t.tab\_id = tmi.tab\_id  WHERE tmi.tab\_id = 732429; |
| **Example of returned rows (cropped screen caption)** |  |

|  |  |
| --- | --- |
| **Query 2** | |
| **English version** | Return a list of all the currently active employees to assist with schedule needs |
| **Source for the query need in the SRS document** | SRS document, page 3, section 1.2, number 7 |
| **SQL sentence** | SELECT u.user\_id, u.first\_name, u.last\_name, u.position  FROM users u  LEFT OUTER JOIN staff s  ON s.staff\_id = u.user\_id  WHERE s.is\_active = 1  ORDER BY u.position |
| **Example of returned rows (cropped screen caption)** |  |

|  |  |
| --- | --- |
| **Query 3** | |
| **English version** | Return a list of each position and the amount of each employee who have the given title to assist with scheduling needs |
| **Source for the query need in the SRS document** | SRS document, page 3, section 1.2, number 7 |
| **SQL sentence** | SELECT u.position, count(\*) as count  FROM users u  LEFT OUTER JOIN staff s on s.staff\_id = u.user\_id  WHERE s.is\_active = 1  GROUP BY u.position |
| **Example of returned rows (cropped screen caption)** |  |

|  |  |
| --- | --- |
| **Query 4** | |
| **English version** | Return a list of all the tabs that have not yet been paid for |
| **Source for the query need in the SRS document** | SRS document, page 5, section 2.2, bullet 5 |
| **SQL sentence** | SELECT t.tab\_id  FROM tabs t  WHERE t.tab\_id NOT IN (  SELECT tr.tab\_id  FROM transactions tr  )  ORDER BY t.tab\_id |
| **Example of returned rows (cropped screen caption)** |  |

|  |  |
| --- | --- |
| **Query 5** | |
| **English version** | Return a list of customers who have made more than one reservation. Helps with repeat customers for analytics |
| **Source for the query need in the SRS document** | Inspired by SRS document, page 5, section 2.2, bullets 8 and 9 |
| **SQL sentence** | SELECT c.customer\_id, count(\*) AS count  FROM customers c, reservations r  WHERE c.customer\_id = r.customer\_id  GROUP BY c.customer\_id  HAVING COUNT(r.customer\_id) > 1 |
| **Example of returned rows (cropped screen caption)** |  |

|  |  |
| --- | --- |
| **Query 6** | |
| **English version** | Return a count of parties that are waiting to be seated for both walkins and reservations. Additional optional filter for those parties who have been waiting longer than 30 minutes |
| **Source for the query need in the SRS document** | SRS document, page 3, section 1.2, number 2 |
| **SQL sentence** | SELECT r.is\_walkin, COUNT(\*) as count  FROM reservations r  WHERE r.wait\_time > 30  GROUP BY r.is\_walkin |
| **Example of returned rows (cropped screen caption)** |  |

|  |  |
| --- | --- |
| **Query 7** | |
| **English version** | Return a list of menu items and the total amount of times that it was ordered |
| **Source for the query need in the SRS document** | SRS document, page 5, section 2.2, bullet 8 |
| **SQL sentence** | SELECT mu.menu\_item\_description, COALESCE(SUM(mu.quantity), 0) as total\_sold  FROM (  SELECT m.menu\_item\_id, m.menu\_item\_description, tmi.quantity  FROM tab\_has\_menu\_item tmi  RIGHT OUTER JOIN menu m  ON tmi.menu\_item\_id = m.menu\_item\_id  ORDER BY tmi.menu\_item\_id  ) mu  GROUP BY mu.menu\_item\_description  ORDER BY total\_sold DESC |
| **Example of returned rows (cropped screen caption)** |  |