TIC2601 Database & Web Applications

AY 2025/26 Semester 1

Database Report: MakanTime

Group 4

Student Name	Administration Number
Chan Zi Yee	A0311192R
Cheng Wei Xian	A0311236U
Ginna Tai Yun Min	A0245465Y
Zheng Shao Bin	A0311223B
Jerin Paul	A0227170M

Table of Contents

1. Introduction

This report presents the database design for MakanTime, a restaurant reservation system that streamlines the booking process for customers and enables efficient reservation management for restaurant staff. The database is designed to support at least ten use cases, including restaurant discovery, booking management, seating plan visualization, and booking analytics.

2. ER Diagram and Mini-World Description

2.1 Mini-World Description

The database schema below supports the key use cases of the proposed restaurant reservation system for both administrators and customers. It consists of tables that capture essential entities such as restaurants, locations, bookings, menus, seating plans, reviews, and promotions, enabling effective reservation management, restaurant browsing, and table assignment functionalities.

There will be constraints on these fields to ensure consistency and target certain use cases.

- ***Restaurants:*** The restaurant table would store all the necessary information for each restaurant. This includes the name of the restaurant, cuisine of the restaurant, a brief write-up of the restaurant (description), as well as the address of the restaurant, which acts as foreign keys to the addresses table. This would help retrieve the necessary information easily.

- ***Addresses:*** The addresses table stores the location information of each restaurant.

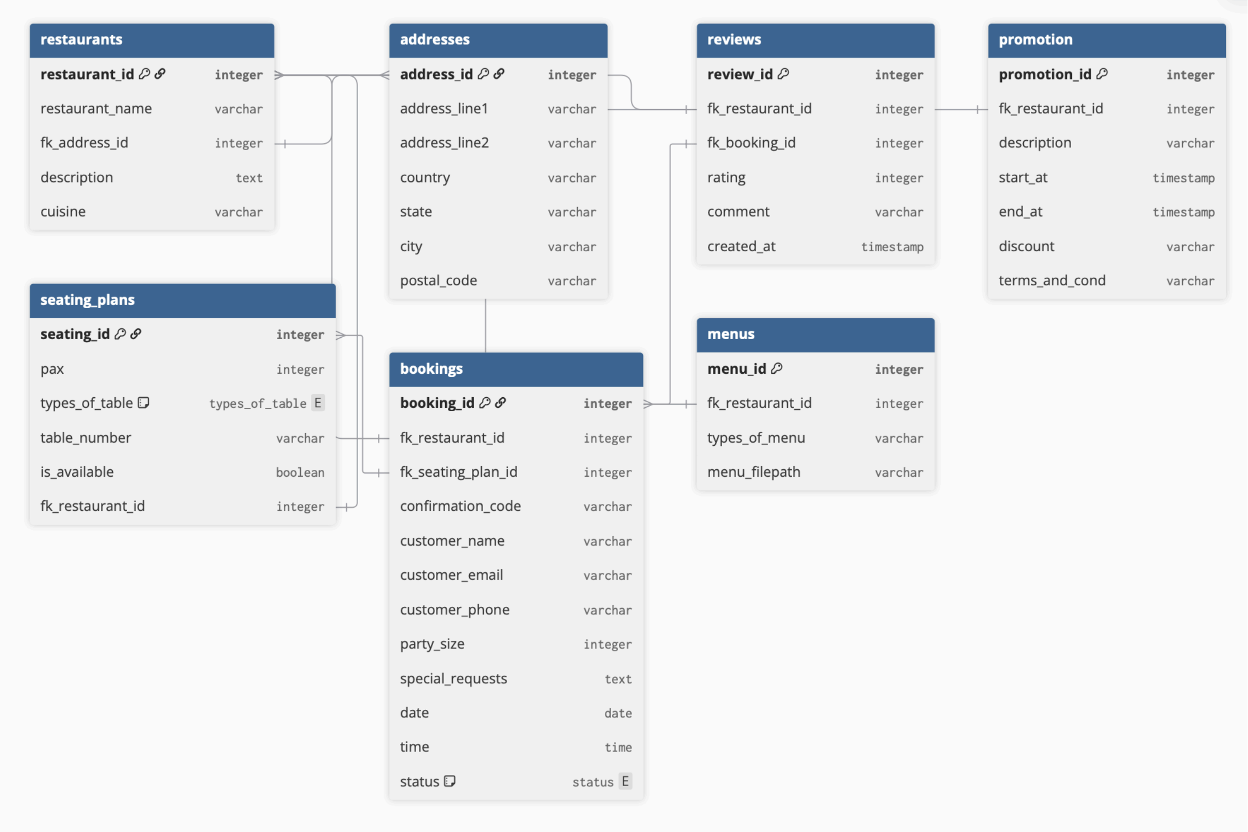
- ***Bookings:*** The bookings table stores reservation information for guest customers. Each booking captures the customer's contact details (name, email, phone), reservation details (date, time, party size), and a unique confirmation code for booking management. The confirmation code allows customers to view, modify, or cancel their reservations without requiring account creation or login. Customers can specify special requests such as dietary restrictions, occasions, or seating preferences. The status field tracks the booking lifecycle from pending confirmation through to completion or cancellation.
    - Status - confirmed, seated, completed, no_show, cancelled

- ***Menus:*** The menus table would store the menus for each restaurant. The types of menus would include dietary considerations like “halal” and “vegan”. This would be helpful in filtering and discovering restaurants through the types of menus a restaurant offers. This is to keep design simple and focused on reservation rather than ordering of items. The table would also store the file path of the menu image to be displayed in the website itself. The menu would be tied to each restaurant through the use of a foreign key (fk_restaurant_id).

- ***Seating Plans:*** The seating plans table stores individual table configurations for each restaurant. Each physical table is represented by a unique table_number (e.g., "T1", "T2", "OUT-1" for outdoor table 1, "VIP-1" for VIP table 1) along with its capacity (pax), type (VIP, indoor, outdoor), and availability status. This design allows restaurants with multiple tables of the same type and size to be accurately represented, enabling proper table assignment and preventing double-booking.
    - Types of tables - vip, indoor, outdoor

- ***Reviews:*** The reviews table would store the reviews for each restaurant. There would be a rating (out of 5) and a comment under each review. This is to help users make sound decisions in which restaurants to choose, to prevent others from making unjustified reviews, and the reviews would be tied to users who have made a booking before. And lastly, there would be a timestamp of when the review was made to help keep track of each review.

- ***Promotions:*** The promotions table would store the promotions each restaurant has. The table would include a simple description of what each promotion entails, a discount code, when it starts and ends, and the terms and conditions for each promotion.



restaurants	
restaurant_id	integer
restaurant_name	varchar
fk_address_id	integer
description	text
cuisine	varchar

addresses	
address_id	integer
address_line1	varchar
address_line2	varchar
country	varchar
state	varchar
city	varchar
postal_code	varchar

reviews	
review_id	integer
fk_restaurant_id	integer
fk_booking_id	integer
rating	integer
comment	varchar
created_at	timestamp

promotion	
promotion_id	integer
fk_restaurant_id	integer
description	varchar
start_at	timestamp
end_at	timestamp
discount	varchar
terms_and_cond	varchar

seating_plans	
seating_id	integer
pax	integer
types_of_table	types_of_table E
table_number	varchar
is_available	boolean
fk_restaurant_id	integer

bookings	
booking_id	integer
fk_restaurant_id	integer
fk_seating_plan_id	integer
confirmation_code	varchar
customer_name	varchar
customer_email	varchar
customer_phone	varchar
party_size	integer
special_requests	text
date	date
time	time
status	status E

menus	
menu_id	integer
fk_restaurant_id	integer
types_of_menu	varchar
menu_filepath	varchar

2.2 ER Diagrams

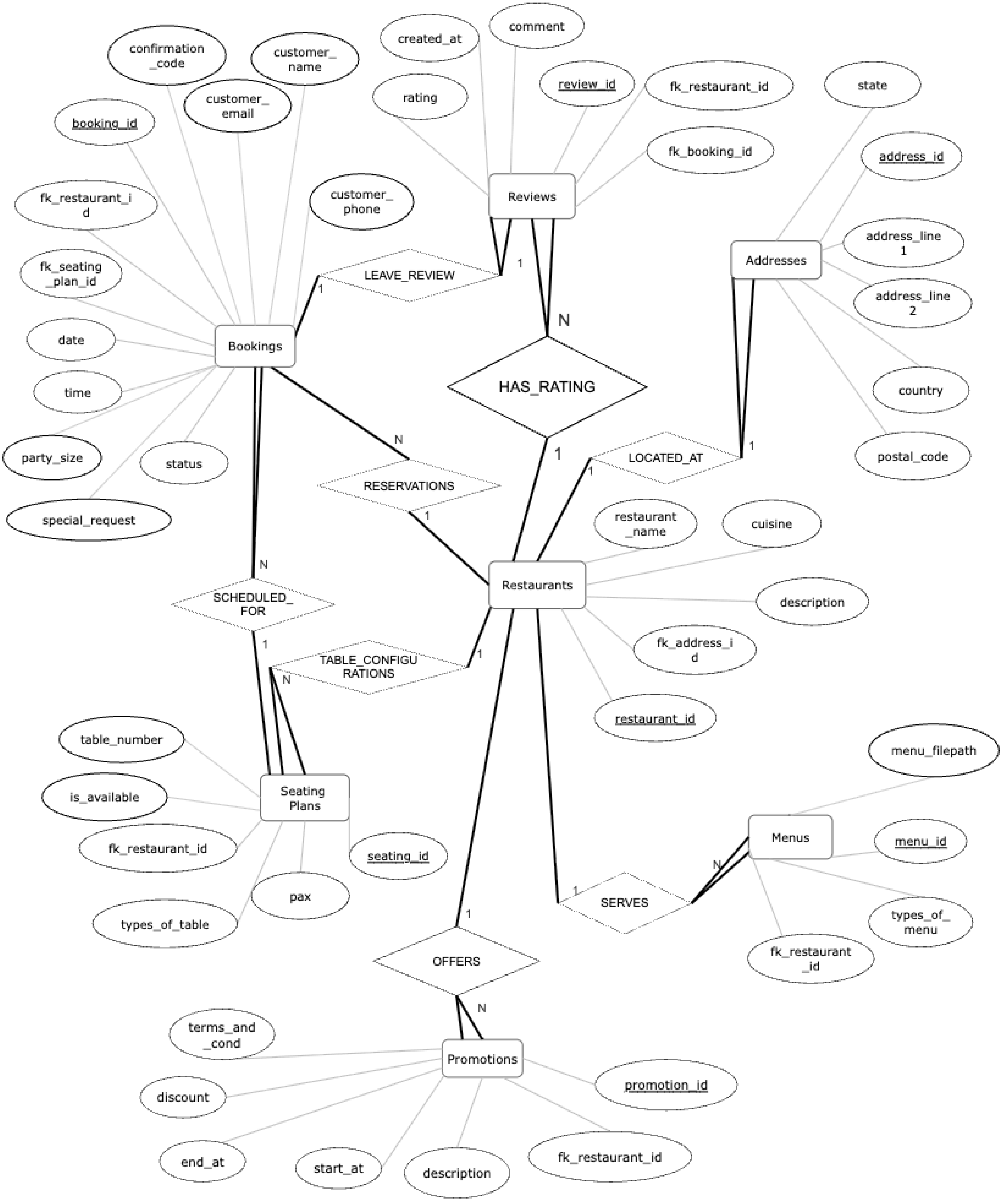
The database schema consists of seven interconnected tables that capture all essential entities and their relationships.

The provided database schema diagram illustrates the complete structure with the following key relationships:

- Restaurant ↔ Address (1:1): Each restaurant has exactly one physical address.
- Restaurant ↔ Menu (1:N): Each restaurant can have multiple menus.
- Restaurant ↔ Seating Plans (1:N): A restaurant can have multiple table configurations.
- Restaurant ↔ Bookings (1:N): A restaurant can have many bookings over time.
- Seating Plan ↔ Bookings (1:N): Each table can be booked multiple times.
- Booking ↔ Review (1:1): Each completed booking can have one associated review.
- Restaurant ↔ Promotions (1:N): A restaurant can run multiple promotional campaigns.
- Restaurant ↔ Review (1:N): A restaurant can have multiple reviews.

Furthermore, total participation is assumed for entities whose existence logically depends on their relationships.

- Each restaurant can have many types of menu.
- Every booking must specify a seating plan (e.g., table for 4). This ensures proper table allocation and supports restaurant capacity management.
- Each review must be linked to a booking, as reviews can only be submitted after a completed booking.
- Each promotion must be associated with a restaurant. Promotions are created by restaurants and cannot exist independently.
- Each restaurant must have at least one seating plan.
- Each restaurant must have an address.


confirmation_code
customer_name
customer_email
booking_id
customer_phone
fk_restaurant_id
fk_seating_plan_id
date
time
party_size
status
special_request
Bookings
created_at
comment
rating
review_id
fk_restaurant_id
fk_booking_id
Reviews
LEAVE_REVIEW
1
1
N
HAS_RATING
1
state
address_id
address_line 1
Addresses
address_line 2
country
postal_code
LOCATED_AT
1
1
N
RESERVATIONS
1
restaurant_name
cuisine
Restaurants
description
fk_address_id
restaurant_id
N
SCHEDULED_FOR
1
TABLE_CONFIGURATIONS
N
1
table_number
is_available
fk_restaurant_id
Seating Plans
seating_id
pax
types_of_table
menu_filepath
Menus
menu_id
N
SERVES
1
types_of_menu
fk_restaurant_id
1
OFFERS
N
terms_and_cond
Promotions
discount
promotion_id
end_at
start_at
description
fk_restaurant_id

3. Relational Database Schema Design

Based on the ER diagram, the following relational schema is proposed for implementation in SQLite. The schema includes seven tables with appropriate constraints, foreign keys (fk), and data types to ensure data integrity.

3.1 Table: restaurants

Column Name	Data Type	Constraints
restaurant_id	INTEGER	PRIMARY KEY, AUTOINCREMENT
restaurant_name	VARCHAR(255)	NOT NULL
fk_address_id	INTEGER	NOT NULL, FOREIGN KEY → address_id
description	TEXT	
cuisine	VARCHAR(100)	NOT NULL

3.2 Table: addresses

Column Name	Data Type	Constraints
address_id	INTEGER	PRIMARY KEY, AUTOINCREMENT
address_line1	VARCHAR(255)	NOT NULL
address_line2	VARCHAR(255)	
country	VARCHAR(100)	NOT NULL
state	VARCHAR(100)	
city	VARCHAR(100)	NOT NULL
postal_code	VARCHAR(20)	NOT NULL

3.3 Table: menus

Column Name	Data Type	Constraints
menu_id	INTEGER	PRIMARY KEY, AUTOINCREMENT
fk_restaurant_id	INTEGER	NOT NULL, FK → restaurant_id
types_of_menu	VARCHAR(50)	
menu_filepath	VARCHAR(50)	

3.4 Table: seating_plans

Column Name	Data Type	Constraints
seating_id	INTEGER	PRIMARY KEY, AUTOINCREMENT
pax	INTEGER	NOT NULL, CHECK (pax > 0)
types_of_table	VARCHAR(50)	CHECK IN (vip, indoor, outdoor)
fk_restaurant_id	INTEGER	NOT NULL, FK → restaurant_id
table_number	VARCHAR(10)	NOT NULL
is_available	BOOLEAN	DEFAULT 1

3.5 Table: bookings

Column Name	Data Type	Constraints
booking_id	INTEGER	PRIMARY KEY, AUTOINCREMENT
confirmation_code	VARCHAR(20)	UNIQUE, NOT NULL
customer_name	VARCHAR(100)	NOT NULL
customer_email	VARCHAR(100)	NOT NULL
customer_phone	VARCHAR(20)	NOT NULL
party_size	INTEGER	NOT NULL, CHECK (party_size > 0)
special_requests	TEXT	
fk_restaurant_id	INTEGER	NOT NULL, FK → restaurant_id
fk_seating_plan_id	INTEGER	NOT NULL, FK → seating_id
date	DATE	NOT NULL
time	TIME	NOT NULL
status	VARCHAR(20)	DEFAULT 'confirm', CHECK IN (pending, confirm, seated, completed, no_show, cancelled)

3.6 Table: reviews

Column Name	Data Type	Constraints
review_id	INTEGER	PRIMARY KEY, AUTOINCREMENT
fk_booking_id	INTEGER	NOT NULL, UNIQUE, FK → booking_id
rating	INTEGER	NOT NULL, CHECK (rating BETWEEN 1 AND 5)
comment	TEXT	
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

3.7 Table: promotions

Column Name	Data Type	Constraints
promotion_id	INTEGER	PRIMARY KEY, AUTOINCREMENT
fk_restaurant_id	INTEGER	NOT NULL, FK → restaurant_id
description	TEXT	NOT NULL
start_at	TIMESTAMP	NOT NULL
end_at	TIMESTAMP	NOT NULL, CHECK (end_at > start_at)
discount	VARCHAR(100)	
terms_and_cond	TEXT	

4. Normalization Strategy

4.1 First Normal Form (1NF)

All tables satisfy 1NF requirements with the following design considerations:

- Each column contains values of a single data type
- Each column has a unique name
- Primary keys are defined for all tables
- The order of rows is not significant

Design Trade-offs:

The **types_of_menu** field in the menus table stores comma-separated text (e.g., "dairy, gluten, nuts") to represent multiple dietary restrictions for a single menu item. While this approach is not strictly atomic according to 1NF principles, we chose this design for the following reasons:

1. **Simplicity:** A properly normalized solution would require two additional tables (allergens and menu_allergens junction table), adding significant complexity to the schema for a relatively simple feature.
2. **Query Simplicity:** The current design allows straightforward text-based searches using LIKE operators (e.g., WHERE types_of_menu LIKE '%gluten%'), which is sufficient for the filtering requirements in Use Case 1.

4.2 Second Normal Form (2NF)

All tables satisfy 2NF requirements by ensuring all non-key attributes are fully functionally dependent on the entire primary key. Since all our tables have single-column primary keys, partial dependencies are automatically prevented.

4.3 Third Normal Form (3NF)

The database design satisfies 3NF requirements by eliminating transitive dependencies. Key design decisions include:

- **Addresses Separation:** Restaurant location information is stored in a separate addresses table. This prevents transitive dependencies where attributes like city, state, and postal_code might depend on each other (e.g., postal_code → city → state) rather than depending directly on the primary key. By separating address information, we ensure that each non-key attribute depends only on address_id.
- **Menu Independence:** Type_of menu_depends only on their menu_id and their associated restaurant (fk_restaurant_id).
- **Seating Plans:** Each seating plan (physical table) depends only on its seating_id and restaurant association (fk_restaurant_id). The table's capacity (pax), type, and number are all directly related to that specific table, with no transitive dependencies.

- **Reviews Simplification:** Reviews are linked directly to bookings via fk_booking_id only. The restaurant information can be derived through the booking relationship (review → booking → restaurant).

4.4 Benefits of 3NF Design

- ***Data Integrity:*** Constraints and foreign keys prevent invalid data states.
- ***Reduced Redundancy:*** Information is stored once and referenced via foreign keys.
- ***Update Anomaly Prevention:*** Changes require updating only one record.
- ***Deletion Anomaly Prevention:*** Cascading deletes are properly managed.
- ***Insertion Anomaly Prevention:*** Entities can be added independently.

5. Use Case Mapping

This section outlines how the implemented database schema effectively enables and supports the functionalities required for each use case as described in the project proposal.

5.1 Customer Use Cases

Use Case 1: Discover & Filter Restaurants

This use case enables users to search and filter restaurants based on various criteria, including cuisine, dietary preferences, price range, customer ratings, location, and availability of promotions. The system leverages multiple related tables to support rich and dynamic filtering.

- The restaurant table stores searchable attributes (restaurant name, cuisine).
- The menu table enables filtering by types of menu.
- The reviews table with the rating field enables filtering by customer ratings.
- The addresses table stores searchable attributes (country, state, city).
- The promotions table enables filtering by whether the restaurant exists in the table.
- Complex filtering is achieved through the creation of SQL VIEWs that combine data from multiple related tables using JOIN operations and WHERE clauses.

Use Case 2: Book Reservation

- The bookings table captures all reservations and customer contact details.
- The party_size field ensures appropriate table allocation.
- The seating_plans table provides available table options based on capacity.
- The confirmation_code field enables booking management without login.

Use Case 3: Manage Reservation

- The confirmation_code field allows customers to access their bookings.
- Customers can search by email to find all their bookings.

- UPDATE operations modify booking details or status.

5.2 Administrator Use Cases

Use Case 4: View All Bookings

- The bookings table with the restaurant association enables viewing all reservations.
- Filtering by date, time, and status is supported through WHERE clauses.
- JOIN with seating_plans provides a complete booking context.

Use Case 5: Update Booking Status

- The status field with CHECK constraints ensures valid state transitions.
- Status values: confirm → seated → completed or no_show.
- UPDATE operations change booking status throughout the lifecycle.

Use Case 6: Manage Seating Plan

- The seating_plans table stores all table configurations.
- Admins can visualize the layout by querying all plans for their restaurant.
- Table assignment is achieved through UPDATE operations on bookings.
- The seating_plans table supports full CRUD operations to help admins plan and update their restaurant's layout.

Use Case 7: Edit Restaurant Information

- The restaurants table stores editable details (description, theme).
- The addresses table allows updating location information independently.

Use Case 8: View Booking Statistics

- Enables restaurant owners to view key business metrics through a simple dashboard in the admin panel. Insights are generated from historical data in the 'bookings' and 'reviews' tables using aggregate functions such as COUNT, AVG, and GROUP BY, along with advanced SQL techniques like Views, CASE statements, window functions, and joins across multiple tables.
- These statistics allow restaurant owners to effectively allocate staff during peak periods, anticipate stock levels, and adjust promotions to attract customers during off-peak hours.
  - ***Monthly booking trends:*** Show the number of bookings each month using a window function to identify seasonal demand patterns (e.g., year-end holidays).
  - ***Peak booking times:*** Show which time slots within each day receive the most bookings to prepare for high-traffic periods.
  - ***No-show and Cancellation Rate:*** Use a CASE statement to calculate the proportion of booking outcomes (e.g., show vs no-show), helping improve operational planning.

    - ***Average review rating over time:*** Track customer satisfaction by calculating the average rating per month, allowing detection of service quality drops and correlation with booking behavior (e.g., rising cancellations).

5.3 Additional Supported Use Cases

Use Case 9: Manage Promotions

- The promotions table stores all promotional campaigns.
- Date range constraints ensure valid promotion periods.
- Multiple active promotions can be queried and displayed.

Use Case 10: Submit and View Reviews

- Reviews link directly to bookings via fk_booking_id.
- Restaurant association is implicit (derived through booking).
- UNIQUE constraint ensures one review per booking.
- Rating constraints ensure valid values (1-5 stars).
- Timestamp tracking for review chronology.
- Average rating calculation through aggregate functions.

6. Conclusion

The MakanTime database design provides a robust, normalized foundation for a comprehensive restaurant reservation system. The schema consists of seven interconnected tables that support all required use cases while maintaining data integrity and minimizing redundancy.

Key Design Strengths

- ***Comprehensive Data Model:*** Captures all essential entities and their relationships for rich functionality.
- ***Normalization to 3NF:*** Eliminates redundancy and prevents data anomalies.
- ***Flexible Querying:*** Supports complex filtering and detailed analytics operations.
- ***Scalability:*** Accommodates future growth without fundamental restructuring.
- ***Data Integrity:*** CHECK constraints and foreign keys enforce business rules.

The MakanTime database provides a solid foundation for delivering a user-friendly, efficient restaurant reservation system that meets both customer needs and business objectives.