Data Dictionary for Restaurant Reservation System

Conventions:
- PK = Primary Key
- FK = Foreign Key
- NN = NOT NULL
- UQ = UNIQUE

Table Overview:

Table	Purpose
app_users	Users who make reservations like profiles or login fields
restaurants	Restaurant directory (name, address, metadata)
restaurant_schedules	Weekly open/close times per restaurant (one row per weekday)
reservations	Bookings connecting a user to a restaurant at a date/time

app_users:

Purpose: Store user identity information for making and viewing reservations

Name	Type	NN	Default	Constraints	Description
id	BIGINT IDENTITY	✓		PK	Surrogate key
full_name	VARCHAR(120)	✓			User's name
email	VARCHAR(120)	✓		UQ	Unique login/contact email
phone	VARCHAR(32)				Optional phone

					number
password_hash	VARCHAR(255)	✓			Hashed password
created_at	TIMESTAMP		CURRENT_TIMESTAMP		Row creation time

Indexes/Constraints:
- Uq_app_users_email -> ensures email is unique

Business Rules:
- Email must have a valid format
- Case-insensitive uniqueness policy can be applied at the application layer

restaurants:

Purpose: Stores restaurants that can be reserved by users

Name	Type	NN	Default	Constraints	Description
id	BIGINT IDENTITY	✓		PK	Surrogate key
name	VARCHAR(120)	✓			Restaurant name displayed to users
address	VARCHAR(255)				Street address of the restaurant
city	VARCHAR(100)				City where the restaurant is located
state	VARCHAR(32)				State/region code
created_at	TIMESTAM	CURRENT_			Creation time

	P	TIMESTAMP			
updated_at	TIMESTAMP				Last update timestamp

Business Rules:

- name is mandatory

restaurant_schedules:

Purpose: Defines open/close times per weekday for each restaurant

Name	Type	NN	Default	Constraints	Description
id	BIGINT IDENTITY	✓		PK	Surrogate key
restaurant_id	BIGINT	✓		FK -> restaurants.id	Parent restaurant
weekday	SMALLINT	✓		UQ with restaurants_id	0 = Monday … Sunday = 6
open_time	TIME	✓			Restaurant opening time
close_time	TIME	✓			Restaurant closing time

Indexes/Constraints:

- Uq_sched_rest_day -> ensures one record per weekday per restaurant

Business Rules:

- open_time < close_time must be true
- Schedule should cover all days (0-6) for complete restaurants

reservations:

Purpose: Stores all booking information connecting users and restaurants

Name	Type	NN	Default	Constraints	Description
id	BIGINT IDENTITY	✓		PK	Surrogate key
user_id	BIGINT	✓		FK -> app.users.id	User making the booking
restaurant_id	BIGINT	✓		FK -> restaurants.id	Restaurant being booked
reservation_at	TIMESTAMP	✓			Combined date/time of booking
party_size	INT	✓			Number of guest (>0)
status	VARCHAR(16)	✓			PENDING/CONFIRMED/CANCELLED
confirmation_code	VARCHAR(32)	✓		UQ	Confirmation code for booking
special_requests	TEXT				Optional notes(allergies, seat preference)
created_at	TIMESTAMP		CURRENT_TIMESTAMP		Time for booking was created

Indexes/Constraints:

- uq_res_confirmation -> unique confirmation code
- idx_res_user_time -> query by user and time

- idx_res_rest_time -> query by restaurant and time

Business Rules:

- Party_size > 0
- Reservation must reference valid user and restaurant
- Must be within restaurant open hours
- Time overlap prevention handled in service logic

Relationships:

- app_users (1) -> reservations (∞) via reservations.user_id
- restaurants (1) -> reservations (∞) via reservations.restaurant_id
- restaurants (1) -> restaurant_schedules (∞) via restaurant_schedules.restaurant_id

Example Queries:

User’s upcoming reservations:

```
SELECT r.id, r.reservation_at, r.party_size, r.status, res.name
FROM reservations r
JOIN restaurants res ON res.id = r.restaurant_id
WHERE r.user_id = :userId
  AND r.reservation_at >= CURRENT_TIMESTAMP
ORDER BY r.reservation_at;
```

Restaurant schedule (Monday -> Sunday):

```
SELECT weekday, open_time, close_time
FROM restaurant_schedules
WHERE restaurant_id = :restId
ORDER BY weekday;
```

Reservations in a date/time range:

```
SELECT *
FROM reservations
WHERE restaurant_id = :restId
  AND reservation_at BETWEEN :startTs AND :endTs
ORDER BY reservation_at;
```