-- Enable the uuid-ossp extension for UUID generation if not already enabled

CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

CREATE TABLE users (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(), -- Auto-generating unique UUID

email VARCHAR(255) NOT NULL UNIQUE, -- Email field with uniqueness constraint

password TEXT NOT NULL, -- Encrypted password

user\_type VARCHAR(50) NOT NULL, -- User type (e.g., admin, regular user)

created\_on TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, -- Auto-populates with the current timestamp

is\_active BOOLEAN DEFAULT TRUE, -- Indicates if the user is active

last\_login TIMESTAMP -- Tracks the last login time

);

———————————————————————————————————————————

CREATE TABLE user\_profiles (

uuid UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(), -- Unique identifier for each profile

first\_name VARCHAR(100) NOT NULL, -- First name of the user

last\_name VARCHAR(100) NOT NULL, -- Last name of the user

user\_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE, -- Foreign key referencing the users table

phone\_number VARCHAR(15) UNIQUE, -- Phone number with uniqueness constraint

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, -- Auto-populates on record creation

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP -- Auto-populates on record creation, updated via trigger

);

-- Create a function to automatically update the `updated\_at` column

CREATE OR REPLACE FUNCTION update\_updated\_at\_column()

RETURNS TRIGGER AS $$

BEGIN

NEW.updated\_at = CURRENT\_TIMESTAMP; -- Set `updated\_at` to the current time

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Create a trigger to call the function before any update

CREATE TRIGGER set\_updated\_at

BEFORE UPDATE ON user\_profiles

FOR EACH ROW

EXECUTE FUNCTION update\_updated\_at\_column();

———————————————————————————————————————————-

CREATE TABLE addresses (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(), -- Unique identifier for each address

user\_profile\_id UUID NOT NULL REFERENCES user\_profiles(uuid) ON DELETE CASCADE, -- Foreign key to user\_profiles

address\_line1 VARCHAR(255) NOT NULL, -- First line of the address

address\_line2 VARCHAR(255), -- Second line of the address (optional)

city VARCHAR(100) NOT NULL, -- City name

state VARCHAR(100) NOT NULL, -- State name

postal\_code VARCHAR(20) NOT NULL, -- Postal/ZIP code

country VARCHAR(100) NOT NULL, -- Country name

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, -- Auto-populates on record creation

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP -- Auto-updated via trigger

);

-- Create a function to update the `updated\_at` column

CREATE OR REPLACE FUNCTION update\_updated\_at\_column\_addresses()

RETURNS TRIGGER AS $$

BEGIN

NEW.updated\_at = CURRENT\_TIMESTAMP; -- Set `updated\_at` to the current time

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Create a trigger to auto-update the `updated\_at` column

CREATE TRIGGER set\_updated\_at\_addresses

BEFORE UPDATE ON addresses

FOR EACH ROW

EXECUTE FUNCTION update\_updated\_at\_column\_addresses();

———————————————————————————————————————————

CREATE TABLE restaurants (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

name VARCHAR(255) NOT NULL,

phone\_number BIGINT,

type VARCHAR(50),

description TEXT,

dollar\_signs SMALLINT,

pickup\_enabled BOOLEAN,

delivery\_enabled BOOLEAN,

is\_open BOOLEAN,

offers\_first\_party\_delivery BOOLEAN,

offers\_third\_party\_delivery BOOLEAN,

miles FLOAT,

weighted\_rating\_value FLOAT,

aggregated\_rating\_count INT

);  
  
-- Alter the restaurants table to add a foreign key reference to the users table

ALTER TABLE restaurants

ADD COLUMN user\_id UUID NOT NULL, -- Add the user\_id column to the restaurants table

ADD CONSTRAINT fk\_restaurant\_user

FOREIGN KEY (user\_id)

REFERENCES users(id) ON DELETE CASCADE; -- Reference the UUID column in users table

CREATE TABLE addresses (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

restaurant\_id UUID NOT NULL REFERENCES restaurants(id) ON DELETE CASCADE,

street\_addr VARCHAR(255),

street\_addr\_2 VARCHAR(255),

city VARCHAR(100),

state VARCHAR(100),

zipcode VARCHAR(20),

country VARCHAR(50),

latitude FLOAT,

longitude FLOAT

);

CREATE TABLE operating\_hours (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

restaurant\_id UUID NOT NULL REFERENCES restaurants(id) ON DELETE CASCADE,

type VARCHAR(50), -- e.g., operational, delivery, pickup, dine\_in

day VARCHAR(15), -- e.g., Monday, Tuesday

hours VARCHAR(50) -- e.g., "07:00AM - 09:00PM"

);

CREATE TABLE cuisines (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

restaurant\_id UUID NOT NULL REFERENCES restaurants(id) ON DELETE CASCADE,

cuisine VARCHAR(100)

);

CREATE TABLE photos (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

restaurant\_id UUID NOT NULL REFERENCES restaurants(id) ON DELETE CASCADE,

type VARCHAR(50), -- e.g., food\_photo, logo\_photo, store\_photo

url TEXT

);

———————————————————————————————————————————

CREATE TABLE menus (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

restaurant\_id UUID NOT NULL REFERENCES restaurants(id) ON DELETE CASCADE,

cuisine\_id UUID NOT NULL REFERENCES cuisines(id) ON DELETE CASCADE,

name VARCHAR(255) NOT NULL, -- e.g., "Breakfast Specials", "Italian Starters"

description TEXT, -- Optional description of the menu

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE menu\_items (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

menu\_id UUID NOT NULL REFERENCES menus(id) ON DELETE CASCADE,

name VARCHAR(255) NOT NULL, -- Name of the dish/item

description TEXT, -- Description of the dish/item

price NUMERIC(10, 2) NOT NULL, -- Price of the item

is\_available BOOLEAN DEFAULT TRUE, -- Indicates if the item is available

calories INT, -- Optional calorie count

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE menu\_item\_options (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

menu\_item\_id UUID NOT NULL REFERENCES menu\_items(id) ON DELETE CASCADE,

name VARCHAR(255) NOT NULL, -- e.g., "Extra Cheese", "Large Size"

price NUMERIC(10, 2) NOT NULL, -- Price for the option

is\_available BOOLEAN DEFAULT TRUE, -- Indicates if the option is available

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

———————————————————————————————————————————  
  
CREATE TABLE cart (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

user\_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE, -- Links cart to a user

status VARCHAR(50) DEFAULT 'active', -- Status: 'active', 'purchased', etc.

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE cart\_items (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

cart\_id UUID NOT NULL REFERENCES cart(id) ON DELETE CASCADE, -- Links to the cart

menu\_item\_id UUID NOT NULL REFERENCES menu\_items(id) ON DELETE CASCADE, -- Links to the menu item

quantity INT NOT NULL CHECK (quantity > 0), -- Number of items

price NUMERIC(10, 2) NOT NULL, -- Price per unit (captured at the time of adding to cart)

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE cart\_item\_options (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

cart\_item\_id UUID NOT NULL REFERENCES cart\_items(id) ON DELETE CASCADE, -- Links to the cart item

menu\_item\_option\_id UUID NOT NULL REFERENCES menu\_item\_options(id) ON DELETE CASCADE, -- Links to the item option

price NUMERIC(10, 2) NOT NULL, -- Price for the option

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Function to update the updated\_at column

CREATE OR REPLACE FUNCTION update\_updated\_at\_column()

RETURNS TRIGGER AS $$

BEGIN

NEW.updated\_at = CURRENT\_TIMESTAMP;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;  
  
———————————————————————————————————————————  
  
CREATE TABLE orders (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(), -- Unique order ID

user\_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE, -- Links to the user who placed the order

total\_price NUMERIC(10, 2) NOT NULL, -- Total price of the order

status VARCHAR(50) DEFAULT 'pending', -- Status of the order: 'pending', 'completed', 'canceled', etc.

payment\_status VARCHAR(50) DEFAULT 'unpaid', -- Status of the payment: 'unpaid', 'paid', 'failed', etc.

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, -- Timestamp when the order was created

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP -- Timestamp when the order was last updated

);

CREATE TABLE order\_items (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(), -- Unique order item ID

order\_id UUID NOT NULL REFERENCES orders(id) ON DELETE CASCADE, -- Links to the order

menu\_item\_id UUID NOT NULL REFERENCES menu\_items(id) ON DELETE CASCADE, -- Links to the menu item ordered

quantity INT NOT NULL CHECK (quantity > 0), -- Quantity of the item ordered

price NUMERIC(10, 2) NOT NULL, -- Price of the menu item at the time of order

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, -- Timestamp when the item was added to the order

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP -- Timestamp when the item details were last updated

);

CREATE TABLE shipping\_details (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(), -- Unique shipping detail ID

order\_id UUID NOT NULL REFERENCES orders(id) ON DELETE CASCADE, -- Links to the order

address\_id UUID NOT NULL REFERENCES addresses(id) ON DELETE CASCADE, -- Links to user's stored address

delivery\_method VARCHAR(50) DEFAULT 'standard', -- Delivery method: 'standard', 'express', etc.

shipping\_status VARCHAR(50) DEFAULT 'pending', -- Shipping status: 'pending', 'shipped', 'delivered', etc.

tracking\_number VARCHAR(100), -- Optional tracking number

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, -- Timestamp when shipping details were created

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, -- Timestamp when shipping details were last updated

CONSTRAINT fk\_shipping\_address FOREIGN KEY (address\_id) REFERENCES addresses(id) ON DELETE CASCADE

);  
  
CREATE TABLE payment\_details (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(), -- Unique payment detail ID

order\_id UUID NOT NULL REFERENCES orders(id) ON DELETE CASCADE, -- Links to the order

payment\_method VARCHAR(50) NOT NULL, -- Payment method: 'credit\_card', 'paypal', etc.

payment\_status VARCHAR(50) DEFAULT 'unpaid', -- Payment status: 'unpaid', 'paid', 'failed', etc.

transaction\_id VARCHAR(100), -- Transaction ID from the payment provider

amount NUMERIC(10, 2) NOT NULL, -- Amount paid for the order

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, -- Timestamp when payment details were created

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP -- Timestamp when payment details were last updated

);

CREATE TABLE order\_history (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(), -- Unique history record ID

order\_id UUID NOT NULL REFERENCES orders(id) ON DELETE CASCADE, -- Links to the order

status VARCHAR(50) NOT NULL, -- The status the order was in

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP -- Timestamp when the status change occurred

);

-- Function to update the updated\_at column

CREATE OR REPLACE FUNCTION update\_updated\_at\_column()

RETURNS TRIGGER AS $$

BEGIN

NEW.updated\_at = CURRENT\_TIMESTAMP;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Trigger for orders table

CREATE TRIGGER set\_updated\_at\_orders

BEFORE UPDATE ON orders

FOR EACH ROW

EXECUTE FUNCTION update\_updated\_at\_column();

-- Trigger for order\_items table

CREATE TRIGGER set\_updated\_at\_order\_items

BEFORE UPDATE ON order\_items

FOR EACH ROW

EXECUTE FUNCTION update\_updated\_at\_column();

-- Trigger for shipping\_details table

CREATE TRIGGER set\_updated\_at\_shipping\_details

BEFORE UPDATE ON shipping\_details

FOR EACH ROW

EXECUTE FUNCTION update\_updated\_at\_column();

-- Trigger for payment\_details table

CREATE TRIGGER set\_updated\_at\_payment\_details

BEFORE UPDATE ON payment\_details

FOR EACH ROW

EXECUTE FUNCTION update\_updated\_at\_column();

———————————————————————————————————————————-