# Capstone Project: Designing RushMore Pizzeria's Enterprise Database System

**Reference Case Study:** `RushMore Pizza Delivery with Python`

## 1. Scenario

Following the initial success of the Python-based ordering tool (from CPP01), RushMore Pizzeria has grown. They now have multiple locations, a growing customer base, and a complex menu.

The simple `orders.json` file system that you first built is now a critical business bottleneck. It's slow, impossible to query for insights, prone to data corruption, and cannot be accessed by multiple cashiers at once.

RushMore's management has brought you back, this time as a professional **Data Engineer & Database Administrator (DBA)**. Your mission is to replace the fragile JSON file system with a "single source of truth": a robust, scalable, and reliable Relational Database Management System (RDBMS).

**The Twist:** Before migrating any real (and sensitive) customer data, management wants to see a proof-of-concept. You must first build the database in the cloud and then populate it with **10,000+ rows of realistic, high-quality *fake* data**. This "masked" data will be used to stress-test the system, ensure data types are correct and demonstrate the power of analytics without compromising any customer privacy.

## 2. Objective

Your primary objective is to design, deploy, and populate a production-ready PostgreSQL database in the cloud. You will model the business's operations, deploy the database to a major cloud provider, and use Python scripting to generate a large, realistic dataset for testing and analysis.

This project will test your skills in:

- **Data Modeling:** Designing a normalized, efficient, and scalable relational database schema (RDBMS).
- **Cloud Deployment:** Provisioning, configuring, and managing a database as a service (DBaaS) on **Azure** or **Google Cloud Platform (GCP)**.
    - Here is how to provision a database on Azure and connect to pgAdmin.

    - Here is how to provision Cloud SQL for PostgreSQL on GCP and connect to pgAdmin.
- **Database Programming:** Writing Python scripts to connect to a remote database, generate synthetic (masked) data using `Faker`, and insert it in the correct order.
- **Data Analysis (Optional):** Connecting an analytics tool to your new database to answer key business questions.

## 3. Instructions

**Part 1: Database Modelling (Design)**

**Table: `Stores`**

- **Description:** Holds information for each physical pizzeria location.
- **Columns:**
    - `store_id` (SERIAL, PRIMARY KEY) - Unique ID for the store.
    - `address` (VARCHAR(255), NOT NULL) - The street address.
    - `city` (VARCHAR(100), NOT NULL) - The city of the store.
    - `phone_number` (VARCHAR(20), UNIQUE NOT NULL) - The store's main phone number.
    - `opened_at` (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP) - When the store opened.

**Table: `Customers`**

- **Description:** Stores PII (Personally Identifiable Information) for all registered customers.
- **Columns:**
    - `customer_id` (SERIAL, PRIMARY KEY) - Unique ID for the customer.
    - `first_name` (VARCHAR(100), NOT NULL) - Customer's first name.
    - `last_name` (VARCHAR(100), NOT NULL) - Customer's last name.
    - `email` (VARCHAR(255), UNIQUE NOT NULL) - Customer's email (for logins, receipts).
    - `phone_number` (VARCHAR(20), UNIQUE NOT NULL) - Customer's phone (for order updates).
    - `created_at` (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP) - When the customer account was created.

**Table: `Ingredients`**

- **Description:** A master list of all raw ingredients used to make menu items.
- **Columns:**
    - `ingredient_id` (SERIAL, PRIMARY KEY) - Unique ID for the ingredient.

    - `name` (VARCHAR(100), UNIQUE NOT NULL) - e.g., "Pepperoni", "Mozzarella Cheese", "Pizza Dough".
    - `stock_quantity` (NUMERIC(10, 2), NOT NULL, DEFAULT 0) - Current amount in stock.
    - `unit` (VARCHAR(20), NOT NULL) - The unit of measure for `stock_quantity` (e.g., 'kg', 'liters', 'units').

**Table: `Menu_Items`**

- **Description:** The master product catalog for all items sold (pizzas, drinks, sides).
- **Columns:**
    - `item_id` (SERIAL, PRIMARY KEY) - Unique ID for the menu item.
    - `name` (VARCHAR(150), NOT NULL) - e.g., "Large Pepperoni Pizza".
    - `category` (VARCHAR(50), NOT NULL) - e.g., 'Pizza', 'Drink', 'Side'.
    - `size` (VARCHAR(20)) - e.g., 'Small', 'Medium', 'Large', '500ml', 'N/A'.

**Table: `Orders`**

- **Description:** This is the master transaction table. Each row represents one complete customer order.
- **Columns:**
    - `order_id` (SERIAL, PRIMARY KEY) - Unique ID for the order.
    - `customer_id` (INTEGER, REFERENCES `Customers`(`customer_id`) ON DELETE SET NULL) - Foreign Key to `Customers`. (Set NULL on delete so order history remains if a customer account is deleted).
    - `store_id` (INTEGER, REFERENCES `Stores`(`store_id`) ON DELETE RESTRICT) - Foreign Key to `Stores`. (Restrict delete to prevent orphaning order data).
    - `order_timestamp` (TIMESTAMP, NOT NULL, DEFAULT CURRENT_TIMESTAMP) - When the order was placed.
    - `total_amount` (NUMERIC(10, 2), NOT NULL) - The final calculated total for the order.

First, you must design the database schema. Your schema should be normalised (aim for 3rd Normal Form) to reduce redundancy and improve data integrity.

**Detailed Schema Design**

Here is a detailed breakdown of the tables, columns, and relationships you are to model.

**Tools and SQL Script**

1. **Tools:** Use a tool like **draw.io** or **Lucidchart** to create an Entity-Relationship Diagram (ERD) of your schema. This ERD must visually represent all tables and their relationships.
2. **SQL Script:** Write the `CREATE TABLE` statements for your entire schema into a single `.sql` file.

**Part 2: Cloud Database Deployment (Deploy)**

You will deploy your empty schema to a managed cloud service.

1. **Choose a Provider:**
    - **On Azure:** Provision an **"Azure Database for PostgreSQL"** flexible server.
    - **On GCP:** Provision a **"Cloud SQL for PostgreSQL"** instance.
2. **Configure:** Create a database (e.g., `rushmore_db`).
3. **Run Schema:** Use a database tool (like DBeaver, pgAdmin, or `psql`) to connect to your new cloud database and run your `.sql` script from Part 1. Your database should now contain all your empty tables.

**Part 3: Data Population with Faker (Populate & Mask)**

This is the core programming task. You will write a Python script to fill your cloud database with realistic, fake data.

1. **Libraries:** Your script must use `psycopg2-binary` (to connect to Postgres) and `Faker` (to generate the data).
2. **Connection:** Your script must read the database credentials (Host, User, Password, DBName) from environment variables or a `.yaml` or `.env` file (do not hardcode them).
3. **Data Generation:** Use `Faker` to create realistic data for PII (Personally Identifiable Information).
    - `fake.name()` for customer names.
    - `fake.phone_number()` for customer phones.
    - `fake.email()` for customer emails.
    - `fake.address()` for store/customer addresses.
    - `fake.date_time_this_year()` for order timestamps.
4. **Target Volume:** Generate a significant amount of data to simulate a real business:
    - `Stores`: 3-5
    - `Menu_Items`: 20-30
    - `Ingredients`: 40-50
    - `Customers`: 1,000+
    - `Orders`: 5,000+
    - `Order_Items`: ~15,000+ (avg. 3 items per order)

5. **Execution Order:** You *must* insert data in the correct order to respect foreign key constraints (e.g., you must create `Customers`, `Stores`, and `Menu_Items` *before* you can create an `Order`).

**Part 5: Analysis (Prove It Works; Optional)**

Finally, prove that your new database is ready for analytics.

1. **Connect:** Connect an analytics tool like **Power BI** or **Looker Studio** or **Tableau** directly to your cloud PostgreSQL database.
2. **Analyze:** Write SQL queries to answer the following key business questions. You can write these directly in your BI tool or in a `.sql` file.

**Key Business Questions (Part 5):**

1. What is the total sales revenue per store?
2. Who are the top 10 most valuable customers (by total spending)?
3. What is the most popular menu item (by quantity sold) across all stores?
4. What is the average order value?
5. What are the busiest hours of the day for orders (e.g., 5 PM, 6 PM)?

## 4. Deliverables

You will submit a **GitHub Repository link** to all your project files. This must also be **documented and presented** as your final capstone presentation.

1. **Entity-Relationship Diagram (ERD):** A `.png` or `.pdf` file of your complete database schema.
2. **Schema SQL File:** The `.sql` file with all your `CREATE TABLE` statements.
3. **Python Population Script:** Your `populate.py` script.
4. **Analytics & Queries (Part 5; again Optional):** A `.sql` file containing the 5 queries you used to answer the business questions, along with the results.
5. **Proof of Cloud Deployment:**
    - A screenshot of your Azure or GCP dashboard showing your active PostgreSQL instance.
    - A screenshot from **pgAdmin** connected to your cloud database, showing the list of tables with their row counts (proving your `Faker` script worked).
6. **Project Documentation & Presentation:**
    - A final presentation slide deck (PowerPoint `.pptx`) summarizing your project's architecture, challenges, and results.
    - **Alternatively**, a professional `README.md` file in a **GitHub repository** containing all your project code, documentation, screenshots, and a clear explanation of how to run your project.