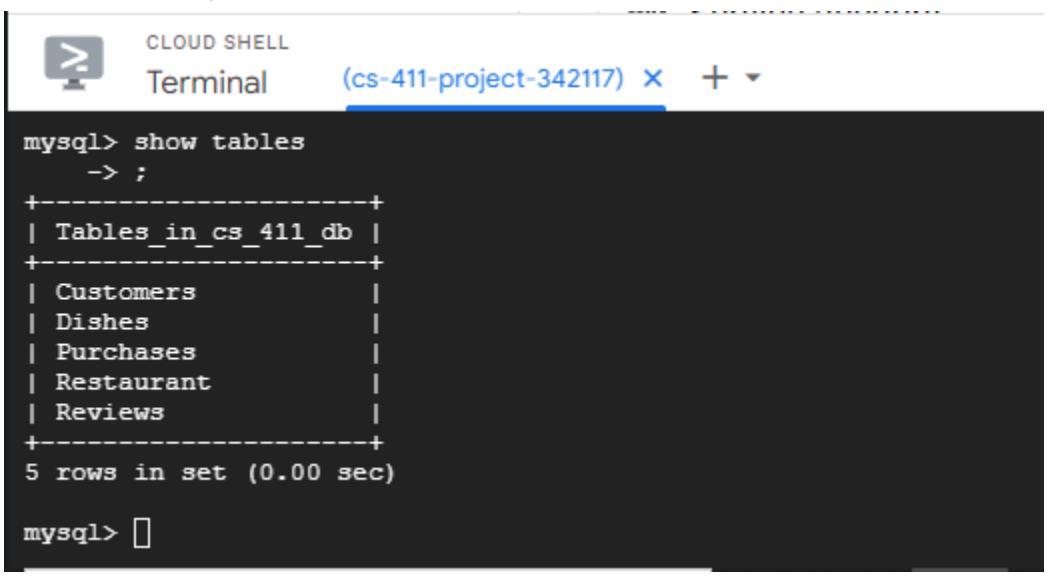
The insert commands for each table are provided in `.sql` files in the root of the repo.

The Restaurant table is populated with 1000+ REAL restaurant data from the Google Maps API using a custom Python script. There are clearly not 1000 restaurants in Champaign, so locations throughout the US are used. Information is fetched from Google Maps, stored as JSON, and then run through a custom converter to create the SQL insert commands. Most of the information is accurate, but for restaurants without an obvious cuisine, it is randomized. Similarly, for restaurants without a listed open/close time, etc, it is set to a default. See the Python code here: https://github.com/shivenk78/GoogleMaps-Fetcher

The remainder of the tables are randomly generated from a data generator site. Otherwise, it would be very time consuming to get 1000+ accurate entries for dishes, purchases, etc. Count query screenshots are provided after each table's DDL schema. Besides Restaurants, everything else has exactly 1000 entries since it's randomly generated. Restaurants fetches 1200 results, and then filters out restaurants with non-ASCII characters in the name to avoid difficulties with mySQL, giving ~1100 results.

We implemented the database using GCP to make it more flexible, and easier to collaborate. It also more closely resembles a production application or project than a local instance.

Tables displayed in GCP Cloud Shell



Restaurants Table

```
DROP TABLE IF EXISTS Restaurant;

CREATE TABLE Restaurant (
 restaurantID INT NOT NULL,
 address varchar(255) NOT NULL,
 name varchar(255) default NULL,
 phoneNumber varchar(255) default NULL,
 cuisine varchar(255) default NULL,
 openingTime varchar(255),
 closingTime varchar(255),
 PRIMARY KEY (restaurantID)
);
```

```
+-----------------------+
| count(r.restaurantID) |
+-----------------------+
|                  1136 |
+-----------------------+
1 row in set (0.01 sec)
```

Purchases Table

```
DROP TABLE IF EXISTS Purchases;

CREATE TABLE Purchases (
 PurchaseID INT NOT NULL,
 RestaurantID INT NOT NULL,
 NetID varchar(255) default NULL,
 DishName varchar(255),
 PRIMARY KEY (PurchaseID),
 FOREIGN KEY (RestaurantID) REFERENCES Restaurant(restaurantID),
 FOREIGN KEY (NetID) REFERENCES Customers(NetID),
 FOREIGN KEY (DishName) REFERENCES Dishes(Name)
);
```

```
+---------------------+
| count(p.PurchaseID) |
+---------------------+
|                1000 |
+---------------------+
```

Dishes Table

```sql
DROP TABLE IF EXISTS Dishes;

CREATE TABLE Dishes (
 RestaurantID INT NOT NULL,
 Name varchar(255) NOT NULL,
 Cuisine varchar(255) default NULL,
 Calories INT default NULL,
 Price REAL,
 PRIMARY KEY (Name),
 FOREIGN KEY (RestaurantID) REFERENCES Restaurant(restaurantID)
);
```

```
+--------------------+
| count(Dishes.Name) |
+--------------------+
|               1000 |
+--------------------+
```

Customers Table

```sql
DROP TABLE IF EXISTS Customers;

CREATE TABLE Customers (
 NetID varchar(255) NOT NULL,
 Name varchar(255) default NULL,
 Email varchar(255) default NULL,
 Password varchar(255),
 PRIMARY KEY (NetID)
);
```

```
+------------------------+
| count(Customers.NetID) |
+------------------------+
|                   1000 |
+------------------------+
```

Reviews Table

```
DROP TABLE IF EXISTS Reviews;

CREATE TABLE Reviews (
 ReviewID INT NOT NULL,
 PurchaseID INT NOT NULL,
 NetID varchar(255) default NULL,
 Rating REAL,
 Title varchar(100),
 Description TEXT(1000),
 PRIMARY KEY (ReviewID),
 FOREIGN KEY (PurchaseID) REFERENCES Purchases(PurchaseID),
 FOREIGN KEY (NetID) REFERENCES Customers(NetID)
);
```

```
+-------------------------+
| count(Reviews.ReviewID) |
+-------------------------+
|                    1000 |
+-------------------------+
```