Creating Fake Customer data

These are the steps that I took in order to create fake customer data with first name, last name, address, secondary address, zipcode, city, and country:

- First I used chatGPT to create list of unique first names, last names, and street names all separated by commas. I copied these lists into their own separate text files.

Names:

- The names were the easiest part. I used the choices method from the random library to create lists of 2500 each. I chose the .choices method because it samples with replacement. It's important to note that the random library isn't truly random, but works well enough for creating fake customer data. These 2 lists were transformed into distinct series objects

Address1:

- I extracted the 200 unique street names from the text file into a list and created another fake list where the complete address would be stored.
- Next I used a for loop that iterated 2500 times, each time selecting a random number between 1 and 10,000 for a house number and a random street name from the street name list. They were parsed together as a single string and appended to the complete address list.

Address2:

- I first created a list of length 2500, where each item was a random selection between secondary address types: 'apt', 'bldg', or 'unit'.
- Next, I used a for-loop that used probability to determine whether each item would have a building number appended to the end of the secondary address type or would be transformed into a null value. This was done because I determined that most of Jugos Rodier's customers would live in houses rather than apartments or the like.
- This was accomplished by using random.choice to select a random number between 1 and 100. If the number was less than 71, the secondary address was transformed into a null value. If the number was not less than 71, then a unit number was assigned. The unit numbers were a random selection of numbers between 1 and 50. In other words, 70% of the time, a null value was assigned, and the other 30% of the time a literal space and unit number were appended .

Zipcode:

There are 7 unique zip codes assigned to customers. In assigning zip codes the the customers, I determined that the restaurant would have significantly higher traffic from customers who lived within the 2 zip codes that were closest to the restaurant. No actual thought or research was put to determining whether the more frequently selected zip codes were actually geographically near to one another since I considered this detail to be beyond the scope of the project. The zip codes that are "closest" to the restaurant were arbitrarily selected from a list of zip codes that are within or near Bakersfield, CA, my hometown.

*Since Addresses, address2 and zip codes are all arbitrarily assigned and selected using a sample with replacement style of random selection, they will not make any logical sense. For example, a customer might have the address "5 Maple Av" with the zipcode 93304 while another customer might have the address "2 Maple Av" with a zipcode 93306. Logically, you would assume that addresses very close to one another would have the same zipcode most of the time, but this will not be the case with this customer database.

- Assigning customers a "near" zipcode or a "further" zipcode was executed using a for-loop and the same probabilistic approach as mentioned above in address2 assignment.
- First, 2 lists of zip codes were created. The "close" zip codes contained 2 zip codes and the "further" zip codes contained 5 zip codes.
- Each of the 2500 iterations of the for-loop follow the same process:

- generate a number between 0 and 100
- If the number is 70 or below, a “close” zip code is assigned.
- if the number is not 70 or below, a “further” zip code is assigned.

City and Country:

All customers were assigned Bakersfield and USA for city and Country.