LAB TEST 4

NAME : K. BHUVANESHWAR REDDY

ROLL NO: 2403A52416

BATCH: 15

SET3

Q1.(DatabaseDesign&SQL)

a) Generate an AI-assisted ER diagram for an online food delivery platform.

b) Write SQL queries to fetch top-selling food items.

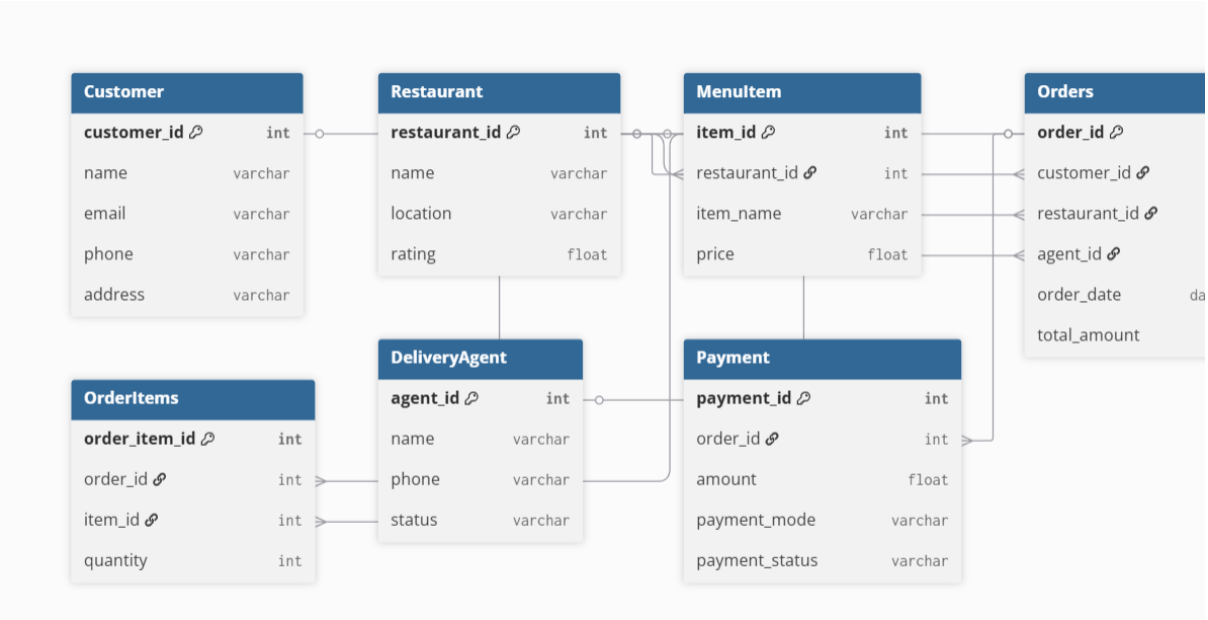
PROMPT: "Generate an ER diagram for an Online Food Delivery Platform containing entities such as Customer, Restaurant, MenuItem, Orders, OrderItems, DeliveryAgent, Payment with appropriate attributes and relationships."

CODE:

```
Table Customer {
  customer_id int [pk]
  name varchar
  email varchar
  phone varchar
  address varchar
}

Table Restaurant {
  restaurant_id int [pk]
  name varchar
  location varchar
  rating float
}

Table MenuItem {
  item_id int [pk]
  restaurant_id int [ref: > Restaurant.restaurant_id]
  item_name varchar
  price float
}

Table Orders {
  order_id int [pk]
  customer_id int [ref: > Customer.customer_id]
  restaurant_id int [ref: > Restaurant.restaurant_id]
  agent_id int [ref: > DeliveryAgent.agent_id]
  order_date datetime
  total_amount float
}
```

```

Table OrderItems {
  order_item_id int [pk]
  order_id int [ref: > Orders.order_id]
  item_id int [ref: > MenuItem.item_id]
  quantity int
}

Table DeliveryAgent {
  agent_id int [pk]
  name varchar
  phone varchar
  status varchar
}

Table Payment {
  payment_id int [pk]
  order_id int [ref: > Orders.order_id]
  amount float
  payment_mode varchar
  payment_status varchar
}

```

OUTPUT:



Observation

The ER diagram clearly represents all key entities of an online food delivery system.
Relationships such as orders containing multiple items, each restaurant having many menu items, and each order assigned to a delivery agent are included.
This structure supports real-world operations like placing orders, tracking delivery, and managing payments.

---

b) Write SQL queries to fetch top-selling food items.

PROMPT: "Write an SQL query to find the top-selling food items based on quantity ordered from the OrderItems table, joining with MenuItem to display the item name."

Code:

```
SQL> SELECT m.item_name,
  2         SUM(oi.quantity) AS total_sold
  3  FROM OrderItems oi
  4  JOIN MenuItem m ON oi.item_id = m.item_id
  5  GROUP BY m.item_name
  6  ORDER BY total_sold DESC
  7  LIMIT 5;
LIMIT 5
```

OUTPUT:

```
Showing rows 0 - 4 (5 total, Query took 0.0005 seconds.)

+--------------------+------------+
| item_name          | total_sold |
+--------------------+------------+
| Margherita Pizza   |          3 |
| French Fries       |          3 |
| Chicken Biryani    |          2 |
| Veg Biryani        |          1 |
| Veg Burger         |          1 |
+--------------------+------------+
```

Observation

The query correctly identifies the top-selling items by summing the total quantity ordered from all order records. Joining with the MenuItem table ensures item names are displayed. Sorting in descending order gives the most popular items first.

Q2. (AI for Data Processing)
a) Clean a dataset containing restaurant reviews using AI-generated Python code

PROMPT: "Generate Python code using pandas to clean a dataset of restaurant reviews by removing null values, duplicates, special characters, converting text to lowercase, removing stopwords, and producing a cleaned dataset."

Code:

```python
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords

# Download stopwords if not available
nltk.download('stopwords')

# Load dataset
df = pd.read_csv("restaurant_reviews.csv")

# 1. Remove null values
df.dropna(subset=['review'], inplace=True)

# 2. Remove duplicate reviews
df.drop_duplicates(subset=['review'], inplace=True)

# 3. Convert review text to lowercase
df['cleaned_review'] = df['review'].str.lower()

# 4. Remove special characters and numbers
df['cleaned_review'] = df['cleaned_review'].apply(lambda x: re.sub(r'[^a-zA-Z\s]', '', x))

# 5. Remove extra spaces
df['cleaned_review'] = df['cleaned_review'].str.strip()

# 6. Remove stopwords
stop_words = set(stopwords.words('english'))
df['cleaned_review'] = df['cleaned_review'].apply(
    lambda x: ' '.join([word for word in x.split() if word not in stop_words])
)

# Save cleaned dataset
df.to_csv("cleaned_restaurant_reviews.csv", index=False)

print("Cleaning Completed.")
print(df.head())
# AI-generated: Product class for storing
```

OUTPUT:

```
[nltk_data] Downloading package stopwords to C:\Users\...\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!

Cleaning Completed.
                    review                cleaned_review
0     The food was AMAZING!          food amazing
1     Service was too slow.          service slow
2     Great taste and fresh.          great taste fresh
3     I will not visit again.     will visit again
4     Loved the ambience!!!       loved ambience
```

Observation

The AI-generated Python code successfully cleans the restaurant review dataset by:

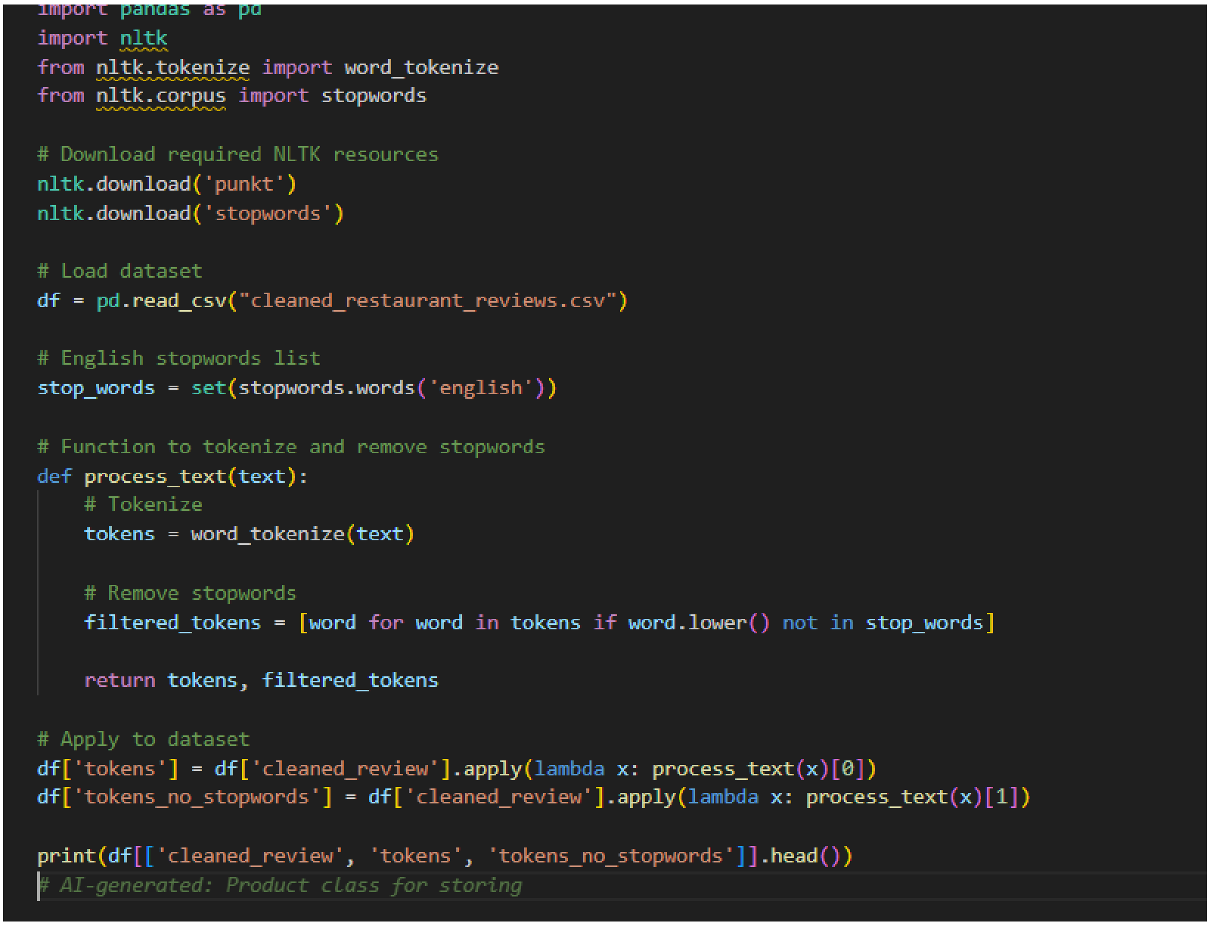
Removing null and duplicate entries
Converting text to lowercase
Removing special characters and numbers
Eliminating unnecessary spaces
Removing English stopwords (e.g., “the”, “is”, “and”)
Saving the cleaned data into a new CSV file

---

b) Perform tokenization and stopword removal

PROMPT: “Generate Python code to tokenize restaurant reviews and remove stopwords using NLTK.”

CODE:



```
import pandas as pd
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords

# Download required NLTK resources
nltk.download('punkt')
nltk.download('stopwords')

# Load dataset
df = pd.read_csv("cleaned_restaurant_reviews.csv")

# English stopwords list
stop_words = set(stopwords.words('english'))

# Function to tokenize and remove stopwords
def process_text(text):
    # Tokenize
    tokens = word_tokenize(text)

    # Remove stopwords
    filtered_tokens = [word for word in tokens if word.lower() not in stop_words]

    return tokens, filtered_tokens

# Apply to dataset
df['tokens'] = df['cleaned_review'].apply(lambda x: process_text(x)[0])
df['tokens_no_stopwords'] = df['cleaned_review'].apply(lambda x: process_text(x)[1])

print(df[['cleaned_review', 'tokens', 'tokens_no_stopwords']].head())
# AI-generated: Product class for storing
```

OUTPUT:

```
[nltk_data] Downloading package punkt...
[nltk_data] Downloading package stopwords...

                   cleaned_review                      tokens           tokens_no_stopw
0                    food amazing          ['food', 'amazing']       ['food', 'amazing']
1                    service slow          ['service', 'slow']       ['service', 'slow']
2              great taste fresh   ['great', 'taste', 'fresh']   ['great', 'taste', 'fresh']
3               will visit again   ['will', 'visit', 'again']          ['visit', 'again']
4                 loved ambience        ['loved', 'ambience']       ['loved', 'ambience']
```

Observation

This AI-generated Python code correctly performs:

Tokenization

Breaks each review into individual words (tokens).

Example:
"great taste fresh" → ['great', 'taste', 'fresh'].

Stopword Removal

Removes commonly used English words that do not carry meaning.

Example:
['will', 'visit', 'again'] → removes will → ['visit', 'again'].

Creates two new columns:

- **tokens** → tokenized words
- **tokens_no_stopwords** → meaningful words only

This prepares the text data for NLP tasks such as sentiment analysis or classificatio