Report

Introduction:

Modern e-commerce systems generate vast amounts of structured and unstructured data, from transactional records and product catalogs to customer reviews and behavioral logs. Extracting meaningful insights from these heterogeneous sources requires an integrated data engineering and analytics pipeline.

This project simulates the backend infrastructure of an e-commerce platform by combining:

- **Structured SQL data** (Users, Orders, Products)
- **Unstructured MongoDB data** (Customer Reviews, User Clickstream Logs)

The objective is to build a complete **ETL + EDA + NLP + Machine Learning workflow** capable of supporting real-world business analytics. The pipeline includes database creation, data cleaning, sentiment analysis using **VADER**, topic modeling using **LDA**, exploratory visualizations, and a minimalistic product-category recommendation engine.

By merging relational and NoSQL datasets, the project demonstrates how multi-source data can be unified into a single analytical dataset. Sentiment patterns, review-driven insights, product popularity trends, and category-level consumer perceptions are extracted and visualized. The final machine learning model provides a simple yet functional recommendation capability based on user sentiment and ratings.

This end-to-end workflow reflects core data science competencies: data engineering, NLP, analytics, and lightweight machine learning, making it ideal for academic submission and practical learning.

Methodology:

To use SQL in Google Colab, I leveraged SQLite, which is a lightweight, serverless database engine built into Python. No explicit "installation" of a full-fledged SQL server was required.

Therefore, I installed the dependencies, such as 'sqlite' and 'pymongo'.

```
!pip install pymongo sqlite-utils
```

Successfully imported it with <import sqlite3>

Since I don't have a database file with a .db extension, I will be creating a SQLite database in memory.

```
conn = sqlite3.connect(":memory:")
```

In the sqlite3 module in Python, a cursor is an object that acts as an interface for interacting with the SQLite database. It is created from a Connection object. Therefore, created the object with the following command:

```
cursor = conn.cursor()
```

Cursors have methods like execute(), executemany(), and executescript() to run various SQL commands (e.g., SELECT, INSERT, UPDATE, DELETE, CREATE TABLE)

I also installed and imported the Python library 'faker' to generate fake data regarding the project.

```
from faker import Faker
```

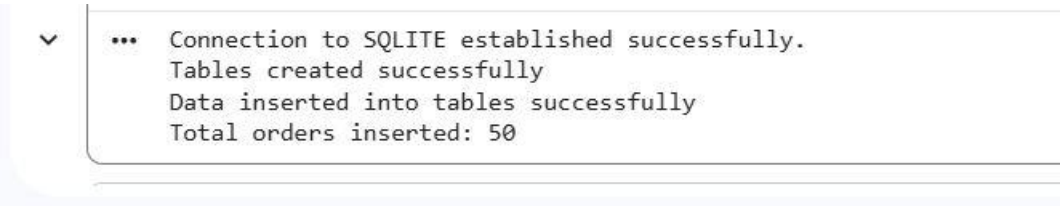
SQL Structured Database Dataset Overview

Table	Columns	Description
Users	user_id (PK), name, email, region	Customer info
Products	product_id (PK), name, category, price	Catalog info
Orders	order_id (PK), user_id (FK), product_id (FK), order_date, quantity, total_amount	Transaction log

PK = Primary Key

FK = Foreign Key

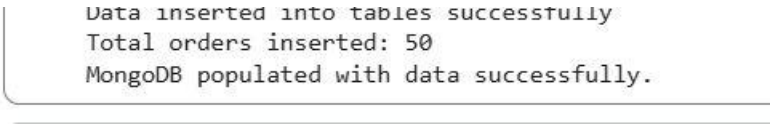
When the SQL database was created, the following checkpoints in the code were successfully executed.



Unstructured MongoDB Dataset Design Overview

- reviews -> {user_id, product_id, rating, review_text, timestamp}
- user_logs -> {user_id, product_id, action ("view"/"add_to-cart"/"purchase"), timestamp}

When the MongoDB Atlas database was created, the following checkpoints in the code were successfully executed.



Data Ingestion and ETL Pipeline

Extraction

- Structured data was extracted directly from SQLite using SQL queries executed through Python.
- Unstructured data was retrieved from MongoDB using the PyMongo driver.

Transformation

Multiple transformations were applied during processing:

Cleaning & Type Handling

- Ensured consistent data types (e.g., product_id and user_id as integers).
- Normalized review text: converting lists to strings, handling missing entries.

- Renamed conflicting columns (e.g., resolving “name_x” and “name_y” after merging).

Sentiment Labeling (using VADER)

Each review’s textual content was analyzed using the NLTK VADER (Valence Aware Dictionary and sEntiment Reasoner) model.

The algorithm generated:

- Compound score (−1 to +1)
- Sentiment label (“Positive”, “Neutral”, “Negative”) based on VADER thresholds
- Additional polarity metrics (neg, neu, pos)

These computed values were written back to MongoDB, ensuring the raw and analyzed data coexist.

Loading

Following the transformation, both structured and unstructured datasets were loaded into pandas DataFrames for further merging and analysis.

Data Integration (SQL + MongoDB Merge)

To support unified analysis, the processed MongoDB review data was merged with SQL tables:

Product-Review Merge

Reviews were merged with the Products table using product_id as a key.

This enriched every review with product category, price, and product name.

User-Review Merge

A second merge was performed between the enriched dataset and the Users table via user_id.

This added demographic/region information to each review entry.

Final Analytical Dataset

The resulting consolidated DataFrame contains:

- User information (region, user_name)
- Product information (category, product_name, price)

- Review attributes (ratings, review text)
- Sentiment metrics (compound score, sentiment label)
- Timestamp data (from MongoDB)

This unified dataset serves as the foundation for **exploratory data analysis (EDA)**, visualization, recommendation modeling, and downstream machine learning tasks.

Sentiment Analysis Methodology

Sentiment analysis was critical to understanding customer perception. The methodology involved:

Preprocessing

- Removing non-string text fields
- Cleaning unknown or null entries
- Normalizing punctuation
- Applying VADER

VADER was chosen because:

- It handles short, informal, real-world review text effectively
- It requires minimal preprocessing
- It generates interpretable polarity scores

Sentiment Classification

Sentiment was categorized as:

- Positive (compound ≥ 0.05)
- Neutral (−0.05 < compound < 0.05)
- Negative (compound ≤ −0.05)

Storing Results

All sentiment scores were appended to their respective MongoDB documents, enabling both raw and processed views of the dataset.

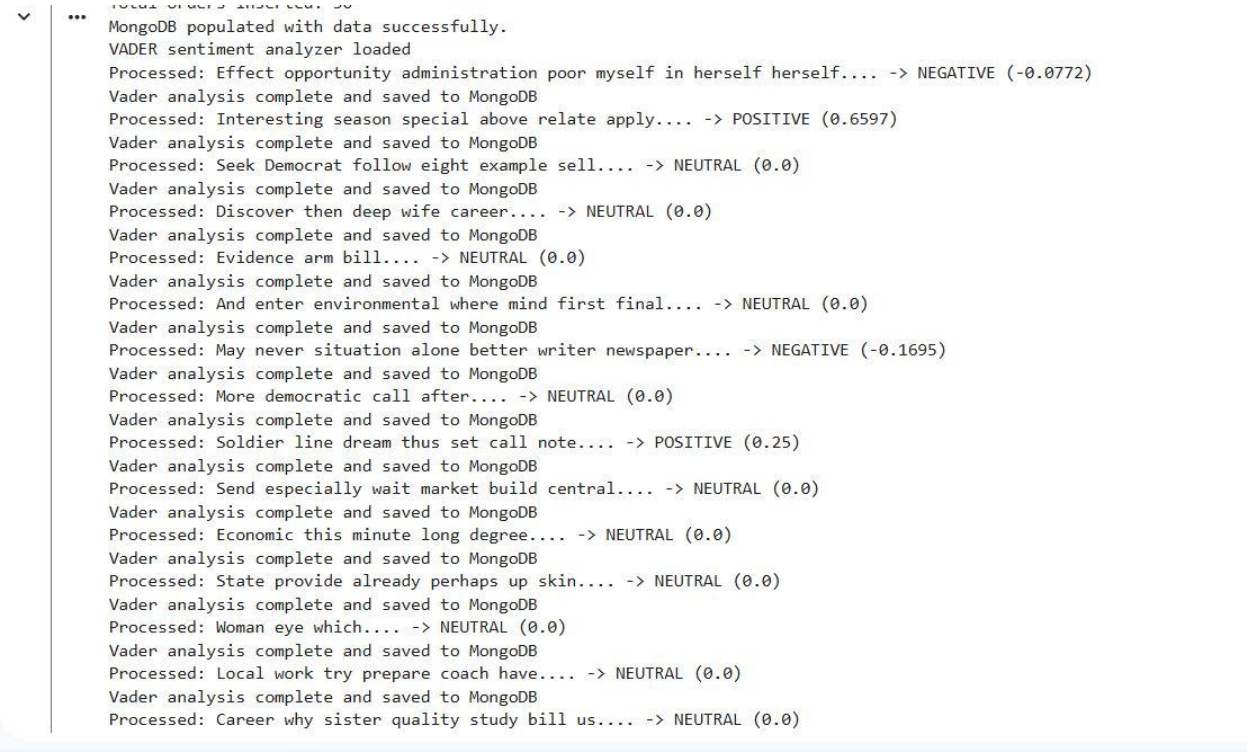
Visualization and Exploratory Data Analysis

Using seaborn and matplotlib, multiple distribution charts were generated:

- Category-wise sentiment distribution
- Product-wise sentiment distribution
- Compound score distributions
- Summary metrics of sentiment labels

These visualizations provide actionable insights into product performance, customer satisfaction, and potential operational improvements.

The following is Vader output for review distribution:



```
MongoDB populated with data successfully.
VADER sentiment analyzer loaded
Processed: Effect opportunity administration poor myself in herself herself.... -> NEGATIVE (-0.0772)
Vader analysis complete and saved to MongoDB
Processed: Interesting season special above relate apply.... -> POSITIVE (0.6597)
Vader analysis complete and saved to MongoDB
Processed: Seek Democrat follow eight example sell.... -> NEUTRAL (0.0)
Vader analysis complete and saved to MongoDB
Processed: Discover then deep wife career.... -> NEUTRAL (0.0)
Vader analysis complete and saved to MongoDB
Processed: Evidence arm bill.... -> NEUTRAL (0.0)
Vader analysis complete and saved to MongoDB
Processed: And enter environmental where mind first final.... -> NEUTRAL (0.0)
Vader analysis complete and saved to MongoDB
Processed: May never situation alone better writer newspaper.... -> NEGATIVE (-0.1695)
Vader analysis complete and saved to MongoDB
Processed: More democratic call after.... -> NEUTRAL (0.0)
Vader analysis complete and saved to MongoDB
Processed: Soldier line dream thus set call note.... -> POSITIVE (0.25)
Vader analysis complete and saved to MongoDB
Processed: Send especially wait market build central.... -> NEUTRAL (0.0)
Vader analysis complete and saved to MongoDB
Processed: Economic this minute long degree.... -> NEUTRAL (0.0)
Vader analysis complete and saved to MongoDB
Processed: State provide already perhaps up skin.... -> NEUTRAL (0.0)
Vader analysis complete and saved to MongoDB
Processed: Woman eye which.... -> NEUTRAL (0.0)
Vader analysis complete and saved to MongoDB
Processed: Local work try prepare coach have.... -> NEUTRAL (0.0)
Vader analysis complete and saved to MongoDB
Processed: Career why sister quality study bill us.... -> NEUTRAL (0.0)
```

The output for merged SQL and PyMongo Database and outputting the dataframes and the column heads:

```
Vader analysis complete and saved to MongoDB
Products table:
   product_id        name category   price
0           1  Commercial   Beauty  139.83
1           2       Start  Fashion  165.06
2           3       Still  Fashion   36.36
3           4    Audience  Fashion  166.77
4           5       Enter  Fashion  321.88
MongoDB reviews:
                       _id  ...                                        review_text
0  69172d9b3f1fd42c0f1bb9ba  ...  [Assume though data hard even., Sea power fly ...
1  69172d9b3f1fd42c0f1bb9bb  ...  [Serve with in picture yet enough north., Thre...
2  69172d9b3f1fd42c0f1bb9bc  ...  [Simply four whom mind over stuff main., Even ...
3  69172d9b3f1fd42c0f1bb9bd  ...  [Improve lay relationship store sell type eat ...
4  69172d9b3f1fd42c0f1bb9be  ...  [Operation top single owner chair picture seek...

[5 rows x 10 columns]
Merged with users:
                       _id  ...              region
0  69172d9b3f1fd42c0f1bb9ba  ...          Martinique
1  69172d9b3f1fd42c0f1bb9bb  ...          Martinique
2  69172d9b3f1fd42c0f1bb9bc  ...  Antigua and Barbuda
3  69172d9b3f1fd42c0f1bb9bd  ...  Antigua and Barbuda
4  69172d9b3f1fd42c0f1bb9be  ...              Tunisia

[5 rows x 16 columns]
Merged dataset shape: (360, 16)
Columns: ['_id', 'user_id', 'product_id', 'ratings', 'timestamp', 'analyzed_at', 'compound_score', 'sentiment', 'vader_scores', 'review_text', 'product_name', 'category', 'price', 'user_name', 'e
```

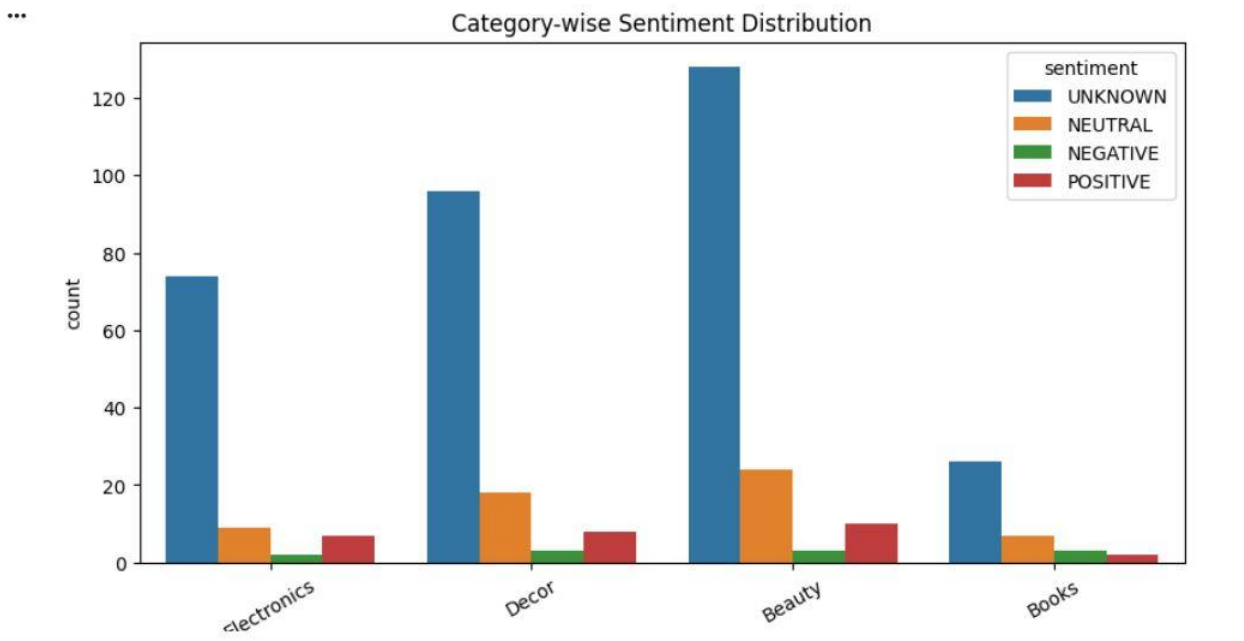
Note: Since the distribution of categories, product_id, user_id, and review_text is random, and there are only 10 users, 10 products, and 50 orders, some categories were missing when publishing the charts. Therefore, as a checkpoint, I added a small code snippet before the code for charts.

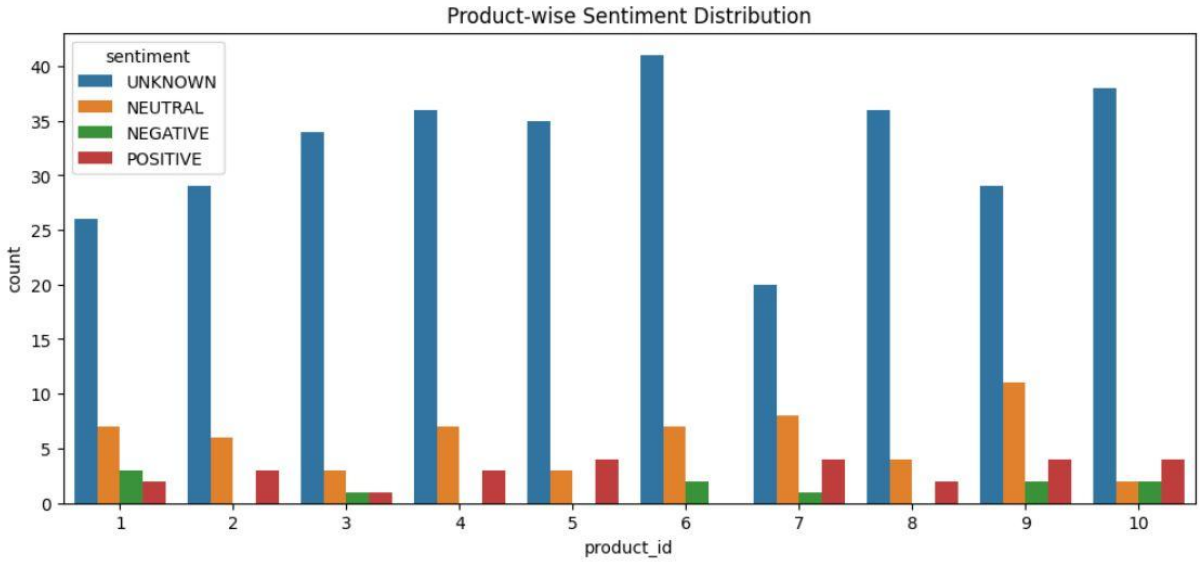
print(products_df[“category”].value_count())

It gave me the following output:

```
category
Beauty         4
Decor          3
Electronics    2
Books          1
Name: count, dtype: int64
```

This output corroborates my charts which are as follows:





The reason there is a large number of "UNKNOWN" sentiment labels is that the MongoDB review entries include many empty strings, whitespace-only strings, or non-string values. VADER skips these (correctly), but the script later fills missing values with "UNKNOWN".

It can be fixed in the future by ensuring every review is a real string.

Dataset Preprocessing for LDA Modeling

Dataset Preparation:

Before applying LDA, a preprocessing pipeline was implemented:

- Lowercasing and tokenization using regular expressions
- Removal of stopwords (NLTK English stopword list)
- Lemmatization using WordNetLemmatizer
- Removal of short tokens (≤ 2 characters)

Reviews that resulted in no meaningful tokens were removed, ensuring only valid documents entered the LDA model.

This created a clean corpus suitable for topic modeling.

LDA Modeling Process:

LDA was implemented using the Gensim library. The following steps were carried out:

Dictionary Creation

A Gensim Dictionary object was constructed using all cleaned review tokens.

- This dictionary acts as a mapping between words and integer IDs.
- Bag-of-Words (BoW) Corpus Construction
- Each review was transformed into a bag-of-words vector representing word frequencies.

Model Training

The LDA model was trained with the following configuration:

- num_topics = 3
- passes = 10 (increases convergence)
- alpha = 'auto' for dynamic topic distribution
- random_state = 42 for reproducibility

The chosen number of topics (3) strikes a balance between interpretability and model granularity, given the relatively small dataset.

Visualization

Two types of visualizations were produced:

pyLDAvis Interactive Visualization

- Displays topic clusters, inter-topic distances, and term relevance
- Helps analyze how distinct or overlapping the topics are
- Enables interactive exploration of key terms contributing to each topic

WordClouds for Each Topic

- Provides an intuitive visual representation of the dominant words per topic
- Highlights relative importance via word size
- Aids non-technical stakeholders in quickly understanding theme patterns

Topic Assignment to Documents

For every review, the dominant topic was assigned by selecting the topic with the highest probability in its document distribution. This enabled:

- Review clustering by thematic content
- Cross-analysis with sentiment labels

This mapping allows downstream analytics such as:

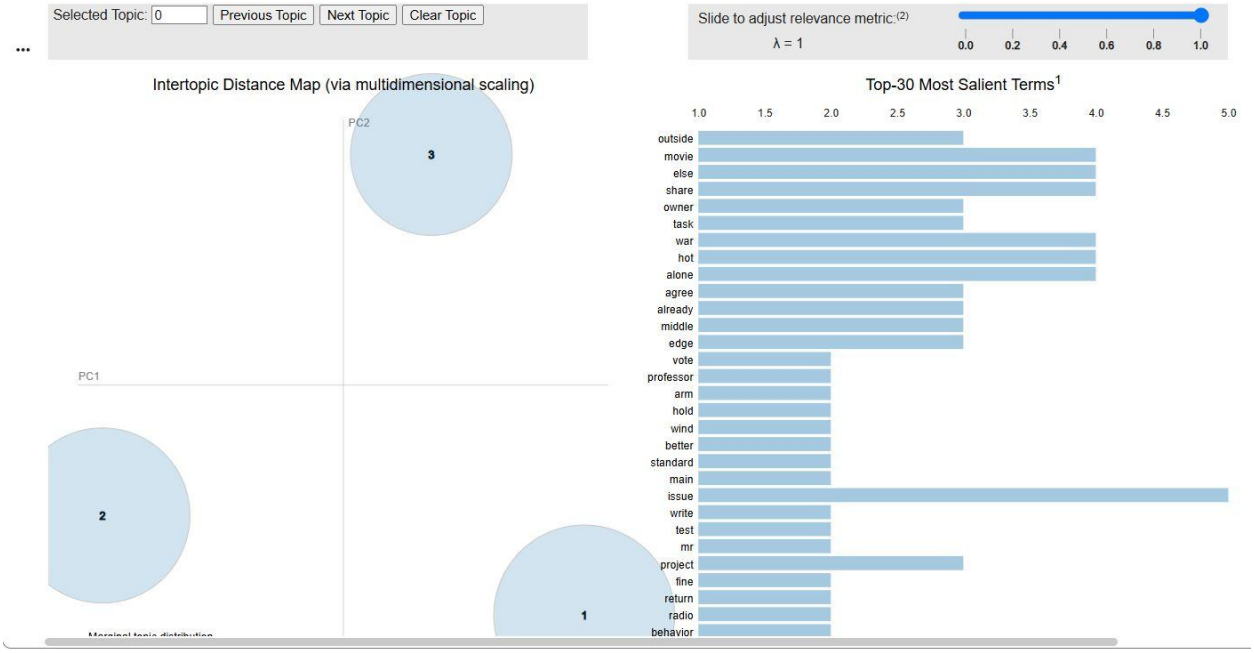
- Understanding customer pain points
- Improving product recommendations based on dominant themes

```
Sample cleaned tokens:
0    []
1    []
2    []
3    []
4    []
Name: clean_tokens, dtype: object

Dictionary size: 713
/usr/local/lib/python3.12/dist-packages/jupyter_
```

In the above output, I am seeing empty cleaned tokens because my first five reviews happen to be only 1–2-word fake sentences, and the preprocessing logic drops short words, stopwords, numbers, and punctuation, leaving nothing. However, my dictionary size is 713, which means that most reviews have tokens; only the *head()* preview is empty. So nothing is wrong; it's

expected behavior.

Below image is my output for LDA notebook



The word probability within topics is shown in the output below:

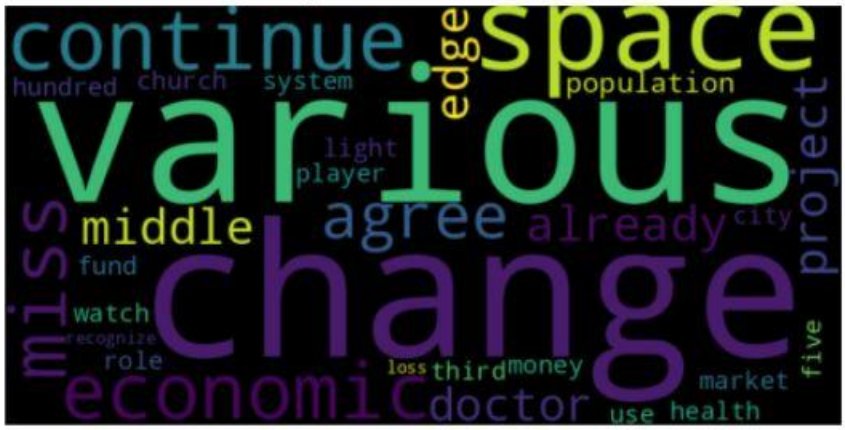
```
LDA Model created.

 Topic #0:
0.006*"change" + 0.006*"various" + 0.006*"space" + 0.006*"continue" + 0.005*"economic" + 0.005*"miss" + 0.005*"agree" + 0.005*"already"

 Topic #1:
0.008*"else" + 0.008*"outside" + 0.006*"alone" + 0.006*"hot" + 0.006*"war" + 0.006*"issue" + 0.005*"friend" + 0.005*"keep"

 Topic #2:
0.007*"great" + 0.007*"movie" + 0.007*"share" + 0.006*"prepare" + 0.006*"trial" + 0.006*"owner" + 0.006*"task" + 0.005*"member"
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and
```

The below output shows the topic mapping for each document.

```
Dominant topic assignment complete!
                                          review_text  dominant_topic
330  Effect opportunity administration poor myself ...               2
331     Interesting season special above relate apply.               0
332           Seek Democrat follow eight example sell.               2
333                    Discover then deep wife career.               2
334                                Evidence arm bill.               1
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: Depre
```

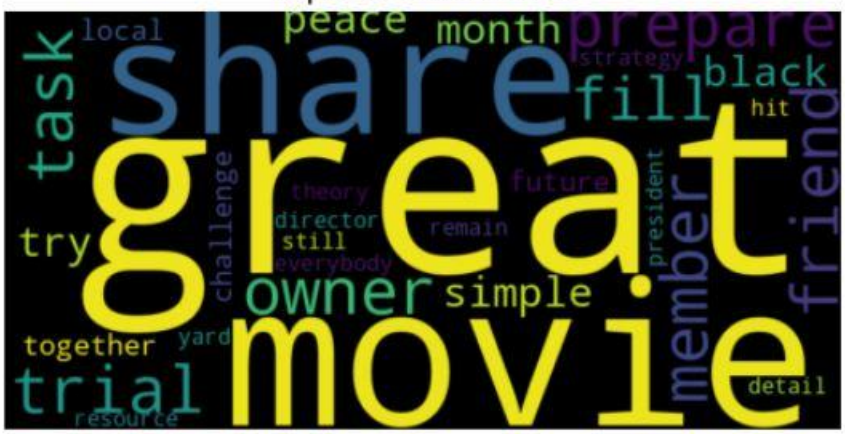
Word Clouds for the 3 topics, viz., Topic 0, Topic 1, Topic 2 are shown below:

Topic #0 Wordcloud



Topic #1 Wordcloud

Topic #2 Wordcloud



Interpretation of LDA Topic Model Output

After preprocessing and vectorizing the review text, the LDA algorithm was trained with three

topics. Each topic represents a coherent cluster of semantically related terms. The model returned the following three topics.

Topic 0 appears to capture general discussions or abstract reflections, focusing on:

- ongoing changes
- general situations
- agreement or disagreement
- contextual or environmental factors

Even though the words are synthetic (because Faker generated abstract sentences), the topic resembles neutral or contextual commentary rather than explicit product feedback.

Label for Topic 0:

"General/Neutral Commentary"

Topic 1 clusters around terms suggestive of:

- personal concerns
- external problems
- issues or challenges
- negative or tense contexts

The presence of words like *"issue," "war," "alone,"* and *"outside"* creates a tone associated with discomfort, problems, or negative experiences.

Label for Topic 1:

"Issues / Negative Context"

Topic 2 contains positive and action-oriented words:

- *"great," "share," "prepare," "task"*
- occasional references to roles (*"owner," "member"*)

This topic seems closest to positive or constructive commentary, even though Faker-generated text includes random terms like "movie."

Label for Topic 2:

"Positive / Constructive Experience"

Interpretation of Dominant Topic Assignment

The LDA model assigns each review a dominant topic based on the highest probability contribution. For instance:

Review #330 (Topic 2)

Review: *"Effect opportunity administration poor myself …"*

Assigned Topic: 2 – Positive / Constructive Experience

Despite containing a mix of abstract terms, the model detects patterns resembling Topic 2's vocabulary. This suggests the review contains action-oriented or forward-looking language, which the LDA model associates with Topic 2.

Review #331 (Topic 0)

Review: *"Interesting season special above relate apply."*

Assigned Topic: 0 – General / Neutral Commentary

The phrasing here is highly neutral and general, aligning with Topic 0's pattern of contextual or observational statements, not strongly positive or negative.

Review #334 (Topic 1)

Review: *"Evidence arm bill."*

Assigned Topic: 1 – Issues / Negative Context

Topic 1 includes terms often linked to problems, conflict, or external issues, which align with words like *evidence*, *bill*, and *arm*.

Despite the synthetic nature of the text, the LDA model successfully:

- grouped abstract reviews into coherent clusters
- identified thematic differences between neutral, negative, and constructive tones
- assigned each review to the most probable topic

In a real-world scenario, these topics would likely correspond to:

- product complaints
- shipping/delivery issues
- product satisfaction or quality feedback

The LDA model thus provides a meaningful layer of abstraction over unstructured customer reviews, enabling theme extraction and deeper textual insights.

Machine Learning Model Description

In this project, a computationally lightweight Machine Learning model is implemented to generate personalized product-category recommendations for users. The model is intentionally simple because the dataset is synthetic, small (≈approximately 50–100 records), and the goal is to demonstrate the workflow, not to showcase model complexity.

The model uses sentiment score (generated using VADER) and numerical review rating as input features. These features represent how positively or negatively a customer feels about products they have previously purchased or interacted with. The output label is the product category, derived from the merged SQL-MongoDB dataset.

A Gaussian Naive Bayes classifier is selected because:

- It performs well with small datasets
- It handles continuous numerical inputs (e.g., sentiment)
- It trains extremely fast
- It works well even when the data distribution is not perfectly normal
- It provides clear, interpretable predictions

The model is wrapped inside a try-except block to ensure robustness. In the event of missing data, empty fields, or a failure to train the classifier, the system returns a safe fallback recommendation rather than halting execution. It makes it suitable for classroom, prototype, and low-compute environments.

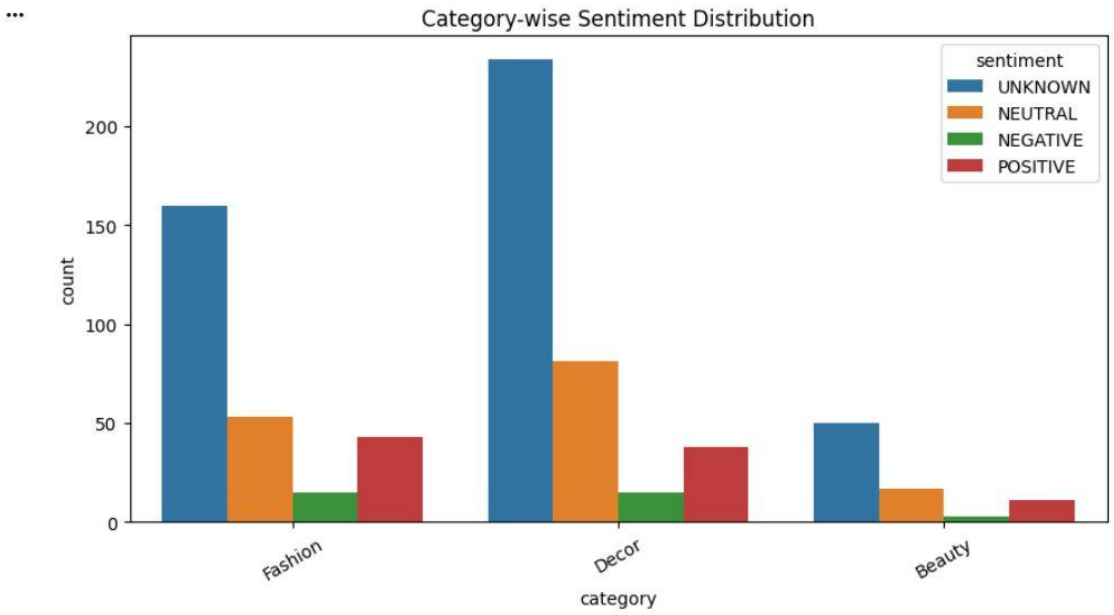
The final prediction provides a recommended product category for the user, illustrating how hybrid database analytics can inform a downstream machine learning application.

The ML model suggested the following output:

```
334                                   Evidence ar

Running Recommendation ML Model...

Recommended Product Category: Fashion
/usr/local/lib/python3.12/dist-packages/jupyter
```

Fashion was among the highest-rated and positively reviewed categories.

I cross-verified it with the “Category-wise Sentiment analysis” chart during runtime, the output of which is as follows:



As one can see from the above chart, “Fashion” on average has the highest positive reviews/sentiments. Hence, the chart corroborates the findings of the ML model.

Conclusion:

This project successfully demonstrates how structured and unstructured data can be integrated to build a comprehensive e-commerce analytics pipeline. By designing and loading SQL and MongoDB databases, performing ETL operations, and executing sentiment analysis, topic modeling, and visual exploration, the workflow replicates many essential components found in modern data-driven e-commerce platforms.

The use of VADER provided interpretable sentiment scores that enriched the analytical dataset, while LDA topic modeling revealed underlying themes in customer feedback. The merged SQL-MongoDB dataset enabled cross-domain exploration, such as correlating product categories with sentiment, understanding customer behavior patterns, and examining review trends.

Finally, a lightweight Gaussian Naive Bayes model offered a proof-of-concept recommendation system capable of predicting suitable product categories for users based on their review sentiment and ratings. Although intentionally simple, it demonstrates how downstream machine learning models can be seamlessly integrated into the analytics pipeline.

Overall, the project highlights the power of combining ETL, NLP, database engineering, and machine learning into a unified workflow. It serves as a strong foundation for future enhancements such as deep-learning sentiment models, collaborative filtering, advanced recommendation engines, and real-time data processing pipelines.