Est. 1841 | YORK ST JOHN UNIVERSITY

York St John University, London

Department of Data Science

Course Name: LDS7005M – Big Data and Cloud Computing

Designing of Cloud Platform Architecture for HARA Consulting.

Student Name: Jamiu Adeyemi Arogundade

Student ID: 240024714

Submission Date: 4th June, 2024

Est. 1841 | YORK ST JOHN UNIVERSITY

Module Title	Big Data and Cloud Computing
Module code	LDS7005M
Student ID	240024714
Submission Date	04– 06 - 2024

Student Declaration:

I confirm that I have read and understood the University Policy and Procedures on Academic Misconduct and that the work submitted is my own.	■
I confirm that I have processed and produced this submission in accordance with the University guidelines and the assessment brief regarding the use of generative AI. Where appropriate, I have acknowledged where and how it has been used. OR Where generative AI is not permitted in this assignment. I confirm that it has not been used in any part of the process or production of this submission https://www.yorksj.ac.uk/policies-and-documents/generative-artificial-intelligence/	■ ■

Table of Contents

Table of Figures

1.0. Introduction

Cloud Computing: Definition, Types, and Importance

Cloud computing is the usage of traditional computing resources over the internet popularly known as cloud. Since its widespread emergence around 2007, cloud computing has changed the way in which IT services are invented, developed, deployed, scaled, updated, maintained, and paid for (Marston S, 2011).

Cloud computing enabled the utilisation of computing services such as servers, storage, networking, databases, software, and more without having to invest in and maintain physical computer's hardware and infrastructure. This technology has allowed organisations and individuals to access and manage data and applications by providing various model solutions. These solutions can be categorised as:

Infrastructure as a Service: Model service that provides virtual computing resources over the internet that offer services like virtual machine, storage, and networks.

Platform as a Service: Model service that enables an on-demand virtual environment for developing, testing, launching, and managing software applications over the internet.

Software as a Service: This is also a model service that delivers a complete software application over the internet. Users can access software from any devices as long as they are connected to the internet.

Advantages of Cloud Computing:

- It offers scalability which allows resources to be scaled up or down according to need.
- It ensures data and software are accessible from anywhere with an internet connection.
- It provides backup and recoveries in the event of loss of data.
- It is cheaper than developing and managing physical servers and data centres.
- It is faster and more efficient than the physical hardware in processing resources.

1.1. Data Types at HARA:

Company Overview: HARA Consulting

HARA is a multinational consulting company specialising in finance, audit, assurance, and advisory services. Comparable to KPMG, PwC, and Deloitte, HARA serves a diverse, global clients across banking, insurance, healthcare, logistics, and technology sectors. The role of the data architect at the firm is to design scalable data architectures that effectively manage the voluminous diverse data and turning it into a strategic asset for the clients.

HARA's operations generate and process a wide array of data types:

Structured Data:

- Financial statements (balance sheets, cash flow statements).
- Transaction logs from banking clients.
- Compliance audit checklists.
- Market data (stock prices, bond yields).
- Customer demographics.

Unstructured Data:

- Audit reports and field notes.
- Client emails and communications.
- News articles affecting client industries.
- Legal contracts and regulations.
- Social media sentiment data.
- Customer reviews and feedback.

This diversity in data types, along with their volume and velocity, presents significant challenges in storage, processing, and analysis. To optimise the firm's big data processing capabilities, we are transitioning to Google's cloud-based architecture, taking opportunities of its comprehensive suite of services available with sustained use discounts and committed use discounts for long-term usage.

1.2. Why Google Cloud Platform (GCP)?

Google Cloud Platform (GCP) offers several intriguing features for big data analytics and machine learning, making it a strong competitor in the cloud market. Here's why GCP stands out:

Scalability

Google Cloud Platform (GCP) excels in scalability, primarily due to its robust global infrastructure and innovative features:

Global Network: GCP leverages Google's extensive global network, which provides lower latency and higher throughput. This is crucial for applications that requires rapid data transfers and large-scale data processing. GCP's network throughput was significantly faster compared to Azure (1.66 faster) and AWS (1.39 faster), enhancing its scalability for big data applications (Al-Said Ahmad, 2019).

Google Kubernetes Engine also known as GKE is a service provided by GCP to help user scaling up their applications and data to meet up with demand and work effectively in scaling down to manage costs. It's a fully managed environment for deploying, scaling, and

managing cloud applications. It’s has been regarded as the most scalable and automated platform that can run more than 10,000 node clusters in less than ten minutes. GKE out scales other cloud platforms up to fifteen times.

Live Migration: A unique feature of GCP is live migration, which allows virtual machines (VMs) to be moved between host systems without downtime. This capability ensures continuous service availability and scalability without interruptions (Inès Krissaane, 2020).

Redundancy

GCP offers robust redundancy features to ensure high availability and data durability:

Global Infrastructure: GCP operates in over 25 regions, providing a strong geo-redundant infrastructure. This extensive network is comparable to AWS (26) and surpasses Azure (22), which operates in fewer regions (Chen, 2021).

Colossus File System: GCP's Colossus file system offers built-in redundancy and high durability, boasting 99.999999999% durability. This level of data protection is achieved without requiring user configuration, making it more user-friendly and reliable (Mesbahi, 2018).

Cost-Effectiveness

GCP is also considered for the company for its cost-effective solutions:

Per-Second Billing: GCP started the per-second billing model for compute instances, allowing for more mealy and cost-effective billing compared to AWS's initial per-hour model and Azure's per-minute model. This approach helps users pay only for the exact usage, reducing unnecessary costs (Al-Said Ahmad, 2019).

Sustained Use Discounts: GCP offers automatic sustained use discounts, which are applied without requiring upfront commitments. This contrasts with AWS's Reserved Instances and Azure's pre-purchase plans, providing more flexibility and cost savings for users who have varying or unpredictable workloads (Inès Krissaane, 2020).

1.3. Types and Sources of Data

HARA consulting handles data from multiple sources, including:

Internal Systems: Enterprise Resource Planning (ERP) systems, Customer Relationships Management (CRM) systems, and Internal databases (sales record, etc.)

External Systems: Client data, social media platforms, third party databases, and customer review websites like Trustpilot.

1.4. Cloud-Based Storage Solution

For scalable, redundant, and cost-effective storage, Google Cloud Platform offers several services suitable for HARA's commitments:
Google Cloud Storage: Ideal for storing large amounts of unstructured data such as text or binary data.

BigQuery: A fully managed, serverless data warehouse that enables scalable analysis over petabytes of data.

Cloud SQL (NoSQL): Managed relational databases for structured data storage.

1.5. Data Ingestion Process

The data ingestion process involves transferring data from various sources to the cloud. Google cloud provides different tools for data movement which are:

BigQuery Data Transfer Service: This service scheduled and managed the transfer of data to the big query automatically. It supports migration of data from different sources and also offers No-code data ingestion (See: Documentation, BigQuery).

Pub/Sub: Shortened from Publishers and Subscribers, Pub/Sub is a scalable serverless messaging service used for streaming analytics that can communicate asynchronously, with latencies on the order of 100 milliseconds. It's mostly used in ingesting real-time event to the databases such as BigQuery and Cloud Storage (See: Documentation, Pub/Sub).

Dataflow: Dataflow is also a serverless unified stream and batch data processing data movement service that is built on Apache Beam for data processing pipeline. Dataflow provides autoscaling and dynamic load balancing, flexibility in running processes other Apache services like Spark and Flink. It also provides real-time AI patterns, streaming engine, horizontal and vertical scaling, data shuffle for fast processing, and other use future at a very low cost (see Documentation, Dataflow).

Datafusion: Cloud Data Fusion is a fully managed, cloud-native, enterprise data integration service for quickly building and managing data pipelines. The Cloud Data Fusion provide a web interface to build scalable data integration solutions and offers

connection to various data sources, transform the data, and then transfer it to various destination systems, without having to manage the infrastructure (see Documentation, Datafusion).

At HARA's consulting, considering the type of data we deal with, we have decided to integrate Cloud BigQuery orchestrating data workflows, and Google Cloud Storage for managing storage accounts. The two tools offer a serverless services for seamless data transfer to the cloud.

2.0. Scalable Processing Architecture

Scalability refers to the cloud-based system's capability to expand the capacity of software service delivery to meet increased demand over a period of time while exposed to fluctuations in demand for the service. This expansion involves increasing the quantity of the software service provided as needed (Lehrig S, 2015).

2.1. Design of Scalable Big Data Processing Architecture

A scalable processing architecture in Google Cloud Platform would consist of the following layers:

Data Ingestion: Cloud BigQuery and Dataflow to efficiently manage the movement of data.

Data Storage: Google Cloud Storage to store raw data, while processed data is stored in BigQuery.

Data Processing: BigQuery for distributed data processing and AI Platform using Google Colab notebook for editing.

Data Serving: BigQuery and Dataproc for data warehousing and analysis purposes.

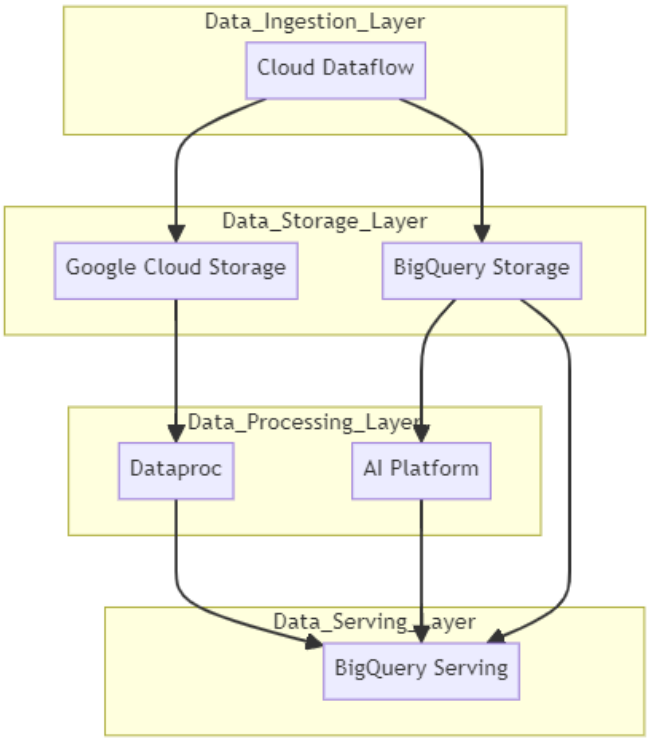


Figure 1: Scalable big Data Processing Architecture Flowchart

2.2. Choice of Cloud Services for Distributed Computing and Parallel Processing

For distributed computing and parallel processing, BigQuery and Dataproc services excels astonishingly.

BigQuery, a serverless tool for big data analysis and warehousing built on SQL and has integrated analytics features. It is excellent at handling big data processing and machine learning. The platform also provides Google Dataproc, a super strong service on Google Cloud Platform that uses Apache Hadoop and Spark for distributed computing and parallel processing. These two tools create a complete solution for the organisation regarding huge datasets and the need for scaled analytics services, all at a very low cost.

2.3. Accommodation of Growing Data Volumes

The designed architecture has been programmed to scale with accommodate data volumes. Google Cloud Platform's global networks enable swift data movement and the cloud's Dataproc service only requires 60 seconds startup which is excellent for auto-scaling. Independent scaling is allowed since the BigQuery storage is separated from the compute service. Moreover, the Cloud Storage and BigQuery services can handle

petabytes of data, this ensures that the architecture can accommodate future growth with ease.

3.0. Data Extraction and Pre-processing

Data Extraction and Pre-processing involves collecting and preparing data for analysis and machine learning. This includes gathering the dataset and importing it into the code-editor as a table. The imported data can be in various formats such as CSV, JSON, TSV, Excel Sheet, or any other file extension.

The preparation of the data involves cleaning it to improve its quality for accurate analysis and machine learning. This includes checking for missing values, outliers, duplicates, fixing spelling errors, and assigning the correct data types to columns.

Data preparation also includes feature engineering for future use and encoding data features.

3.1.

Taqwa Store, one of the clients at HARA, has consulted the firm to provide analysis and insights regarding their customers purchase activities. They want to know about the customers' satisfaction shopping at the store to understand what they are doing well and the areas for improvement.

Taqwa Store formally provide a platform where customers can provide feedback on their experiences after making an online purchase and this is the resources to leverage on in analysing the customer experiences.

The dataset was downloaded from Kaggle, created by CYNTHIA REMPEL.

Extraction of Customer Reviews

Taqwa store has a 'TSV' file containing reviews text of their customers for analysis. Converting the file to a 'CSV' is required to organise the file contents in an accurate manner.

For this project,

Jupyter Notebook will be used in importing the file from the local storage, converting the file to a 'CSV' extension and exporting to the Cloud Storage.

The Cloud Storage will serve as the container to store the file and other clients' files.

Cloud BigQuery will be leveraged for the analysis and visualisation.

Step one: Creating a Cloud new project.

Created a new project for Taqwa Store's Customer Reviews

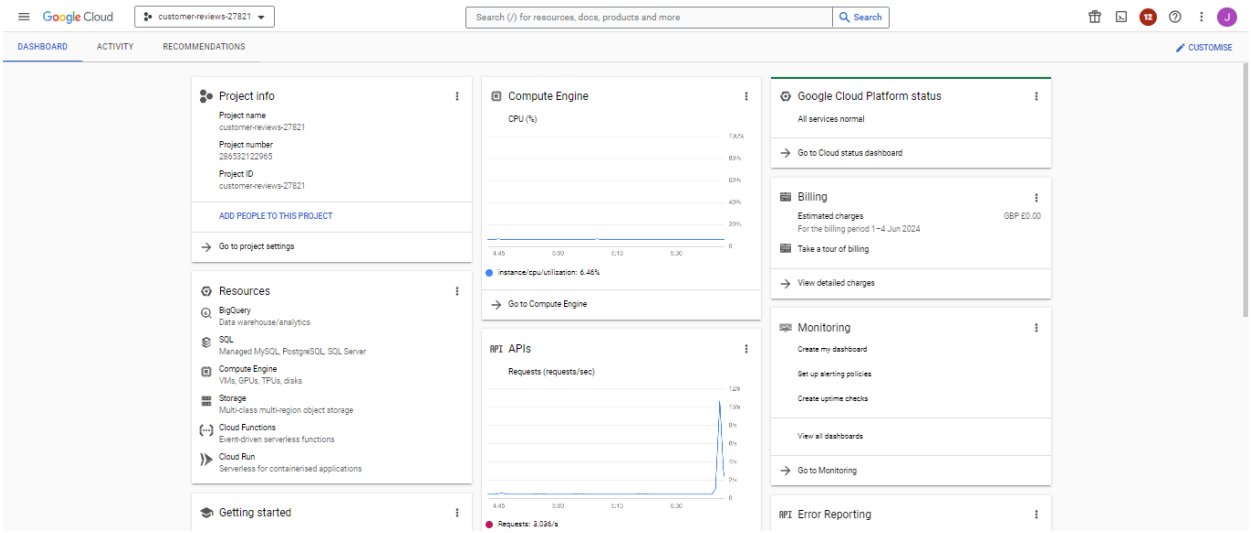


Figure 2: Dashboard after Creating a New Project

Created Virtual Machine for computing. Virtual Machine is the representation of the physical computer in the cloud.

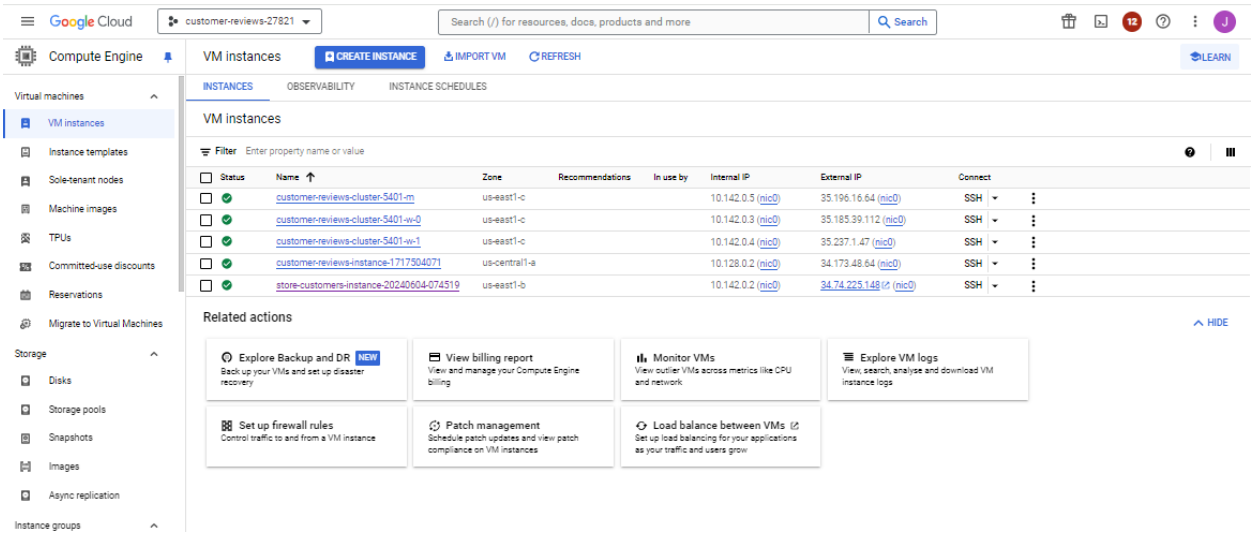


Figure 3: Virtual Machine for The Project

Step Two: Creating a Cloud Storage and Bucket

The storage account serves as the data store for keeping the data and can be accessed by any service both in the cloud platform and externally. After creating a storage account, a bucket was also created inside the storage account as the container for the dataset.

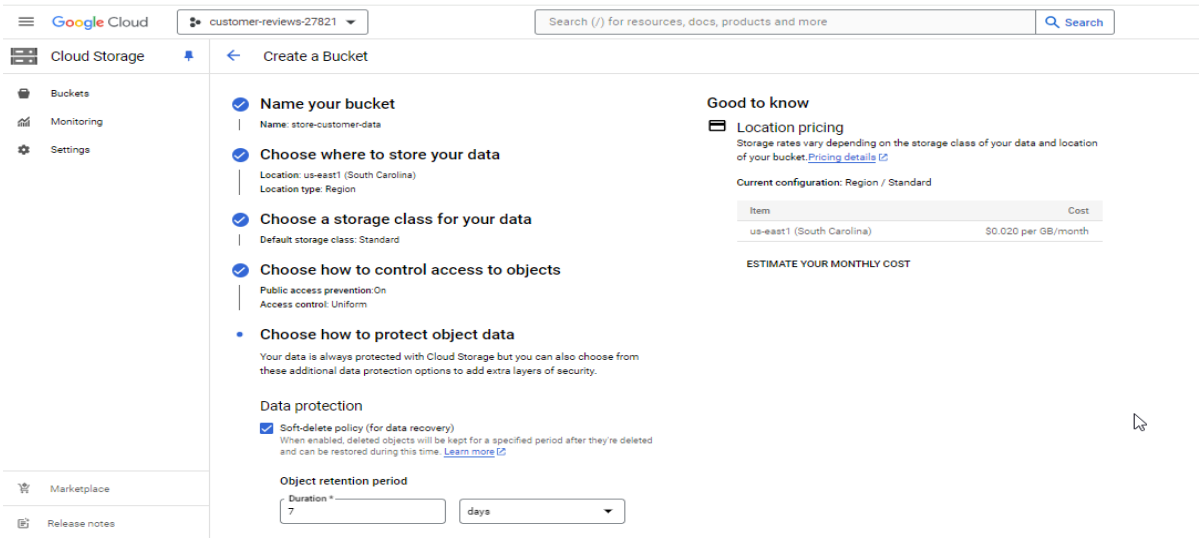


Figure 4: Created Storage Account and Bucket

Step Three: Conversion of file to CSV and exporting to the Cloud Storage on the Local Environment.

As explained above, the file was converted from TSV to CSV format for smooth data ingestion and transformation using the Python Pandas library.

```python
import pandas as pd
import csv
import warnings

# Ignore warnings
warnings.filterwarnings('ignore')

# Set pandas to display all columns
pd.set_option('display.max_columns', None)

# Define the file path
file_path = 'amazon_reviews_multilingual_US_v1_00.tsv'

# Read the data into a pandas DataFrame
df = pd.read_csv(file_path, sep='\t', error_bad_lines=False, warn_bad_lines=True, quoting=csv.QUOTE_NONE)

# Display the DataFrame
print(df.head())
```

```
  marketplace  customer_id       review_id  product_id  product_parent  \
0          US     53096384   R63J84G1LOX6R  1563890119       763187671
1          US     53096399  R1BALOA11Z06MT  1559947608       381720534
2          US     53096332  R1LLAY5W5PZUS4  0671701800       860650224
3          US     53096335  R3R9VTJ82FXECQ  0425132153       624269601
4          US     51747709  R1P5J3FNBWTFXY  0517122707       161411385

                              product_title product_category  star_rating  \
0  The Sandman Vol. 1: Preludes and Nocturnes            Books            4
1          The 22 Immutable Laws of Marketing            Books            4
2                                     Contact            Books            5
3                                  Good Omens            Books            5
4                     A Confederacy of Dunces            Books            5

   helpful_votes  total_votes vine verified_purchase  \
0              0            1    N                N
1              0            0    N                N
2              1            2    N                N
3              0            0    N                N
4              0            0    N                N

                                        review_headline  \
```

Figure 5: Conversion of file from TSV to CSV.

After the conversion, the newly converted CSV file was exported to the Cloud Storage's Bucket.

```python
from google.oauth2 import service_account

# Load service account credentials
credentials = service_account.Credentials.from_service_account_file(
    'customer-reviews-27821-d3e04d90f4f7.json')

# Initialize the Google Cloud Storage client with credentials
client = storage.Client(credentials=credentials)

# Define the bucket name and the destination file name
bucket_name = 'store-customer-data'
destination_blob_name = 'customer_reviews.csv'

# Get the bucket
bucket = client.bucket(bucket_name)

# Create a new blob and upload the file's content
blob = bucket.blob(destination_blob_name)

# Upload to cloud
try:
    blob.upload_from_filename(export_file_path)
    print(f'File {export_file_path} uploaded to {bucket_name}/{destination_blob_name}.')
except Exception as e:
    print(f'Failed to upload file: {e}')
```

```
File customer_reviews.csv uploaded to store-customer-data/customer_reviews.csv.
```

Figure 6: Exported CSV file to the Cloud Storage Bucket

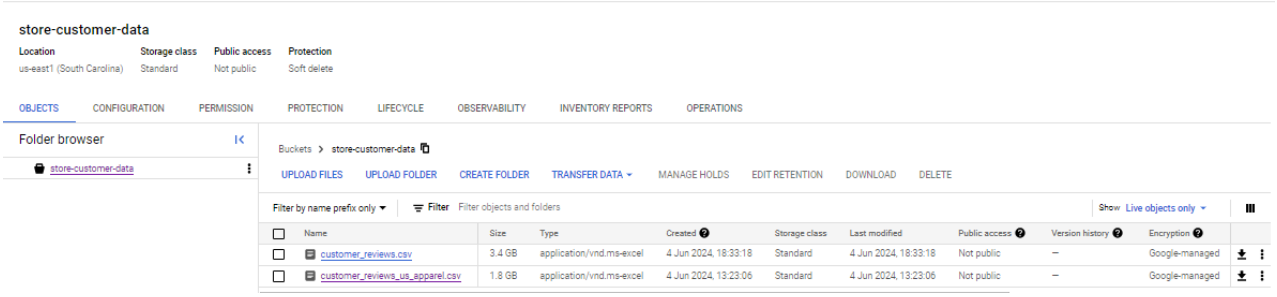


Figure 7: Bucket State after Exporting files

Step 4: Data Transfer from Cloud Storage to BigQuery for Analysis

The next step is the data movement from the Cloud Storage to the BigQuery for data cleaning, transformation, and analysis.

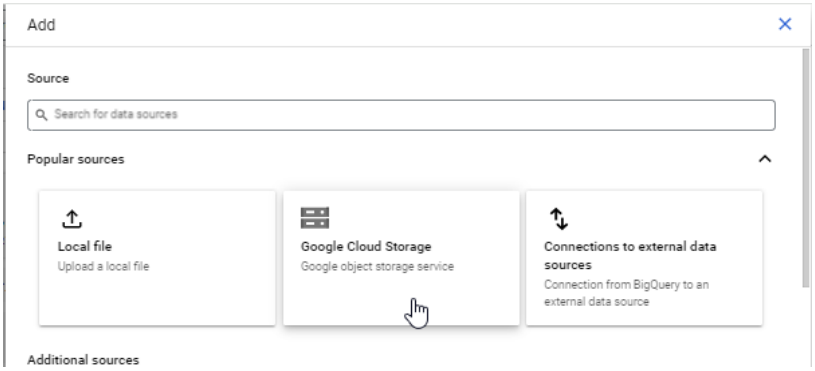


Figure 8: Adding Data to the BigQuery

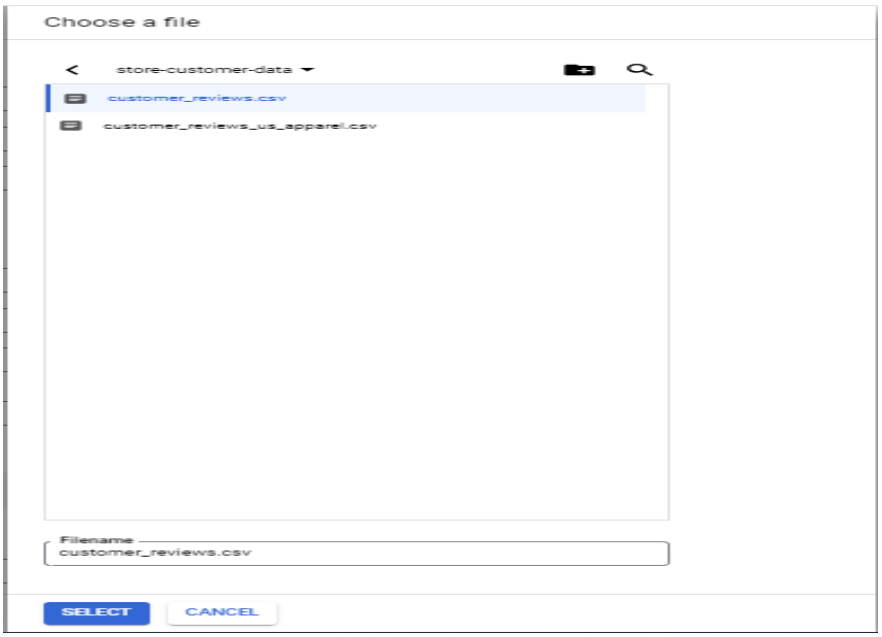


Figure 9: Select the file from Cloud Storage

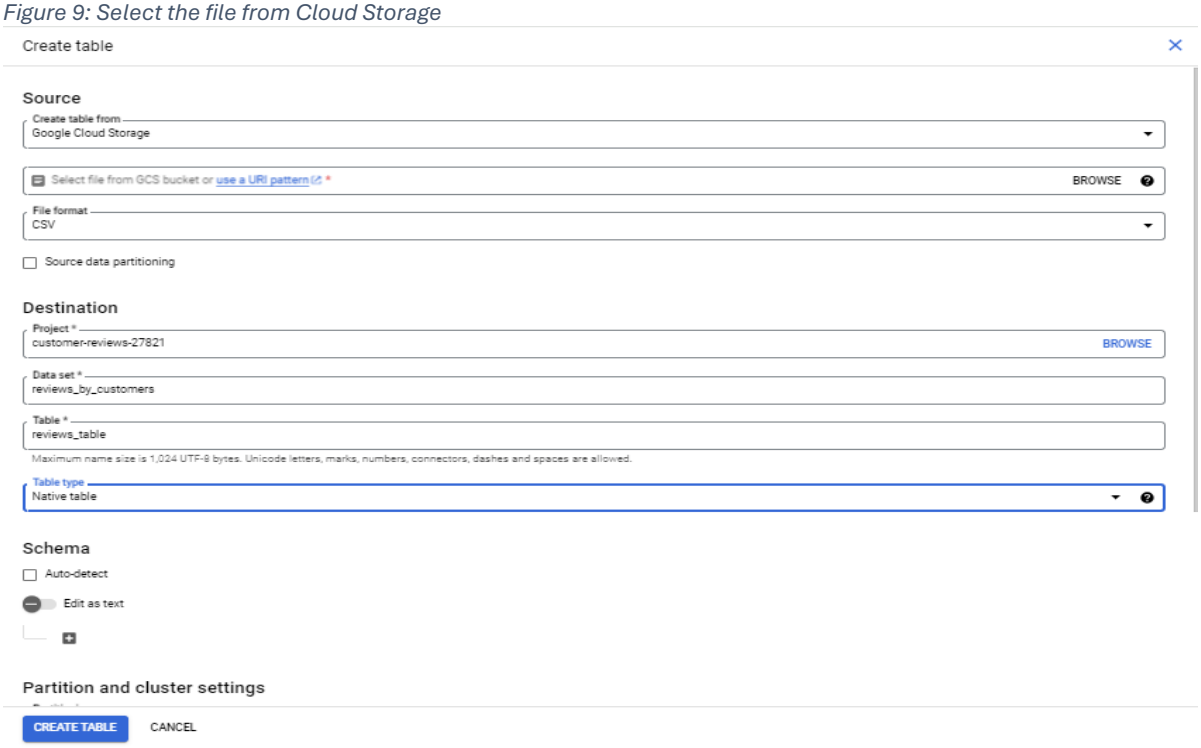


Figure 10: Created a BigQuery Table

Step 5: Launching JupyterLab Notebook from Vertex AI

Google Cloud Platform provides Vertex AI service which is used in creating instances for code-editor. An instance of Jupyter Notebook was created to connect to the BigQuery Table in order to do data analysis from the Notebbok.

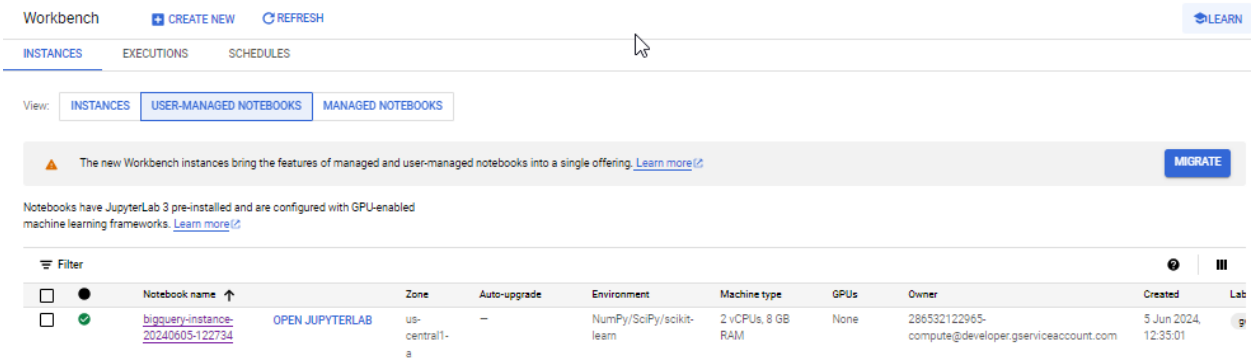


Figure 11: Launced JupyterLab Notebook from Vertex AI

Step 6: Connecting to BigQuery and Importing Table into JupyLab Notebook

Instantiate Service Access to BigQuery Client

```
from google.cloud import bigquery

key_path = 'customer-reviews-27821-ce7f459d0a0a.json'

client = bigquery.Client.from_service_account_json(key_path)
```

Import Dataset from Big

```
query = """
SELECT *
FROM `customer-reviews-27821.reviews_by_customers.reviews_table`
WHERE RAND() < (1150000 / (SELECT COUNT(*) FROM `customer-reviews-27821.reviews_by_customers.reviews_table`))
"""

query_data = client.query(query)

df = query_data.to_dataframe()

df.head()
```

	marketplace	customer_id	review_id	product_id	product_parent	product_title	product_category	star_rating	helpful_votes	total_votes	vine	verified_purchase	review_headline	review_body	review_date
0	US	52529112	R19O08234WZ3M0	B0042S51H8	386765193	Plugable USB to RS-232 DB9 Serial Adapter (Pro...	PC	3	0	1	False	True	Worked but required an additional adapter	The adapter came with female threaded nuts tha...	2013-01-09
1	US	43856501	R2NOPGMUC39HO	B006GWO5NE	564071965	Amazon 5ft USB to Micro-USB Cable (works with ...	PC	1	0	0	False	False	problem with kindle	This did not solve my kindle problem and i don...	2013-08-19
2	US	33278723	R2MW15DTHRRMSB	B008HK3XKO	893861753	Marware Vibe Standing Case for Kindle Fire HD ...	PC	5	0	0	False	True	kindle cover	Fit the kindle fire hd perfectly. The black wa...	2013-01-28
3	US	21951802	R3KDC1O75FRIUS	B008RXYOKY	766086893	Apple Thunderbolt to FireWire Adapter	PC	5	0	0	False	True	Would recommend to a friend.	Fits my needs. Happy with my purchase. Well c...	2014-03-05
4	US	43248065	R3EXUH02885PU6	B0095VP8D4	792792668	AMD A10-5800K APU 3.8Ghz Processor AD580KWOHJBOX	PC	5	1	2	False	True	Nice Product	I love this processor. It's so fast and quite ...	2013-03-24

Figure 12: Connected BigQuery Table

Data Exploration and Analysis

```
print(f"Dataset has {df.shape[0]} rows and {df.shape[1]} columns")
```

```
Dataset has 3250246 rows and 15 columns
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3250246 entries, 0 to 3250245
Data columns (total 15 columns):
 #   Column             Dtype
---  ------             -----
 0   marketplace        object
 1   customer_id        Int64
 2   review_id          object
 3   product_id         object
 4   product_parent     Int64
 5   product_title      object
 6   product_category   object
 7   star_rating        Int64
 8   helpful_votes      Int64
 9   total_votes        Int64
 10  vine               boolean
 11  verified_purchase  boolean
 12  review_headline    object
 13  review_body        object
 14  review_date        dbdate
dtypes: Int64(5), boolean(2), dbdate(1), object(7)
memory usage: 350.3+ MB
```

```
df.isnull().sum()
```

Figure 13: Dataframe Shape and Info.

Dropping Null Values:

Dropped Null Values from the dataset.

```
df.isnull().sum()
```

```
marketplace           0
customer_id           0
review_id             0
product_id            0
product_parent        0
product_title         0
product_category      0
star_rating           0
helpful_votes         0
total_votes           0
vine                  0
verified_purchase     0
review_headline      40
review_body          29
review_date           0
dtype: int64
```

```
df.dropna(subset=['review_headline','review_body'])
```

	marketplace	customer_id	review_id	product_id	product_parent	product_title	product_category
0	US	28184012	R3T0JMIE2WOW53	B00687PWQG	661839716	John Adams Season 1	Digital_Video_Download
1	US	46377182	R1DSO3VMV63MSY	B009NPKTN6	877921727	Rock of Ages (2012)	Digital_Video_Download
2	US	47659188	R3IOQVWIU1JCM2	B00BF11STU	203465797	Zero Dark Thirty	Digital_Video_Download
3	US	47503411	R21A120094I1JE	B00GX59QO2	645382359	Red 2	Digital_Video_Download
4	US	1700003	R373ARADQ1OMI	B00HY7YXIM	592331046	The White Queen, Season 1	Digital_Video_Download
...	...	...	...	...	...	...	...

Figure 14: Drooped Null Values

4.0. Analysis and Insights

Sentiment analysis was done using Textblob library to analyse the customer views on the products.

```python
import pandas as pd
import numpy as np
from multiprocessing import Pool, cpu_count
from textblob import TextBlob

# Define a function to analyze sentiment using TextBlob
def analyze_sentiment(text):
    if text is not None:
        blob = TextBlob(text)
        return blob.sentiment.polarity
    else:
        return None

# Define a function to map sentiment score to category
def map_sentiment_category(score):
    if score is None:
        return 'Neutral'
    elif score > 0:
        return 'Positive'
    elif score < 0:
        return 'Negative'
    else:
        return 'Neutral'

# Function to apply sentiment analysis to a chunk of data
def process_chunk(chunk):
    chunk['sentiment_score'] = chunk['review_body'].apply(analyze_sentiment)
    chunk['sentiment_category'] = chunk['sentiment_score'].apply(map_sentiment_category)
    return chunk

# Function to parallelize the sentiment analysis
def parallelize_dataframe(df, func, num_partitions):
    df_split = np.array_split(df, num_partitions)
    pool = Pool(cpu_count())
    df = pd.concat(pool.map(func, df_split))
    pool.close()
    pool.join()
    return df

# Number of partitions for parallel processing
num_partitions = cpu_count() * 2

# Apply the sentiment analysis in parallel
df_result = parallelize_dataframe(df, process_chunk, num_partitions)

print(df_result.head())

# Save the result to a CSV file
df_result.to_csv('sentiment_analysis_results.csv', index=False)
```

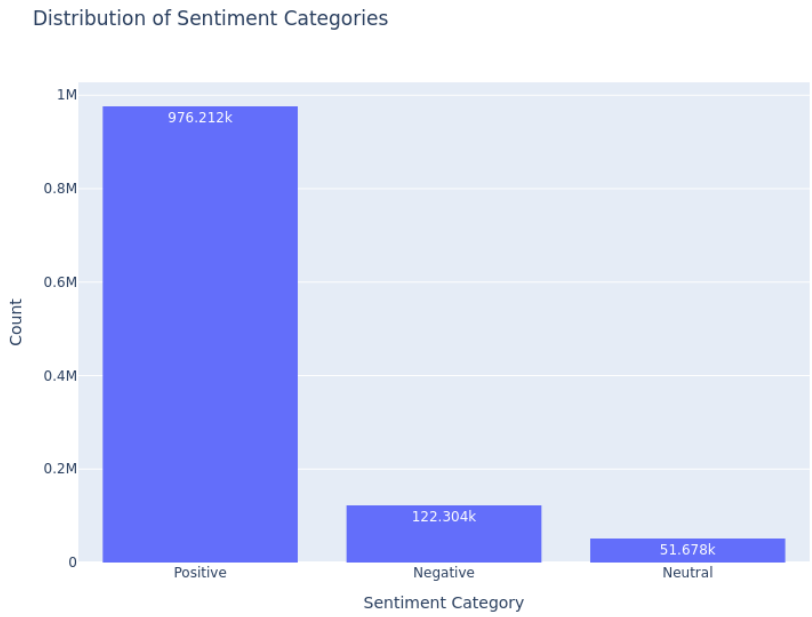
Figure 15: Sentiment Analysis

```python
df.sentiment_category.value_counts()
```

```
sentiment_category
Positive    976212
Negative    122304
Neutral      51678
Name: count, dtype: int64
```

Figure 16: Sentiment Result.

Distribution of Sentiment Category

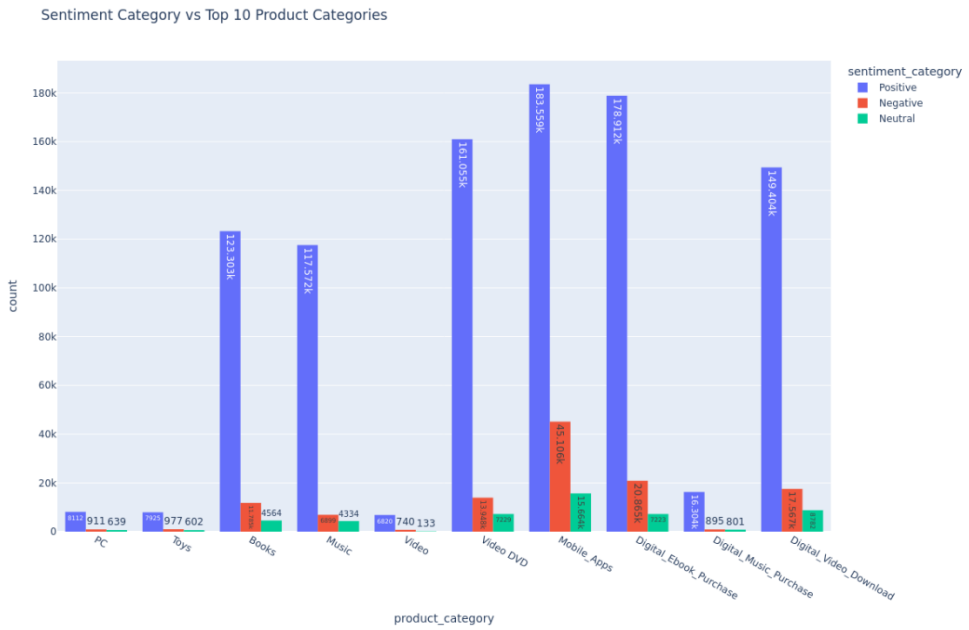


Positive Reviews: There are significantly more positive reviews (976.212k) compared to negative and neutral reviews. This shows that the many of the customers had a good experience with their purchases.

Negative Reviews: There are 122.304k negative reviews, indicating a smaller but notable portion of customers were dissatisfied.

Neutral Reviews: There are 51.678k neutral reviews, suggesting that a relatively small portion of customers had a mixed experience.

Distribution of Product Category by Sentiment Category



High Positive Reviews: Product categories like Mobile Apps, Digital Ebook, Video DVD, Digital Video Downloaded, Books, and Music have high counts of positive reviews, indicating high customer satisfaction.

Negative Reviews: Categories such as Books, Music, Mobile Apps, and Digital Ebook Purchase have a higher number of negative reviews compared to other categories. This indicates some level of dissatisfaction in these areas. But they also have high positive reviews too compared to the negatives.

Neutral Reviews: The distribution of neutral reviews is relatively consistent across all categories, with no significant peaks, suggesting mixed experiences are less common across different product types.

Distribution of Star Rating and Sentiment Category

Five-Star Reviews: The highest number of five-star reviews (604.752k) is positive, which aligns with the high count of positive sentiment reviews.

Four-Star Reviews: Four-star reviews (183.588k) are also predominantly positive.

Three-Star Reviews: Three-star reviews show a balanced mix but lean towards positive sentiment.

One and Two-Star Reviews: These ratings are more likely to be negative, as expected, with significant counts for one-star (29.471k) and two-star (31.119k) ratings being negative.

These visualizations can help the client to understand customers' sentiment, the value of reviews, and areas where products or services might need improvement.

The dataset shows a dominance of positive reviews, indicating overall customer satisfaction. Both positive and negative reviews are considered helpful, with positive reviews being the most valued. Some product categories like PC, Toys, and Videos have higher negative reviews, which could indicate areas for improvement.

In addition, there's a clear correlation between higher star ratings and positive sentiment, as well as lower star ratings and negative sentiment, which validates the sentiment analysis process.

5. Cost Optimization Strategies

To optimize price in running a cloud platform, most of the analysis from data preparation and analysis were done outside the Cloud console. Using both Jupyter and JupyterLab Notebook, connection was made to the cloud console using the generated service access key and the services API and are as efficient as carrying out the process in the Cloud console.

Stopping of processes when not in used also help in optimizing price for example the Compute Engine.

Selecting the low cost and balanced machines also played crucial roles in saving cost but the have their limitations of slow computing and limited resources unlike the high-performance machines. Locations also played an important role too.

6. Security and Compliance

The IAM service provided by Google Cloud Storage helped in restriction permission from outside to the firm Virtual Machine. It allowed the cloud architect to be able to grants and restricts access. Only approved individuals have access to the Cloud resources.

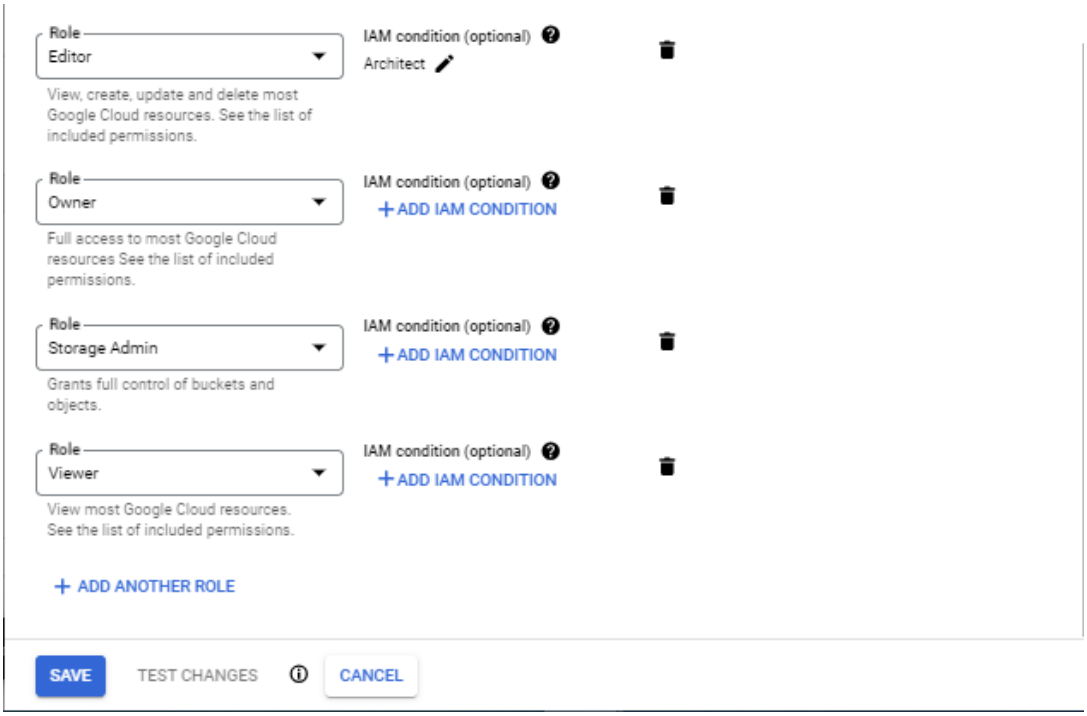


Figure 17: Google Cloud IAM.

Google Cloud Platform also provides option to grantor restrict access to the datasets stored in the Cloud Storage for public use.

7. Conclusion

Google Cloud Platform services such as BigQuery and Dataflow for ingestion, Cloud Storage for storage, Dataproc for processing, and AI Platform for analysis and visualisation were efficient enough to handle the overall data pipeline for sentiment analysis of Taqwa Store's customers' reviews. This scalable and cost-effective Cloud solution ensures that HARA Consulting can derive meaningful insights from customer reviews to enhance decision-making, improve client services, and secure client data from malicious attackers by leveraging the Cloud service IAM security. All these services provided by Cloud Platforms are more efficient, durable, scalable, secure, and cheaper than the traditional physical computer and storage.

8. Recommendations

Incorporating advanced machine learning models provided by Google Cloud Platform to improve sentiment analysis and other modelling accuracy.

Schedule Dataflow jobs to regularly ingest new data for continuous analysis. Streaming of data using services like Pub/Sub and Dataflow are ideal for timely dataflow into the Cloud Storage and BigQuery Storage.

Regular monitoring of resources will help against managing cost and resources and also protect against data theft.

9. Limitations and Caveats

Google Cloud Platform offers a variety of solutions for Cloud Computing, many of which provide similar functionalities. To effectively utilize these services, a deep understanding of the platform is necessary.

It is also important to note that users with limited funds may encounter certain limitations in terms of what they can achieve on the platform.

References

Al-Said Ahmad, A. A. P., 2019. Scalability analysis comparisons of cloud-based software services.. *Journal of Cloud Computing,* 8(1)(https://rdcu.be/dJzKe), pp. 1-17.

Chen, J. D. T. &. X. G. .. 1. 2. (., 2021. A multi-objective optimization for resource allocation of emergent demands in cloud computing. *Journal Cloud Computing,* Volume 10.

Inès Krissaane, C. D. N. A. G.-S. G. K. N. E. R. K. J. L. A. M. C. P. I. K. P. A., 2020. Scalability and cost-effectiveness analysis of whole genome-wide association studies on Google Cloud Platform and Amazon Web Services. *Journal of the American Medical Informatics Association,* 27(9), p. 1425–1430.

Lehrig S, E. H. B. S., 2015. Scalability, elasticity, and efficiency in cloud computing: a systematic literature review of definitions and metrics.. *Proceedings of the 11th international ACM SIGSOFT conference on quality of software architectures ,* p. 83–92.

Marston S, L. Z. B. S. Z. J. G. A., 2011. Cloud computing — the business perspective.. *Decis Support Syst 51,* p. 176–189.

Mesbahi, M. R. A. &. H. M., 2018. Reliability and high availability in cloud computing environments: a reference roadmap. *Human-centric Computing and Information Sciences,* Volume 8.

https://cloud.google.com/bigquery/docs

https://cloud.google.com/dataflow/docs

https://cloud.google.com/dataproc/docs

https://cloud.google.com/datafusion/docs

https://cloud.google.com/storage/docs

https://cloud.google.com/vertex-ai/docs

In [21]:

```
import pandas as pd
import csv
import warnings

# Ignore warnings
warnings.filterwarnings('ignore')

# Set pandas to display all columns
pd.set_option('display.max_columns', None)

# Define the file path
file_path = 'amazon_reviews_multilingual_US_v1_00.tsv'

# Read the data into a pandas DataFrame
df = pd.read_csv(file_path, sep='\t', error_bad_lines=False, warn_bad_lin

# Display the DataFrame
print(df.head())
```

```
  marketplace  customer_id       review_id  product_id  product_parent  \
0          US     53096384   R63J84G1LOX6R  1563890119       763187671
1          US     53096399  R1BALOA11Z06MT  1559947608       381720534
2          US     53096332  R1LLAY5W5PZUS4  0671701800       860650224
3          US     53096335  R3R9VTJ82FXECQ  0425132153       624269601
4          US     51747709  R1P5J3FNBWTFXY  0517122707       161411385

                                product_title product_category  star_rati
ng  \
0  The Sandman Vol. 1: Preludes and Nocturnes            Books
4
1          The 22 Immutable Laws of Marketing            Books
4
2                                     Contact            Books
5
3                                  Good Omens            Books
5
4                     A Confederacy of Dunces            Books
5

   helpful_votes  total_votes vine verified_purchase  \
0              0            1    N                 N
1              0            0    N                 N
2              1            2    N                 N
3              0            0    N                 N
4              0            0    N                 N

                                   review_headline  \
0                           ignore the review below
1                                           awesome
2                         Read the book. It's good.
3   Funniest book ever written about the Anti-Christ
4  A winner that didn't last. Only a mothers love...

                                        review_body review_date
0  this is the first 8 issues of the series. it i...  1995-08-13
1  I've always been partial to immutable laws. Th...  1995-08-17
2  This is a book about first contact with aliens...  1995-08-30
3  This is quite possibly *the* funniest book I h...  1995-09-11
4  The story behind the book is almost better tha...  1995-10-17
```

In [29]:

```
print(f"The Dataframe has {df.shape[0]} and {df.shape[1]} columns")
```

```
The Dataframe has 6931166 and 15 columns
```

In [30]:

```
from google.oauth2 import service_account

# Load service account credentials
credentials = service_account.Credentials.from_service_account_file(
    'customer-reviews-27821-d3e04d90f4f7.json')

# Initialize the Google Cloud Storage client with credentials
client = storage.Client(credentials=credentials)

# Define the bucket name and the destination file name
bucket_name = 'store-customer-data'
destination_blob_name = 'customer_reviews.csv'

# Get the bucket
bucket = client.bucket(bucket_name)

# Create a new blob and upload the file's content
blob = bucket.blob(destination_blob_name)

# Upload to cloud
try:
    blob.upload_from_filename(export_file_path)
    print(f'File {export_file_path} uploaded to {bucket_name}/{destinatio
except Exception as e:
    print(f'Failed to upload file: {e}')
```

```
File customer_reviews.csv uploaded to store-customer-data/customer_review
s.csv.
```

Import Libraries

In [1]:
```
!pip list | grep google-cloud-bigquery
```

```
google-cloud-bigquery                   3.22.0
google-cloud-bigquery-storage           2.25.0
```

In [2]:
```
!pip install google-cloud-storage google-cloud-bigquery pandas textblob plotly
```

```
Requirement already satisfied: google-cloud-storage in /opt/conda/lib/python3.10/site-packages (2.14.0)
Requirement already satisfied: google-cloud-bigquery in /opt/conda/lib/python3.10/site-packages (3.22.0)
Requirement already satisfied: pandas in /opt/conda/lib/python3.10/site-packages (2.0.3)
Collecting textblob
  Downloading textblob-0.18.0.post0-py3-none-any.whl.metadata (4.5 kB)
Requirement already satisfied: plotly in /opt/conda/lib/python3.10/site-packages (5.22.0)
Requirement already satisfied: google-auth<3.0dev,>=2.23.3 in /opt/conda/lib/python3.10/site-packages (from google-clo
ud-storage) (2.29.0)
Requirement already satisfied: google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,>=1.31.5 in /opt/conda/lib/pyt
hon3.10/site-packages (from google-cloud-storage) (1.34.1)
Requirement already satisfied: google-cloud-core<3.0dev,>=2.3.0 in /opt/conda/lib/python3.10/site-packages (from googl
e-cloud-storage) (2.4.1)
Requirement already satisfied: google-resumable-media>=2.6.0 in /opt/conda/lib/python3.10/site-packages (from google-c
loud-storage) (2.7.0)
Requirement already satisfied: requests<3.0.0dev,>=2.18.0 in /opt/conda/lib/python3.10/site-packages (from google-clou
d-storage) (2.31.0)
Requirement already satisfied: google-crc32c<2.0dev,>=1.0 in /opt/conda/lib/python3.10/site-packages (from google-clou
d-storage) (1.5.0)
Requirement already satisfied: packaging>=20.0.0 in /opt/conda/lib/python3.10/site-packages (from google-cloud-bigquer
y) (24.0)
Requirement already satisfied: python-dateutil<3.0dev,>=2.7.2 in /opt/conda/lib/python3.10/site-packages (from google-
cloud-bigquery) (2.9.0)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.10/site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in /opt/conda/lib/python3.10/site-packages (from pandas) (2024.1)
Requirement already satisfied: numpy>=1.21.0 in /opt/conda/lib/python3.10/site-packages (from pandas) (1.25.2)
Collecting nltk>=3.8 (from textblob)
  Downloading nltk-3.8.1-py3-none-any.whl.metadata (2.8 kB)
Requirement already satisfied: tenacity>=6.2.0 in /opt/conda/lib/python3.10/site-packages (from plotly) (8.3.0)
Requirement already satisfied: googleapis-common-protos<2.0dev,>=1.56.2 in /opt/conda/lib/python3.10/site-packages (fr
om google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,>=1.31.5->google-cloud-storage) (1.63.0)
Requirement already satisfied: protobuf!=3.20.0,!=3.20.1,!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<4.0.0d
ev,>=3.19.5 in /opt/conda/lib/python3.10/site-packages (from google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,
>=1.31.5->google-cloud-storage) (3.20.3)
Requirement already satisfied: grpcio<2.0dev,>=1.33.2 in /opt/conda/lib/python3.10/site-packages (from google-api-core
[grpc]!=2.0.*,!=2.1.*,!=2.10.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.*,!=2.9.*,<3.0.0dev,>=1.34.1->goo
gle-cloud-bigquery) (1.63.0)
Requirement already satisfied: grpcio-status<2.0dev,>=1.33.2 in /opt/conda/lib/python3.10/site-packages (from google-a
pi-core[grpc]!=2.0.*,!=2.1.*,!=2.10.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.*,!=2.9.*,<3.0.0dev,>=1.3
4.1->google-cloud-bigquery) (1.48.2)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /opt/conda/lib/python3.10/site-packages (from google-auth<3.0
dev,>=2.23.3->google-cloud-storage) (5.3.3)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /opt/conda/lib/python3.10/site-packages (from google-auth<3.0d
ev,>=2.23.3->google-cloud-storage) (0.4.0)
Requirement already satisfied: rsa<5,>=3.1.4 in /opt/conda/lib/python3.10/site-packages (from google-auth<3.0dev,>=2.2
3.3->google-cloud-storage) (4.9)
Requirement already satisfied: click in /opt/conda/lib/python3.10/site-packages (from nltk>=3.8->textblob) (8.1.7)
Requirement already satisfied: joblib in /opt/conda/lib/python3.10/site-packages (from nltk>=3.8->textblob) (1.4.2)
Collecting regex>=2021.8.3 (from nltk>=3.8->textblob)
  Downloading regex-2024.5.15-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (40 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 40.9/40.9 kB 771.8 kB/s eta 0:00:00 0:00:01
Requirement already satisfied: tqdm in /opt/conda/lib/python3.10/site-packages (from nltk>=3.8->textblob) (4.66.4)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.10/site-packages (from python-dateutil<3.0dev,>=2.7.
2->google-cloud-bigquery) (1.16.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/conda/lib/python3.10/site-packages (from requests<3.0.
0dev,>=2.18.0->google-cloud-storage) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.10/site-packages (from requests<3.0.0dev,>=2.18.
0->google-cloud-storage) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/conda/lib/python3.10/site-packages (from requests<3.0.0dev,>
=2.18.0->google-cloud-storage) (1.26.18)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.10/site-packages (from requests<3.0.0dev,>
=2.18.0->google-cloud-storage) (2024.2.2)
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in /opt/conda/lib/python3.10/site-packages (from pyasn1-modules>=
0.2.1->google-auth<3.0dev,>=2.23.3->google-cloud-storage) (0.6.0)
Downloading textblob-0.18.0.post0-py3-none-any.whl (626 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 626.3/626.3 kB 4.9 MB/s eta 0:00:00a 0:00:01
Downloading nltk-3.8.1-py3-none-any.whl (1.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.5/1.5 MB 24.8 MB/s eta 0:00:00a 0:00:01
Downloading regex-2024.5.15-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (775 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 775.1/775.1 kB 57.0 MB/s eta 0:00:00
Installing collected packages: regex, nltk, textblob
Successfully installed nltk-3.8.1 regex-2024.5.15 textblob-0.18.0.post0
```

Instantiate Service Access to BigQuery Client

In [3]:
```
from google.cloud import bigquery

key_path = 'customer-reviews-27821-ce7f459d0a0a.json'

client = bigquery.Client.from_service_account_json(key_path)
```

Import Dataset from Big

In [4]:
```
query = """
SELECT *
FROM `customer-reviews-27821.reviews_by_customers.reviews_table`
WHERE RAND() < (1150000 / (SELECT COUNT(*) FROM `customer-reviews-27821.reviews_by_customers.reviews_table`))
"""

query_data = client.query(query)

df = query_data.to_dataframe()

df.head()
```

Out[4]:

	marketplace	customer_id	review_id	product_id	product_parent	product_title	product_category	star_rating	helpful_votes	to
0	US	52529112	R19O08234WZ3M0	B00425S1H8	386765193	Plugable USB to RS-232 DB9 Serial Adapter (Pro...	PC	3	0	
1	US	43856501	R2NOPGMUC39HO	B006GWO5NE	564071965	Amazon 5ft USB to Micro-USB Cable (works with ...	PC	1	0	
2	US	33278723	R2MW15DTHRRMSB	B008HK3XKO	893861753	Marware Vibe Standing Case for Kindle Fire HD ...	PC	5	0	
3	US	21951802	R3KOC1O75FRIUS	B008RXYOKY	766086893	Apple Thunderbolt to FireWire Adapter	PC	5	0	
4	US	43248065	R3EXUH0288SPU6	B0095VP8D4	792792668	AMD A10-5800K APU 3.8Ghz Processor AD580KWOHJBOX	PC	5	1	

In [5]:
```
print(f"Dataset has {df.shape[0]} rows and {df.shape[1]} columns")
```

```
Dataset has 3250246 rows and 15 columns
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3250246 entries, 0 to 3250245
Data columns (total 15 columns):
 #   Column             Dtype
---  ------             -----
 0   marketplace        object
 1   customer_id        Int64
 2   review_id          object
 3   product_id         object
 4   product_parent     Int64
 5   product_title      object
 6   product_category   object
 7   star_rating        Int64
 8   helpful_votes      Int64
 9   total_votes        Int64
 10  vine               boolean
 11  verified_purchase  boolean
 12  review_headline    object
 13  review_body        object
 14  review_date        dbdate
dtypes: Int64(5), boolean(2), dbdate(1), object(7)
memory usage: 350.3+ MB
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
marketplace           0
customer_id           0
review_id             0
product_id            0
product_parent        0
product_title         0
product_category      0
star_rating           0
helpful_votes         0
total_votes           0
vine                  0
verified_purchase     0
review_headline      40
review_body          29
review_date           0
dtype: int64
```

In [8]:
```
df.dropna(subset=['review_headline','review_body'])
```

Out[8]:

	marketplace	customer_id	review_id	product_id	product_parent	product_title	product_category	star_rating	helpful_votes
0	US	28184012	R3T0JMIE2WOW53	B00687PWQG	661839716	John Adams Season 1	Digital_Video_Download	5	(
1	US	46377182	R1DSO3VMV63MSY	B009NPKTN6	877921727	Rock of Ages (2012)	Digital_Video_Download	4	(
2	US	47659188	R3IOQVWIU1JCM2	B00BF11STU	203465797	Zero Dark Thirty	Digital_Video_Download	2	(
3	US	47503411	R21A120094I1JE	B00GX59QO2	645382359	Red 2	Digital_Video_Download	5	(
4	US	1700003	R373ARADQ1OMI	B00HY7YXIM	592331046	The White Queen, Season 1	Digital_Video_Download	2	(
...	...	...	...	...	...	...	...	...	..
3250241	US	1430874	R1G86DIFDQA88W	B0047HYMHE	520198341	Test Asin: A Dance with Governator 2	Books	5	
3250242	US	44600026	R279NY81DJTZLN	B000000OQF	485103504	Appetite For Destruction	Music	5	(
3250243	US	51086982	R1SNPD3SIN2RH4	B000000XCK	616654004	Pendulum	Music	5	
3250244	US	13606832	R1OFQXH4K6A52C	B000002KCC	468801826	Tonight's the Night	Music	1	
3250245	US	41992983	R13DQT4LRJLQVG	B000002LQR	982029524	Blood Sugar Sex Magik	Music	5	3

3250177 rows × 15 columns

Data Exploration

In [9]:
```
df['product_title'].value_counts()
```

Out[9]:
```
product_title
Candy Crush Saga                                                                    23431
The Secret Society® - Hidden Mystery                                                19524
Google Chromecast HDMI Streaming Media Player                                       17020
The Fault in Our Stars                                                              16898
Minecraft                                                                           16335
                                                                                    ...
Laundry Service (+ Bonus Avcd)                                                          1
Seiko SKX007K Automatic 200m Diver Rubber Band Men's Watch                              1
Voyager Outlander 03                                                                    1
SWEENEY TODD                                                                            1
I Am Legend (Widescreen) (2 Disc Edition w/Limited Edition Steelbook Packaging + Digital Copy of Film)  1
Name: count, Length: 44383, dtype: int64
```

In [10]:
```
df['product_category'].value_counts()
```

Out[10]:
```
product_category
Mobile_Apps                 690062
Digital_Ebook_Purchase      585378
Video DVD                   514603
Digital_Video_Download      496675
Books                       393728
Music                       365240
Digital_Music_Purchase       50837
Toys                         27449
PC                           26713
Video                        21736
Home Entertainment           17288
Wireless                     10687
Camera                        7964
Video Games                   7223
Electronics                   5376
Musical Instruments           5079
Watches                       4927
Tools                         3496
Shoes                         3487
Baby                          2792
Sports                        1935
Home Improvement              1751
Outdoors                      1468
Office Products               1080
Home                           939
Kitchen                        882
Lawn and Garden                569
Health & Personal Care         505
Automotive                     117
Mobile_Electronics              88
Apparel                         61
Luggage                         38
Beauty                          29
Software                        27
Grocery                          9
Personal_Care_Appliances         3
Furniture                        3
Pet Products                     2
Name: count, dtype: int64
```

Star Rating

In [11]:
```
df.star_rating.value_counts()
```

Out[11]:
```
star_rating
5    2083072
4     593908
3     251457
1     190183
2     131626
Name: count, dtype: Int64
```

Helpful Votes

In [12]:
```
df.helpful_votes.value_counts()
```

Out[12]:
```
helpful_votes
0       2254854
1        458081
2        169673
3         93267
4         57924
         ...
909           1
915           1
494           1
1708          1
2968          1
Name: count, Length: 1150, dtype: Int64
```

Total Votes

In [13]:
```
df.total_votes.value_counts()
```

Out[13]:
```
total_votes
0       1833671
1        572370
2        247623
3        135490
4         88411
         ...
947           1
728           1
1539          1
1605          1
1047          1
Name: count, Length: 1273, dtype: Int64
```

Verified Purchases

In [14]:
```
df.verified_purchase.value_counts()
```

Out[14]:
```
verified_purchase
True     2372151
False     878095
Name: count, dtype: Int64
```

Reviews Headline

In [15]:

```
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from sklearn.feature_extraction.text import CountVectorizer

# Function to get most common words
def get_most_common_words(text_series, n=50):
    vectorizer = CountVectorizer(stop_words='english')
    word_count = vectorizer.fit_transform(text_series.dropna())
    sum_words = word_count.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vectorizer.vocabulary_.items()]
    words_freq = sorted(words_freq, key=lambda x: x[1], reverse=True)
    return words_freq[:n]

common_words_headline = get_most_common_words(df['review_headline'])

def plot_word_cloud(common_words):
    wordcloud = WordCloud(width=800, height=400).generate_from_frequencies(dict(common_words))
    plt.figure(figsize=(12, 7))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.show()

plot_word_cloud(common_words_headline)
```

In [16]:
```
plt.savefig('wordcloud_headline .png')

common_words_headline
```

Out[16]:
```
[('stars', 460189),
 ('great', 323640),
 ('good', 187142),
 ('book', 139131),
 ('movie', 134191),
 ('love', 132078),
 ('game', 111222),
 ('best', 108725),
 ('fun', 107655),
 ('read', 105354),
 ('awesome', 83833),
 ('amazing', 51296),
 ('excellent', 51179),
 ('story', 49628),
 ('like', 44410),
 ('time', 39808),
 ('loved', 37425),
 ('album', 37166),
 ('app', 36643),
 ('better', 35077),
 ('just', 33891),
 ('series', 32758),
 ('classic', 31312),
 ('star', 30403),
 ('really', 28875),
 ('wonderful', 26347),
 ('nice', 21814),
 ('worth', 21335),
 ('bad', 20831),
 ('favorite', 20779),
 ('film', 20557),
 ('don', 20409),
 ('cd', 20325),
 ('fantastic', 20117),
 ('review', 20089),
 ('new', 19908),
 ('music', 19755),
 ('interesting', 19170),
 ('ok', 17424),
 ('beautiful', 16820),
 ('perfect', 16616),
 ('watch', 16385),
 ('entertaining', 15773),
 ('wow', 15731),
 ('cool', 15420),
 ('life', 15267),
 ('dvd', 15253),
 ('easy', 14818),
 ('buy', 14651),
 ('little', 14567)]

<Figure size 640x480 with 0 Axes>
```

Reviews Body

In [17]:

```
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from sklearn.feature_extraction.text import CountVectorizer

# Function to get most common words
def get_most_common_words(text_series, n=50):
    vectorizer = CountVectorizer(stop_words='english')
    word_count = vectorizer.fit_transform(text_series.dropna())
    sum_words = word_count.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vectorizer.vocabulary_.items()]
    words_freq = sorted(words_freq, key=lambda x: x[1], reverse=True)
    return words_freq[:n]

common_words_body = get_most_common_words(df['review_headline'])

def plot_word_cloud(common_words):
    wordcloud = WordCloud(width=800, height=400).generate_from_frequencies(dict(common_words))
    plt.figure(figsize=(12, 7))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.show()

plot_word_cloud(common_words_headline)
```

In [18]:
```
plt.savefig('wordcloud_headline .png')

common_words_body
```

Out[18]:
```
[('stars', 460189),
 ('great', 323640),
 ('good', 187142),
 ('book', 139131),
 ('movie', 134191),
 ('love', 132078),
 ('game', 111222),
 ('best', 108725),
 ('fun', 107655),
 ('read', 105354),
 ('awesome', 83833),
 ('amazing', 51296),
 ('excellent', 51179),
 ('story', 49628),
 ('like', 44410),
 ('time', 39808),
 ('loved', 37425),
 ('album', 37166),
 ('app', 36643),
 ('better', 35077),
 ('just', 33891),
 ('series', 32758),
 ('classic', 31312),
 ('star', 30403),
 ('really', 28875),
 ('wonderful', 26347),
 ('nice', 21814),
 ('worth', 21335),
 ('bad', 20831),
 ('favorite', 20779),
 ('film', 20557),
 ('don', 20409),
 ('cd', 20325),
 ('fantastic', 20117),
 ('review', 20089),
 ('new', 19908),
 ('music', 19755),
 ('interesting', 19170),
 ('ok', 17424),
 ('beautiful', 16820),
 ('perfect', 16616),
 ('watch', 16385),
 ('entertaining', 15773),
 ('wow', 15731),
 ('cool', 15420),
 ('life', 15267),
 ('dvd', 15253),
 ('easy', 14818),
 ('buy', 14651),
 ('little', 14567)]

<Figure size 640x480 with 0 Axes>
```

Sentiment Analysis

In [14]:

```
import pandas as pd
import numpy as np
from multiprocessing import Pool, cpu_count
from textblob import TextBlob

# Define a function to analyze sentiment using TextBlob
def analyze_sentiment(text):
    if text is not None:
        blob = TextBlob(text)
        return blob.sentiment.polarity
    else:
        return None

# Define a function to map sentiment score to category
def map_sentiment_category(score):
    if score is None:
        return 'Neutral'
    elif score > 0:
        return 'Positive'
    elif score < 0:
        return 'Negative'
    else:
        return 'Neutral'

# Function to apply sentiment analysis to a chunk of data
def process_chunk(chunk):
    chunk['sentiment_score'] = chunk['review_body'].apply(analyze_sentiment)
    chunk['sentiment_category'] = chunk['sentiment_score'].apply(map_sentiment_category)
    return chunk

# Function to parallelize the sentiment analysis
def parallelize_dataframe(df, func, num_partitions):
    df_split = np.array_split(df, num_partitions)
    pool = Pool(cpu_count())
    df = pd.concat(pool.map(func, df_split))
    pool.close()
    pool.join()
    return df

# Number of partitions for parallel processing
num_partitions = cpu_count() * 2

# Apply the sentiment analysis in parallel
df_result = parallelize_dataframe(df, process_chunk, num_partitions)

print(df_result.head())

# Save the result to a CSV file
df_result.to_csv('sentiment_analysis_results.csv', index=False)
```

```
  marketplace  customer_id       review_id  product_id  product_parent  \
0          US     52529112  R19O08234WZ3M0  B00425S1H8       386765193
1          US     43856501   R2NOPGMUC39HO  B006GWO5NE       564071965
2          US     33278723  R2MW15DTHRRMSB  B008HK3XKO       893861753
3          US     21951802  R3KOC1O75FRIUS  B008RXYOKY       766086893
4          US     43248065  R3EXUH0288SPU6  B0095VP8D4       792792668

                                      product_title product_category  \
0  Plugable USB to RS-232 DB9 Serial Adapter (Pro...               PC
1  Amazon 5ft USB to Micro-USB Cable (works with ...               PC
2  Marware Vibe Standing Case for Kindle Fire HD ...               PC
3             Apple Thunderbolt to FireWire Adapter               PC
4   AMD A10-5800K APU 3.8Ghz Processor AD580KWOHJBOX               PC

   star_rating  helpful_votes  total_votes   vine  verified_purchase  \
0            3              0            1  False               True
1            1              0            0  False              False
2            5              0            0  False               True
3            5              0            0  False               True
4            5              1            2  False               True

                             review_headline  \
0  Worked but required an additional adapter
1                       problem with kindle
2                              kindle cover
3              Would recommend to a friend.
4                              Nice Product

                                       review_body review_date  \
0  The adapter came with female threaded nuts tha...  2013-01-09
1  This did not solve my kindle problem and i don...  2013-08-19
2  Fit the kindle fire hd perfectly. The black wa...  2013-01-28
3  Fits my needs. Happy with my purchase.  Well c...  2014-03-05
4  I love this processor. It's so fast and quite ...  2013-03-24

   sentiment_score sentiment_category
0         0.047619           Positive
1        -0.100000           Negative
2         0.405556           Positive
3         0.400000           Positive
4         0.652500           Positive
```

In [22]:

```
df = df_result.copy()
```

In [34]:

```
df.sentiment_category.value_counts()
```

Out[34]:

```
sentiment_category
Positive    976212
Negative    122304
Neutral      51678
Name: count, dtype: int64
```

Review Date

In [7]:

```
# Convert review_date to datetime format and extract the year
df['review_date'] = pd.to_datetime(df['review_date'], errors='coerce')
df['review_year'] = df['review_date'].dt.year
```

Visulisation

In [15]:

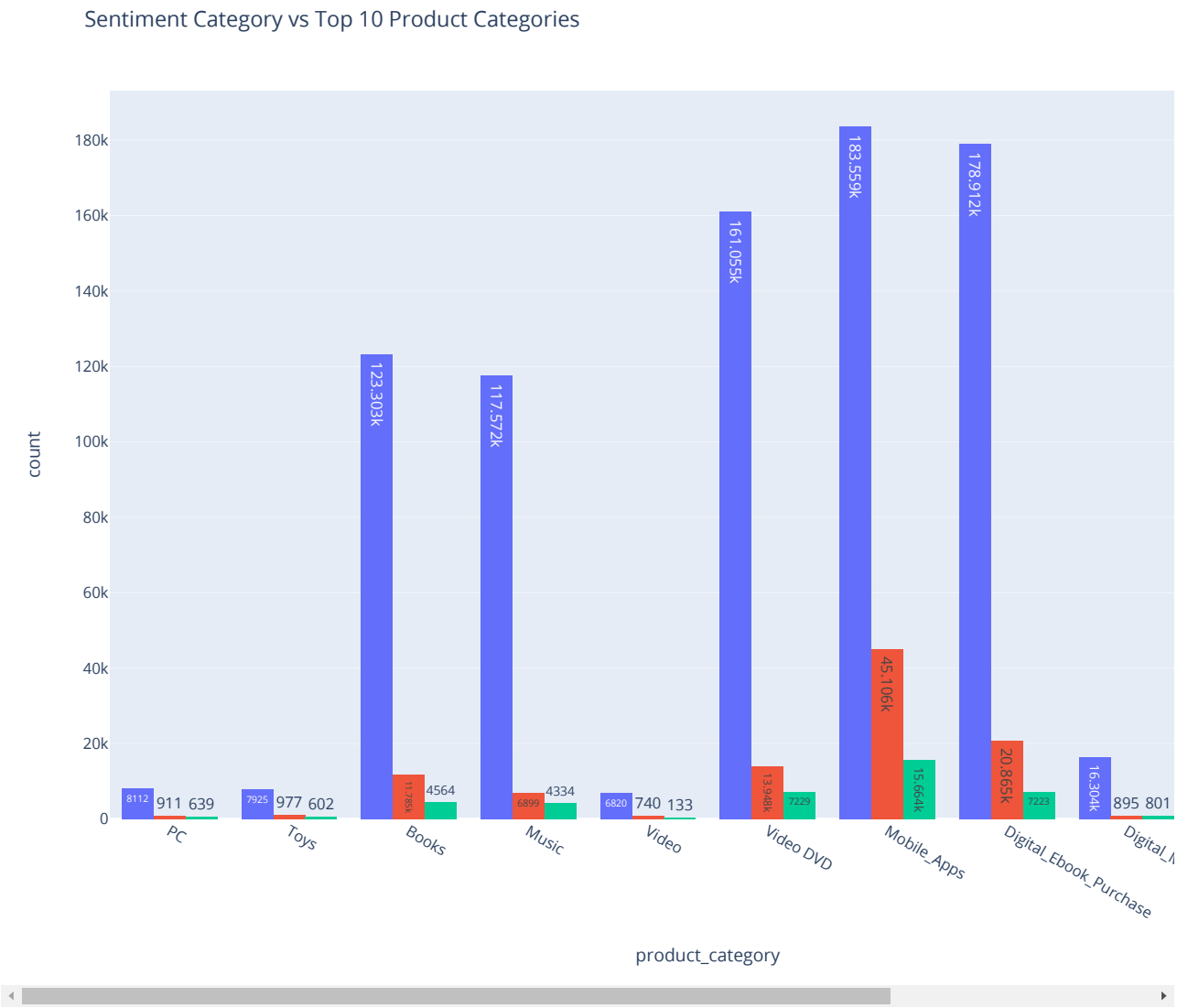
```
!pip install kaleido
```

```
Collecting kaleido
  Downloading kaleido-0.2.1-py2.py3-none-manylinux1_x86_64.whl.metadata (15 kB)
Downloading kaleido-0.2.1-py2.py3-none-manylinux1_x86_64.whl (79.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.9/79.9 MB 13.4 MB/s eta 0:00:0000:0100:01
Installing collected packages: kaleido
Successfully installed kaleido-0.2.1
```

Sentiment Distribution by Product Category

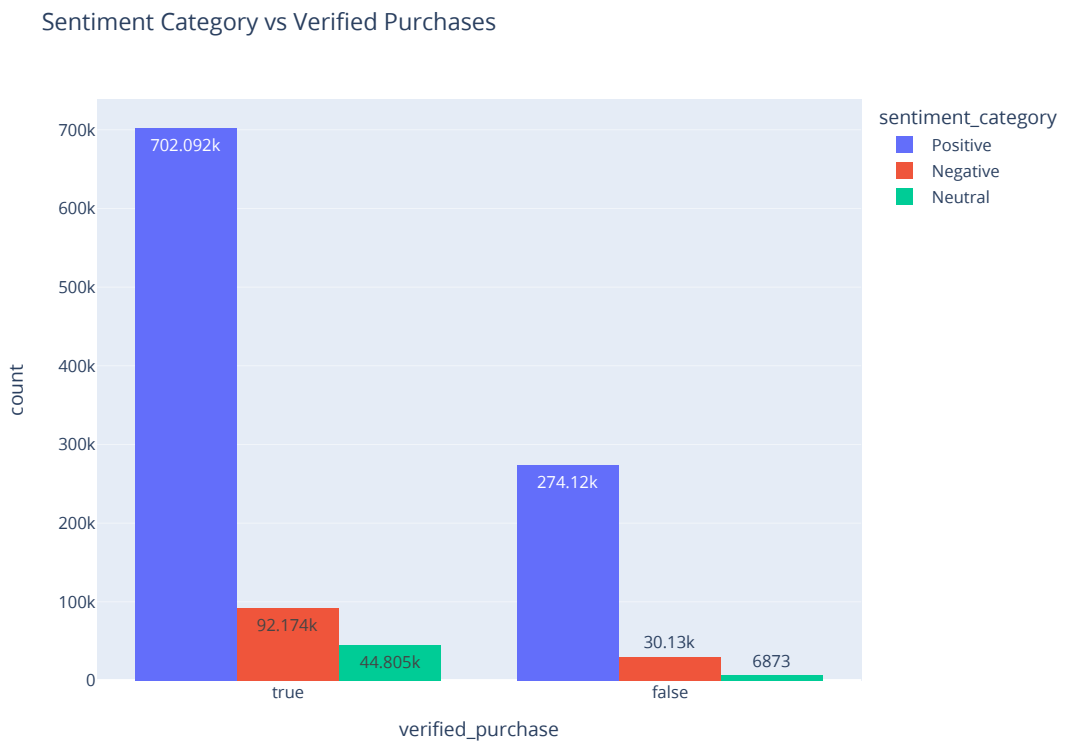
In [3]:

```
import plotly.graph_objs as go
import plotly.express as px
import kaleido

# Get the top 10 product categories
top_categories = df['product_category'].value_counts().nlargest(10).index

# Filter DataFrame to include only the top 10 product categories
df_top_categories = df[df['product_category'].isin(top_categories)]

# Plot Sentiment Category vs Product Category for top 10 categories
fig_product_category = px.histogram(df_top_categories, x='product_category', color='sentiment_category', barmode='grou
                                    title='Sentiment Category vs Top 10 Product Categories',
                                    text_auto=True, # Add text value
                                    width=1200,
                                    height=800)

fig_product_category.write_image('sentiment_category_vs_product_category.png')
fig_product_category.show()
```



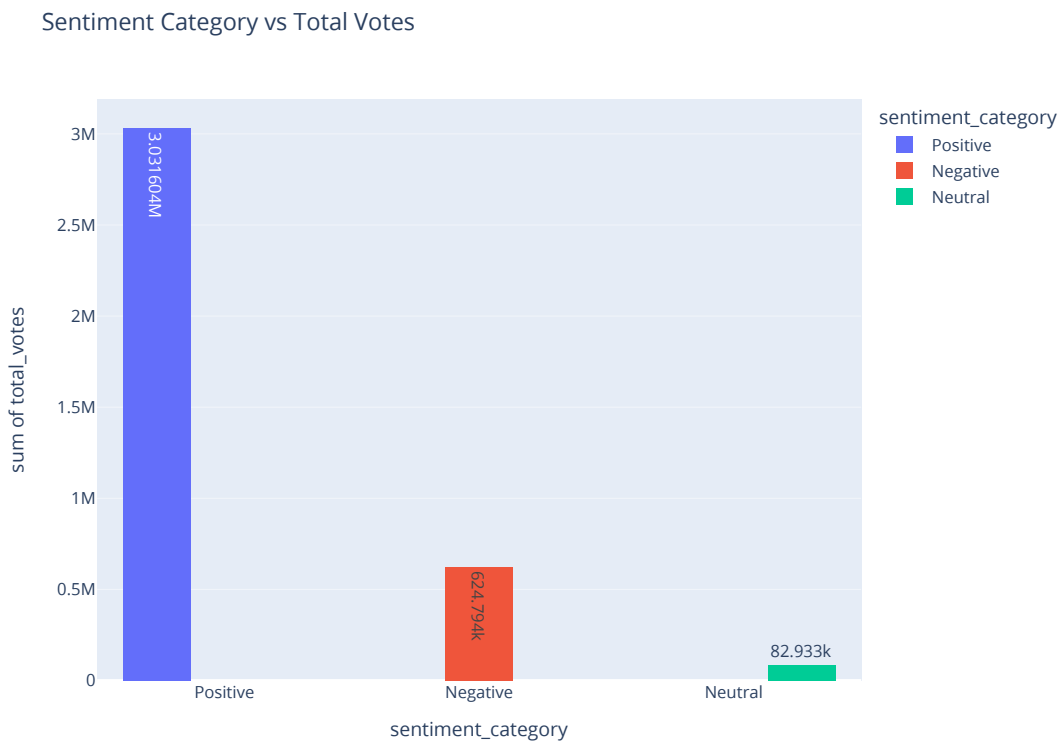
Sentiment Category vs Verified Purchases

In [4]:
```
fig_verified_purchases = px.histogram(df, x='verified_purchase', color='sentiment_category', barmode='group',
                                      title='Sentiment Category vs Verified Purchases',
                                      text_auto=True, # Add text value
                                      width=800,
                                      height=600)

fig_verified_purchases.write_image('sentiment_category_vs_verified_purchases.png')
fig_verified_purchases.show()
```



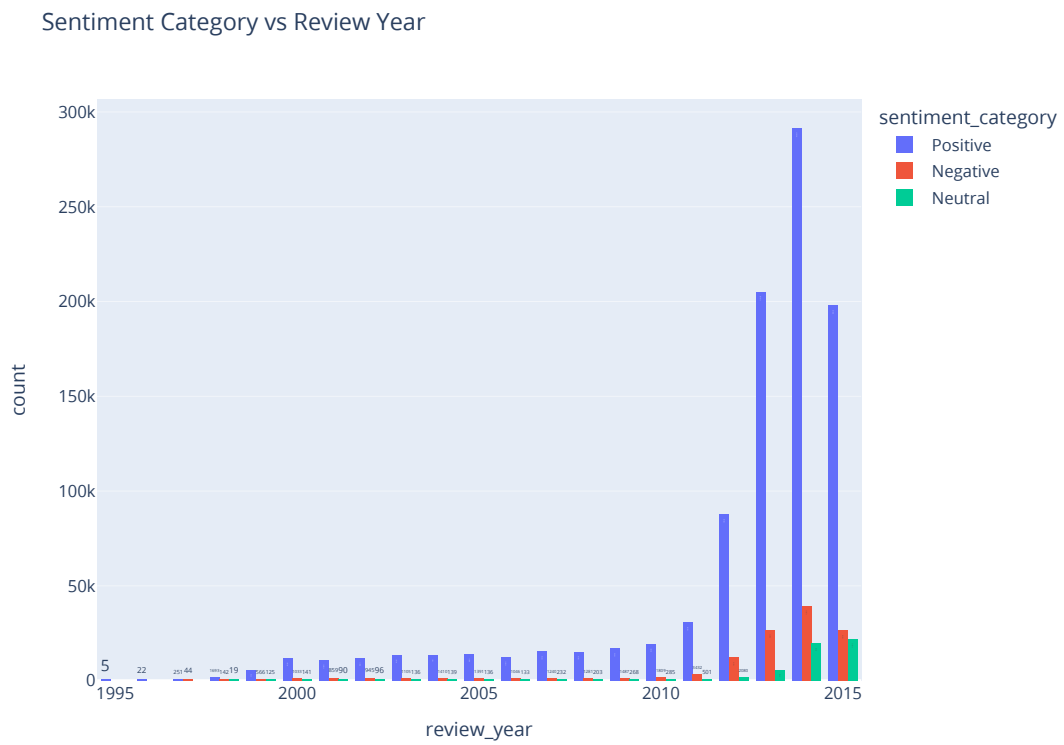
Sentiment Category vs Total Votes

In [5]:
```
fig_total_votes = px.histogram(df, x='sentiment_category', y='total_votes', color='sentiment_category', barmode='group
                               title='Sentiment Category vs Total Votes',
                               text_auto=True, # Add text value
                               width=800,
                               height=600)

fig_total_votes.write_image('sentiment_category_vs_total_votes.png')
fig_total_votes.show()
```



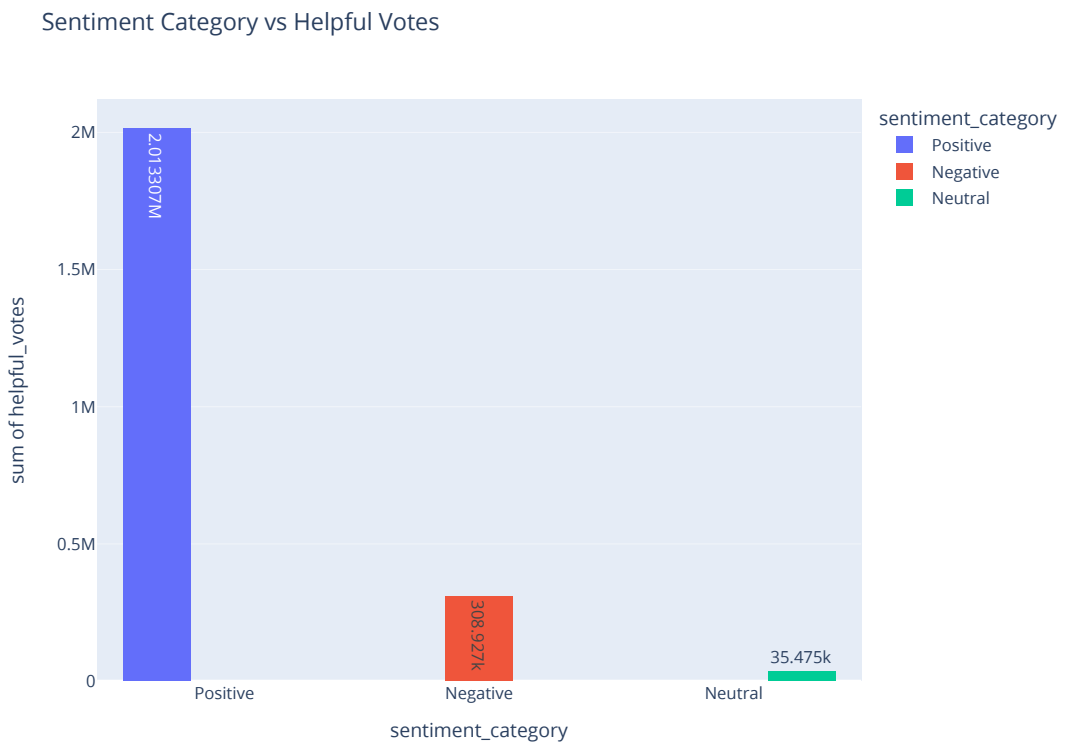
Sentiment Category vs Review Year

In [8]:
```
fig_review_year = px.histogram(df, x='review_year', color='sentiment_category', barmode='group',
                               title='Sentiment Category vs Review Year',
                               text_auto=True, # Add text value
                               width=800,
                               height=600)

fig_review_year.write_image('sentiment_category_vs_review_year.png')
fig_review_year.show()
```



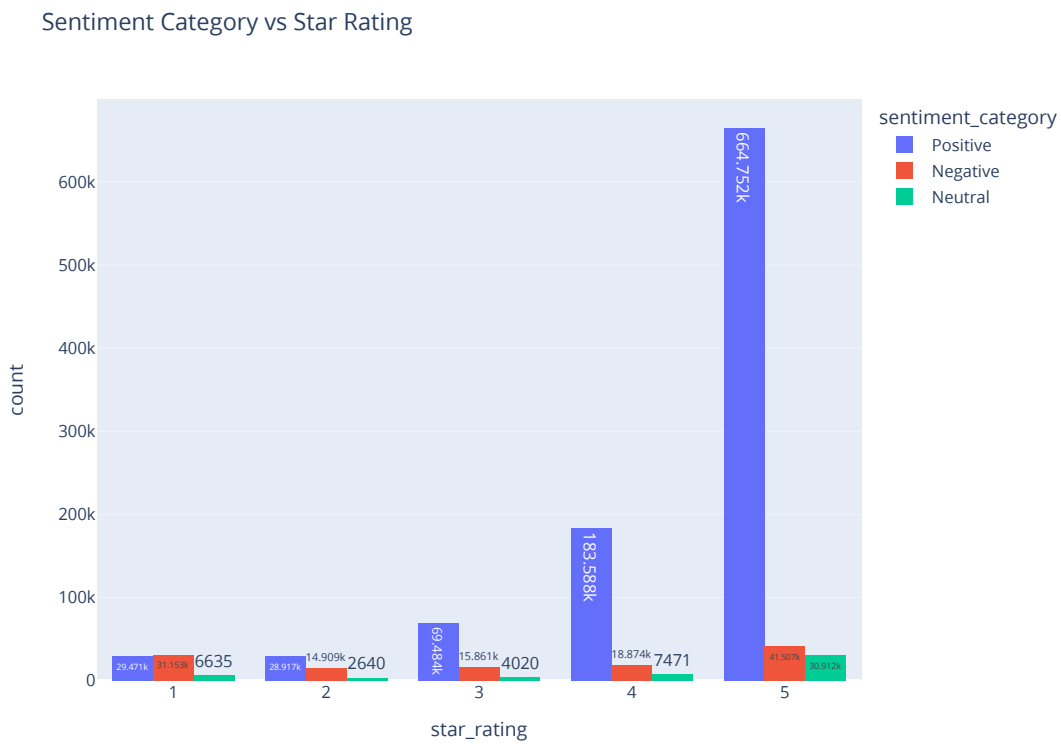
Sentiment Category vs Helpful Votes

In [9]:
```
fig_helpful_votes = px.histogram(df, x='sentiment_category', y='helpful_votes', color='sentiment_category', barmode='g
                                 title='Sentiment Category vs Helpful Votes',
                                 text_auto=True, # Add text value
                                 width=800,
                                 height=600)

fig_helpful_votes.write_image('sentiment_category_vs_helpful_votes.png')
fig_helpful_votes.show()
```



Sentiment Category vs Star Rating

In [10]:
```
fig_star_rating = px.histogram(df, x='star_rating', color='sentiment_category', barmode='group',
                               title='Sentiment Category vs Star Rating',
                               text_auto=True, # Add text value
                               width=800,
                               height=600)

fig_star_rating.write_image('sentiment_category_vs_star_rating.png')
fig_star_rating.show()
```



Save to Cloud Storage

In [ ]:
```
# Save to Google Cloud Storage
storage_client = storage.Client()
bucket_name = 'store-customer-data'
bucket = storage_client.bucket(bucket_name)
blob = bucket.blob('sentiment_analysis_results.csv')
blob.upload_from_filename('sentiment_analysis_results.csv')
```