



BUSINESS REVIEWS

Harnessing the Power of Data

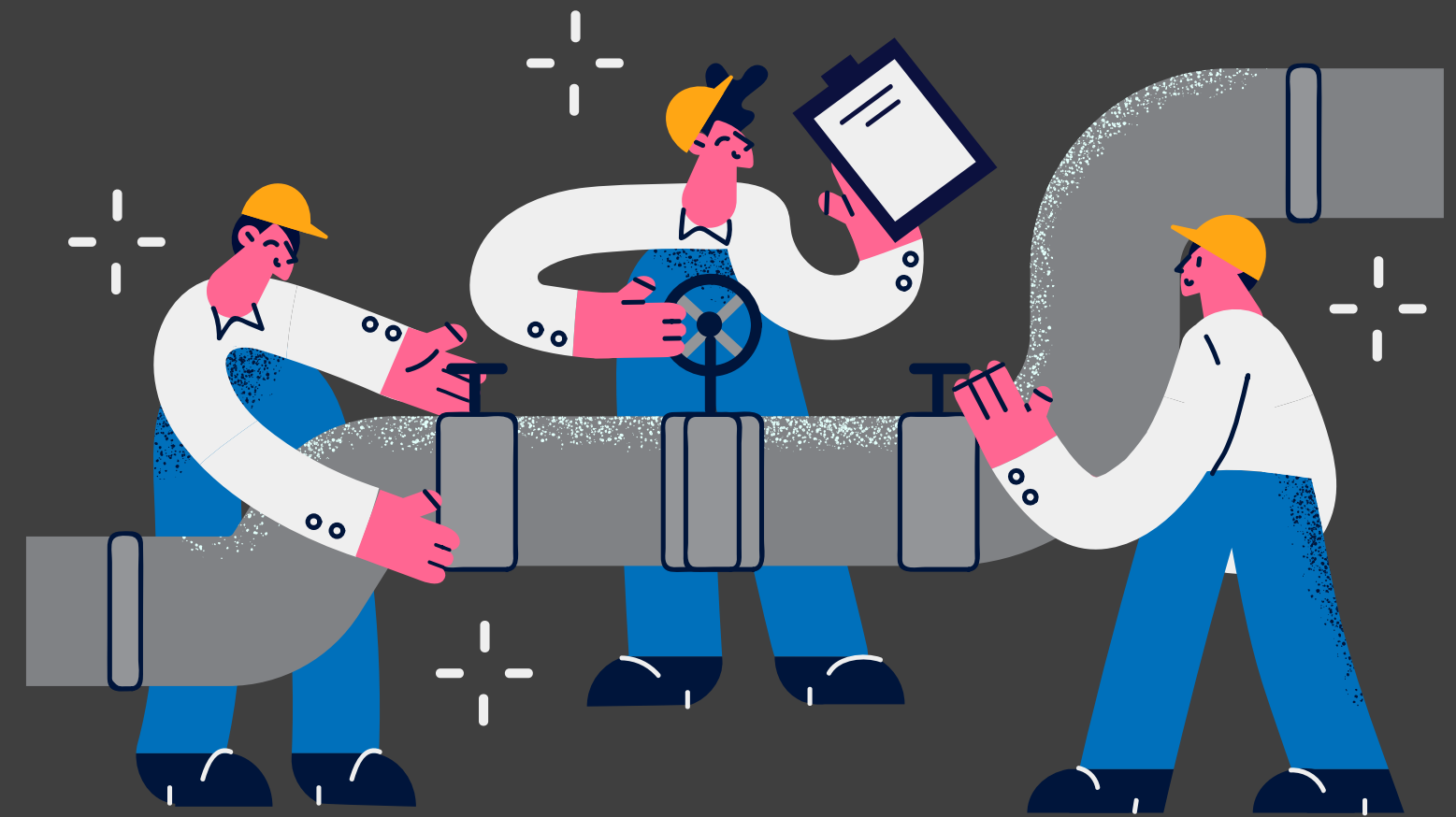
Imagine a world where your favorite streaming platform knows exactly what you want to watch before you even search. Or where online retailers predict your next purchase with uncanny accuracy.

This is the reality of recommendation systems, powerful tools that leverage data to personalize user experiences.



Data Engineering – The Hidden Engine of Recommendation Systems

- Building a successful recommendation system goes beyond collecting data and applying algorithms.
- Data engineering serves as the hidden powerhouse, meticulously transforming data into insightful recommendations.



Impact of Recommendation Systems:

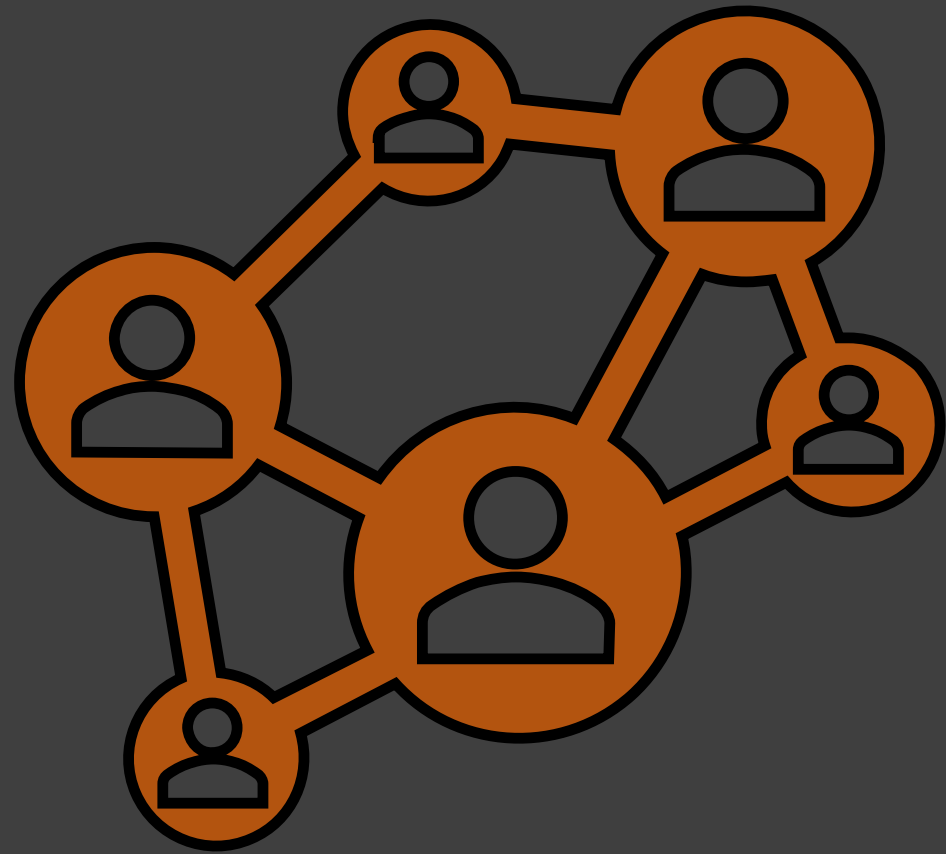
- Increased User Engagement
- Improved Sales and Conversions
- Enhanced Decision-Making





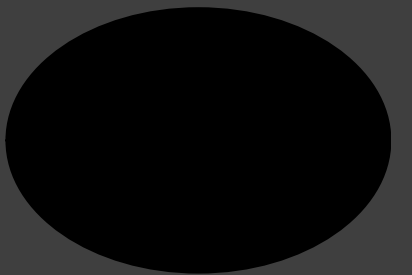
Business Reviews Case Study

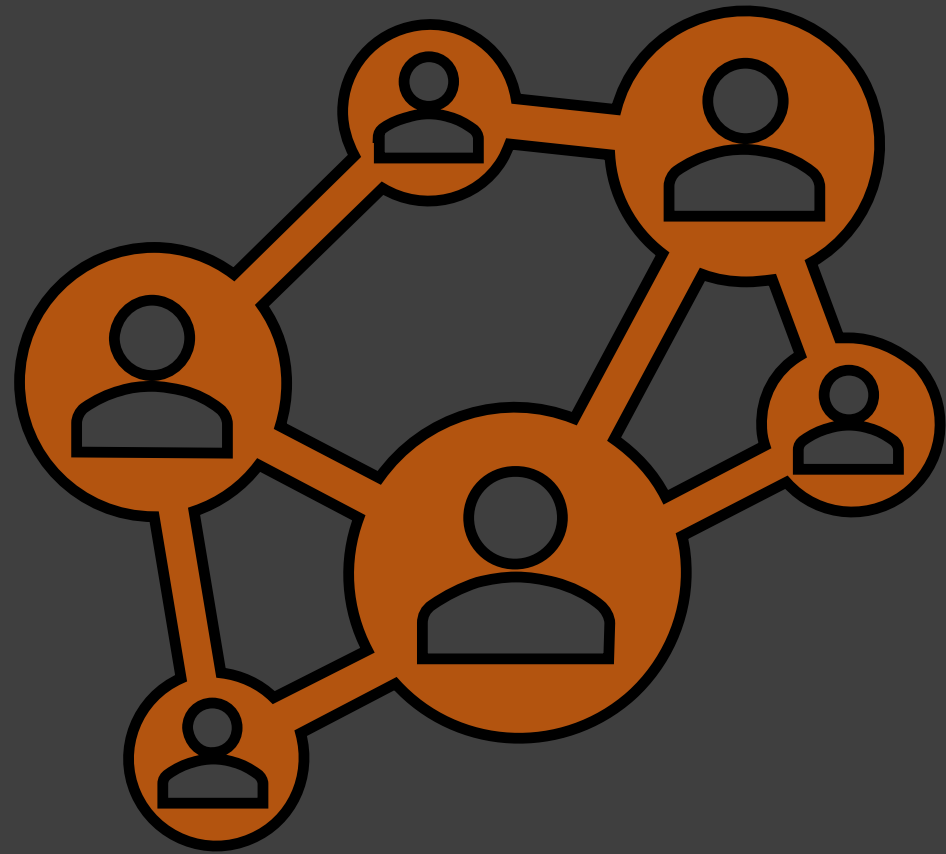
Yelp Business Review Data Pipeline



Agenda

1



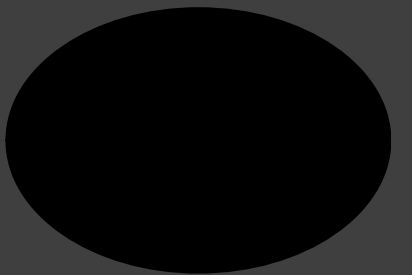


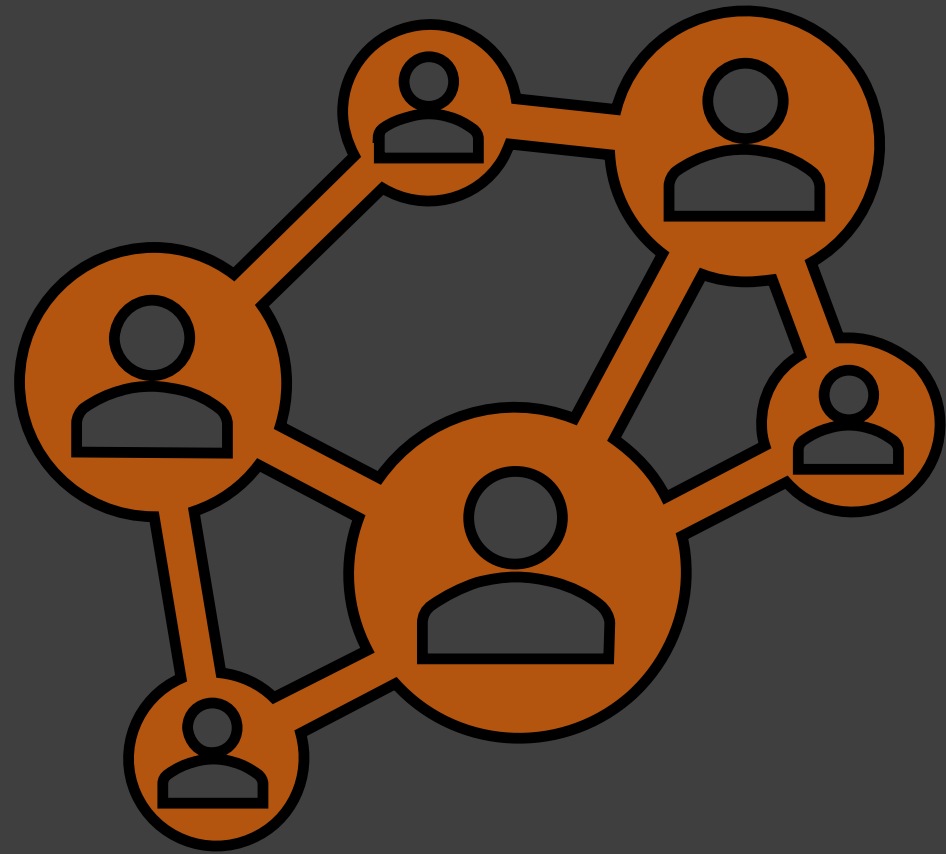
Agenda

1

Overview

2





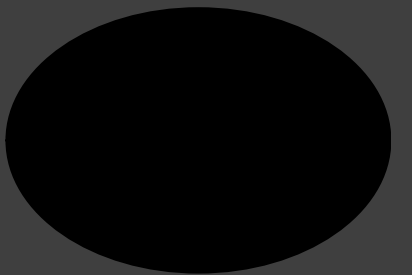
Agenda

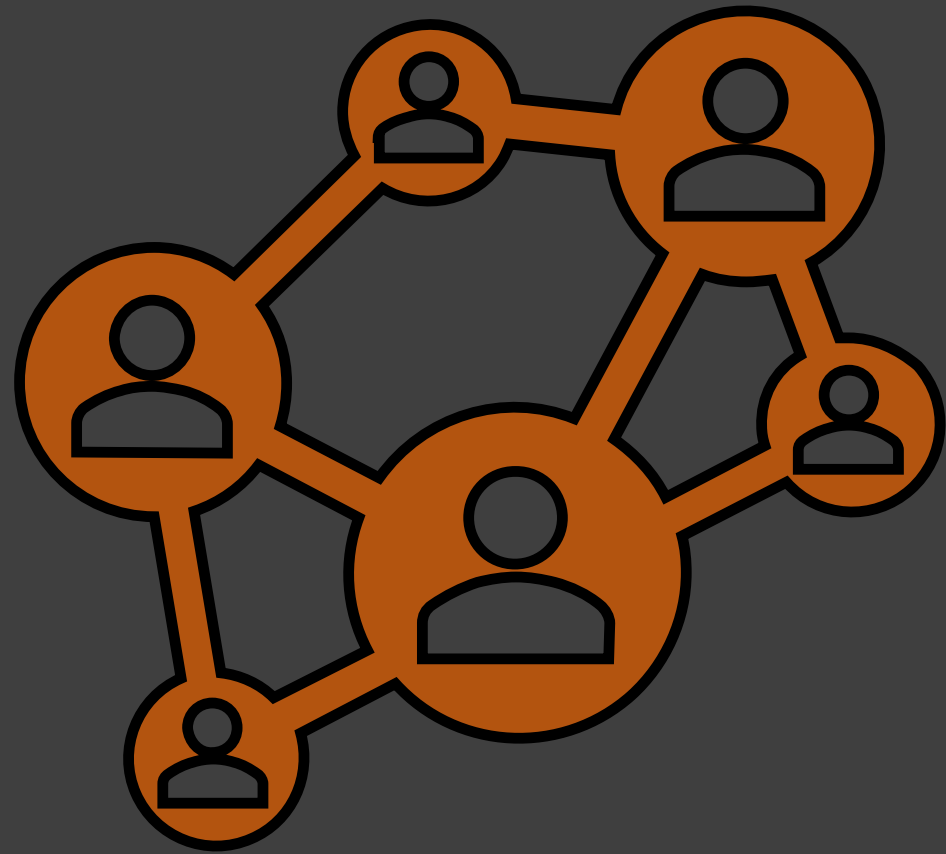
1

Overview

2

**Key
Components**





Agenda

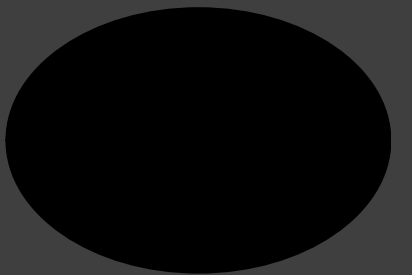
1

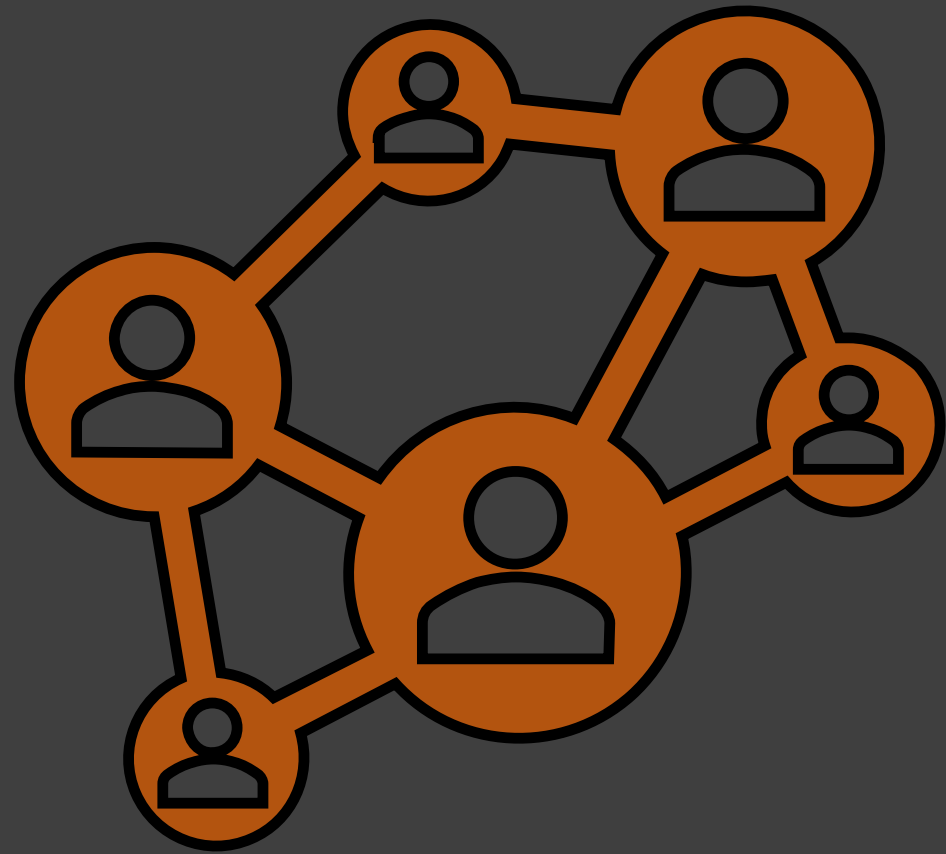
Overview

2

Key
Components

3





Agenda

1

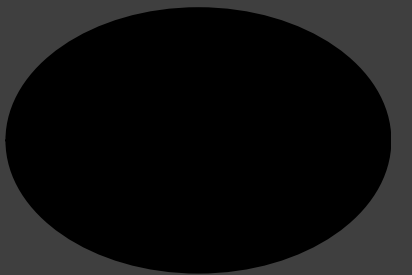
Overview

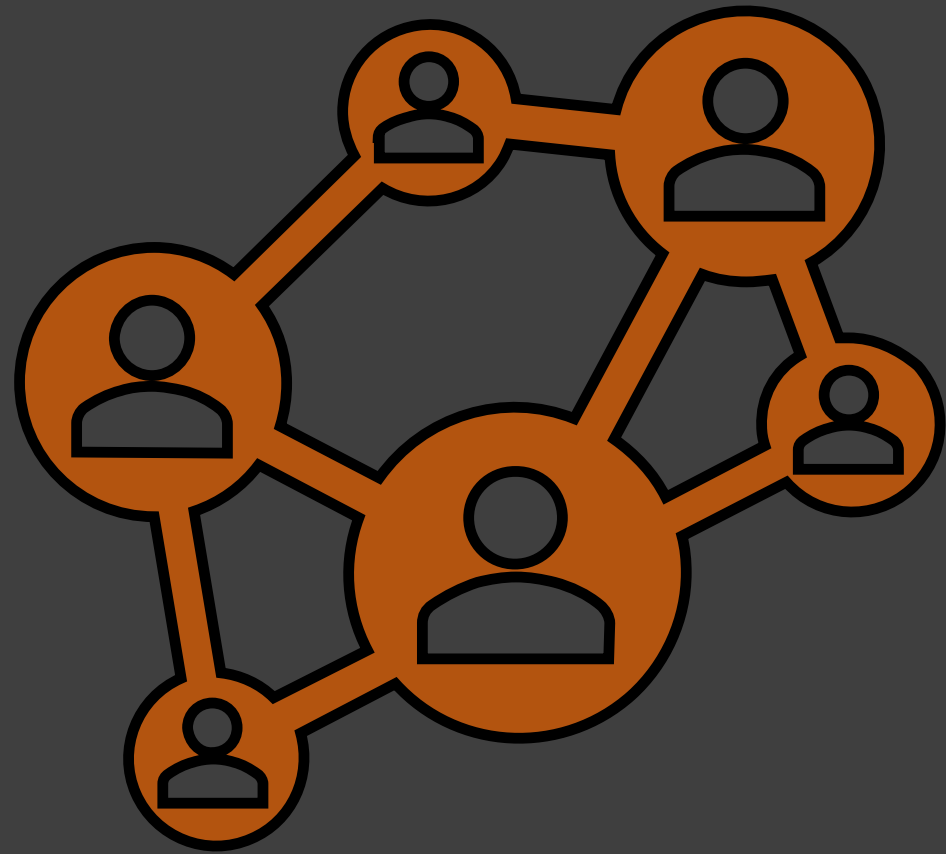
2

**Key
Components**

3

**Challenges
&
Benefits**





Agenda

1

Overview

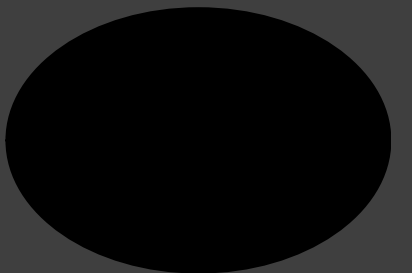
2

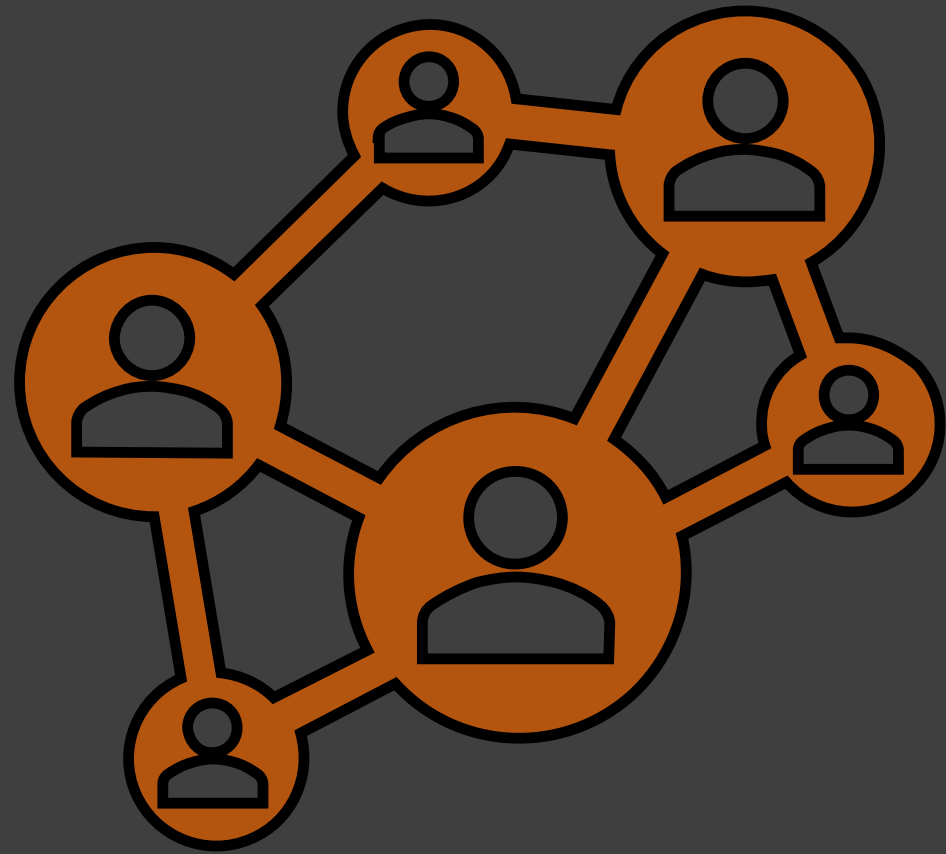
**Key
Components**

3

**Challenges
&
Benefits**

4





Agenda

1

Overview

2

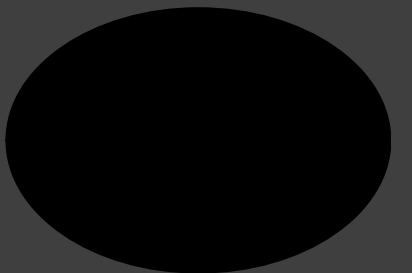
**Key
Components**

3

**Challenges
&
Benefits**

4

Demo

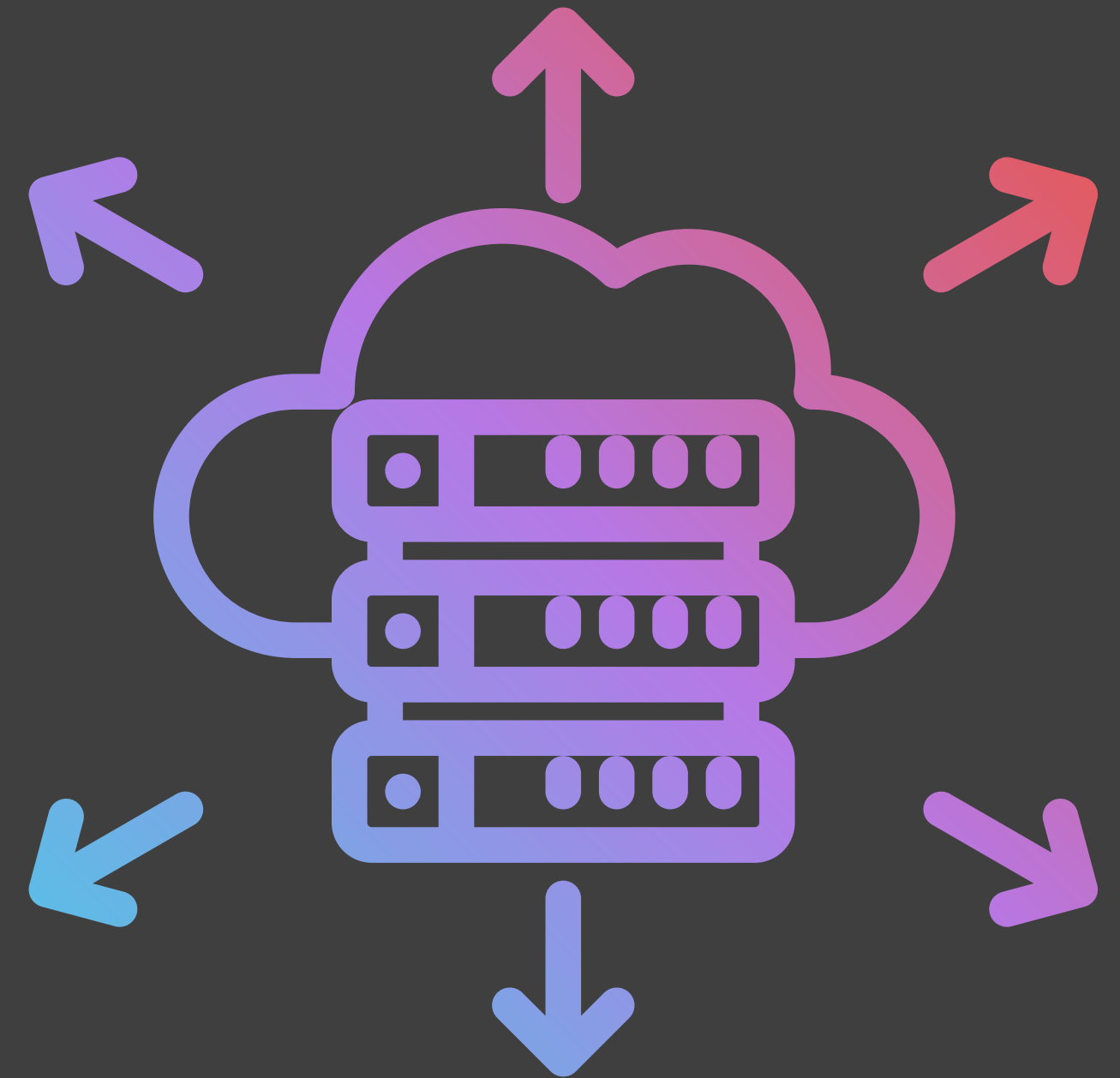


Project Overview

Process and analyze Yelp business review data to gain insights into customer sentiment, business performance, and industry trends.

Technology Stack:

- Python
- Kafka
- AWS services (DynamoDB, S3, Redshift)
- PySpark
- AWS Lambda
- Power BI



Project Pipeline

Data Source

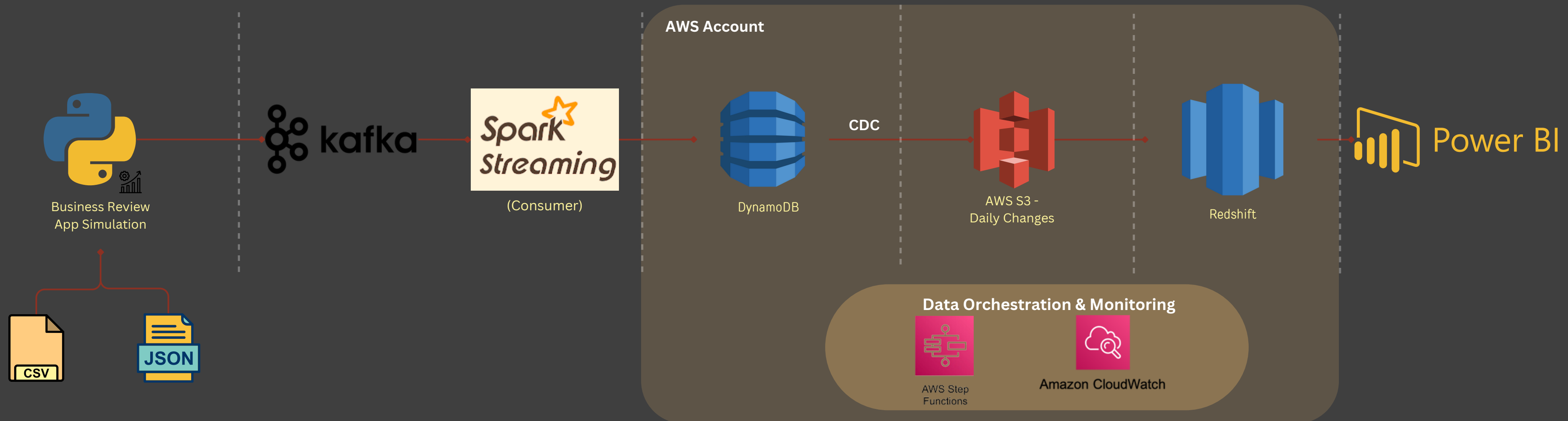
Data Ingestion & Preparation

Data Staging

Data Processing

Data Warehouse

Data Visualization



Data Sources

The primary data sources for the project were three key tables from Yelp's dataset :

Business.csv (150,346 businesses)

Contains business data including location data, attributes, and categories. We take all records.

Review.json (6,990,280 reviews)

Contains full review text data including the user_id that wrote the review and the business_id the review is written for. We take 6M of records.

User.csv (1,987,897 users)

User data including the user's friend mapping and all the metadata associated with the user. We take 1M of records.



Streaming Producer

Read the file by Python



Streaming Producer



Read the file by Python

Convert the DataFrame to a list of dictionaries

Streaming Producer



Read the file by Python

Convert the DataFrame to a list of dictionaries

Send the records in loop to Kafka topics

Streaming Producer



Read the file by Python

Convert the DataFrame to a list of dictionaries

Send the records in loop to Kafka topics

Wait one second to send the next record

Streaming Engine

Kafka, a distributed streaming platform, revolutionizes real-time data processing and communication.



Streaming Engine



Kafka, a distributed streaming platform, revolutionizes real-time data processing and communication.

Key Features (Scalability, High Throughput)

Streaming Engine



Kafka, a distributed streaming platform, revolutionizes real-time data processing and communication.

Key Features (Scalability, High Throughput)

Use Cases: (Real-Time Analytics, Event Sourcing)

Streaming Consumer

The Kafka consumer, implemented using PySpark



Streaming Consumer

The Kafka consumer, implemented using PySpark



Decodes the data and parses it as JSON

Streaming Consumer



The Kafka consumer, implemented using PySpark

Decodes the data and parses it as JSON

Processed and transformed incoming data.

Streaming Consumer



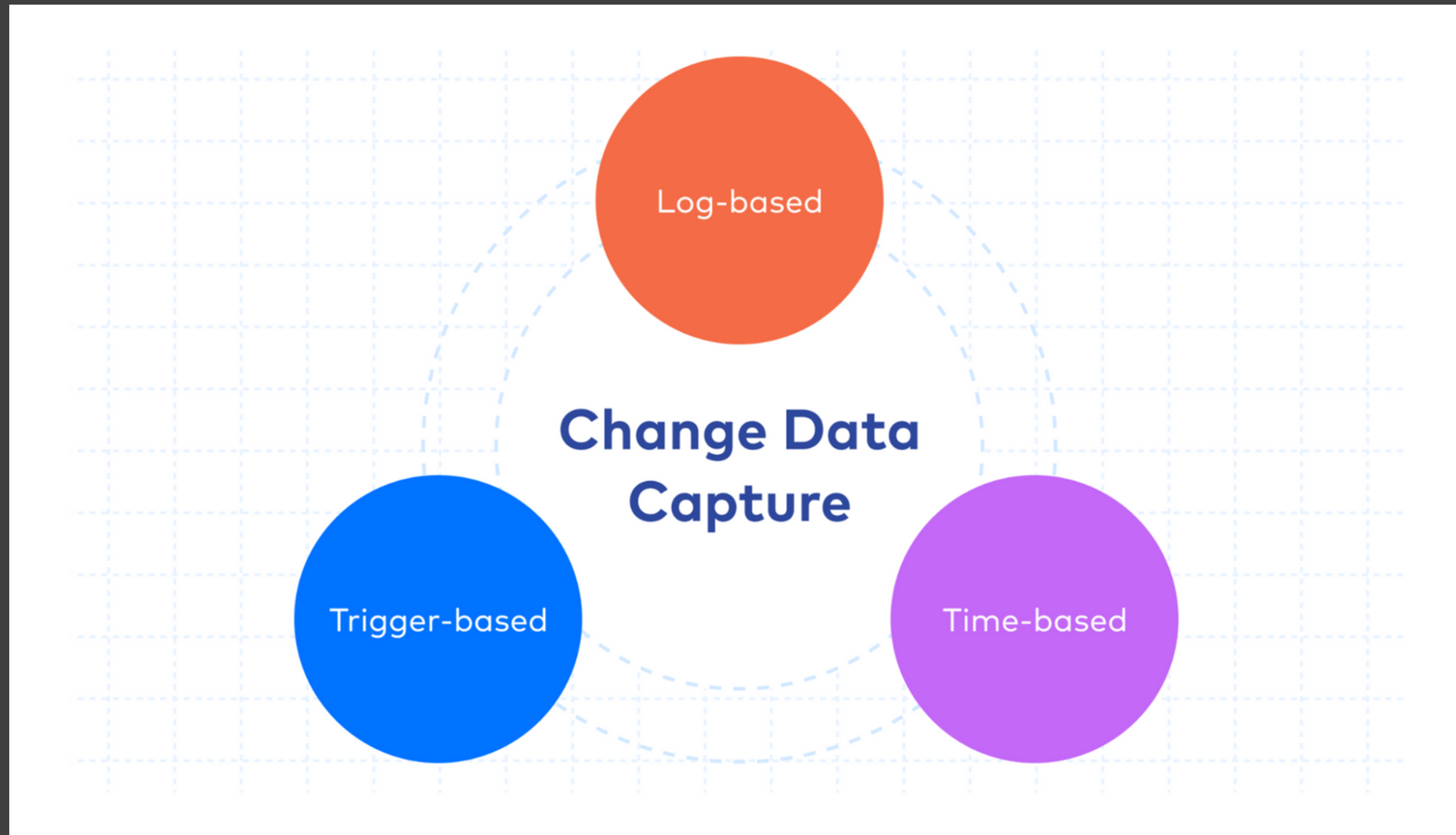
The Kafka consumer, implemented using PySpark

Decodes the data and parses it as JSON

Processed and transformed incoming data.

PySpark jobs enriched the data and prepared it for storage in DynamoDB.

Change Data Capture (CDC)



DynamoDB Tables



DynamoDB was used as a NoSQL database for real-time storage of processed Yelp data.

DynamoDB Tables



DynamoDB was used as a NoSQL database for real-time storage of processed Yelp data.

AWS Lambda Triggers in DynamoDB were configured to capture changes in the data, enabling CDC.

AWS Lambda Triggers



Lambda functions were implemented to trigger CDC processes in DynamoDB whenever changes occurred in the data.

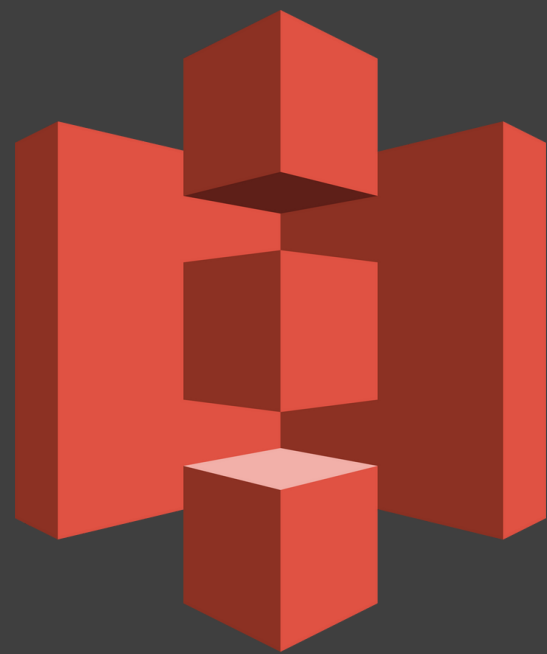
AWS Lambda Triggers



Lambda functions were implemented to trigger CDC processes in DynamoDB whenever changes occurred in the data.

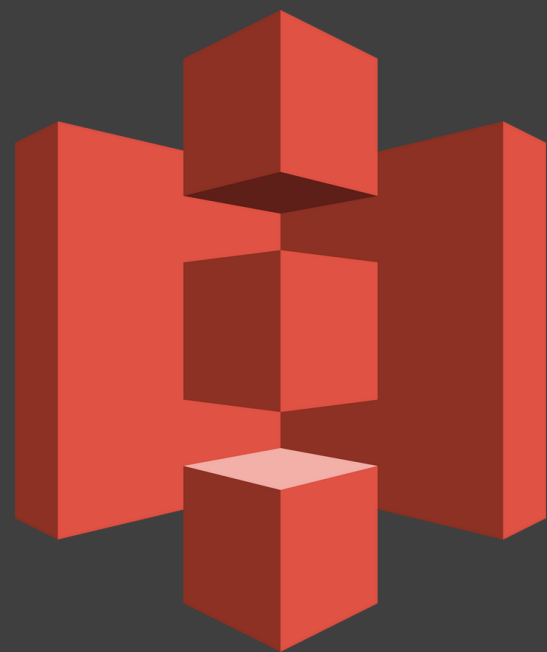
These triggers captured and propagated changes to an S3 bucket for daily batch processing.

S3 Batch Processing



Daily batch processing was performed on the data stored in S3 to aggregate, clean, and transform the data.

S3 Batch Processing



Daily batch processing was performed on the data stored in S3 to aggregate, clean, and transform the data.

This processed data was then made ready for loading into the Redshift data warehouse.

AWS RedShift



AWS Redshift served as the central data warehouse for storing the Yelp data in a structured and optimized format.

AWS RedShift



AWS Redshift served as the central data warehouse for storing the Yelp data in a structured and optimized format.

Daily batches of processed data from S3 were loaded into Redshift for analytical queries.

Analytical Dashboard



Power BI was utilized for data visualization, creating interactive dashboards and reports.

Analytical Dashboard



Power BI was utilized for data visualization, creating interactive dashboards and reports.

Direct queries were made to Redshift to pull the latest data for analytical visualization and analysis.



Business Reviews Dashboard



75K

Businesses



200K

user



3M

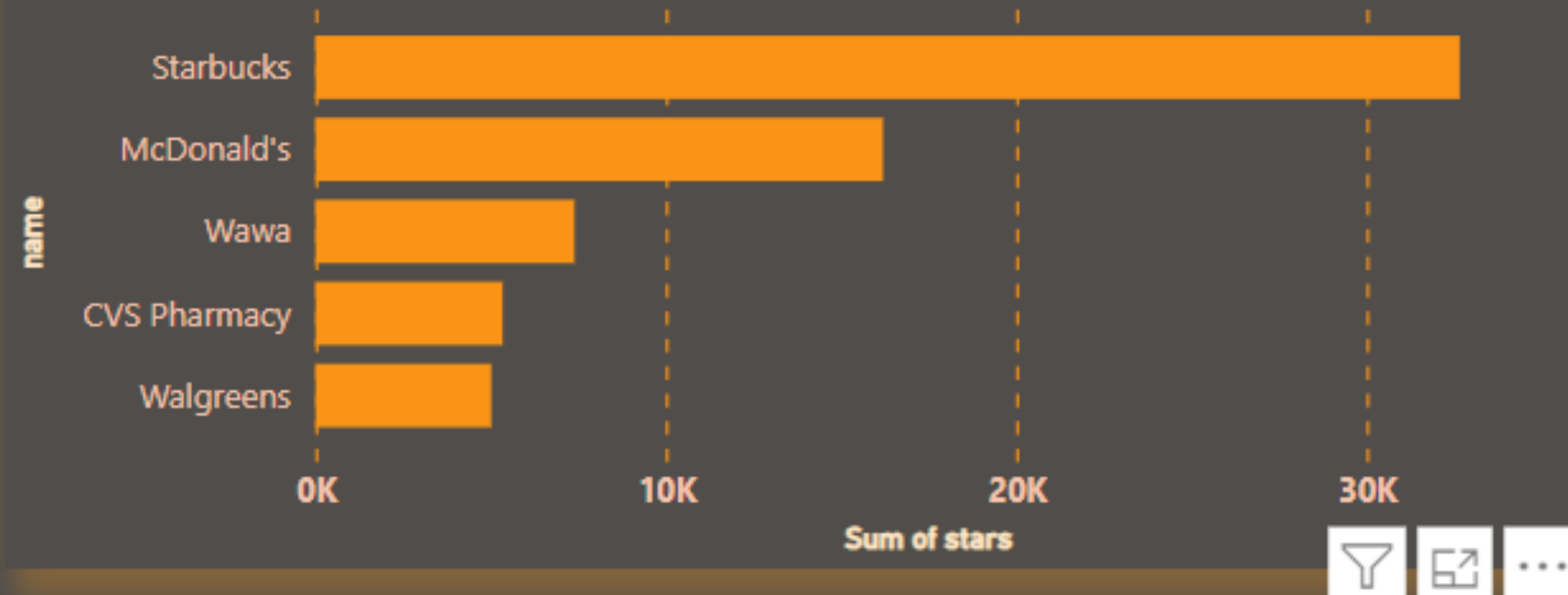
Review

stars

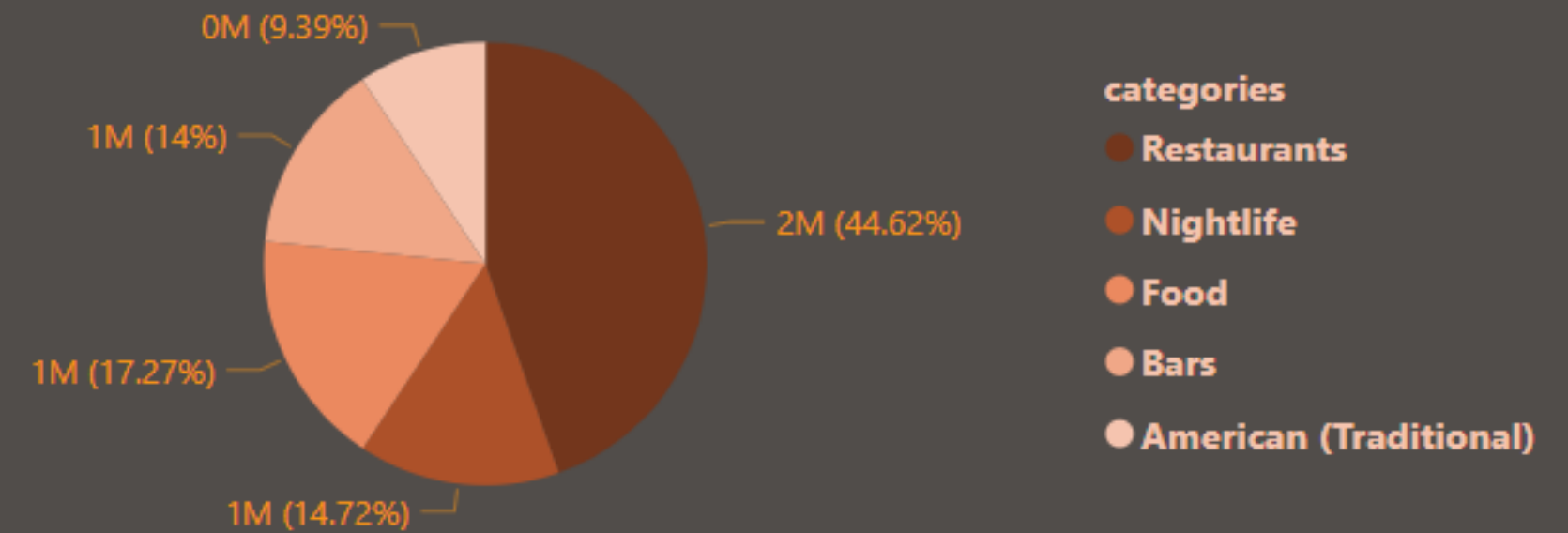
All



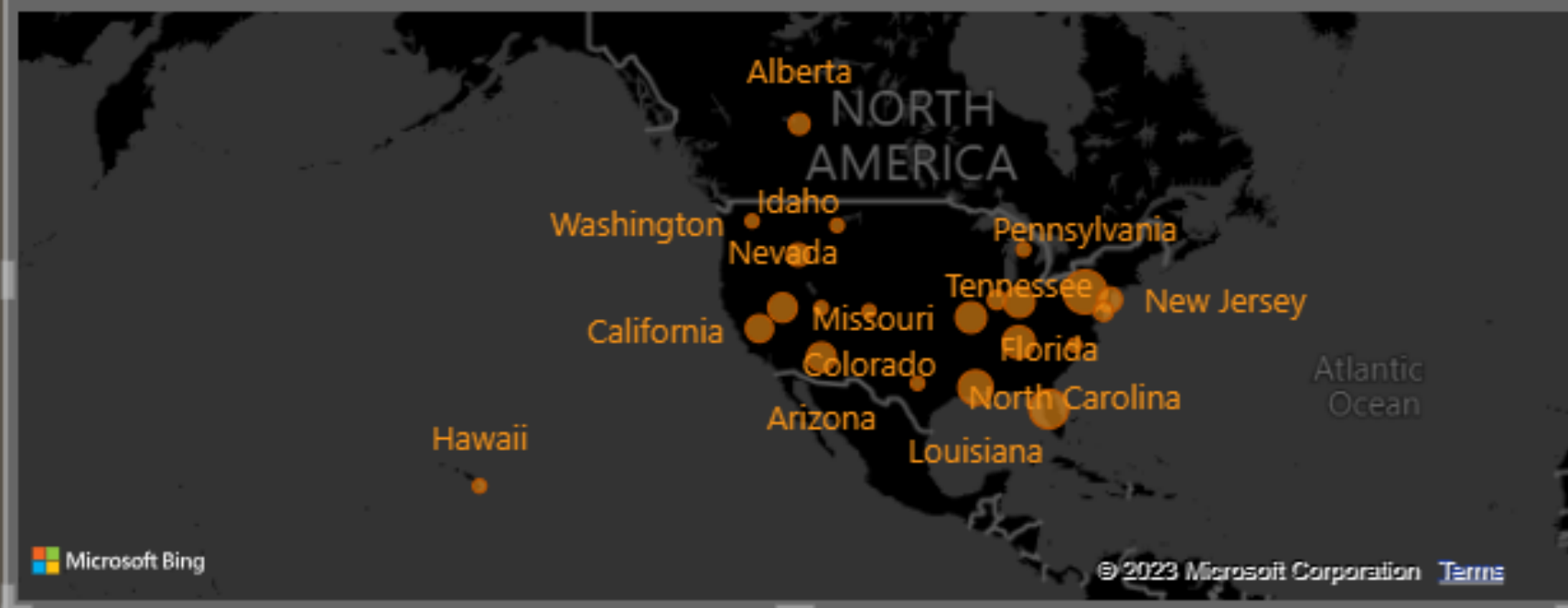
What are the top 5 business having stars ?



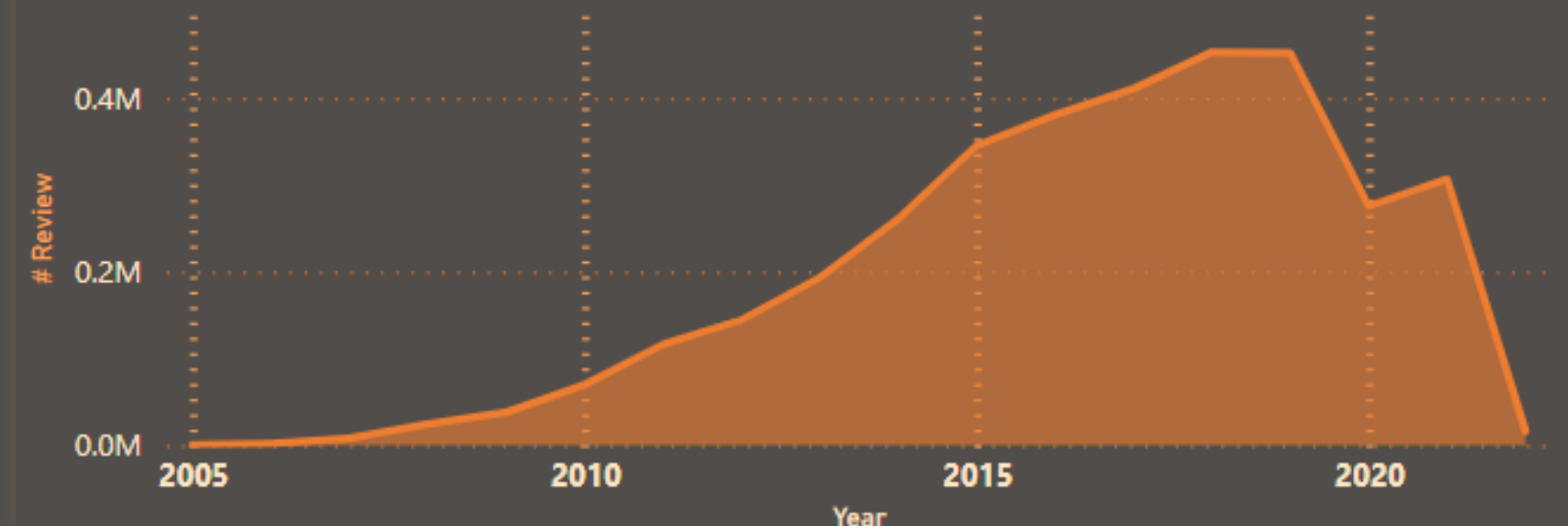
How many reviews does each category have?

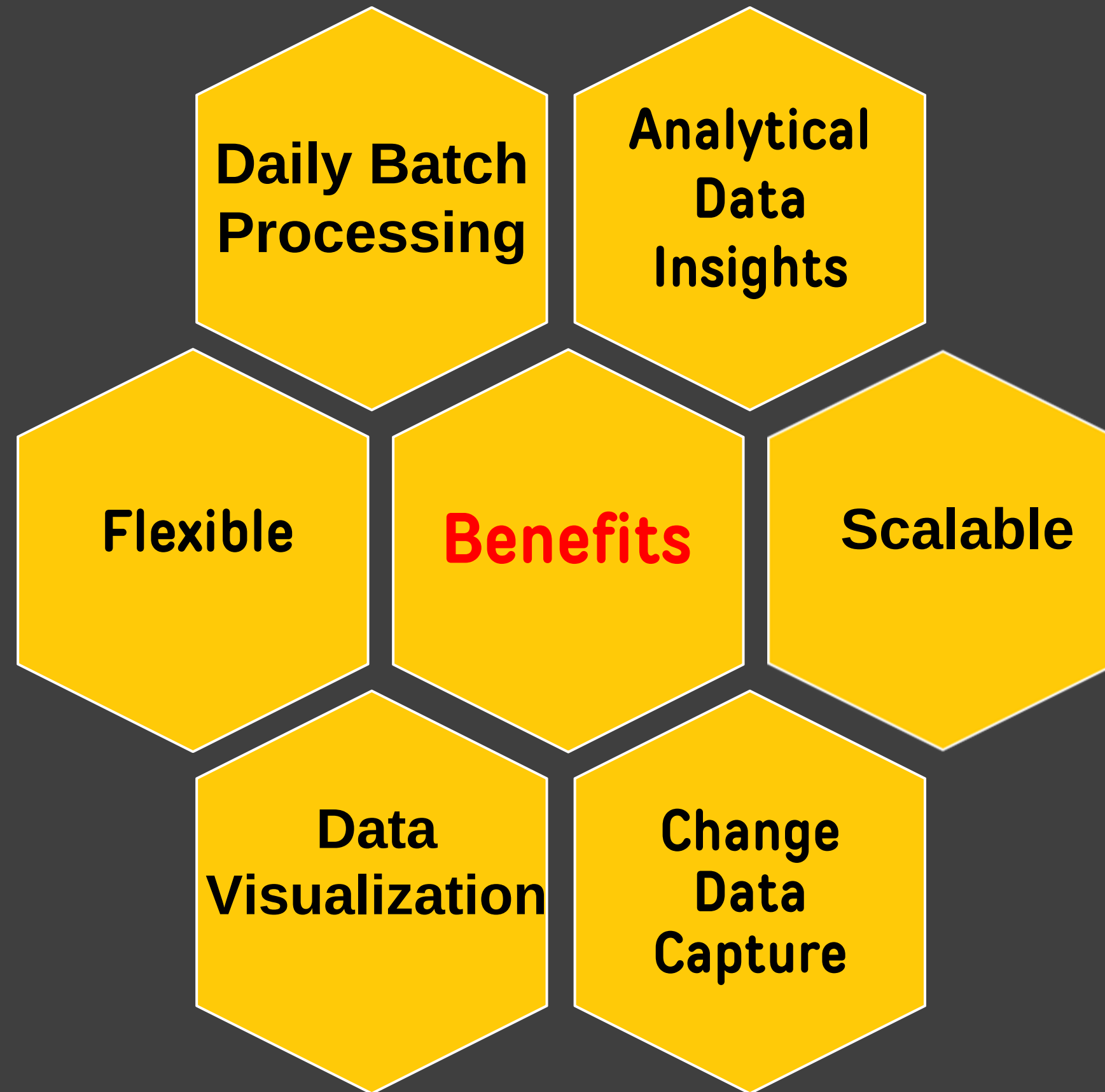


what is the number of reviews in each state?



How many reviews are by year?



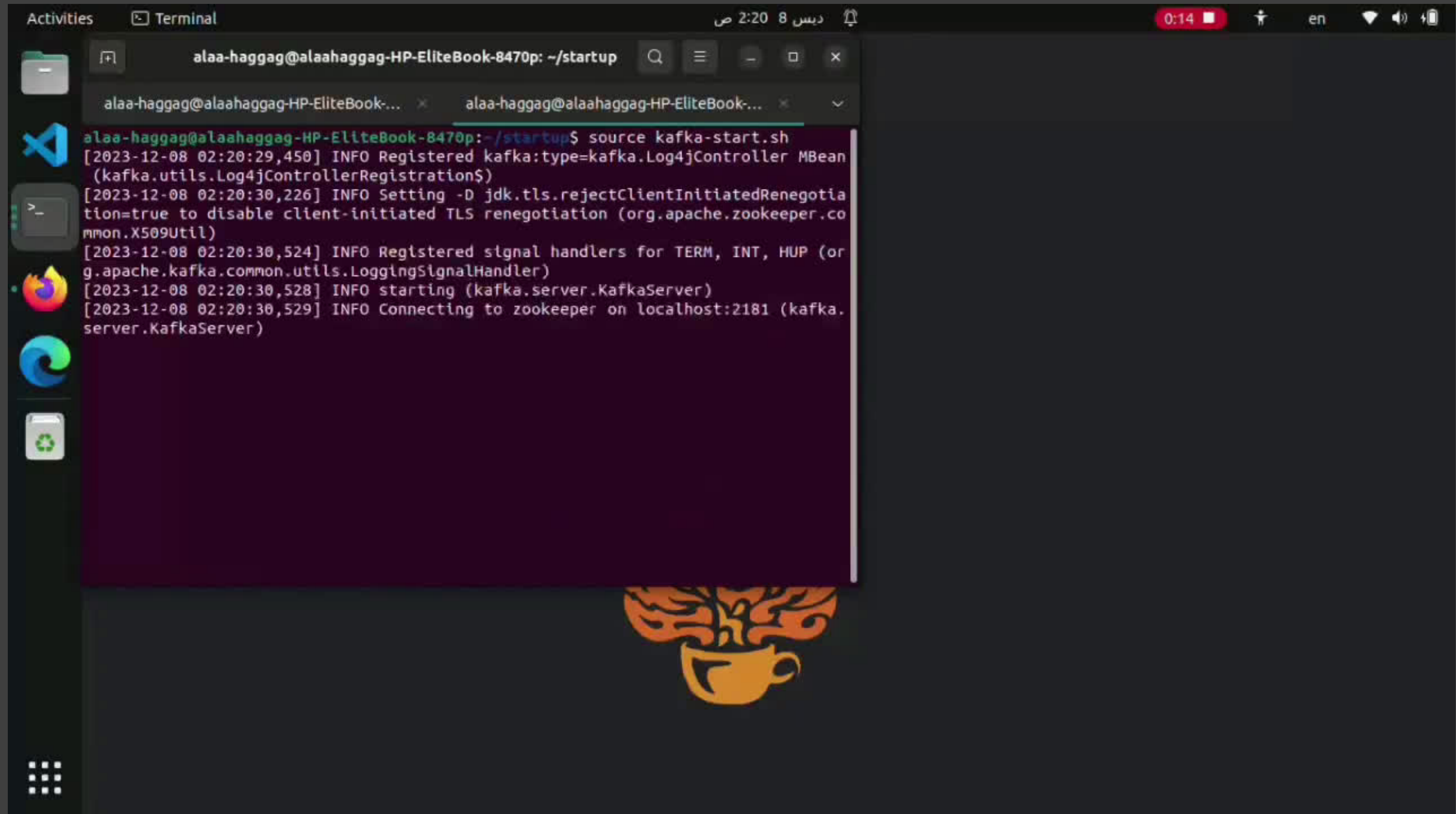


Challenges

- **AWS Account:**
 - Any holder of prepaid payment cards bearing the logos of Visa or Mastercard for electronic payments will be ineffective for use outside.
- **Service Dependencies:**
 - Services like Kafka have dependencies on other services (e.g., Zookeeper for Kafka)
- **Spark Dependencies:**
 - Ensuring all required JARs are available and compatible is essential for Spark's streaming job. Missing or incompatible JARs can lead to job failures.
- **Configuration Challenges:**
 - Ensuring environment variables and configurations are correctly set can be tricky.



Project Demo



The screenshot shows a terminal window titled "alaa-haggag@alaahaggag-HP-EliteBook-8470p: ~/startup". The terminal displays the output of the command `source kafka-start.sh`. The logs indicate that Kafka is starting successfully, including registering the Log4j controller, setting JVM options, and connecting to Zookeeper on localhost:2181. The system's top bar shows the time as 2:20 on December 8th, and the terminal window has a search bar and standard window controls. A decorative orange cup of coffee with steam is visible at the bottom center of the terminal window.

```
alaa-haggag@alaahaggag-HP-EliteBook-8470p: ~/startup
alaa-haggag@alaahaggag-HP-EliteBook-8470p:~/startup$ source kafka-start.sh
[2023-12-08 02:20:29,450] INFO Registered kafka:type=kafka.Log4jController MBean
(kafka.utils.Log4jControllerRegistration$)
[2023-12-08 02:20:30,226] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotia
tion=true to disable client-initiated TLS renegotiation (org.apache.zookeeper.co
mmon.X509Util)
[2023-12-08 02:20:30,524] INFO Registered signal handlers for TERM, INT, HUP (or
g.apache.kafka.common.utils.LoggingSignalHandler)
[2023-12-08 02:20:30,528] INFO starting (kafka.server.KafkaServer)
[2023-12-08 02:20:30,529] INFO Connecting to zookeeper on localhost:2181 (kafka.
server.KafkaServer)
```

Thank You!

Any Questions?

