

Demographically -Enhanced Movie Recommendation System for Personalized Book Suggestion in Big Data



Team Medusa

Priya Varahan
Priyanka Akula
Pooja Manjunatha
Shreya Chikatmarla
Nandini Sreekumaran Nair



Introduction



- Massive amount of data and material have made personalized recommendations more important than ever in today's world.
 - We're trying something new with our project: we're using big data to give personalized suggestions for both books and movies.
 - The project is based on AWS's stable infrastructure, which provides a solid base for users to have engaging and flexible experiences.
 - Using AWS and big data to make personalized suggestions is possible, which takes into account each person's tastes and makes users more interested.
 - The project wants to use AWS's strong technology to do more than just make suggestions. It also wants to give users an immersive and responsive experience.
- 
- 

Objectives

- Scalable Recommendation Engine
- Personalized User Experiences
- Cross-Domain Recommendation
- Optimized Resource Utilization





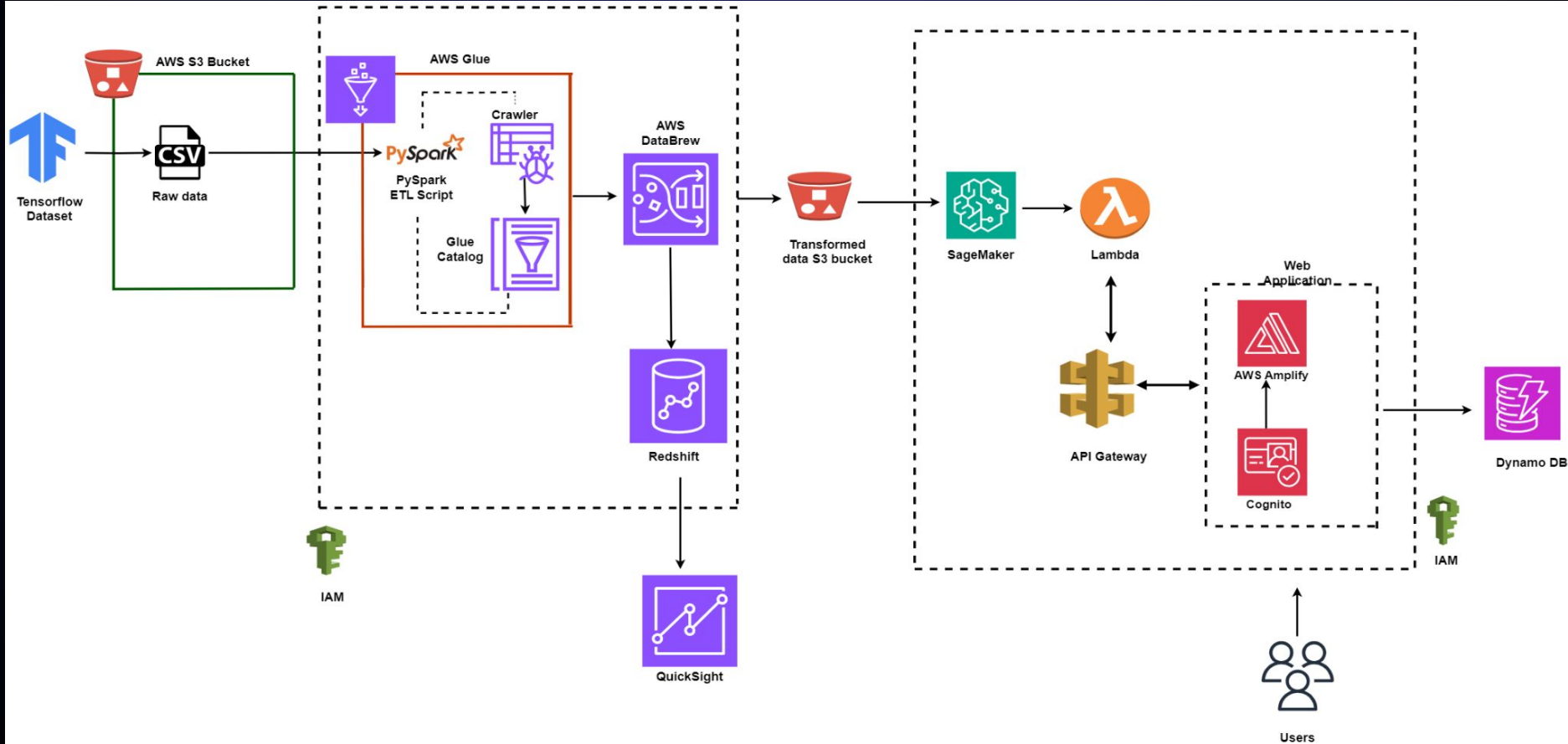
Dataset Description

- The MovieLens dataset comprises user ratings for movies, collected by the GroupLens research group.
- The "1m-ratings" version, demographic features include user gender, bucketized user age, user occupation label, user occupation text, and user zip code.
- The dataset comprises information on books, including unique identifiers, titles, author details, publication information, and reader engagement metrics.
- The unnecessary columns from the movie and book dataset are removed.

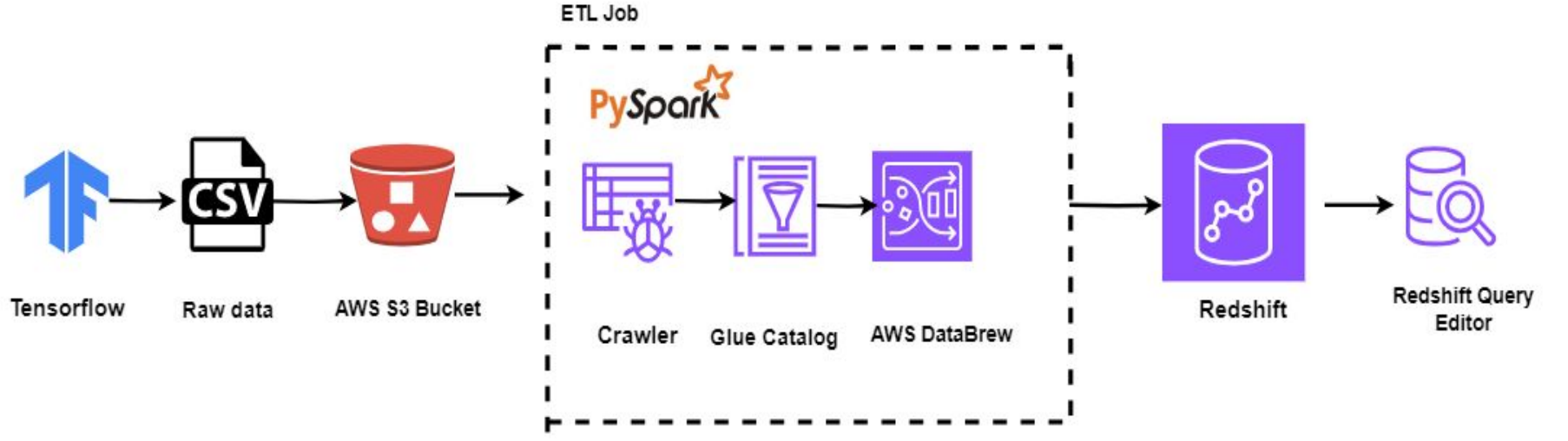
https://www.tensorflow.org/datasets/catalog/movie_lens#movie_lens1m-ratings



System Architecture



Data flow Diagram



1)The MovieLens Tensorflow dataset is initially stored in an **S3 bucket**.

2)ETL using **PySpark** with **AWS Glue Catalog** and **Crawlers**.

3)AWS **DataBrew** for Data Preparation.

4)Transformed data is stored in **Redshift**.

5)**Redshift Query editor** for queries and to visualize results.

Data Frame of Movie Dataset

```
# Display the entire DataFrame
print("Displaying the DataFrame:")
print(df)
```

Displaying the DataFrame:

	age	genres	movieid	title
0	45	[7]	b'357'	b"One Flew Over the Cuckoo's Nest (1975)"
1	25	[4 14]	b'709'	b'Strictly Ballroom (1992)'
2	18	[4]	b'412'	b'Very Brady Sequel, A (1996)'
3	50	[5 7]	b'56'	b'Pulp Fiction (1994)'
4	50	[10 16]	b'895'	b'Scream 2 (1997)'
...
99995	25	[0 1 15]	b'228'	b'Star Trek: The Wrath of Khan (1982)'
99996	35	[13 16]	b'333'	b'Game, The (1997)'
99997	18	[10]	b'567'	b"Wes Craven's New Nightmare (1994)"
99998	35	[0 10 15 16]	b'183'	b'Alien (1979)'
99999	18	[4]	b'1140'	b'Road to Wellville, The (1994)'

	usage	timestamp	gender	userid	occupation
0	46	879024327	True	b'138'	b'doctor'
1	32	875654590	True	b'92'	b'entertainment'
2	24	882075110	True	b'301'	b'student'
3	50	883326919	True	b'60'	b'healthcare'
4	55	891409199	True	b'197'	b'technician'
...
99995	30	888557237	True	b'774'	b'student'
99996	41	891012877	True	b'313'	b'marketing'
99997	19	879795430	False	b'262'	b'student'
99998	37	892839492	False	b'911'	b'writer'
99999	21	874791894	True	b'276'	b'student'

[100000 rows x 9 columns]

Data Frame of Book Dataset

```
print("Displaying the DataFrame:")
print(df)
```

Displaying the DataFrame:

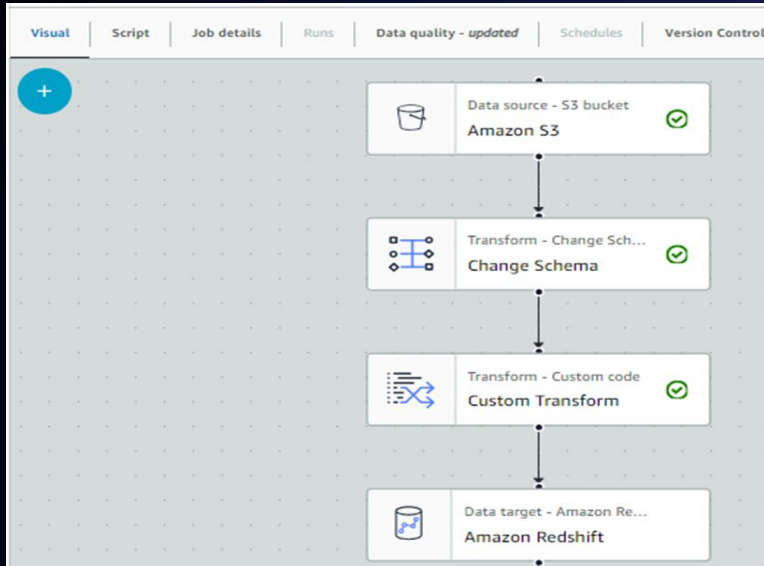
	id	title
0	0	The Martian
1	1	Under the Banner of Heaven: A Story of Violent...
2	2	Cutting for Stone
3	3	We Need to Talk About Kevin
4	4	The Immortal Life of Henrietta Lacks
...
10013	10013	Edge of Passion (Stealth Guardians, #1)
10014	10014	Tales from a Not-So-Popular Party Girl (Dork D...
10015	10015	Bottoms
10016	10016	The Wedding Dress
10017	10017	The Immortal Hunter (Argeneau #11; Rogue Hunte...

	book_link
0	https://www.goodreads.com/book/show/18007564-t...
1	https://www.goodreads.com/book/show/10847.Unde...
2	https://www.goodreads.com/book/show/3591262-cu...
3	https://www.goodreads.com/book/show/80660.We_N...
4	https://www.goodreads.com/book/show/6493208-th...
...	...
10013	https://www.goodreads.com/book/show/13516444-e...
10014	https://www.goodreads.com/book/show/8274537-ta...
10015	https://www.goodreads.com/book/show/2024071.Bo...
10016	https://www.goodreads.com/book/show/783968.The...
10017	https://www.goodreads.com/book/show/3942622-th...

	genre
0	Science Fiction,Fiction,Audiobook,Adventure,Sp...
1	Nonfiction,Religion,History,Crime,True Crime,M...
2	Fiction,Historical,Historical Fiction,Cultural...
3	Fiction,Contemporary,Thriller,Horror,Mystery,C...
4	Nonfiction,Science,History,Biography,Health,Me...
...	...
10013	Fantasy,Paranormal,Romance,Romance,Paranormal ...
10014	Childrens,Middle Grade,Realistic Fiction,Child...
10015	NaN
10016	NaN
10017	Romance,Paranormal Romance,Paranormal,Vampires...

[10018 rows x 4 columns]

ETL Process



AWS Crawlers and Glue Catalog

[AWS Glue](#) > Crawlers

Crawlers

A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables.

Crawlers (1) Info

View and manage all available crawlers.

<input type="checkbox"/>	Name	State	Schedule	Last run
<input type="checkbox"/>	newcrawler	Ready		Succeeded

[AWS Glue](#) > Tables

Tables

A table is the metadata definition that represents your data, including its schema. A table can be used as a source or target in a job definition.

Tables (1)

View and manage all available tables.

<input type="checkbox"/>	Name	Database	Location	Classification
<input type="checkbox"/>	movieuser_cleann_25nov2023_17C	sampledatabase	s3://dataset-bucket-project/movie	CSV

1. Initializes Spark and Glue contexts, and creates a Glue job.
2. Reads a CSV file from an S3 bucket into a dynamic frame .
3. Applies schema mapping to change data types of selected columns in the dynamic frame.
4. Runs a Glue crawler to discover and catalog data from the transformed dynamic frame.
5. Stores the transformed dynamic frame to Redshift for further analysis.

PySpark Script for Schema Transformation

Script (Locked) [Info](#)

```
15 # Script generated for node Amazon S3
16 AmazonS3_node1701323994859 = glueContext.create_dynamic_frame.from_options(
17     format_options={
18         "quoteChar": "'",
19         "withHeader": True,
20         "separator": ",",
21         "optimizePerformance": False,
22     },
23     connection_type="s3",
24     format="csv",
25     connection_options={
26         "paths": [
27             "s3://dataset-bucket-project/movieuser-cleann_25Nov2023_1700940945052/movie_clean.csv"
28         ],
29         "recurse": True,
30     },
31     transformation_ctx="AmazonS3_node1701323994859",
32 )
33
34 # Script generated for node Change Schema
35 ChangeSchema_node1701324004477 = ApplyMapping.apply(
36     frame=AmazonS3_node1701323994859,
37     mappings=[
38         ("bucketized_user_age", "string", "bucketized_user_age", "int"),
39         ("movie_genres", "string", "movie_genres", "string"),
40         ("movie_id", "string", "movie_id", "int"),
41         ("movie_title", "string", "movie_title", "string"),
42         ("raw_user_age", "string", "raw_user_age", "int"),
43         ("timestamp", "string", "timestamp", "timestamp"),
44         ("user_gender", "string", "user_gender", "char"),
45         ("user_id", "string", "user_id", "int"),
46         ("user_occupation_label", "string", "user_occupation_label", "string"),
47         ("user_occupation_text", "string", "user_occupation_text", "string"),
48         ("user_rating", "string", "user_rating", "int"),
49         ("user_zip_code", "string", "user_zip_code", "int"),
50     ],
51     transformation_ctx="ChangeSchema_node1701324004477",
52 )
53
```

Applied schema mapping to the dynamic frame, transforming the data types of selected columns.

AWS Glue Data Brew

The screenshot displays the AWS Glue Data Brew interface. The left pane shows a data catalog table profile for a table with 497 rows and 12 columns. The columns are: # bucketized_u..., ABC movie_genres, ABC movie_id, ABC movie_title, # raw_user_age, ABC timestamp, ABC user_gender, and ABC user_id. Each column has a histogram and summary statistics. The right pane shows a recipe editor for a recipe named 'movie-user-recipe' with 11 steps. The steps are:

1. Delete empty rows with missing values in **bucketized_user_age**
2. Delete empty rows with missing values in **movie_genres**
3. Delete empty rows with missing values in **movie_id**
4. Remove invalid values from **bucketized_user_age**
5. Remove letters, quotation marks from **movie_id**
6. Remove custom value from **movie_title**
7. Remove quotation marks from **movie_title**
8. Remove letters from **user_id**
9. Remove quotation marks from **user_id**
10. Remove custom value, quotation marks from **user_occupation_text**
11. Remove letters, quotation marks from **user_zip_code**

- 1.DataBrew will automatically profile and provide insights into the data structure and quality.
- 2.“Recipe” is used to build a series of transformations for data preparation.
- 3.Applied transformations like duplicate records, handling missing values, and formatting columns.
- 4.Exported the cleaned data to Redshift.

Querying Data in Redshift Query Editor



The screenshot displays the Redshift Query Editor v2 interface. On the left is a file explorer showing the hierarchy: Serverless: default-workg... > awsdatacatalog > sampledatabase > Tables > movieu... Below this are folders for bookuser, dev, and sample_data_dev. The top bar includes a search bar, a 'Run all' button, and a toggle for 'Isolated session'. The main area contains three query blocks, each with a 'Run' button and a 'Limit 100' toggle.

Top Movie Genres by Rating Count

```
1 SELECT movie_genres, COUNT(user_rating) AS rating_count
2 FROM your_table_name
3 GROUP BY movie_genres
4 ORDER BY rating_count DESC
5 LIMIT 5;
```

Movies with the Most Ratings

```
1 SELECT movie_title, COUNT(user_rating) AS rating_count
2 FROM your_table_name
3 GROUP BY movie_title
4 ORDER BY rating_count DESC
5 LIMIT 10;
```

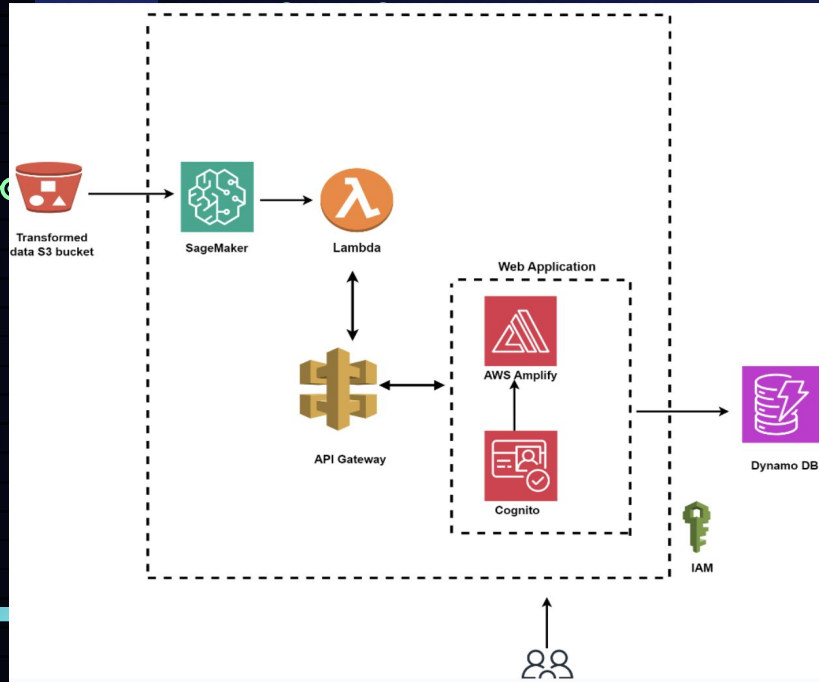
Top-Rated Movies

```
1 SELECT movie_title, AVG(user_rating) AS average_rating
2 FROM your_table_name
3 GROUP BY movie_title
4 ORDER BY average_rating DESC
5 LIMIT 10;
```

- Redshift query editor is a Web -Based SQL interface.
- The data after transformation is analysed and queried for meaningful insights.
- Queries were visualized using Built-in chart in redshift editor.

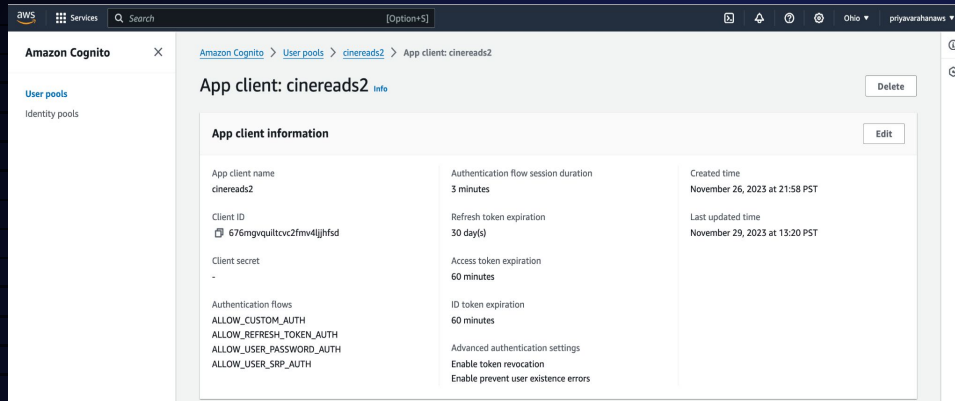


Workflow-Web Application



1. Setting up Authentication with **AWS Cognito**
2. Frontend Development with **AWS Amplify**
3. Backend API with **API Gateway** and **AWS Lambda**
4. Machine Learning with **SageMaker**
5. Connecting the Components using **IAM**
6. Storing the data in **DynamoDB**

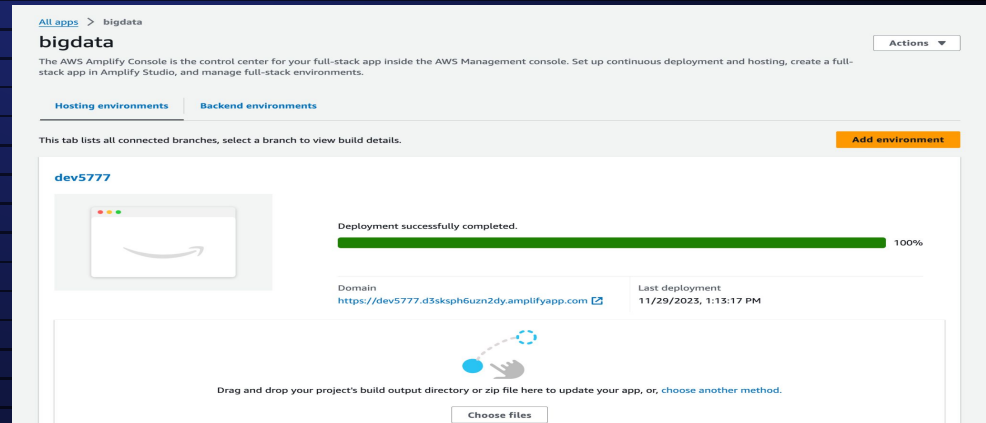
1. Setting up Authentication with AWS Cognito



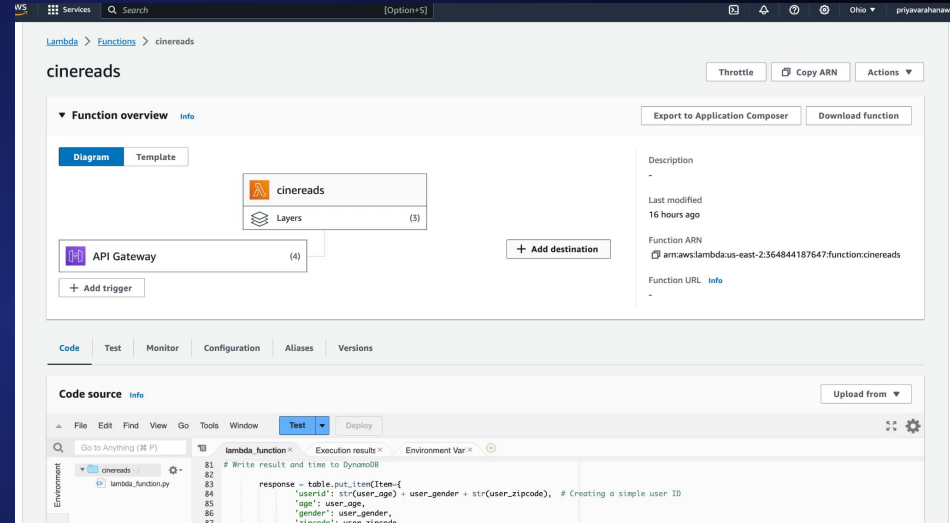
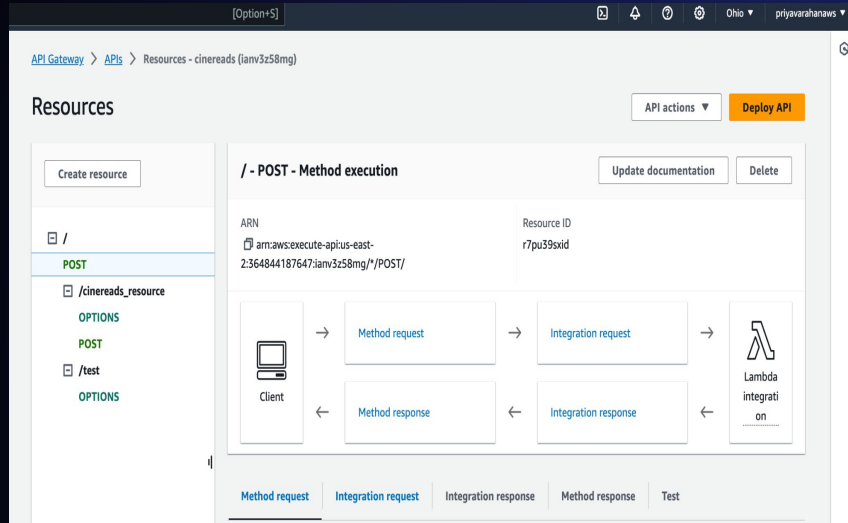
- User pool creation
- Identity pool creation
- Integration with front end

2. Frontend Development with AWS Amplify

- Initialize Amplify
- Hosting
- Frontend Frameworks
- Amplify Libraries



3.Backend API with API Gateway and AWS Lambda



- API Gateway
- Lambda functions to handle API requests
- integration between API Gateway and Lambda functions.

4. Machine Learning with SageMaker

Amazon SageMaker > Notebook instances > Medusa

Medusa

Delete Stop Open Jupyter Open JupyterLab

Notebook instance settings Edit

Name	Status	Notebook instance type	Platform identifier
Medusa	InService	ml.t3.medium	Amazon Linux 2, Jupyter Lab 3 (notebook-ai2-v2)
ARN	Creation time	Elastic Inference	Minimum IMDS Version
arn:aws:sagemaker:us-east-2:364844187647:notebook-instance/Medusa	Nov 26, 2023 04:58 UTC	-	2
Lifecycle configuration	Last updated	Volume Size	
	Nov 26, 2023 05:00 UTC	5GB EBS	

- Model Training for book and movie recommendation

5. Connecting the Components using IAM

- Secure Access
- Data Flow

IAM > Roles > s3access4lambda

s3access4lambda Info Delete

Summary Edit

Creation date	ARN
November 26, 2023, 12:09 (UTC-08:00)	arn:aws:iam::364844187647:role/service-role/s3access4lambda
Last activity	Maximum session duration
7 hours ago	1 hour

Permissions Trust relationships Tags Access Advisor Revoke sessions

Permissions policies (4) Info

You can attach up to 10 managed policies.

Filter by Type

Search All types

Policy name	Type	Attached entities
AWSConfigRulesExecutionRole	AWS managed	2
AWSLambdaBasicExecutionRole-f9a...	Customer managed	1
AWSLambdaS3ExecutionRole-3612...	Customer managed	1

6. Storing the Data in DynamoDB

You can add, remove, or edit the attributes of an item. You can nest attributes inside other attributes up to 32 levels deep. [Learn more](#)

Attributes

[Add new attribute](#)

Attribute name	Value	Type	
userid - Partition key	18M94607	String	
age	18	Number	Remove
gender	M	String	Remove
LatestGreetingTime	Wed, 29 Nov 2023 03:49:53 +0000	String	Remove
Recommendations	Insert a field	Map	Remove
Recommended Books based	Insert a field	List	Remove
0	Tea Cups & Tiger Claws	String	Remove
Recommended Movie Titles	Insert a field	List	Remove
0	Shakespeare in Love (1998)	String	Remove
1	Fight Club (1999)	String	Remove
2	Gladiator (2000)	String	Remove
3	Chinatown (1974)	String	Remove
4	Star Wars: Episode V - The Empire Strikes Back (1980)	String	Remove
zipcode	94607	Number	Remove

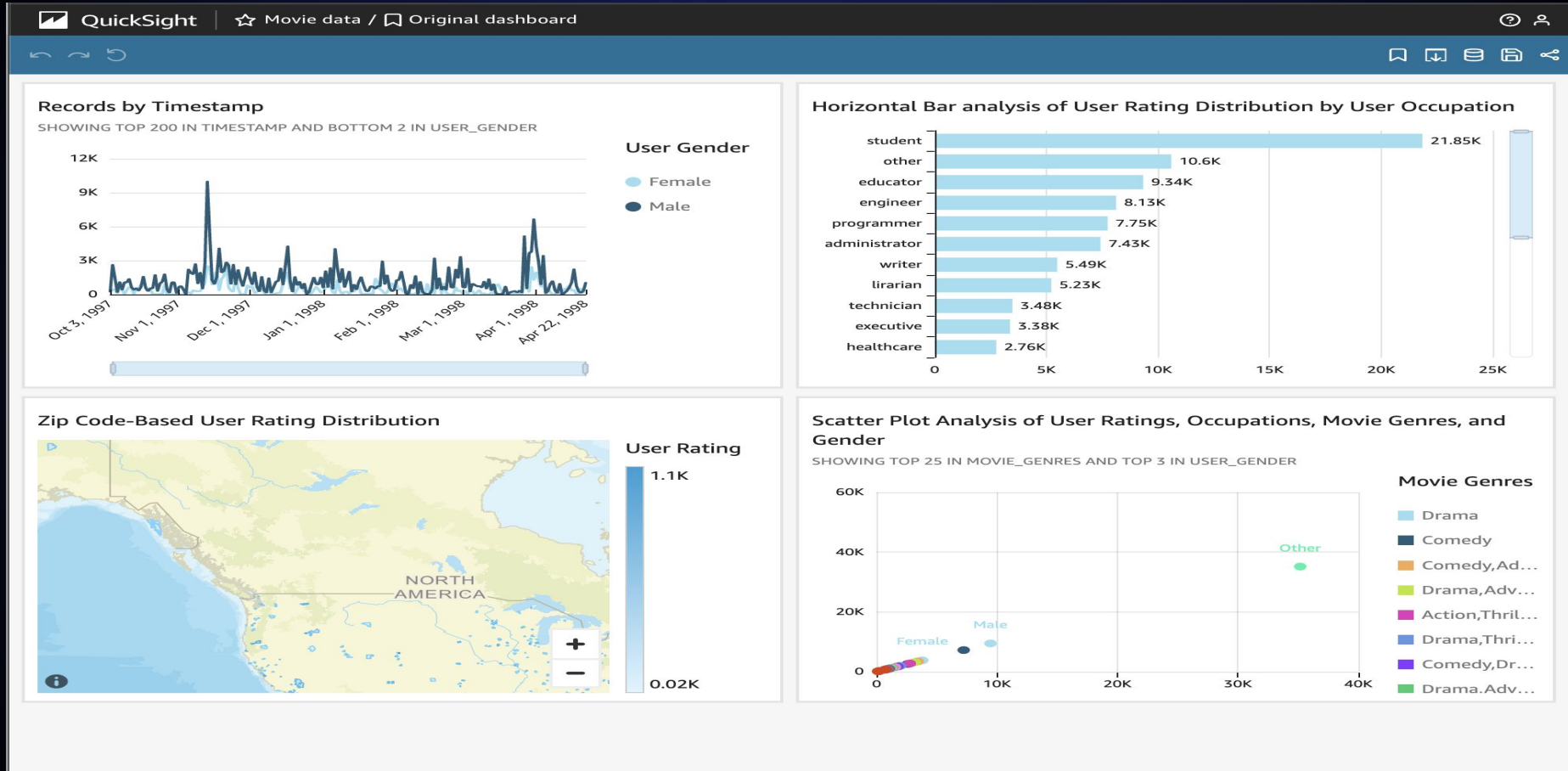
[Cancel](#) [Save](#) [Save and close](#)

- store the recommendations made by the system



Demo

Visualization



Project Impact

- The project has the capacity to significantly influence multiple sectors and demography. These are the main areas that have been impacted.

Consumer Personalization and Experience

- Enhanced User Experience
- Cross-Media Recommendations

Data Analytics and Big Data

- Leveraging Big Data

Demographic Insights

- Understanding Consumer Trends

Future Work



- Integration with Other Technologies like virtual and augmented reality, voice assistants, and other emerging technologies could further enhance user interaction and engagement.
- Avoiding Filter Bubbles
 - Serendipity in Recommendations strictly based on the user's past behavior.
 - Multi-Faceted Profiling based on a variety of factors (e.g., mood, recent news, time of day) in addition to historical preferences.
- Ethical Considerations



Conclusion



- Analyzed the data set and identified the descriptive features required for our book and movie recommendation system
- Identified and implemented algorithms to predict movie and books based on the user inputs.
- Visualisation of the available data
- Implemented a system to accept the users details to recommend movies and books

Reference

- <https://github.com/Priyankaakula/Medusa/blob/main/README.md>
- <https://us-east-2.quicksight.aws.amazon.com/sn/dashboards/0b1b3d74-f84b-4e13-af08-4cff4d4fcbfe>
- <https://dev5777.d3sksph6uzn2dy.amplifyapp.com>
- <https://app.diagrams.net/?src=about>
- https://www.tensorflow.org/datasets/catalog/movie_lens#movie_lens1m-ratings
- <https://ianv3z58mg.execute-api.us-east-2.amazonaws.com/cinereads>



The background is a dark blue gradient with various decorative elements. In the top left, there are two vertical cyan lines of different lengths. To their right is a vertical stack of five cyan chevrons pointing up. In the top right, there is a green circuit line with a small circle at its end, and a dark blue arrow pointing downwards. On the right side, there is a horizontal green circuit line with two small circles. Below it is a vertical stack of five cyan chevrons pointing down. In the bottom right, there is a horizontal row of six cyan chevrons pointing right. At the bottom, there are two horizontal cyan lines of different lengths, and a dark blue arrow pointing to the right on the left side.

Thank You