MOVIE TICKET RESERVATION SYSTEM
TEAM 7

Team Members	NUID
Krutik Kanakia	002787847
Kumar Mehul	002761391
Mansi Sanjeev Upadhyay	002766397
Tanuj Verma	002726506

Data Model: Document (NoSQL)

Target Platform: Arango DB

Objective/Scope:

- Create a Arango Database System to store movie reservation system information
- Implement Data Validation to ensure that the data entered in the database is accurate and consistent
- Use indexing to improve the performance and scalability of the database
- Use complex queries to extract maximum information from our database
- Use visualizations to discover the trends and movie popularity among the customers

Visualization Tool: Tableau

CONTENTS

I. **Architecture Diagram**
  a. Overview of system components
  b. Data flow between components

II. **Entity Relationship Diagram**
  a. Collections definitions
  b. Edges definitions

III. **Graph Diagram**
  a. Edge definitions

IV. **Data Previews**
  a. Sample data for each Collections
  b. Data formats

V. **ETL Process**
  a. Data extraction techniques
  b. Data transformation methods
  c. Data loading procedures

VI. **Importing Data into Arango DB**
  a. Collection and edge definitions with indexes
  b. Data import process
  c. Creating indexes in database collections

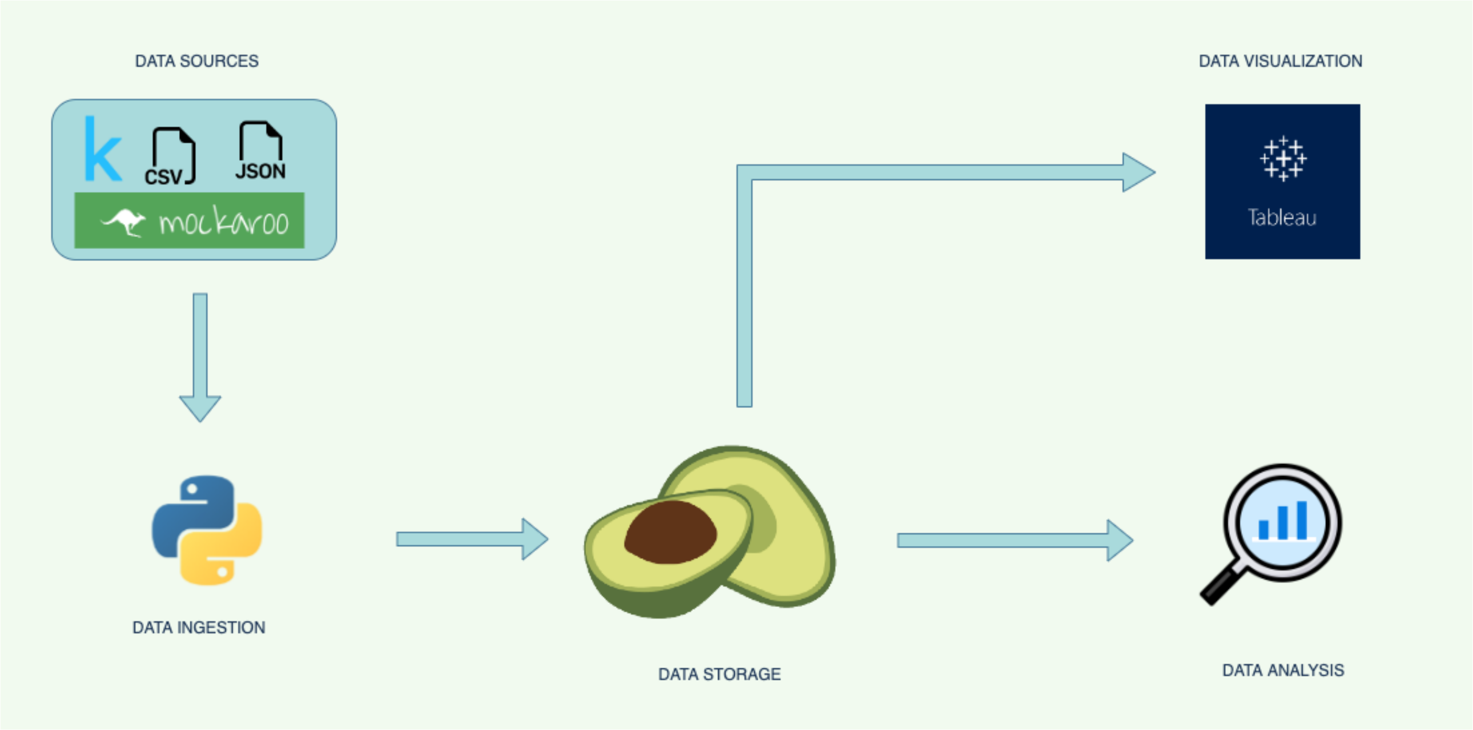
VII. **Data Refresh**
  a. Data update frequency
  b. Processes for updating the database and automating it

VIII. **Queries Executed on the Implemented Database-Reports**
  a. Query explanations and results

IX. **Visualizations**
  a. Queries for data visualization
  b. Dashboard design and layout
  c. Interpretation of visualization results

X. **Conclusion**

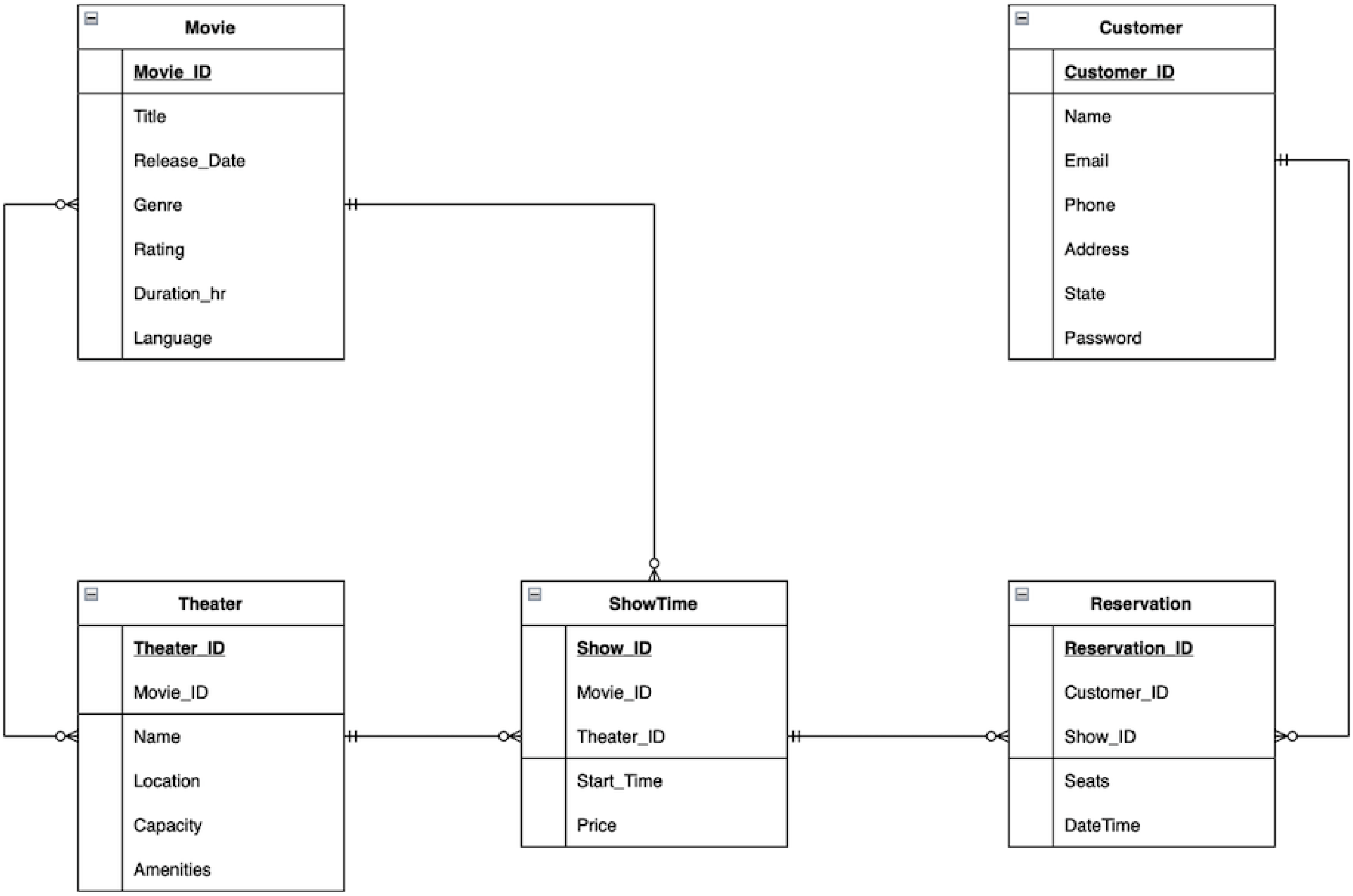
ARCHITECTURE DIAGRAM



Key takeaways:

- **Data Sources:** We used Mockaroo as our primary data source for our project.
- **Data Ingestion:** For pre-processing the data we used python script and “arangoimport” command-line tool as our source to push data into Arango DB.
- **Data Storage:** We used Arango DB as our source for data Storage.
- **Visualization:** Tableau was used for visualizing different parameters of our Database.
- **Data Analysis:** Complex DB queries was used for data analysis

ENTITY RELATIONSHIP DIAGRAM



Following are the collections used in our ER Diagram:

Collection: Customer
Attributes: ‘Customer_ID’, ‘Name’, ‘Email’, ‘Phone’, ‘Address’, ‘State’, ‘Password’
Description: Customer Entity contains the information of the Customers that book the movie ticket.
Relation: Customer is related to Reservation as the customer makes a reservation after looking at the showtimes.

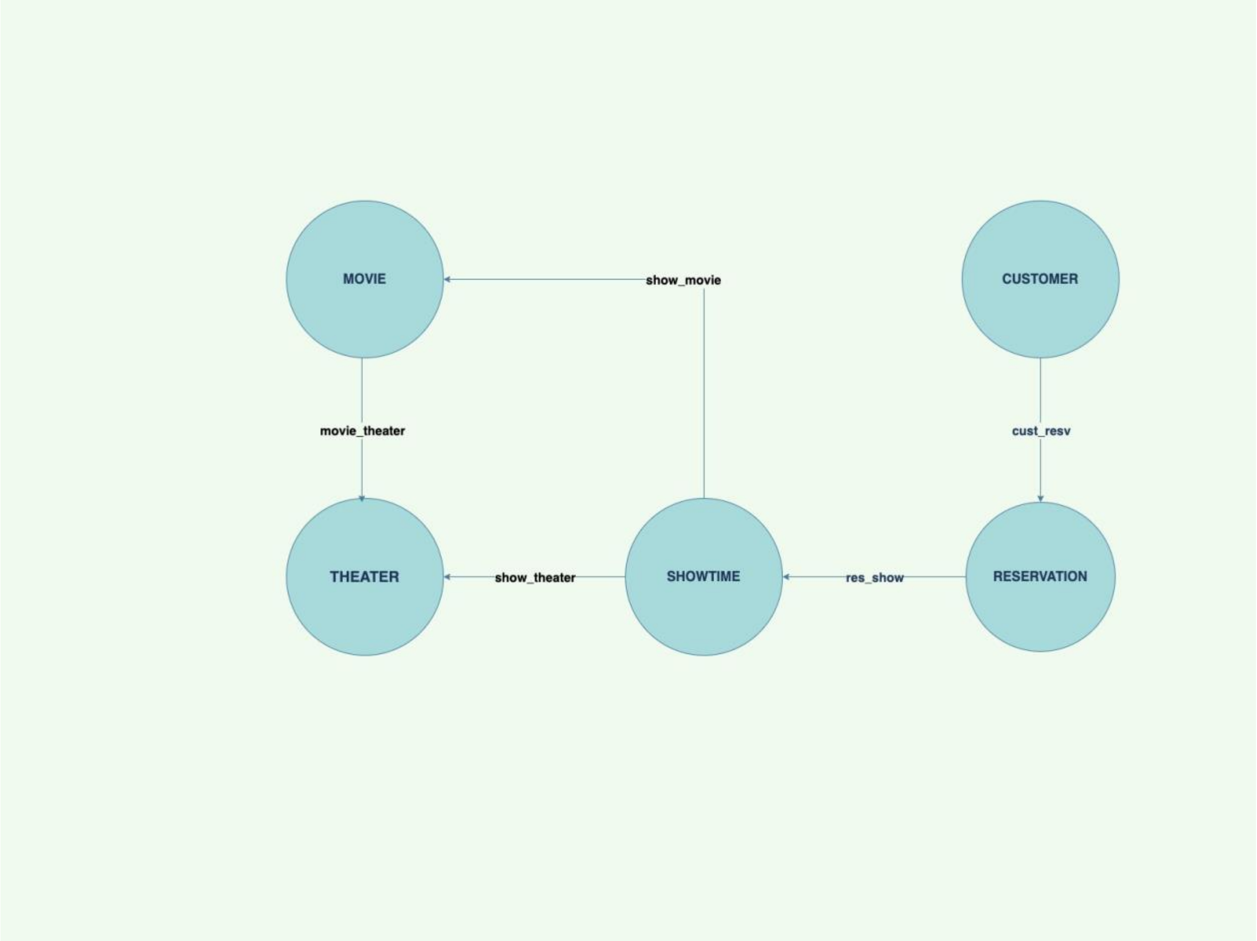
Collection: Reservation
Attributes:’Reservation_ID’,’Customer_ID’,’Show_ID’,’Seats’,’DateTime’
Description: Reservations entity contains the details of the reservation made by the customers
Relation: Reservation is related to Customer as Customer is the one making the reservation. It is also related to the Showtime collection from which it derives the details of the showtime for which the reservation has been made.

Collection: ShowTime
Attributes:’Show_ID’,’Movie_ID’,’Theater_ID’,’Start_Time’,’Price’
Description: Showtimes entity contains the details of the different showtimes of the movie according to the theaters in which the movie is being shown.
Relation: Showtimes is related to Movie and Theaters entities from which it will derive the details of the movie and the theater in which the movie is being shown.

Collection: Movie
Attributes:’Movie_ID’,’Title’,’Release_Date’,’Genre’’,Rating’,’Duration_hr’,’Language’
Description: The Movie entity stores the different details about the movie like the movie name, movie duration, genre,rating,etc.
Relation: Movie entity is related to the Theaters entity in which means the theaters in which the movie is being shown, and to the entity Showtimes which has the details about the showtimes of the different movies.

Collection: Theater
Attributes:’Theater_ID’,’Movie_ID’,’Name’,’Location’,’Capacity’,’Amenities’
Description: Theaters has the details of different theaters like the theater name, location, capacity, etc.
Relation: Theater is related to Movies from which it derives the details of the movies being shown there, and to Showtimes, showing the showtimes at the different theaters.

GRAPH DIAGRAM



EDGE COLLECTIONS

Customer and Reservation (**cust_resv**): This edge collection represents the relationship between customers and their reservations. Each edge connects a customer to their reservation, indicating which customer made which reservation.

Reservation and ShowTime (**res_show**): This edge collection represents the relationship between reservations and showtimes. Each edge connects a reservation to a showtime, indicating for which showtime the reservation has been made.

ShowTime and Movie (**show_movie**): This edge collection represents the relationship between showtimes and movies. Each edge connects a showtime to a movie, indicating which movie is being played at a specific showtime.

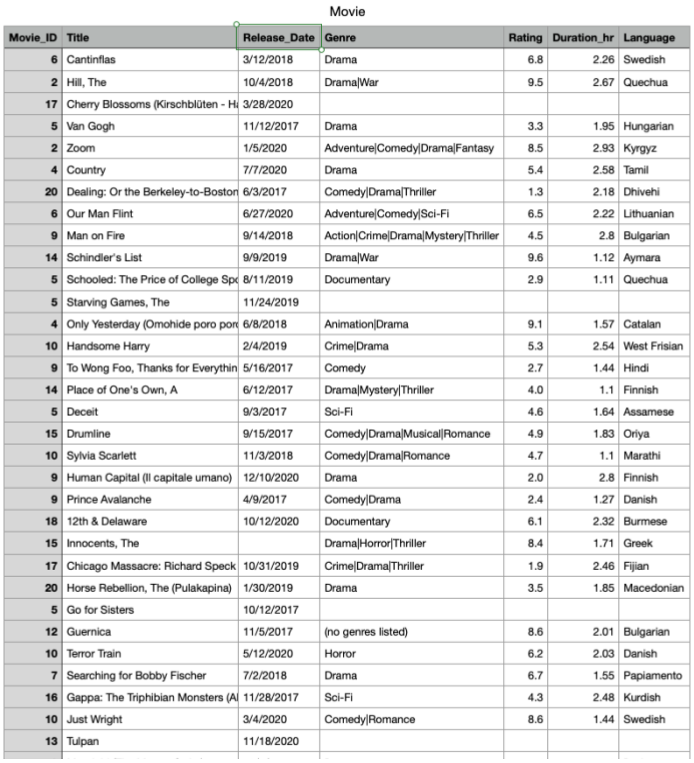
ShowTime and Theater (**show_theater**): This edge collection represents the relationship between showtimes and theaters. Each edge connects a showtime to a theater, indicating in which theater a specific movie is being played at a particular showtime.

Movie and Theater (**movie_theater**): This edge collection represents the relationship between movies and theaters. Each edge connects a movie to a theater, indicating which theaters are playing a specific movie.

DATA PREVIEWS

Below screenshots tell us the nature of data we got from our data source for all collections:

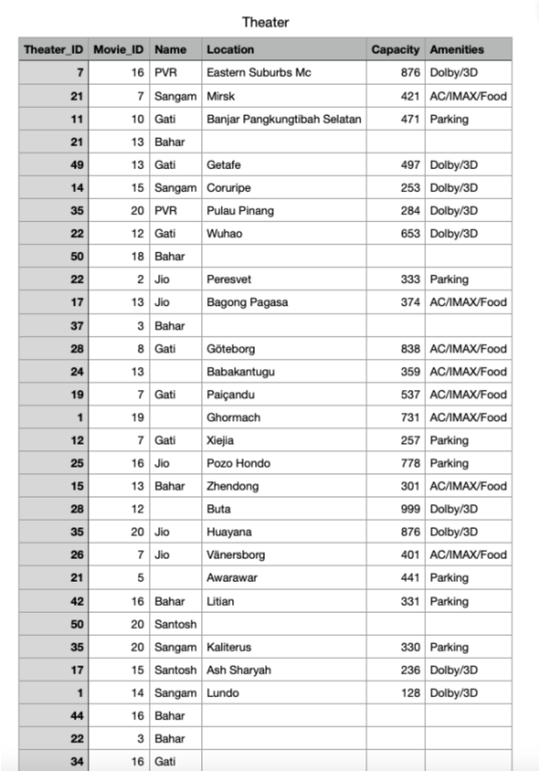
Movie



Movie

Movie_ID	Title	Release_Date	Genre	Rating	Duration_hr	Language
6	Cantinflas	3/12/2018	Drama	6.8	2.26	Swedish
2	Hill, The	10/4/2018	Drama|War	9.5	2.67	Quechua
17	Cherry Blossoms (Kirschblüten - Ha	3/28/2020				
5	Van Gogh	11/12/2017	Drama	3.3	1.95	Hungarian
2	Zoom	1/5/2020	Adventure|Comedy|Drama|Fantasy	8.5	2.93	Kyrgyz
4	Country	7/7/2020	Drama	5.4	2.58	Tamil
20	Dealing: Or the Berkeley-to-Boston	6/3/2017	Comedy|Drama|Thriller	1.3	2.18	Dhivehi
6	Our Man Flint	6/27/2020	Adventure|Comedy|Sci-Fi	6.5	2.22	Lithuanian
9	Man on Fire	9/14/2018	Action|Crime|Drama|Mystery|Thriller	4.5	2.8	Bulgarian
14	Schindler's List	9/9/2019	Drama|War	9.6	1.12	Aymara
5	Schooled: The Price of College Spo	8/11/2019	Documentary	2.9	1.11	Quechua
5	Starving Games, The	11/24/2019				
4	Only Yesterday (Omohide poro poro	6/8/2018	Animation|Drama	9.1	1.57	Catalan
10	Handsome Harry	2/4/2019	Crime|Drama	5.3	2.54	West Frisian
9	To Wong Foo, Thanks for Everythin	5/16/2017	Comedy	2.7	1.44	Hindi
14	Place of One's Own, A	6/12/2017	Drama|Mystery|Thriller	4.0	1.1	Finnish
5	Deceit	9/3/2017	Sci-Fi	4.6	1.64	Assamese
15	Drumline	9/15/2017	Comedy|Drama|Musical|Romance	4.9	1.83	Oriya
10	Sylvia Scarlett	11/3/2018	Comedy|Drama|Romance	4.7	1.1	Marathi
9	Human Capital (Il capitale umano)	12/10/2020	Drama	2.0	2.8	Finnish
9	Prince Avalanche	4/9/2017	Comedy|Drama	2.4	1.27	Danish
18	12th & Delaware	10/12/2020	Documentary	6.1	2.32	Burmese
15	Innocents, The		Drama|Horror|Thriller	8.4	1.71	Greek
17	Chicago Massacre: Richard Speck	10/31/2019	Crime|Drama|Thriller	1.9	2.46	Fijian
20	Horse Rebellion, The (Pulakapina)	1/30/2019	Drama	3.5	1.85	Macedonian
5	Go for Sisters	10/12/2017				
12	Guernica	11/5/2017	(no genres listed)	8.6	2.01	Bulgarian
10	Terror Train	5/12/2020	Horror	6.2	2.03	Danish
7	Searching for Bobby Fischer	7/2/2018	Drama	6.7	1.55	Papiamento
16	Gappa: The Triphibian Monsters (Al	11/28/2017	Sci-Fi	4.3	2.48	Kurdish
10	Just Wright	3/4/2020	Comedy|Romance	8.6	1.44	Swedish
13	Tulpan	11/18/2020				

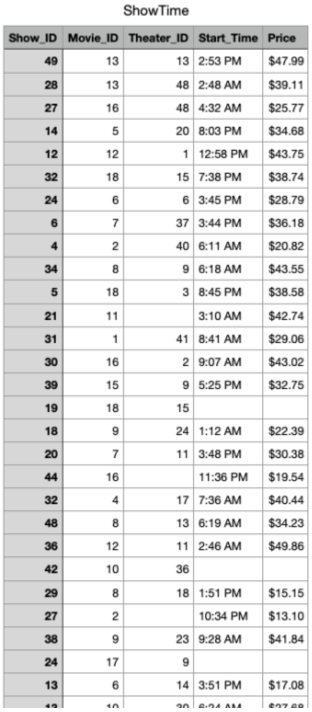
Theater



Theater

Theater_ID	Movie_ID	Name	Location	Capacity	Amenities
7	16	PVR	Eastern Suburbs Mc	876	Dolby/3D
21	7	Sangam	Mirsk	421	AC/IMAX/Food
11	10	Gati	Banjar Pangkungtibah Selatan	471	Parking
21	13	Bahar			
49	13	Gati	Getafe	497	Dolby/3D
14	15	Sangam	Coruripe	253	Dolby/3D
35	20	PVR	Pulau Pinang	284	Dolby/3D
22	12	Gati	Wuhao	653	Dolby/3D
50	18	Bahar			
22	2	Jio	Peresvet	333	Parking
17	13	Jio	Bagong Pagasa	374	AC/IMAX/Food
37	3	Bahar			
28	8	Gati	Göteborg	838	AC/IMAX/Food
24	13		Babakantugu	359	AC/IMAX/Food
19	7	Gati	Paiçandu	537	AC/IMAX/Food
1	19		Ghormach	731	AC/IMAX/Food
12	7	Gati	Xiejia	257	Parking
25	16	Jio	Pozo Hondo	778	Parking
15	13	Bahar	Zhendong	301	AC/IMAX/Food
28	12		Buta	999	Dolby/3D
35	20	Jio	Huayana	876	Dolby/3D
26	7	Jio	Vänersborg	401	AC/IMAX/Food
21	5		Awarawar	441	Parking
42	16	Bahar	Litian	331	Parking
50	20	Santosh			
35	20	Sangam	Kaliterus	330	Parking
17	15	Santosh	Ash Sharyah	236	Dolby/3D
1	14	Sangam	Lundo	128	Dolby/3D
44	16	Bahar			
22	3	Bahar			
34	16	Gati			

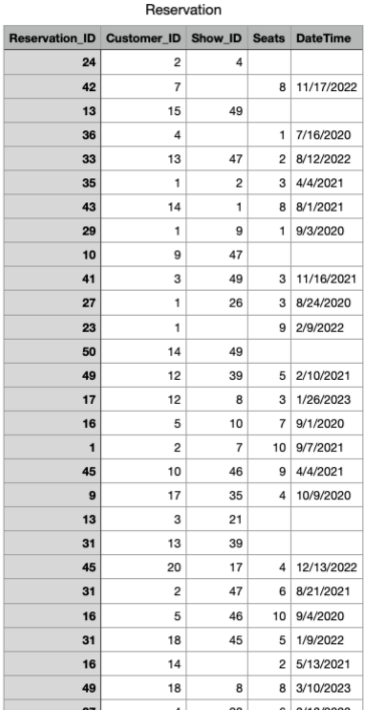
ShowTime



ShowTime

Show_ID	Movie_ID	Theater_ID	Start_Time	Price
49	13	13	2:53 PM	$47.99
28	13	48	2:48 AM	$39.11
27	16	48	4:32 AM	$25.77
14	5	20	8:03 PM	$34.68
12	12	1	12:58 PM	$43.75
32	18	15	7:38 PM	$38.74
24	6	6	3:45 PM	$28.79
6	7	37	3:44 PM	$36.18
4	2	40	6:11 AM	$20.82
34	8	9	6:18 AM	$43.55
5	18	3	8:45 PM	$38.58
21	11		3:10 AM	$42.74
31	1	41	8:41 AM	$29.06
30	16	2	9:07 AM	$43.02
39	15	9	5:25 PM	$32.75
19	18	15		
18	9	24	1:12 AM	$22.39
20	7	11	3:48 PM	$30.38
44	16		11:36 PM	$19.54
32	4	17	7:36 AM	$40.44
48	8	13	6:19 AM	$34.23
36	12	11	2:46 AM	$49.86
42	10	36		
29	8	18	1:51 PM	$15.15
27	2		10:34 PM	$13.10
38	9	23	9:28 AM	$41.84
24	17	9		
13	6	14	3:51 PM	$17.08

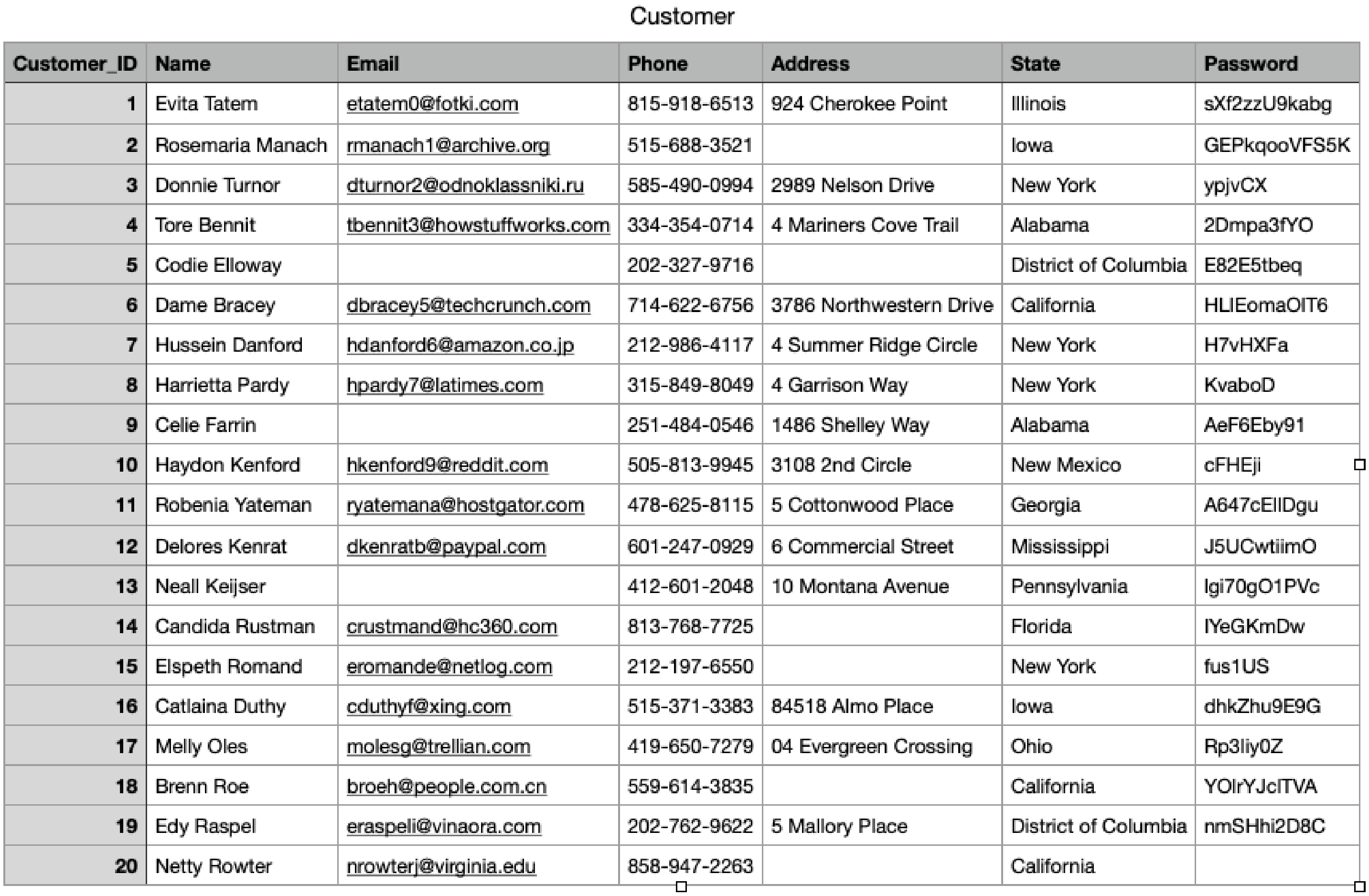
Reservations



Reservation

Reservation_ID	Customer_ID	Show_ID	Seats	DateTime
24	2	4		
42	7		8	11/17/2022
13	15	49		
36	4		1	7/16/2020
33	13	47	2	8/12/2022
35	1	2	3	4/4/2021
43	14	1	8	8/1/2021
29	1	9	1	9/3/2020
10	9	47		
41	3	49	3	11/16/2021
27	1	26	3	8/24/2020
23	1		9	2/9/2022
50	14	49		
49	12	39	5	2/10/2021
17	12	8	3	1/26/2023
16	5	10	7	9/1/2020
1	2	7	10	9/7/2021
45	10	46	9	4/4/2021
9	17	35	4	10/9/2020
13	3	21		
31	13	39		
45	20	17	4	12/13/2022
31	2	47	6	8/21/2021
16	5	46	10	9/4/2020
31	18	45	5	1/9/2022
16	14		2	5/13/2021
49	18	8	8	3/10/2023

Customer



Customer

Customer_ID	Name	Email	Phone	Address	State	Password
1	Evita Tatem	etatem0@fotki.com	815-918-6513	924 Cherokee Point	Illinois	sXf2zzU9kabg
2	Rosemaria Manach	rmanach1@archive.org	515-688-3521		Iowa	GEPkqooVFS5K
3	Donnie Turnor	dturnor2@odnoklassniki.ru	585-490-0994	2989 Nelson Drive	New York	ypjvCX
4	Tore Bennit	tbennit3@howstuffworks.com	334-354-0714	4 Mariners Cove Trail	Alabama	2Dmpa3fYO
5	Codie Elloway		202-327-9716		District of Columbia	E82E5tbeq
6	Dame Bracey	dbracey5@techcrunch.com	714-622-6756	3786 Northwestern Drive	California	HLIEomaOIT6
7	Hussein Danford	hdanford6@amazon.co.jp	212-986-4117	4 Summer Ridge Circle	New York	H7vHXFa
8	Harrietta Pardy	hpardy7@latimes.com	315-849-8049	4 Garrison Way	New York	KvaboD
9	Celie Farrin		251-484-0546	1486 Shelley Way	Alabama	AeF6Eby91
10	Haydon Kenford	hkenford9@reddit.com	505-813-9945	3108 2nd Circle	New Mexico	cFHEji
11	Robenia Yateman	ryatemana@hostgator.com	478-625-8115	5 Cottonwood Place	Georgia	A647cEllDgu
12	Delores Kenrat	dkenratb@paypal.com	601-247-0929	6 Commercial Street	Mississippi	J5UCwtiimO
13	Neall Keijser		412-601-2048	10 Montana Avenue	Pennsylvania	Igi70gO1PVc
14	Candida Rustman	crustmand@hc360.com	813-768-7725		Florida	lYeGKmDw
15	Elspeth Romand	eromande@netlog.com	212-197-6550		New York	fus1US
16	Catlaina Duthy	cduthyf@xing.com	515-371-3383	84518 Almo Place	Iowa	dhkZhu9E9G
17	Melly Oles	molesg@trellian.com	419-650-7279	04 Evergreen Crossing	Ohio	Rp3liy0Z
18	Brenn Roe	broeh@people.com.cn	559-614-3835		California	YOlrYJclTVA
19	Edy Raspel	eraspeli@vinaora.com	202-762-9622	5 Mallory Place	District of Columbia	nmSHhi2D8C
20	Netty Rowter	nrowterj@virginia.edu	858-947-2263		California	

- In above Table we observe that there are multiple columns with null values and duplicate data.
- We will clean that data using python. After cleaning data, we will be updating our original Data files with processed data.

  .

ETL PROCESS

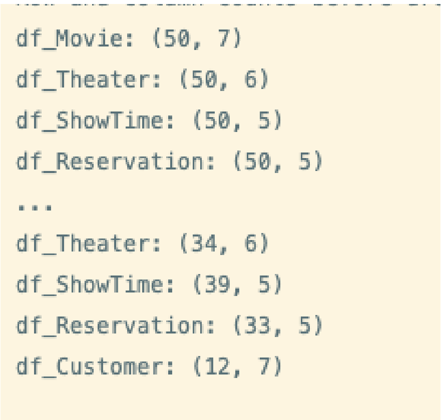
Below code reads data from five CSV files into pandas Data Frames, removes null and duplicate rows, and then saves the cleaned and transformed data back to the same CSV files

Python Script

```
import pandas as pd
import csv

#Read all the CSV files
df_Movie = pd.read_csv("/Users/tanujverma/Desktop/NEU/ADBMS/ADBMS_project/Movie.csv")
df_Theater = pd.read_csv("/Users/tanujverma/Desktop/NEU/ADBMS/ADBMS_project/Theater.csv")
df_ShowTime = pd.read_csv("/Users/tanujverma/Desktop/NEU/ADBMS/ADBMS_project/ShowTime.csv")
df_Reservation = pd.read_csv("/Users/tanujverma/Desktop/NEU/ADBMS/ADBMS_project/Reservation.csv")
df_Customer = pd.read_csv("/Users/tanujverma/Desktop/NEU/ADBMS/ADBMS_project/Customer.csv")


# Print row and column counts before dropping null and duplicate values
print("Row and column counts before dropping null and duplicate values:")
print("df_Movie:", df_Movie.shape)
print("df_Theater:", df_Theater.shape)
print("df_ShowTime:", df_ShowTime.shape)
print("df_Reservation:", df_Reservation.shape)
print("df_Customer:", df_Customer.shape)

# Drop null and duplicate values from each data frame
df_Movie.dropna(inplace=True)
df_Movie.drop_duplicates(inplace=True)

df_Theater.dropna(inplace=True)å
df_Theater.drop_duplicates(inplace=True)

df_ShowTime.dropna(inplace=True)
df_ShowTime.drop_duplicates(inplace=True)

df_Reservation.dropna(inplace=True)
df_Reservation.drop_duplicates(inplace=True)

df_Customer.dropna(inplace=True)
df_Customer.drop_duplicates(inplace=True)

# Print row and column counts after dropping null and duplicate values
```

```
print("\nRow and column counts after dropping null and duplicate values:")
print("df_Movie:", df_Movie.shape)
print("df_Theater:", df_Theater.shape)
print("df_ShowTime:", df_ShowTime.shape)
print("df_Reservation:", df_Reservation.shape)
print("df_Customer:", df_Customer.shape)
```

- **Output:**



```
df_Movie: (50, 7)
df_Theater: (50, 6)
df_ShowTime: (50, 5)
df_Reservation: (50, 5)
...
df_Theater: (34, 6)
df_ShowTime: (39, 5)
df_Reservation: (33, 5)
df_Customer: (12, 7)
```

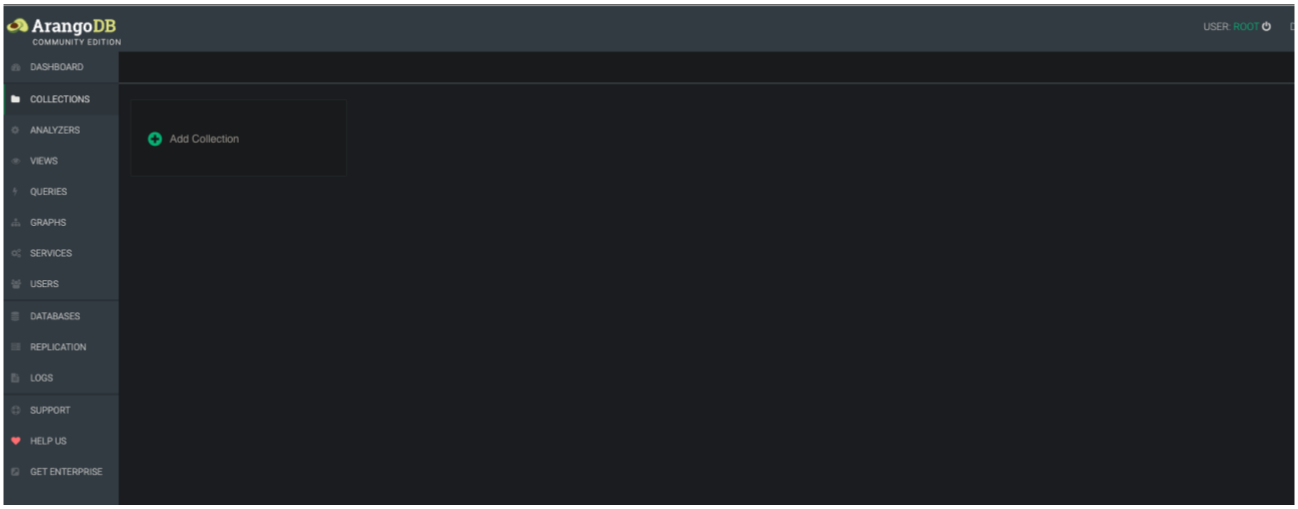
We can see here all columns with null and duplicate values are remove in latter output (reduced row counts)

- **Updated our original CSV files:**

```
df_Movie.to_csv('/Users/tanujverma/Desktop/NEU/ADBMS/ADBMS_project/Movie.csv', index=False)
df_Theater.to_csv('/Users/tanujverma/Desktop/NEU/ADBMS/ADBMS_project/Theater.csv', index=False)
df_ShowTime.to_csv('/Users/tanujverma/Desktop/NEU/ADBMS/ADBMS_project/ShowTime.csv', index=False)
df_Reservation.to_csv('/Users/tanujverma/Desktop/NEU/ADBMS/ADBMS_project/Reservation.csv', index=False)
df_Customer.to_csv('/Users/tanujverma/Desktop/NEU/ADBMS/ADBMS_project/Customer.csv', index=False)
```

Now our data is clean and can be imported to Arango DB WebUI.

WebUI before Importing the Data



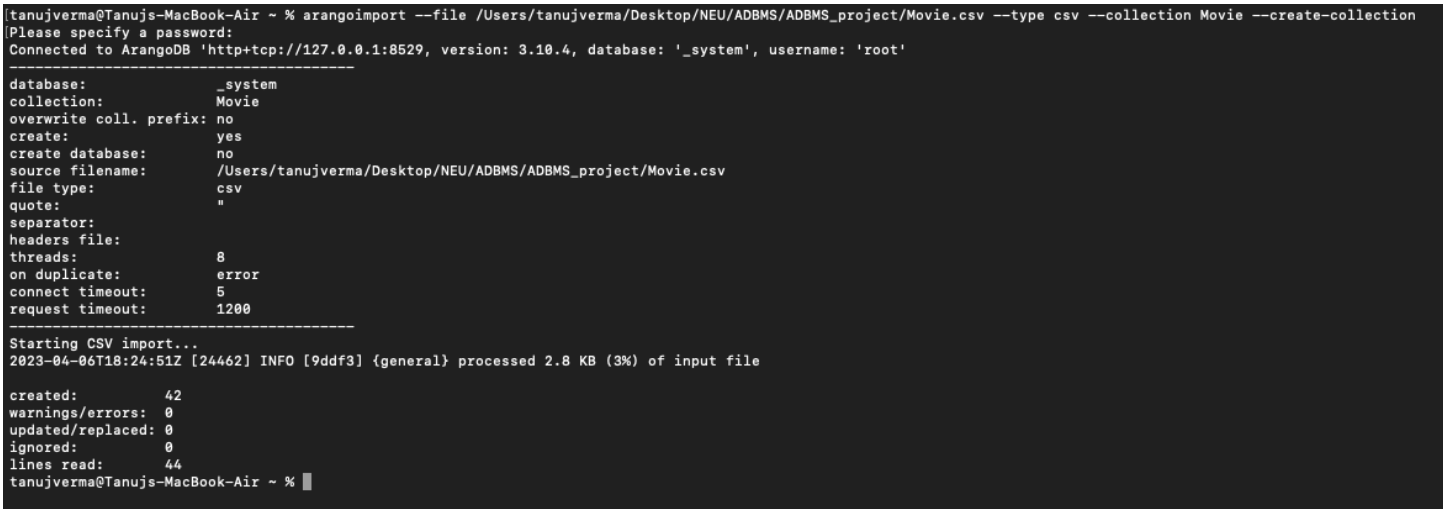
IMPORTING DATA IN ARANGO-DB: Collection and Edges

Now we will import all our **collection** Files using Arangoimport.

Queries:

We will write 5 queries for 5 collection documents which we cleaned earlier

1.arangoimport –file /Users/tanujverma/Desktop/NEU/ADBMS/ADBMS_project/Movie.csv –type csv –collection **Movie** –create-collection



```
tanujverma@Tanujs-MacBook-Air ~ % arangoimport --file /Users/tanujverma/Desktop/NEU/ADBMS/ADBMS_project/Movie.csv --type csv --collection Movie --create-collection
Please specify a password:
Connected to ArangoDB 'http+tcp://127.0.0.1:8529, version: 3.10.4, database: '_system', username: 'root'
----------------------------------------
database:               _system
collection:             Movie
overwrite coll. prefix: no
create:                 yes
create database:        no
source filename:        /Users/tanujverma/Desktop/NEU/ADBMS/ADBMS_project/Movie.csv
file type:              csv
quote:                  "
separator:
headers file:
threads:                8
on duplicate:           error
connect timeout:        5
request timeout:        1200
----------------------------------------
Starting CSV import...
2023-04-06T18:24:51Z [24462] INFO [9ddf3] {general} processed 2.8 KB (3%) of input file

created:          42
warnings/errors:  0
updated/replaced: 0
ignored:          0
lines read:       44
tanujverma@Tanujs-MacBook-Air ~ %
```

Similarly, we will use below commands to import data to our DB for remaining all collections.

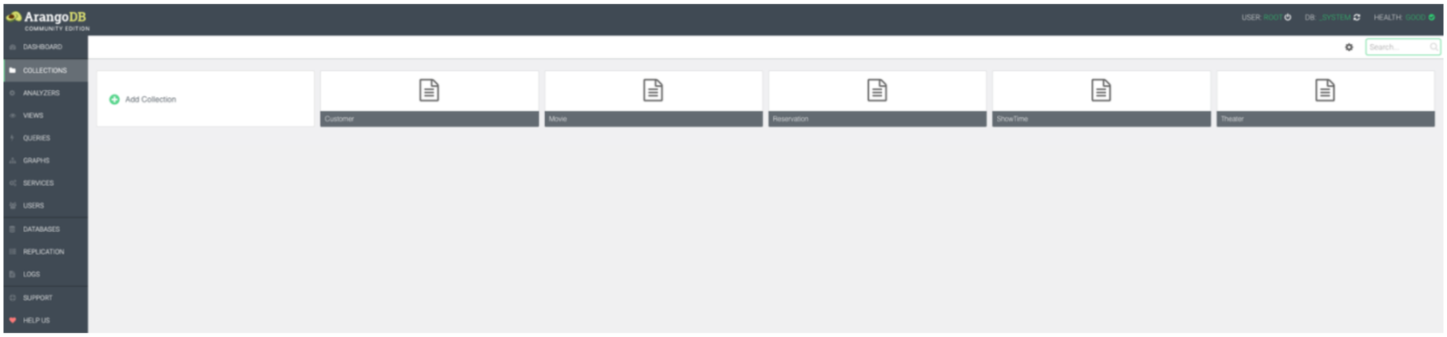
2.arangoimport –file /Users/tanujverma/Desktop/NEU/ADBMS/ADBMS_project/Reservation.csv –type csv –collection **Reservation** –create-collection

3.arangoimport –file /Users/tanujverma/Desktop/NEU/ADBMS/ADBMS_project/Theater.csv –type csv –collection **Theater** –create-collection

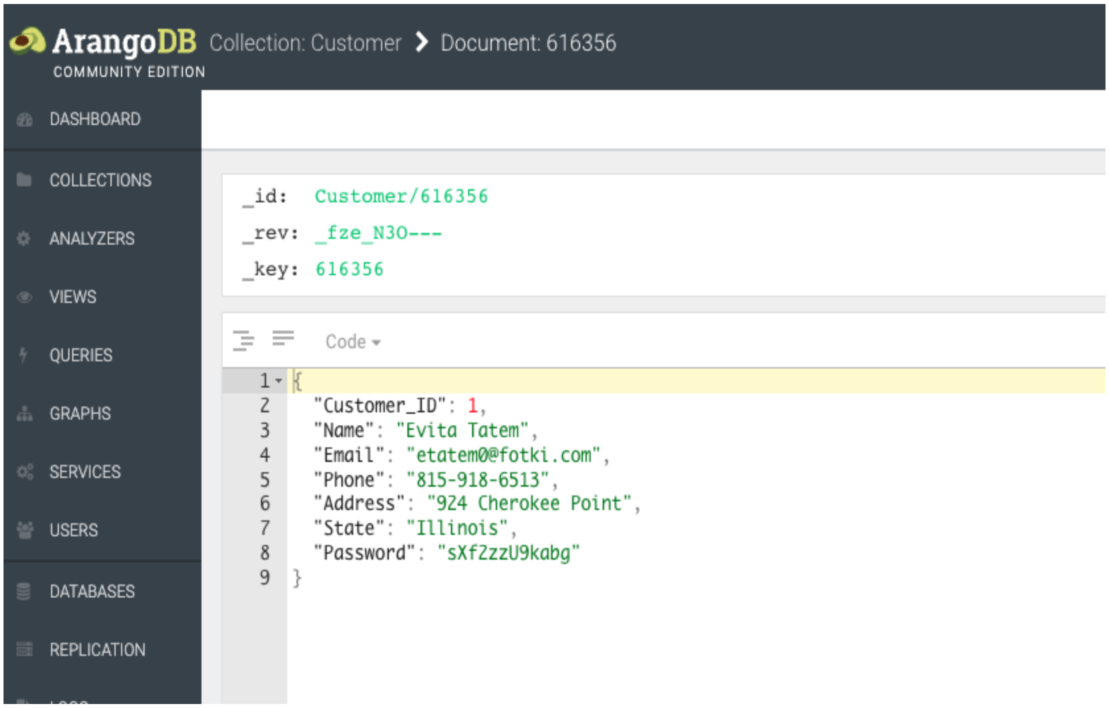
4.arangoimport –file /Users/tanujverma/Desktop/NEU/ADBMS/ADBMS_project/Customer.csv –type csv –collection **Customer** –create-collection

5.arangoimport –file /Users/tanujverma/Desktop/NEU/ADBMS/ADBMS_project/ShowTime.csv –type csv –collection **ShowTime** –create-collection

After importing data out UI will look like:



Customer_ID :1 data



_id: Customer/616356

_rev: _fze_N3O---

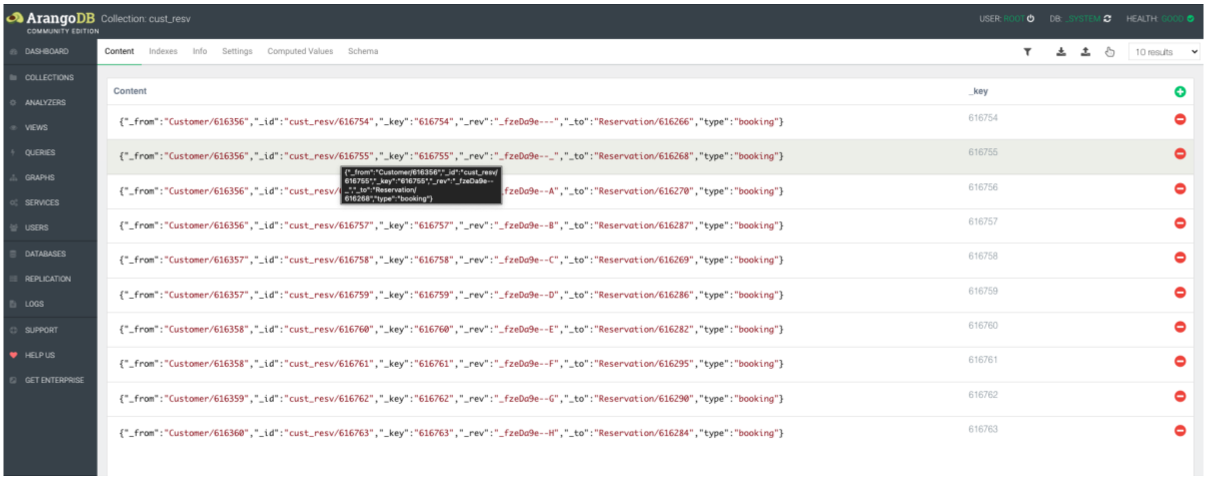
_key: 616356

```
{
  "Customer_ID": 1,
  "Name": "Evita Tatem",
  "Email": "etatem0@fotki.com",
  "Phone": "815-918-6513",
  "Address": "924 Cherokee Point",
  "State": "Illinois",
  "Password": "sXf2zzU9kabg"
}
```

Now we will write AQL queries for all our Edges(total:5) in Web UI

1.Customer and Reservation:cust_resv

```
FOR c IN Customer
  FOR r IN Reservation
    FILTER r.Customer_ID == c.Customer_ID
    INSERT {
      "_from": c._id,
      "_to": r._id,
      "type": "booking"
    } INTO cust_resv
```

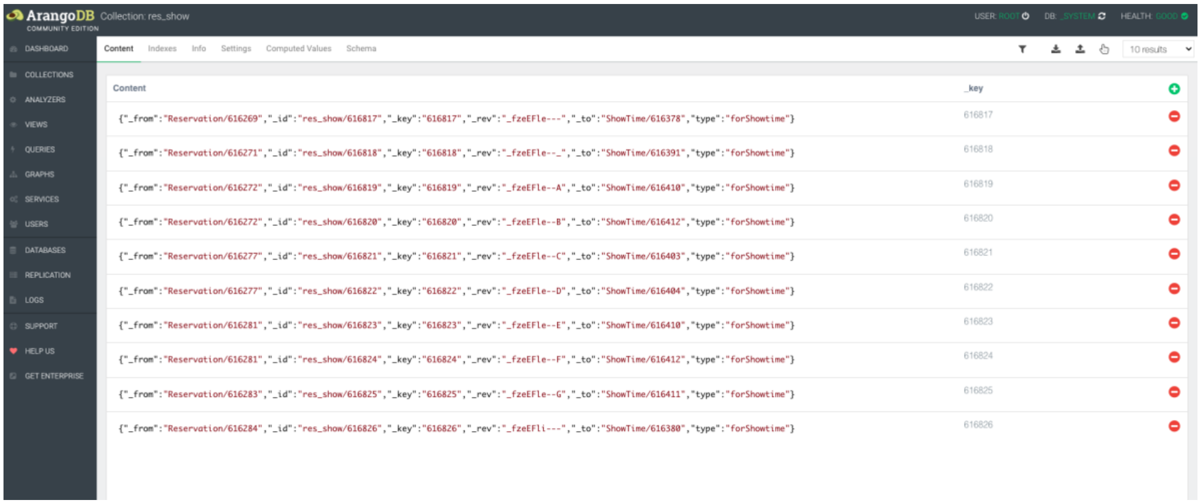


```
Content                                                                                                                                                        _key
{"_from":"Customer/616356","_id":"cust_resv/616754","_key":"616754","_rev":"_fzeDa9e---","_to":"Reservation/616266","type":"booking"}   616754
{"_from":"Customer/616356","_id":"cust_resv/616755","_key":"616755","_rev":"_fzeDa9e--_","_to":"Reservation/616268","type":"booking"}   616755
{"_from":"Customer/616356","_id":"cust_resv/  fzeDa9e--A","_to":"Reservation/616270","type":"booking"}   616756
{"_from":"Customer/616356","_id":"cust_resv/616757","_key":"616757","_rev":"_fzeDa9e--B","_to":"Reservation/616287","type":"booking"}   616757
{"_from":"Customer/616357","_id":"cust_resv/616758","_key":"616758","_rev":"_fzeDa9e--C","_to":"Reservation/616269","type":"booking"}   616758
{"_from":"Customer/616357","_id":"cust_resv/616759","_key":"616759","_rev":"_fzeDa9e--D","_to":"Reservation/616286","type":"booking"}   616759
{"_from":"Customer/616358","_id":"cust_resv/616760","_key":"616760","_rev":"_fzeDa9e--E","_to":"Reservation/616282","type":"booking"}   616760
{"_from":"Customer/616358","_id":"cust_resv/616761","_key":"616761","_rev":"_fzeDa9e--F","_to":"Reservation/616295","type":"booking"}   616761
{"_from":"Customer/616359","_id":"cust_resv/616762","_key":"616762","_rev":"_fzeDa9e--G","_to":"Reservation/616290","type":"booking"}   616762
{"_from":"Customer/616360","_id":"cust_resv/616763","_key":"616763","_rev":"_fzeDa9e--H","_to":"Reservation/616284","type":"booking"}   616763
```

```
{"_from":"Customer/616356","_id":"cust_resv/
616755","_key":"616755","_rev":"_fzeDa9e--
_","_to":"Reservation/
616268","type":"booking"}
```

2.Reservation and ShowTime: res_show

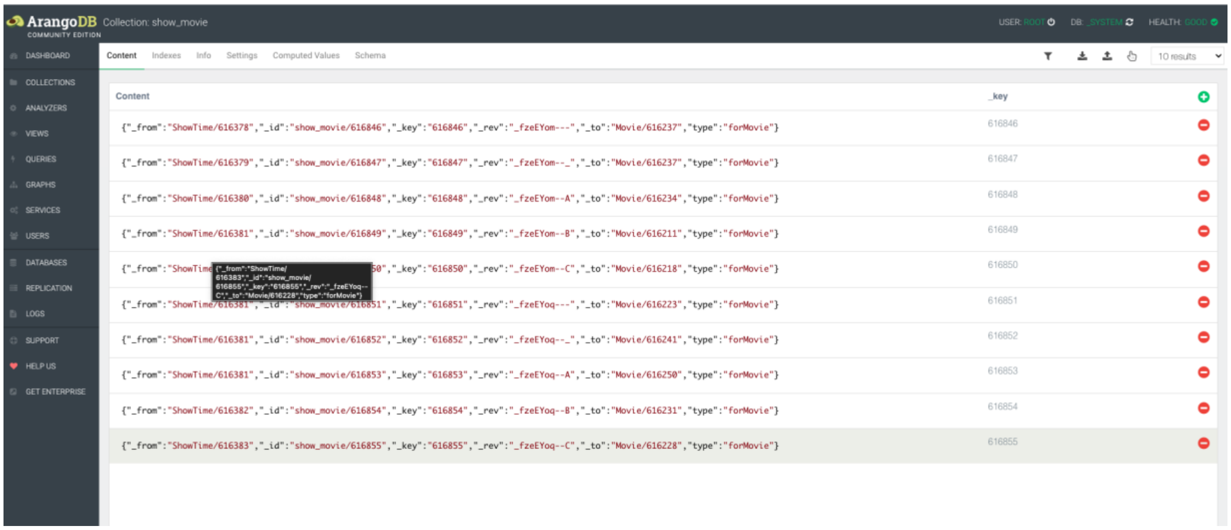
```
FOR r IN Reservation
  FOR s IN ShowTime
    FILTER r.Show_ID == s.Show_ID
    INSERT {
      “_from”: r._id,
      “_to”: s._id,
      “type”: “forShowtime”
    } INTO res_show
```



```
Content                                                                                                                                                            _key
{"_from":"Reservation/616269","_id":"res_show/616817","_key":"616817","_rev":"_fzeEFle---","_to":"ShowTime/616378","type":"forShowtime"}   616817
{"_from":"Reservation/616271","_id":"res_show/616818","_key":"616818","_rev":"_fzeEFle--_","_to":"ShowTime/616391","type":"forShowtime"}   616818
{"_from":"Reservation/616272","_id":"res_show/616819","_key":"616819","_rev":"_fzeEFle--A","_to":"ShowTime/616410","type":"forShowtime"}   616819
{"_from":"Reservation/616272","_id":"res_show/616820","_key":"616820","_rev":"_fzeEFle--B","_to":"ShowTime/616412","type":"forShowtime"}   616820
{"_from":"Reservation/616277","_id":"res_show/616821","_key":"616821","_rev":"_fzeEFle--C","_to":"ShowTime/616403","type":"forShowtime"}   616821
{"_from":"Reservation/616277","_id":"res_show/616822","_key":"616822","_rev":"_fzeEFle--D","_to":"ShowTime/616404","type":"forShowtime"}   616822
{"_from":"Reservation/616281","_id":"res_show/616823","_key":"616823","_rev":"_fzeEFle--E","_to":"ShowTime/616410","type":"forShowtime"}   616823
{"_from":"Reservation/616281","_id":"res_show/616824","_key":"616824","_rev":"_fzeEFle--F","_to":"ShowTime/616412","type":"forShowtime"}   616824
{"_from":"Reservation/616283","_id":"res_show/616825","_key":"616825","_rev":"_fzeEFle--G","_to":"ShowTime/616411","type":"forShowtime"}   616825
{"_from":"Reservation/616284","_id":"res_show/616826","_key":"616826","_rev":"_fzeEFli---","_to":"ShowTime/616380","type":"forShowtime"}   616826
```

3.ShowTime and Movie : show_movie

```
FOR s IN ShowTime
  FOR m IN Movie
    FILTER s.Movie_ID == m.Movie_ID
    INSERT {
      “_from”: s._id,
      “_to”: m._id,
      “type”: “forMovie”
    } INTO show_movie
```



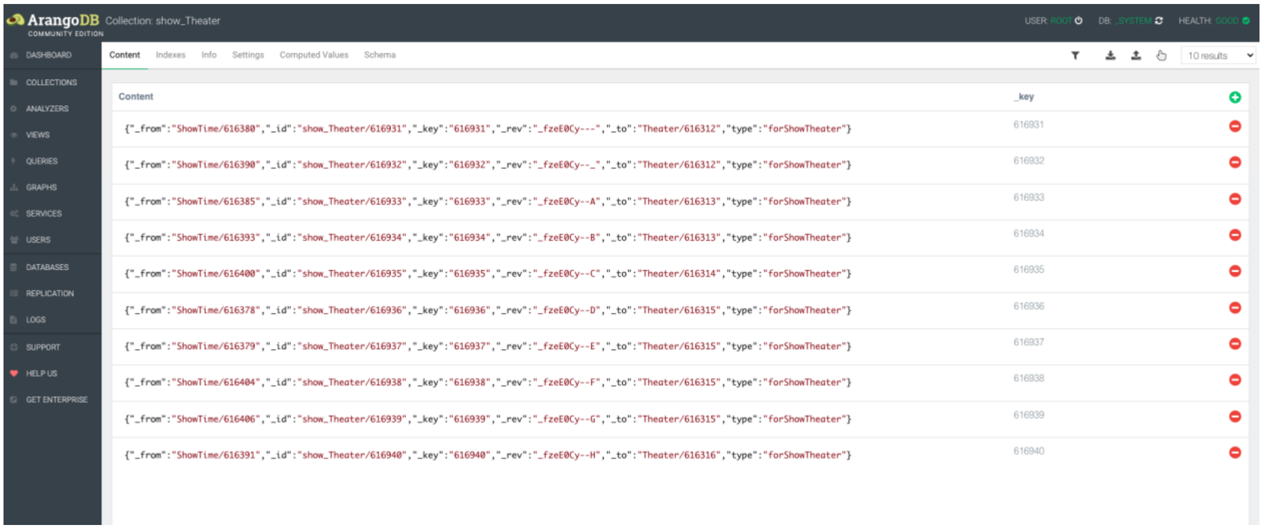
Content	_key
{"_from":"ShowTime/616378","_id":"show_movie/616846","_key":"616846","_rev":"_fzeEYom---","_to":"Movie/616237","type":"forMovie"}	616846
{"_from":"ShowTime/616379","_id":"show_movie/616847","_key":"616847","_rev":"_fzeEYom--_","_to":"Movie/616237","type":"forMovie"}	616847
{"_from":"ShowTime/616380","_id":"show_movie/616848","_key":"616848","_rev":"_fzeEYom--A","_to":"Movie/616234","type":"forMovie"}	616848
{"_from":"ShowTime/616381","_id":"show_movie/616849","_key":"616849","_rev":"_fzeEYom--B","_to":"Movie/616211","type":"forMovie"}	616849

{"_from":"ShowTime
{"_from":"ShowTime/616383","_id":"show_movie/616855","_key":"616855","_rev":"_fzeEYoq--C","_to":"Movie/616228","type":"forMovie"}
50","_key":"616850","_rev":"_fzeEYom--C","_to":"Movie/616218","type":"forMovie"}
	616850

{"_from":"ShowTime/616381","_id":"show_movie/616851","_key":"616851","_rev":"_fzeEYoq---","_to":"Movie/616223","type":"forMovie"}	616851
{"_from":"ShowTime/616381","_id":"show_movie/616852","_key":"616852","_rev":"_fzeEYoq--_","_to":"Movie/616241","type":"forMovie"}	616852
{"_from":"ShowTime/616381","_id":"show_movie/616853","_key":"616853","_rev":"_fzeEYoq--A","_to":"Movie/616250","type":"forMovie"}	616853
{"_from":"ShowTime/616382","_id":"show_movie/616854","_key":"616854","_rev":"_fzeEYoq--B","_to":"Movie/616231","type":"forMovie"}	616854
{"_from":"ShowTime/616383","_id":"show_movie/616855","_key":"616855","_rev":"_fzeEYoq--C","_to":"Movie/616228","type":"forMovie"}	616855

4.ShowTime and Theater: show_theater

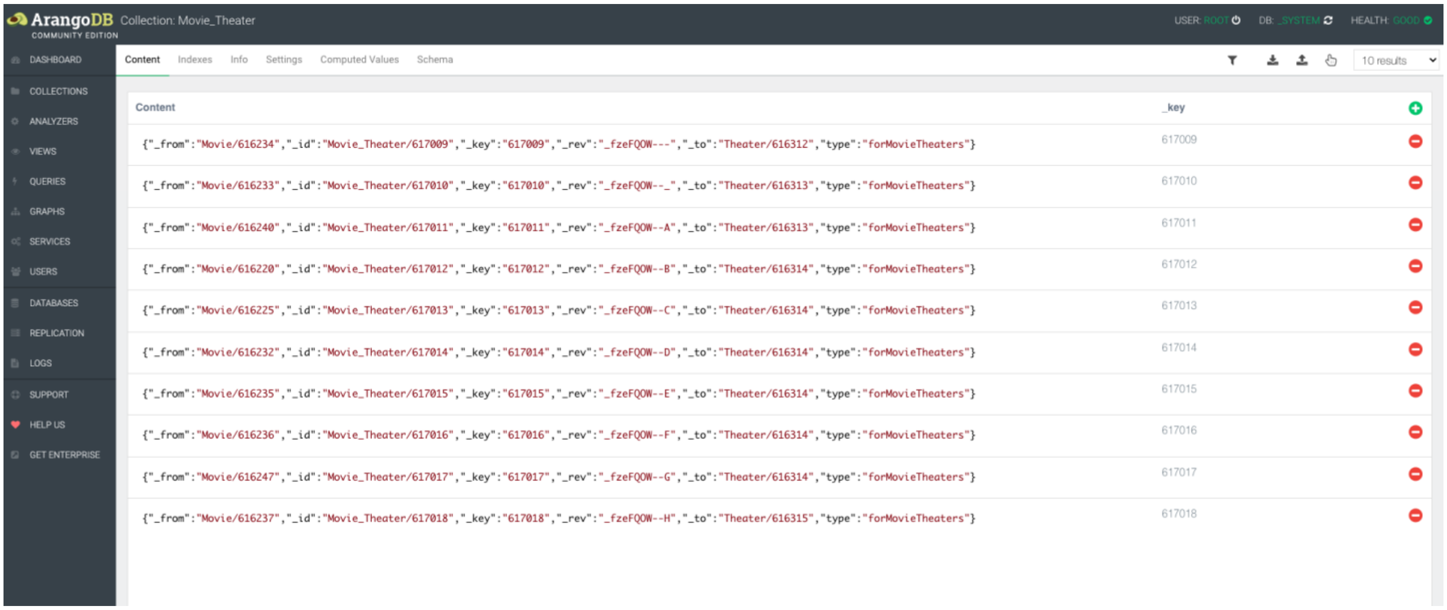
```
FOR s IN ShowTime
  FOR t IN Theater
    FILTER s.Movie_ID == t.Movie_ID
    INSERT {
      “_from”: s._id,
      “_to”: t._id,
      “type”: “forshowtheater”
    } INTO show_theater
```



Content	_key
{"_from":"ShowTime/616380","_id":"show_Theater/616931","_key":"616931","_rev":"_fzeE0Cy---","_to":"Theater/616312","type":"forShowTheater"}	616931
{"_from":"ShowTime/616390","_id":"show_Theater/616932","_key":"616932","_rev":"_fzeE0Cy--_","_to":"Theater/616312","type":"forShowTheater"}	616932
{"_from":"ShowTime/616385","_id":"show_Theater/616933","_key":"616933","_rev":"_fzeE0Cy--A","_to":"Theater/616313","type":"forShowTheater"}	616933
{"_from":"ShowTime/616393","_id":"show_Theater/616934","_key":"616934","_rev":"_fzeE0Cy--B","_to":"Theater/616313","type":"forShowTheater"}	616934
{"_from":"ShowTime/616400","_id":"show_Theater/616935","_key":"616935","_rev":"_fzeE0Cy--C","_to":"Theater/616314","type":"forShowTheater"}	616935
{"_from":"ShowTime/616378","_id":"show_Theater/616936","_key":"616936","_rev":"_fzeE0Cy--D","_to":"Theater/616315","type":"forShowTheater"}	616936
{"_from":"ShowTime/616379","_id":"show_Theater/616937","_key":"616937","_rev":"_fzeE0Cy--E","_to":"Theater/616315","type":"forShowTheater"}	616937
{"_from":"ShowTime/616404","_id":"show_Theater/616938","_key":"616938","_rev":"_fzeE0Cy--F","_to":"Theater/616315","type":"forShowTheater"}	616938
{"_from":"ShowTime/616406","_id":"show_Theater/616939","_key":"616939","_rev":"_fzeE0Cy--G","_to":"Theater/616315","type":"forShowTheater"}	616939
{"_from":"ShowTime/616391","_id":"show_Theater/616940","_key":"616940","_rev":"_fzeE0Cy--H","_to":"Theater/616316","type":"forShowTheater"}	616940

5.Movie and Theater : movie_theater

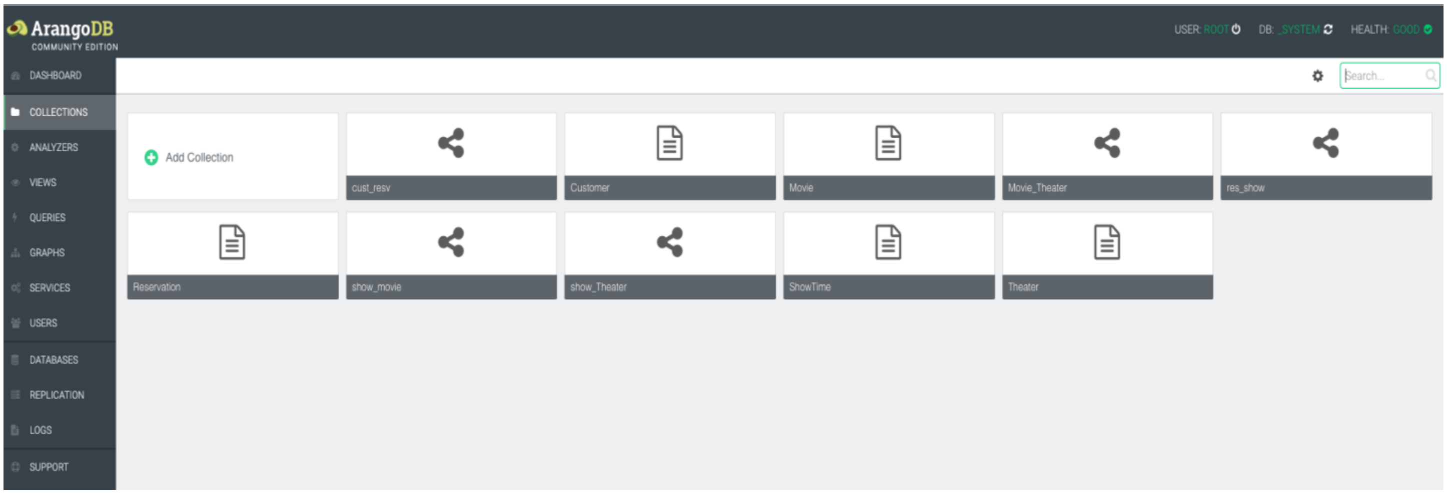
```
FOR m IN Movie
  FOR t IN Theater
    FILTER m.Movie_ID == t.Movie_ID
    INSERT {
      “_from”: m._id,
      “_to”: t._id,
      “type”: “formovietheaters”
          } INTO movie_theater
```



Collection: Movie_Theater

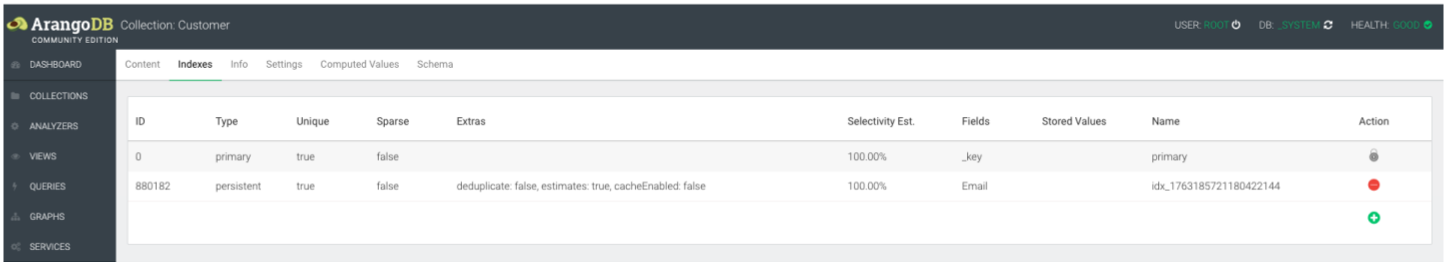
Content	_key
{"_from":"Movie/616234","_id":"Movie_Theater/617009","_key":"617009","_rev":"_fzeFQOW---","_to":"Theater/616312","type":"forMovieTheaters"}	617009
{"_from":"Movie/616233","_id":"Movie_Theater/617010","_key":"617010","_rev":"_fzeFQOW--_","_to":"Theater/616313","type":"forMovieTheaters"}	617010
{"_from":"Movie/616240","_id":"Movie_Theater/617011","_key":"617011","_rev":"_fzeFQOW--A","_to":"Theater/616313","type":"forMovieTheaters"}	617011
{"_from":"Movie/616220","_id":"Movie_Theater/617012","_key":"617012","_rev":"_fzeFQOW--B","_to":"Theater/616314","type":"forMovieTheaters"}	617012
{"_from":"Movie/616225","_id":"Movie_Theater/617013","_key":"617013","_rev":"_fzeFQOW--C","_to":"Theater/616314","type":"forMovieTheaters"}	617013
{"_from":"Movie/616232","_id":"Movie_Theater/617014","_key":"617014","_rev":"_fzeFQOW--D","_to":"Theater/616314","type":"forMovieTheaters"}	617014
{"_from":"Movie/616235","_id":"Movie_Theater/617015","_key":"617015","_rev":"_fzeFQOW--E","_to":"Theater/616314","type":"forMovieTheaters"}	617015
{"_from":"Movie/616236","_id":"Movie_Theater/617016","_key":"617016","_rev":"_fzeFQOW--F","_to":"Theater/616314","type":"forMovieTheaters"}	617016
{"_from":"Movie/616247","_id":"Movie_Theater/617017","_key":"617017","_rev":"_fzeFQOW--G","_to":"Theater/616314","type":"forMovieTheaters"}	617017
{"_from":"Movie/616237","_id":"Movie_Theater/617018","_key":"617018","_rev":"_fzeFQOW--H","_to":"Theater/616315","type":"forMovieTheaters"}	617018

Now our database is fully implemented. Screenshot of all Edge and collection files are below:



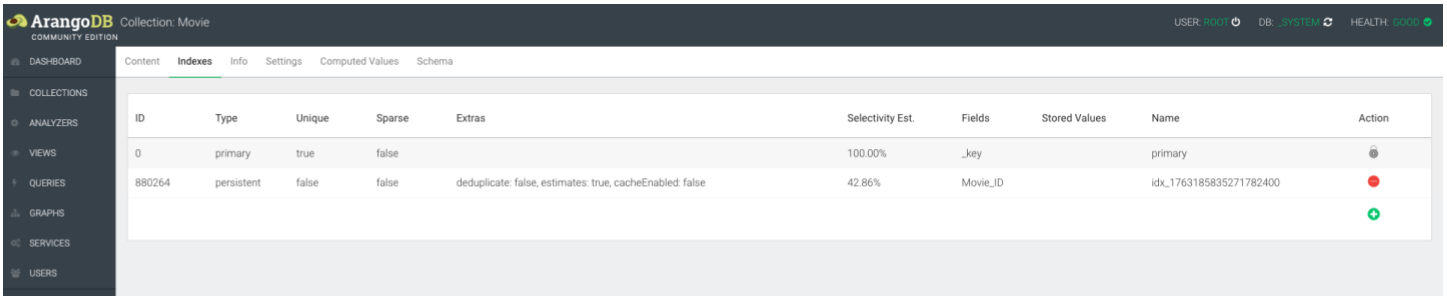
We now create a few **indexes** which will be used for query optimization.

Customer: Persistent index on Email



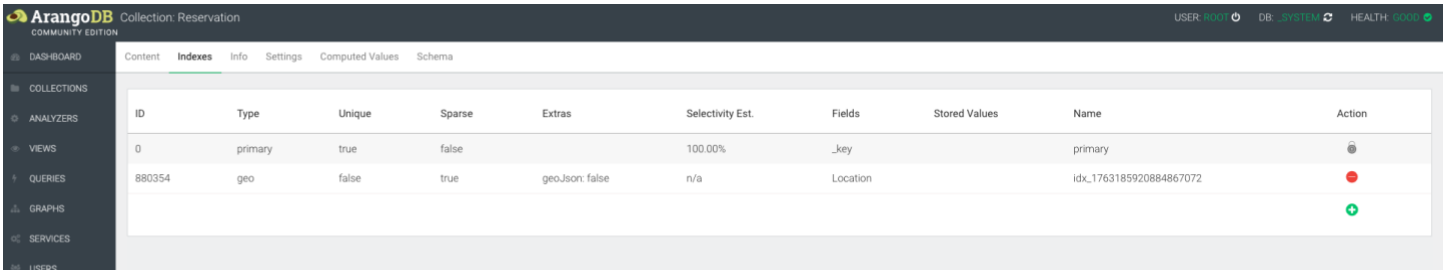
ID	Type	Unique	Sparse	Extras	Selectivity Est.	Fields	Stored Values	Name	Action
0	primary	true	false		100.00%	_key		primary	🔒
880182	persistent	true	false	deduplicate: false, estimates: true, cacheEnabled: false	100.00%	Email		idx_1763185721180422144	⊖
									⊕

Movie: Persistent Index on Movie_ID



ID	Type	Unique	Sparse	Extras	Selectivity Est.	Fields	Stored Values	Name	Action
0	primary	true	false		100.00%	_key		primary	🔒
880264	persistent	false	false	deduplicate: false, estimates: true, cacheEnabled: false	42.86%	Movie_ID		idx_1763185835271782400	⊖
									⊕

Reservation: Geoindex on Location



ID	Type	Unique	Sparse	Extras	Selectivity Est.	Fields	Stored Values	Name	Action
0	primary	true	false		100.00%	_key		primary	🔒
880354	geo	false	true	geoJson: false	n/a	Location		idx_1763185920884867072	⊖
									⊕

Now we converted our csv files with same data to Json files for future implementations (same data in now in json format)

DATA REFRESH

Python-based Data Refresh Implementation

- Utilizing Python to monitor and update data in the Arango DB web UI when changes are made to the node files.
- JSON Files Storage: Storing JSON files for all collections in the following directory: "/Users/tanujverma/Desktop/NEU/ADBMS/ArangoDB".
- File Monitoring: Actively monitoring JSON files in the specified directory for any modifications or updates.

Implementation Example

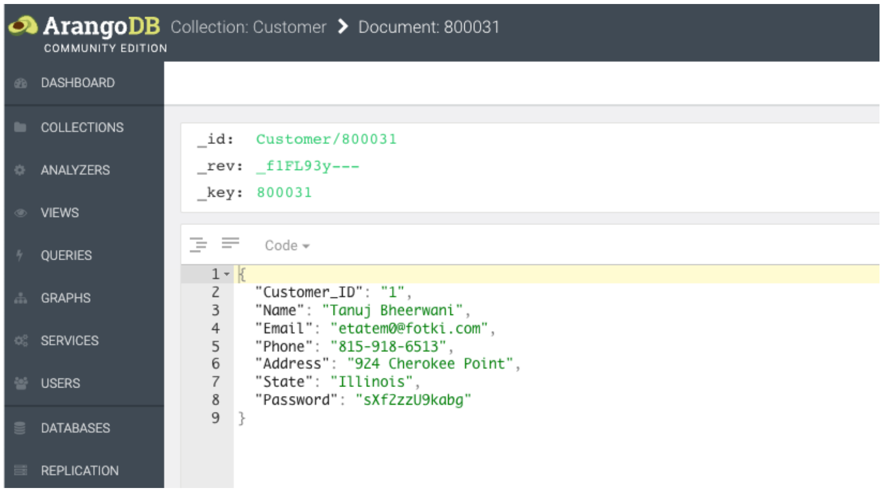
We have created below: **ONGOING DATA REFRESH**
(we can create a script of below code and run it via terminal(cronjob) for constant monitoring or at intervals)

```
import json
import time
from arango import ArangoClient
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler
class MyHandler(FileSystemEventHandler):
    def on_modified(self, event):
        if event.src_path.endswith('.json'):
            update_arango_db(event.src_path)
def update_arango_db(json_file_path):
    file_name = json_file_path.split('/')[-1].split('.')[0]

    if file_name in ['Movie', 'Theater', 'ShowTime', 'Reservation', 'Customer']:
        update_collection(json_file_path)
        print(f"Collection updated: {file_name}")
def update_collection(json_file_path):
    collection_name = json_file_path.split('/')[-1].split('.')[0]
    # Connect to the ArangoDB server
    client = ArangoClient(hosts='http://localhost:8529')
    db = client.db('_system', username='root', password='')
    # Clear the existing collection
    if db.has_collection(collection_name):
        collection = db.collection(collection_name)
        collection.truncate()
    else:
        # Create the collection if it doesn't exist
        db.create_collection(collection_name)
        collection = db.collection(collection_name)
    # Import data from the JSON file
    with open(json_file_path, 'r') as f:
        data = json.load(f)
        for document in data:
            collection.insert(document)
def main():
    path = "/Users/tanujverma/Desktop/NEU/ADBMS/ArangoDB"  # Set your path to the JSON files
    event_handler = MyHandler()
    observer = Observer()
    observer.schedule(event_handler, path, recursive=False)
    observer.start()
    try:
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        observer.stop()
    observer.join()
if __name__ == "__main__":
    main()
```

Now whenever we change any attributes in the code that collection will be reflected.

Example of Implementation

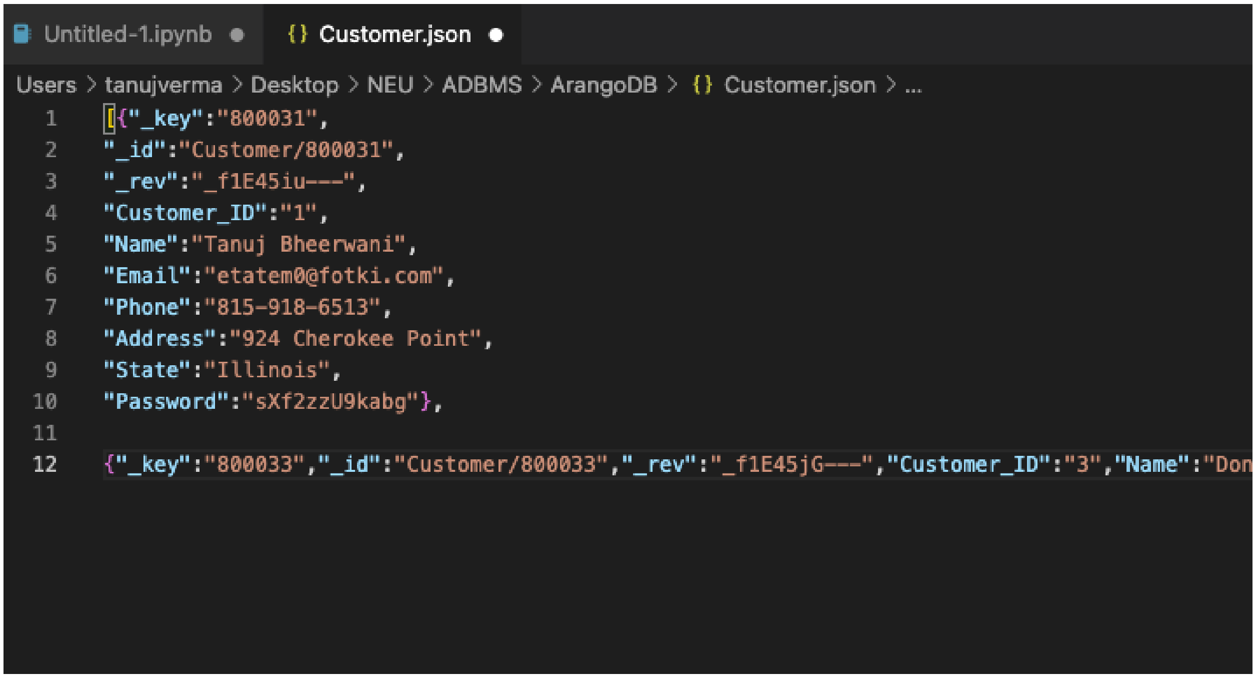
Below are the customer details for Customer_ID 1 :



```
_id:  Customer/800031
_rev: _f1FL93y---
_key: 800031
```

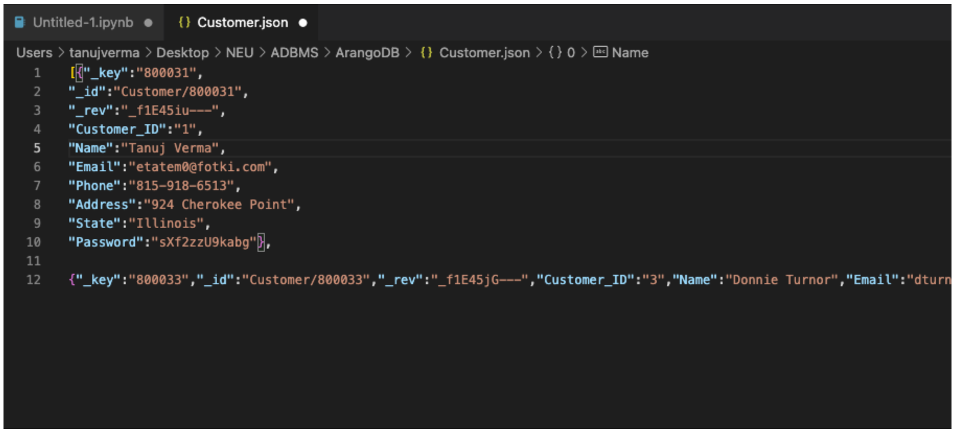
```
{
  "Customer_ID": "1",
  "Name": "Tanuj Bheerwani",
  "Email": "etatem0@fotki.com",
  "Phone": "815-918-6513",
  "Address": "924 Cherokee Point",
  "State": "Illinois",
  "Password": "sXf2zzU9kabg"
}
```

We are going to change the Customer Name. Now we make change into the JSON file of Customer. Originally in our json file, we have the name as “Tanuj Bheerwani”


Untitled-1.ipynb
Customer.json
Users > tanujverma > Desktop > NEU > ADBMS > ArangoDB > Customer.json > ...

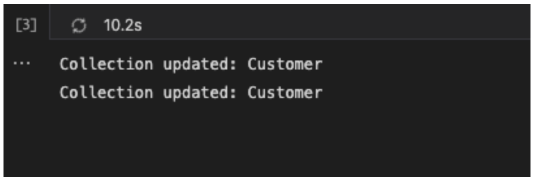
```
[{"_key":"800031",
"_id":"Customer/800031",
"_rev":"_f1E45iu---",
"Customer_ID":"1",
"Name":"Tanuj Bheerwani",
"Email":"etatem0@fotki.com",
"Phone":"815-918-6513",
"Address":"924 Cherokee Point",
"State":"Illinois",
"Password":"sXf2zzU9kabg"},

{"_key":"800033","_id":"Customer/800033","_rev":"_f1E45jG---","Customer_ID":"3","Name":"Don
```

We change it to “Tanuj Verma” and save it.


Untitled-1.ipynb
Customer.json
Users > tanujverma > Desktop > NEU > ADBMS > ArangoDB > Customer.json > 0 > Name

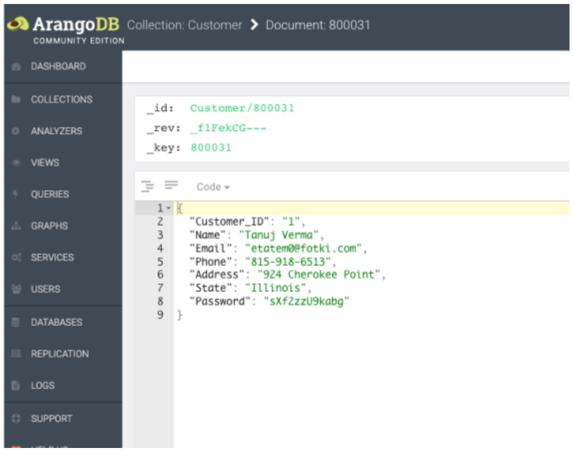
```
[{"_key":"800031",
"_id":"Customer/800031",
"_rev":"_f1E45iu---",
"Customer_ID":"1",
"Name":"Tanuj Verma",
"Email":"etatem0@fotki.com",
"Phone":"815-918-6513",
"Address":"924 Cherokee Point",
"State":"Illinois",
"Password":"sXf2zzU9kabg"},

{"_key":"800033","_id":"Customer/800033","_rev":"_f1E45jG---","Customer_ID":"3","Name":"Donnie Turnor","Email":"dturn
```

Our Python script immediately found which collection was updated and pushed that change into the web UI.



```
Collection updated: Customer
Collection updated: Customer
```

Now we check our web UI. We can see that Name is automatically updated. That’s how we implemented ongoing data refresh.



Collection: Customer > Document: 800031

```
_id:  Customer/800031
_rev: _f1FekCG---
_key: 800031
```

```
{
  "Customer_ID": "1",
  "Name": "Tanuj Verma",
  "Email": "etatem0@fotki.com",
  "Phone": "815-918-6513",
  "Address": "924 Cherokee Point",
  "State": "Illinois",
  "Password": "sXf2zzU9kabg"
}
```

CronJob automation steps we can use for automation.

- Opened a terminal and found the path to the Python interpreter on my system
- Then opened the crontab file for the current user by typing crontab -e in the terminal.
- Set set the script to run every hour by adding the following line to the crontab file:
  0 * * * * /usr/bin/python3 /Users/tanujverma/Desktop/data_refresh.py
- Saved the crontab file and exited the editor. The cron daemon is now set to execute the Python script at the specified intervals.

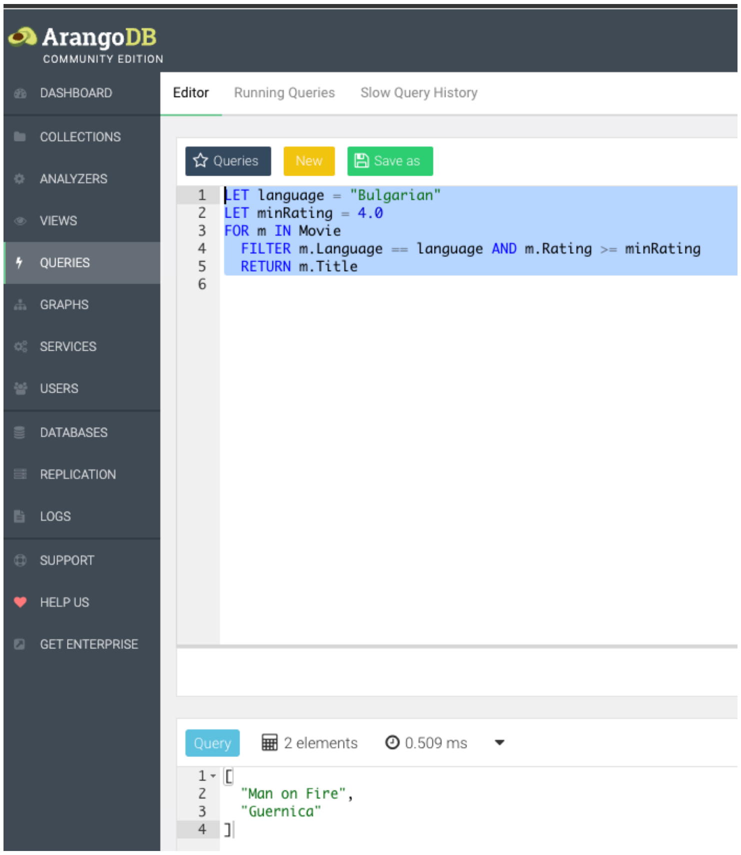
QUERIES WHICH WE EXECUTED ON OUR IMPLEMENTED DATABASE-REPORTS

1. Find all movies in a specific language: “Bulgarian” with a minimum rating:

Query:

```
LET language = "Bulgarian"
LET minRating = 4.0
FOR m IN Movie
FILTER m.Language == language AND m.Rating >= minRating RETURN m.Title
```

Output:



```
LET language = "Bulgarian"
LET minRating = 4.0
FOR m IN Movie
  FILTER m.Language == language AND m.Rating >= minRating
  RETURN m.Title
```

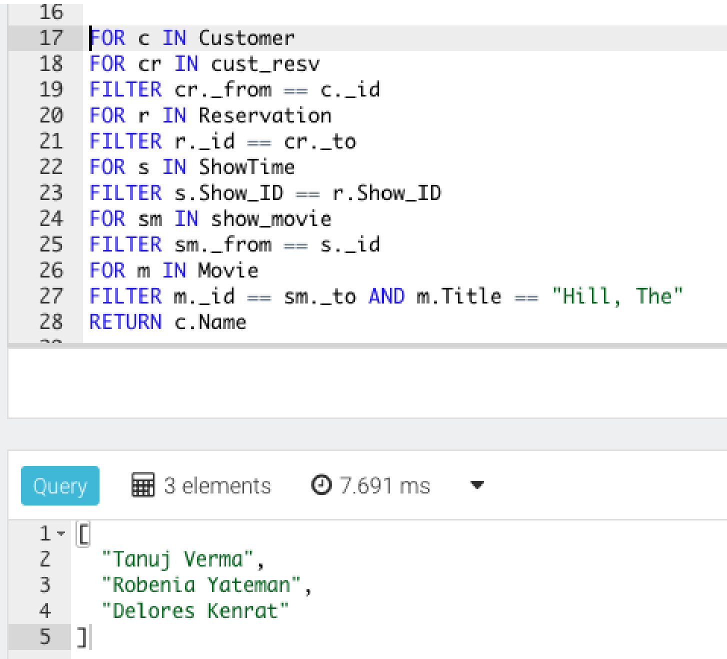
```
[
  "Man on Fire",
  "Guernica"
]
```

2. Find customers who have made reservations for a specific movie: “Hill, The”

Query:

```
FOR c IN Customer
FOR cr IN cust_resv
FILTER cr._from == c._id
FOR r IN Reservation
FILTER r._id == cr._to
FOR s IN ShowTime
FILTER s.Show_ID == r.Show_ID
FOR sm IN show_movie
FILTER sm._from == s._id
FOR m IN Movie
FILTER m._id == sm._to AND m.Title == "Hill, The"
RETURN c.Name
```

Output:



```
FOR c IN Customer
FOR cr IN cust_resv
FILTER cr._from == c._id
FOR r IN Reservation
FILTER r._id == cr._to
FOR s IN ShowTime
FILTER s.Show_ID == r.Show_ID
FOR sm IN show_movie
FILTER sm._from == s._id
FOR m IN Movie
FILTER m._id == sm._to AND m.Title == "Hill, The"
RETURN c.Name
```

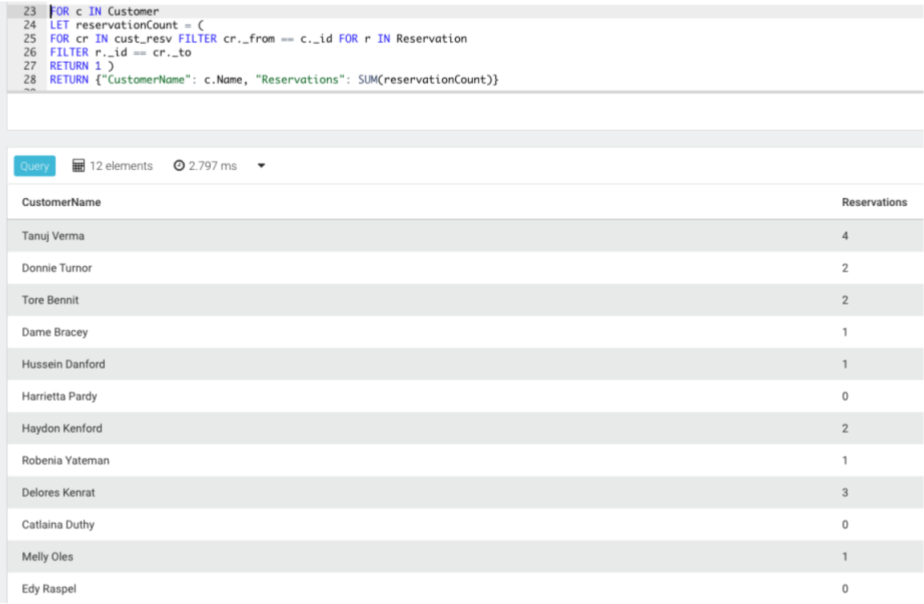
```
[
  "Tanuj Verma",
  "Robenia Yateman",
  "Delores Kenrat"
]
```

3.Show count of reservations made by each customer :

Query:

FOR c IN Customer
LET reservationCount = (
FOR cr IN cust_resv FILTER cr._from == c._id FOR r IN Reservation
FILTER r._id == cr._to
RETURN 1)
RETURN {"CustomerName": c.Name, "Reservations": SUM(reservationCount)}

Output:



```
FOR c IN Customer
LET reservationCount = (
FOR cr IN cust_resv FILTER cr._from == c._id FOR r IN Reservation
FILTER r._id == cr._to
RETURN 1 )
RETURN {"CustomerName": c.Name, "Reservations": SUM(reservationCount)}
```

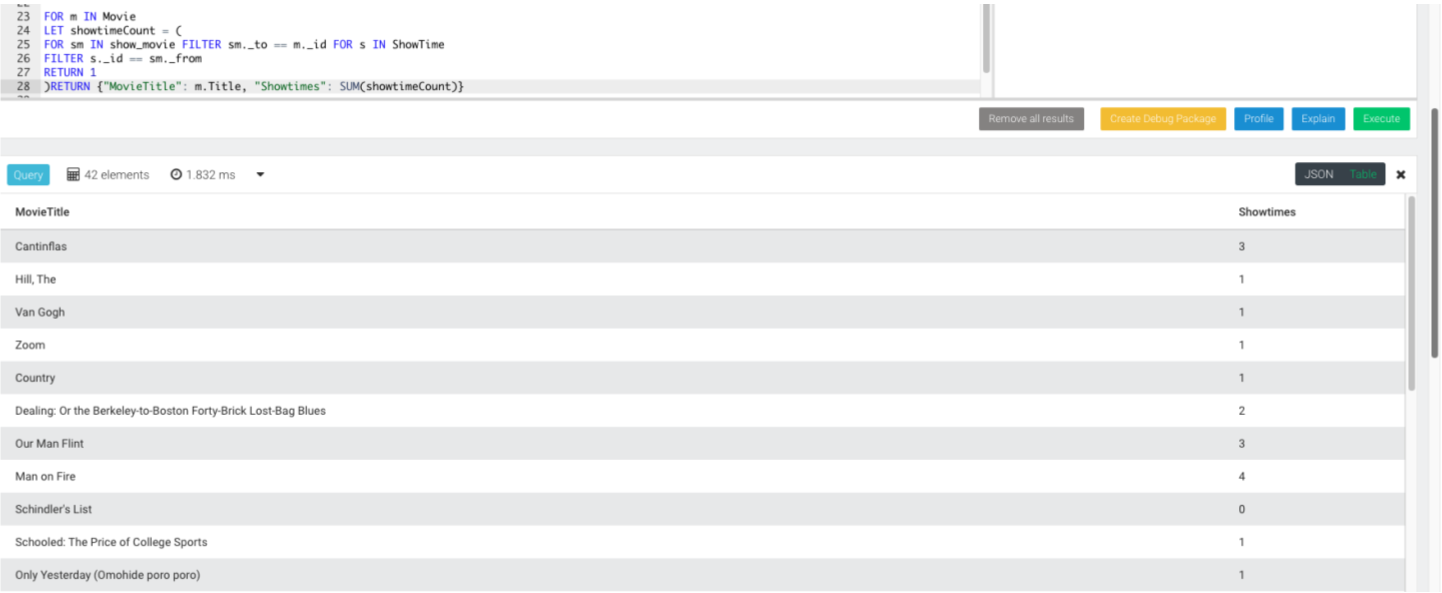
CustomerName	Reservations
Tanuj Verma	4
Donnie Turnor	2
Tore Bennit	2
Dame Bracey	1
Hussein Danford	1
Harrietta Pardy	0
Haydon Kenford	2
Robenia Yateman	1
Delores Kenrat	3
Catlaina Duthy	0
Melly Oles	1
Edy Raspel	0

4. Show count of showtimes for all the movies

Query:

```
FOR m IN Movie
LET showtimeCount = (
FOR sm IN show_movie FILTER sm._to == m._id FOR s IN ShowTime
FILTER s._id == sm._from
RETURN 1
)RETURN {"MovieTitle": m.Title, "Showtimes": SUM(showtimeCount)}
```

Output:



```
FOR m IN Movie
LET showtimeCount = (
FOR sm IN show_movie FILTER sm._to == m._id FOR s IN ShowTime
FILTER s._id == sm._from
RETURN 1
)RETURN {"MovieTitle": m.Title, "Showtimes": SUM(showtimeCount)}
```

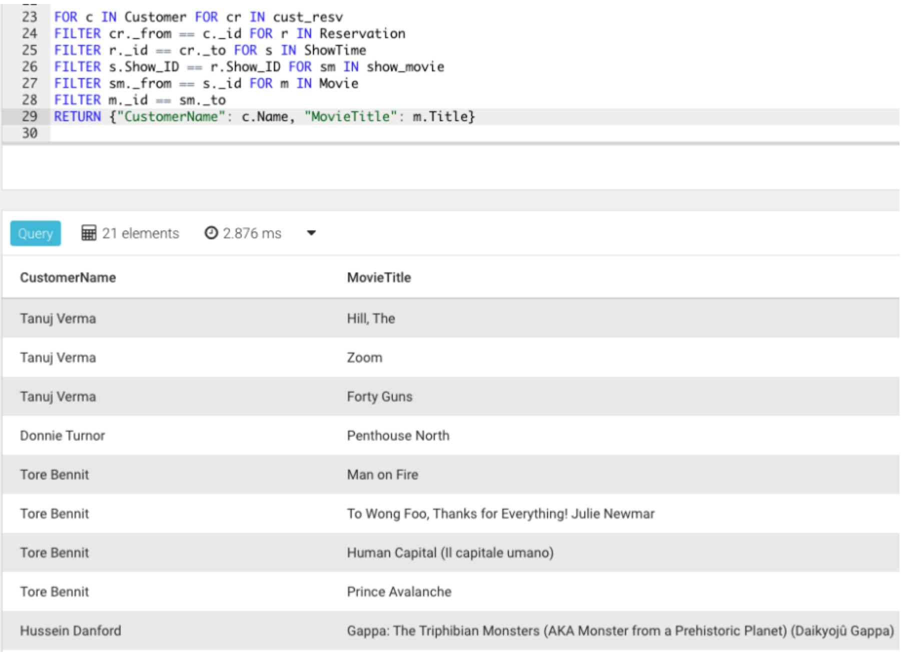
MovieTitle	Showtimes
Cantinflas	3
Hill, The	1
Van Gogh	1
Zoom	1
Country	1
Dealing: Or the Berkeley-to-Boston Forty-Brick Lost-Bag Blues	2
Our Man Flint	3
Man on Fire	4
Schindler's List	0
Schooled: The Price of College Sports	1
Only Yesterday (Omohide poro poro)	1

5. Show all the customers who have booked movies

Query :

```
FOR c IN Customer FOR cr IN cust_resv
FILTER cr._from == c._id FOR r IN Reservation
FILTER r._id == cr._to FOR s IN ShowTime
FILTER s.Show_ID == r.Show_ID FOR sm IN show_movie
FILTER sm._from == s._id FOR m IN Movie
FILTER m._id == sm._to
RETURN {"CustomerName": c.Name, "MovieTitle": m.Title}
```

Output:



```
FOR c IN Customer FOR cr IN cust_resv
FILTER cr._from == c._id FOR r IN Reservation
FILTER r._id == cr._to FOR s IN ShowTime
FILTER s.Show_ID == r.Show_ID FOR sm IN show_movie
FILTER sm._from == s._id FOR m IN Movie
FILTER m._id == sm._to
RETURN {"CustomerName": c.Name, "MovieTitle": m.Title}
```

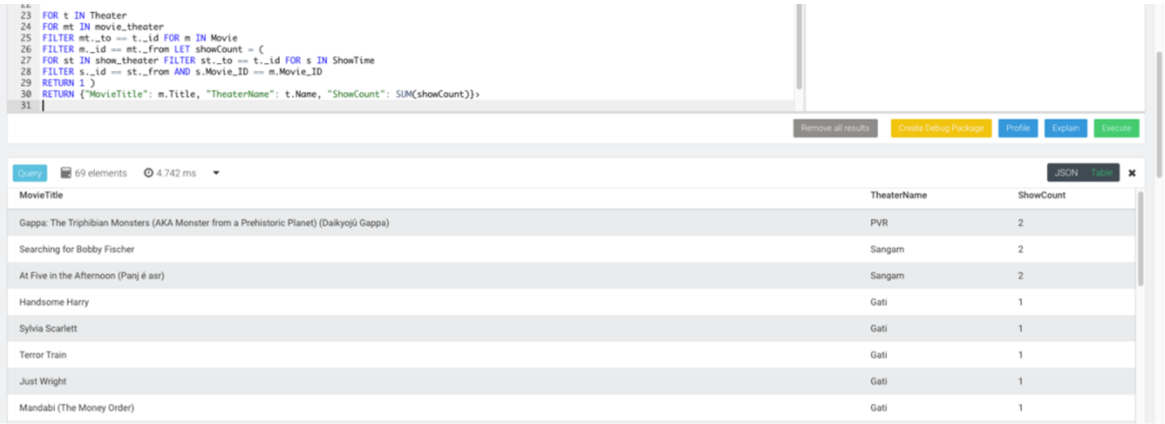
CustomerName	MovieTitle
Tanuj Verma	Hill, The
Tanuj Verma	Zoom
Tanuj Verma	Forty Guns
Donnie Turnor	Penthouse North
Tore Bennit	Man on Fire
Tore Bennit	To Wong Foo, Thanks for Everything! Julie Newmar
Tore Bennit	Human Capital (Il capitale umano)
Tore Bennit	Prince Avalanche
Hussein Danford	Gappa: The Triphibian Monsters (AKA Monster from a Prehistoric Planet) (Daikyojû Gappa)

6. Show all the movies running in a theater with the movie name, theater name, and the count of shows in that theater

Query:

FOR t IN Theater
FOR mt IN movie_theater
FILTER mt._to == t._id FOR m IN Movie
FILTER m._id == mt._from LET showCount = (
FOR st IN show_theater FILTER st._to == t._id FOR s IN ShowTime
FILTER s._id == st._from AND s.Movie_ID == m.Movie_ID
RETURN 1)
RETURN {"MovieTitle": m.Title, "TheaterName": t.Name, "ShowCount": SUM(showCount)}›

Output:



```
FOR t IN Theater
FOR mt IN movie_theater
FILTER mt._to == t._id FOR m IN Movie
FILTER m._id == mt._from LET showCount = (
FOR st IN show_theater FILTER st._to == t._id FOR s IN ShowTime
FILTER s._id == st._from AND s.Movie_ID == m.Movie_ID
RETURN 1 )
RETURN {"MovieTitle": m.Title, "TheaterName": t.Name, "ShowCount": SUM(showCount)}›
```

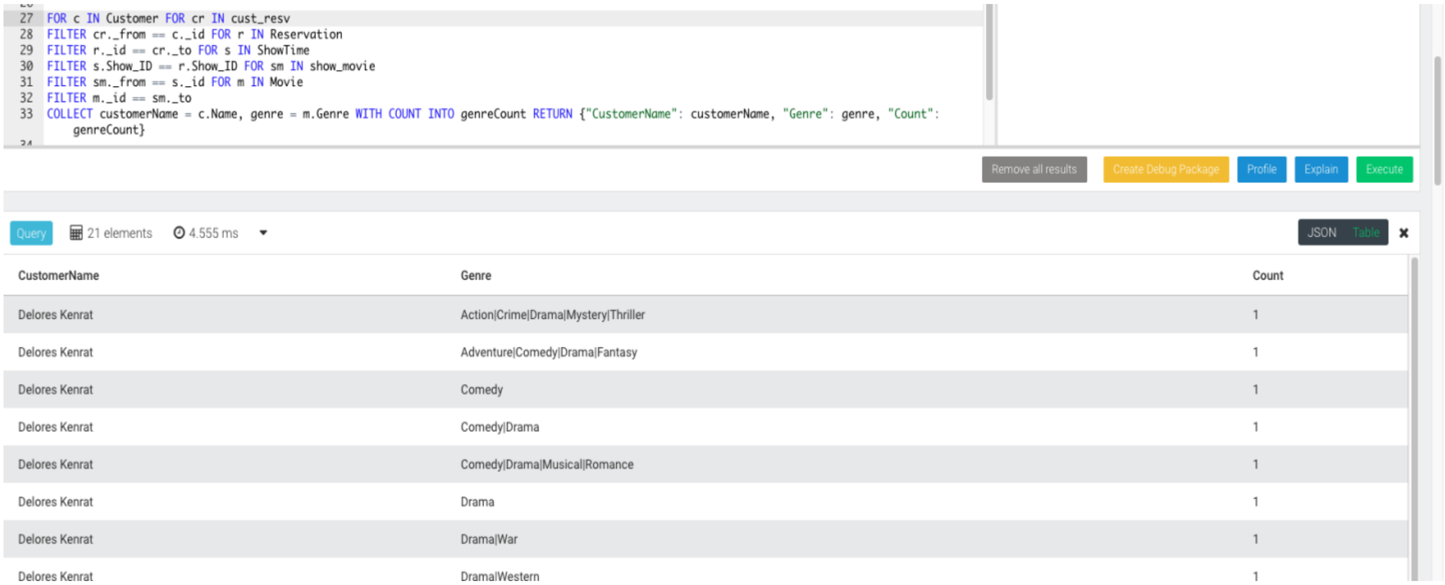
MovieTitle	TheaterName	ShowCount
Gappa: The Triphibian Monsters (AKA Monster from a Prehistoric Planet) (Daikyojû Gappa)	PVR	2
Searching for Bobby Fischer	Sangam	2
At Five in the Afternoon (Panj é asr)	Sangam	2
Handsome Harry	Gati	1
Sylvia Scarlett	Gati	1
Terror Train	Gati	1
Just Wright	Gati	1
Mandabi (The Money Order)	Gati	1

7. Show the customer’s name, watched genres, and count of times they watched those genres

Query:

FOR c IN Customer FOR cr IN cust_resv
FILTER cr._from == c._id FOR r IN Reservation
FILTER r._id == cr._to FOR s IN ShowTime
FILTER s.Show_ID == r.Show_ID FOR sm IN show_movie
FILTER sm._from == s._id FOR m IN Movie
FILTER m._id == sm._to
COLLECT customerName = c.Name, genre = m.Genre WITH COUNT INTO genreCount RETURN {"CustomerName": customerName, "Genre": genre, "Count": genreCount}

Output:



```
FOR c IN Customer FOR cr IN cust_resv
FILTER cr._from == c._id FOR r IN Reservation
FILTER r._id == cr._to FOR s IN ShowTime
FILTER s.Show_ID == r.Show_ID FOR sm IN show_movie
FILTER sm._from == s._id FOR m IN Movie
FILTER m._id == sm._to
COLLECT customerName = c.Name, genre = m.Genre WITH COUNT INTO genreCount RETURN {"CustomerName": customerName, "Genre": genre, "Count":
    genreCount}
```

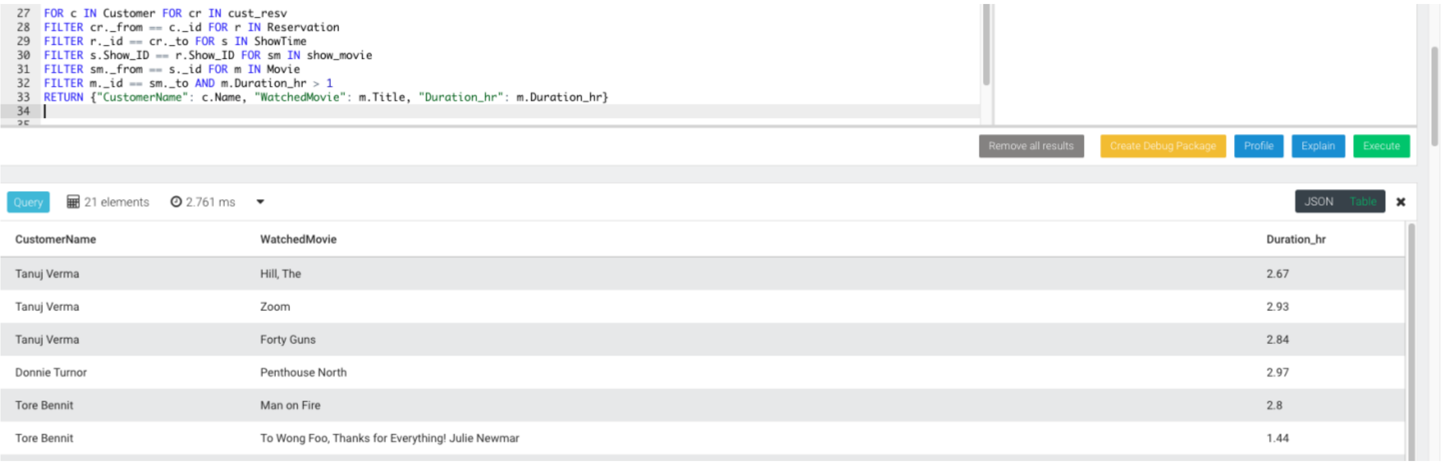
CustomerName	Genre	Count
Delores Kenrat	Action|Crime|Drama|Mystery|Thriller	1
Delores Kenrat	Adventure|Comedy|Drama|Fantasy	1
Delores Kenrat	Comedy	1
Delores Kenrat	Comedy|Drama	1
Delores Kenrat	Comedy|Drama|Musical|Romance	1
Delores Kenrat	Drama	1
Delores Kenrat	Drama|War	1
Delores Kenrat	Drama|Western	1

8. Show all the movies watched by the customer whose duration is more that 1 hour

Query:

FOR c IN Customer FOR cr IN cust_resv
FILTER cr._from == c._id FOR r IN Reservation
FILTER r._id == cr._to FOR s IN ShowTime
FILTER s.Show_ID == r.Show_ID FOR sm IN show_movie
FILTER sm._from == s._id FOR m IN Movie
FILTER m._id == sm._to AND m.Duration_hr > 1
RETURN {"CustomerName": c.Name, "WatchedMovie": m.Title, "Duration_hr": m.Duration_hr}

Output:



```
FOR c IN Customer FOR cr IN cust_resv
FILTER cr._from == c._id FOR r IN Reservation
FILTER r._id == cr._to FOR s IN ShowTime
FILTER s.Show_ID == r.Show_ID FOR sm IN show_movie
FILTER sm._from == s._id FOR m IN Movie
FILTER m._id == sm._to AND m.Duration_hr > 1
RETURN {"CustomerName": c.Name, "WatchedMovie": m.Title, "Duration_hr": m.Duration_hr}
```

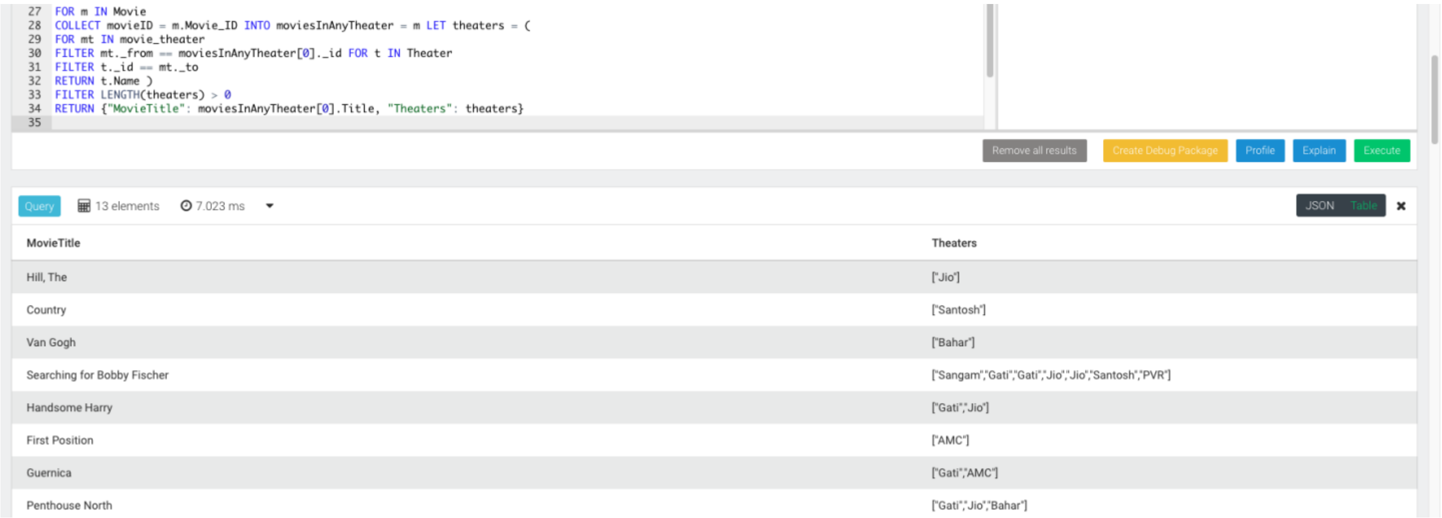
CustomerName	WatchedMovie	Duration_hr
Tanuj Verma	Hill, The	2.67
Tanuj Verma	Zoom	2.93
Tanuj Verma	Forty Guns	2.84
Donnie Turnor	Penthouse North	2.97
Tore Bennit	Man on Fire	2.8
Tore Bennit	To Wong Foo, Thanks for Everything! Julie Newmar	1.44

9.Show movies running in all theaters

Query:

FOR m IN Movie
COLLECT movieID = m.Movie_ID INTO moviesInAnyTheater = m LET theaters = (
FOR mt IN movie_theater
FILTER mt._from == moviesInAnyTheater[0]._id FOR t IN Theater
FILTER t._id == mt._to
RETURN t.Name)
FILTER LENGTH(theaters) > 0
RETURN {"MovieTitle": moviesInAnyTheater[0].Title, "Theaters": theaters}

Output:



```
FOR m IN Movie
COLLECT movieID = m.Movie_ID INTO moviesInAnyTheater = m LET theaters = (
FOR mt IN movie_theater
FILTER mt._from == moviesInAnyTheater[0]._id FOR t IN Theater
FILTER t._id == mt._to
RETURN t.Name )
FILTER LENGTH(theaters) > 0
RETURN {"MovieTitle": moviesInAnyTheater[0].Title, "Theaters": theaters}
```

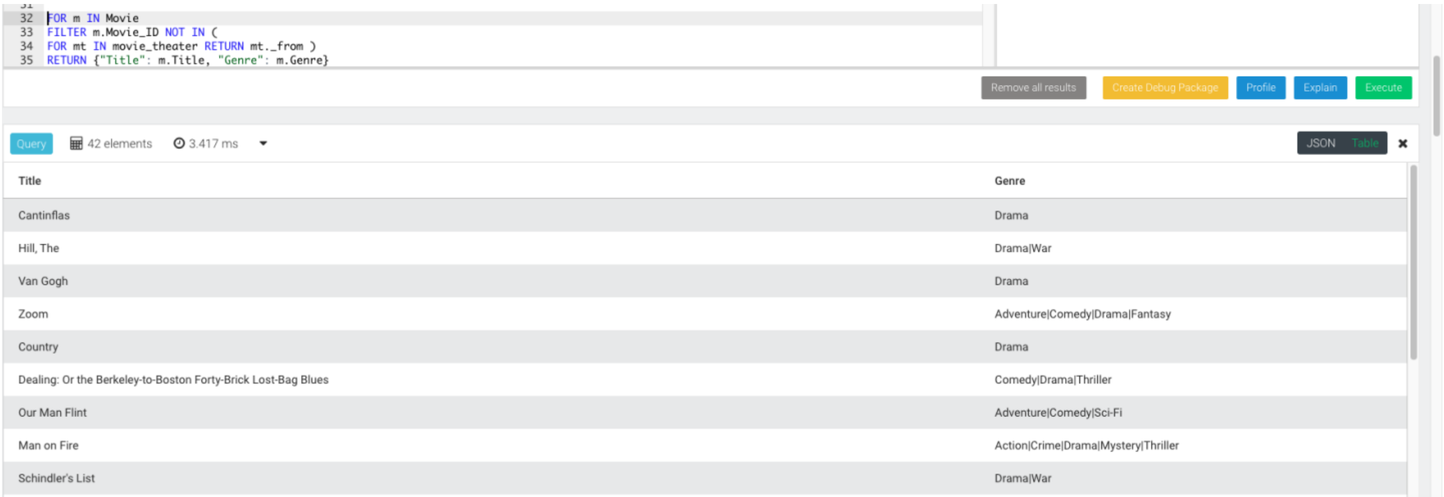
MovieTitle	Theaters
Hill, The	["Jio"]
Country	["Santosh"]
Van Gogh	["Bahar"]
Searching for Bobby Fischer	["Sangam","Gati","Gati","Jio","Jio","Santosh","PVR"]
Handsome Harry	["Gati","Jio"]
First Position	["AMC"]
Guernica	["Gati","AMC"]
Penthouse North	["Gati","Jio","Bahar"]

10. Show movies not running in any theater

Query:

FOR m IN Movie
FILTER m.Movie_ID NOT IN (
FOR mt IN movie_theater RETURN mt._from)
RETURN {"Title": m.Title, "Genre": m.Genre}

Output:



```
FOR m IN Movie
FILTER m.Movie_ID NOT IN (
FOR mt IN movie_theater RETURN mt._from )
RETURN {"Title": m.Title, "Genre": m.Genre}
```

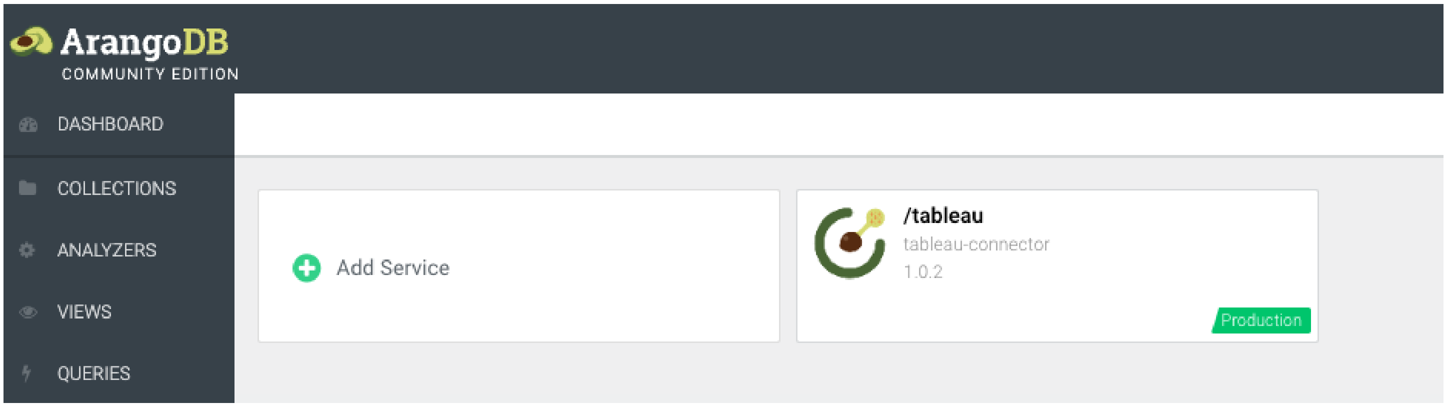
Title	Genre
Cantinflas	Drama
Hill, The	Drama|War
Van Gogh	Drama
Zoom	Adventure|Comedy|Drama|Fantasy
Country	Drama
Dealing: Or the Berkeley-to-Boston Forty-Brick Lost-Bag Blues	Comedy|Drama|Thriller
Our Man Flint	Adventure|Comedy|Sci-Fi
Man on Fire	Action|Crime|Drama|Mystery|Thriller
Schindler's List	Drama|War

VISUALIZATIONS

We have used tableau for visualization. In total we have created 6 visualizations and 2 dashboards.

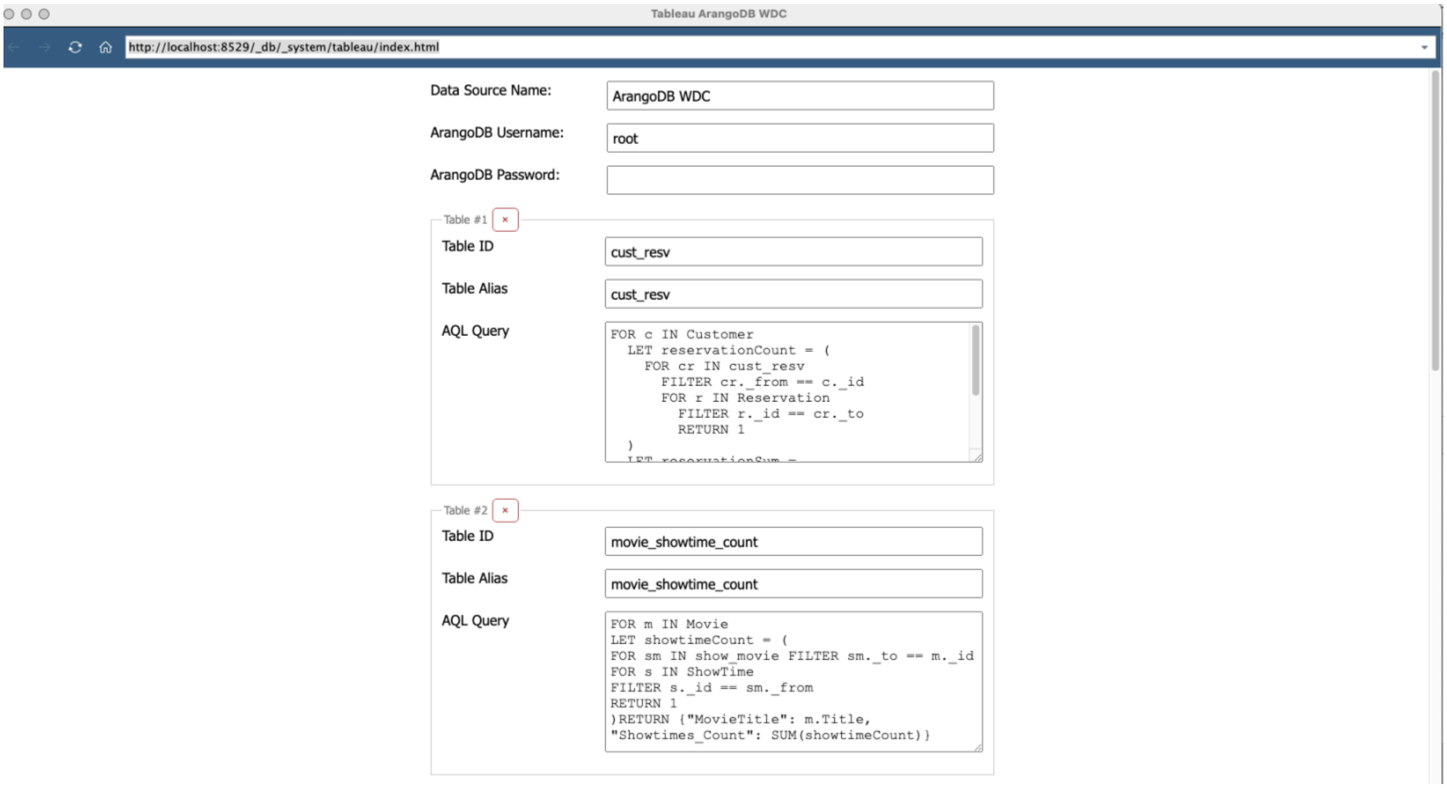
Steps to Connect to tableau:

- Install tableau services from web UI



- After installation is completed, we open Tableau and go to Connections from web Server and put in the URL : http://localhost:8529/_db/_system/tableau/index.html.This will make sure that we are connecting directly our Arango DB with Tableau and can run queries from tableau directly.

- After that we will put all our queries in TABLES show below. In total we have used 6 complex queries for visualizations



Data Source Name: ArangoDB WDC

ArangoDB Username: root

ArangoDB Password:

Table #1

Table ID: cust_resv

Table Alias: cust_resv

AQL Query:

```
FOR c IN Customer
  LET reservationCount = (
    FOR cr IN cust_resv
      FILTER cr._from == c._id
      FOR r IN Reservation
        FILTER r._id == cr._to
        RETURN 1
  )
  LET reservationSum =
```

Table #2

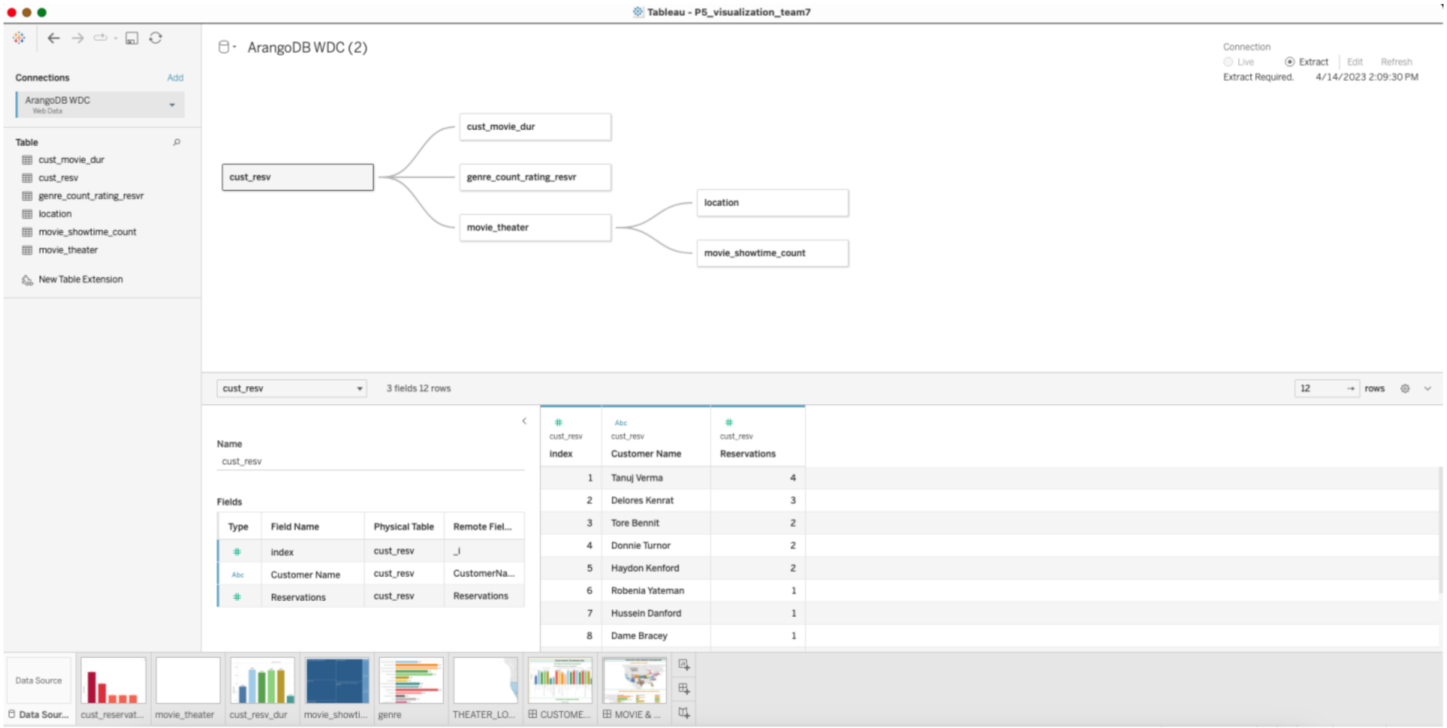
Table ID: movie_showtime_count

Table Alias: movie_showtime_count

AQL Query:

```
FOR m IN Movie
LET showtimeCount = (
FOR sm IN show_movie FILTER sm._to == m._id
FOR s IN ShowTime
FILTER s._id == sm._from
RETURN 1
)RETURN {"MovieTitle": m.Title,
"Showtimes_Count": SUM(showtimeCount)}
```

After writing queries we click Extract. We then connect all tables logically and our data is ready for visualization.



Connection ○ Live ● Extract | Edit Refresh

Extract Required. 4/14/2023 2:09:30 PM

cust_resv 3 fields 12 rows

12 → rows

Name: cust_resv

Fields

Type	Field Name	Physical Table	Remote Fiel...
#	index	cust_resv	_i
Abc	Customer Name	cust_resv	CustomerNa...
#	Reservations	cust_resv	Reservations

# cust_resv index	Abc cust_resv Customer Name	# cust_resv Reservations
1	Tanuj Verma	4
2	Delores Kenrat	3
3	Tore Bennit	2
4	Donnie Turnor	2
5	Haydon Kenford	2
6	Robenia Yateman	1
7	Hussein Danford	1
8	Dame Bracey	1

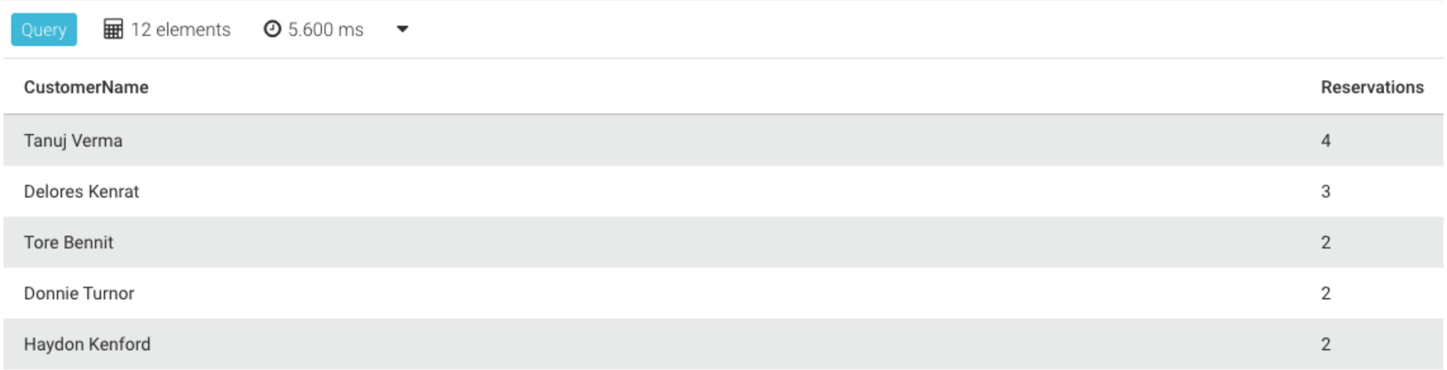
QUERIES FOR VISUALIZATION

1. CUSTOMERS WITH RESERVATION COUNTS

Query:

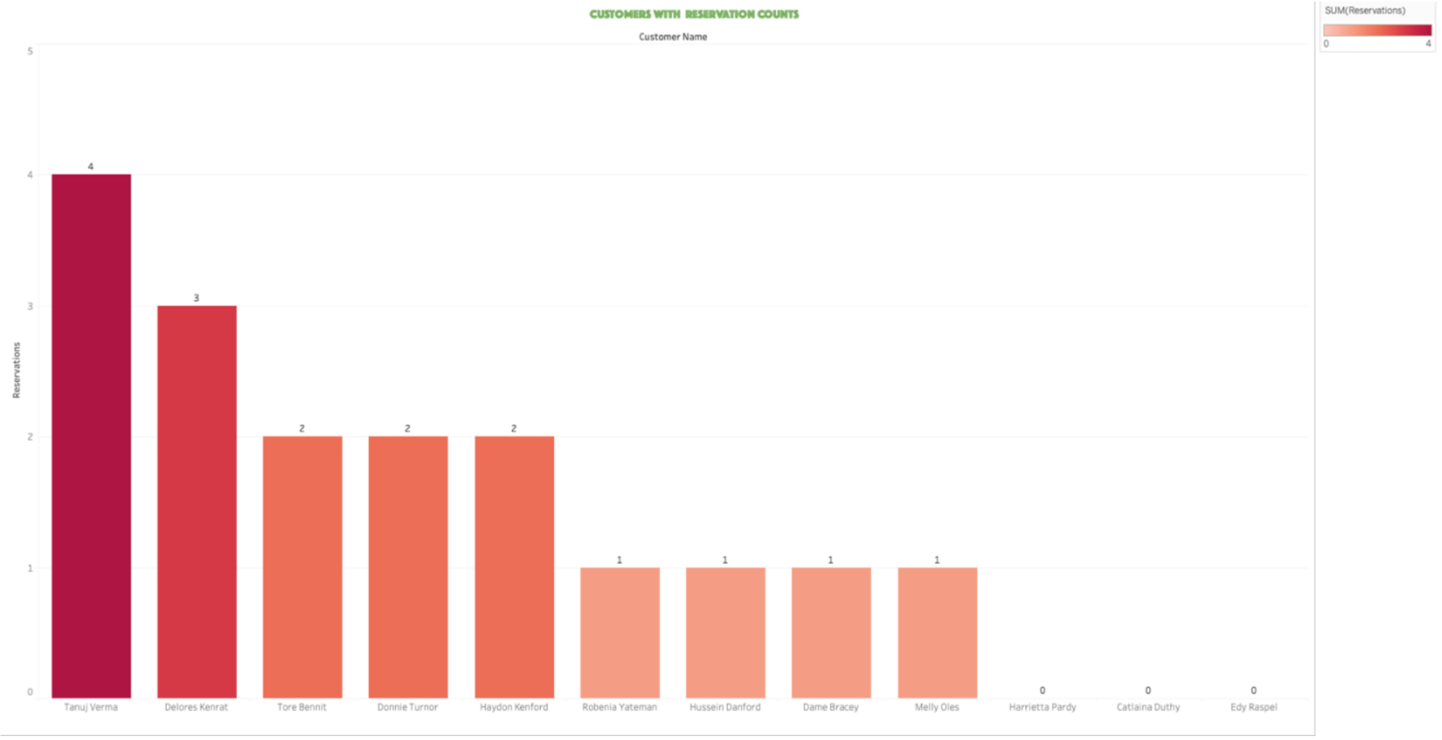
```
FOR c IN Customer
  LET reservationCount = (
    FOR cr IN cust_resv
      FILTER cr._from == c._id
      FOR r IN Reservation
        FILTER r._id == cr._to
        RETURN 1)
  LET reservationSum = SUM(reservationCount)
  SORT reservationSum desc
  RETURN {"CustomerName": c.Name, "Reservations": reservationSum}
```

Output in Arango DB:



CustomerName	Reservations
Tanuj Verma	4
Delores Kenrat	3
Tore Bennit	2
Donnie Turnor	2
Haydon Kenford	2

Visualization (bar plot):

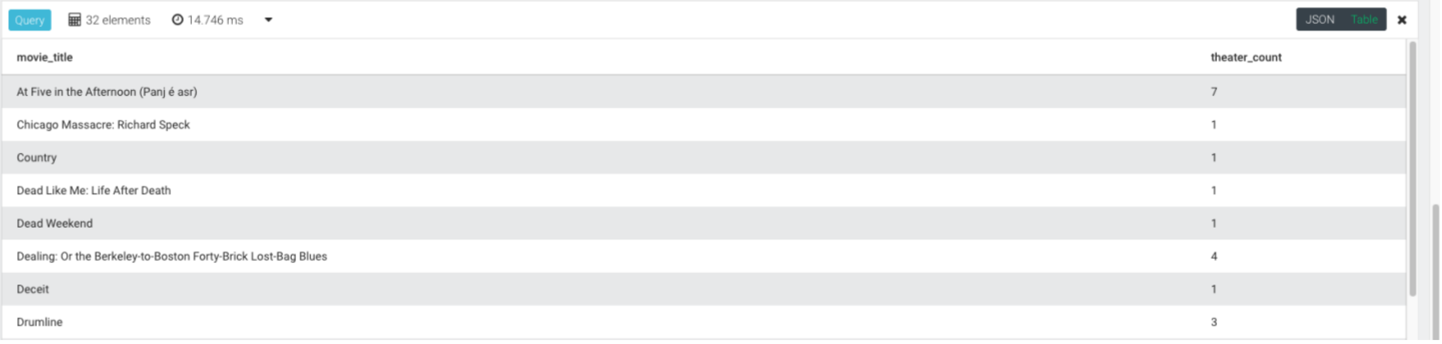


2. NUMBER OF THEATERS SCREENING EACH MOVIE

Query:

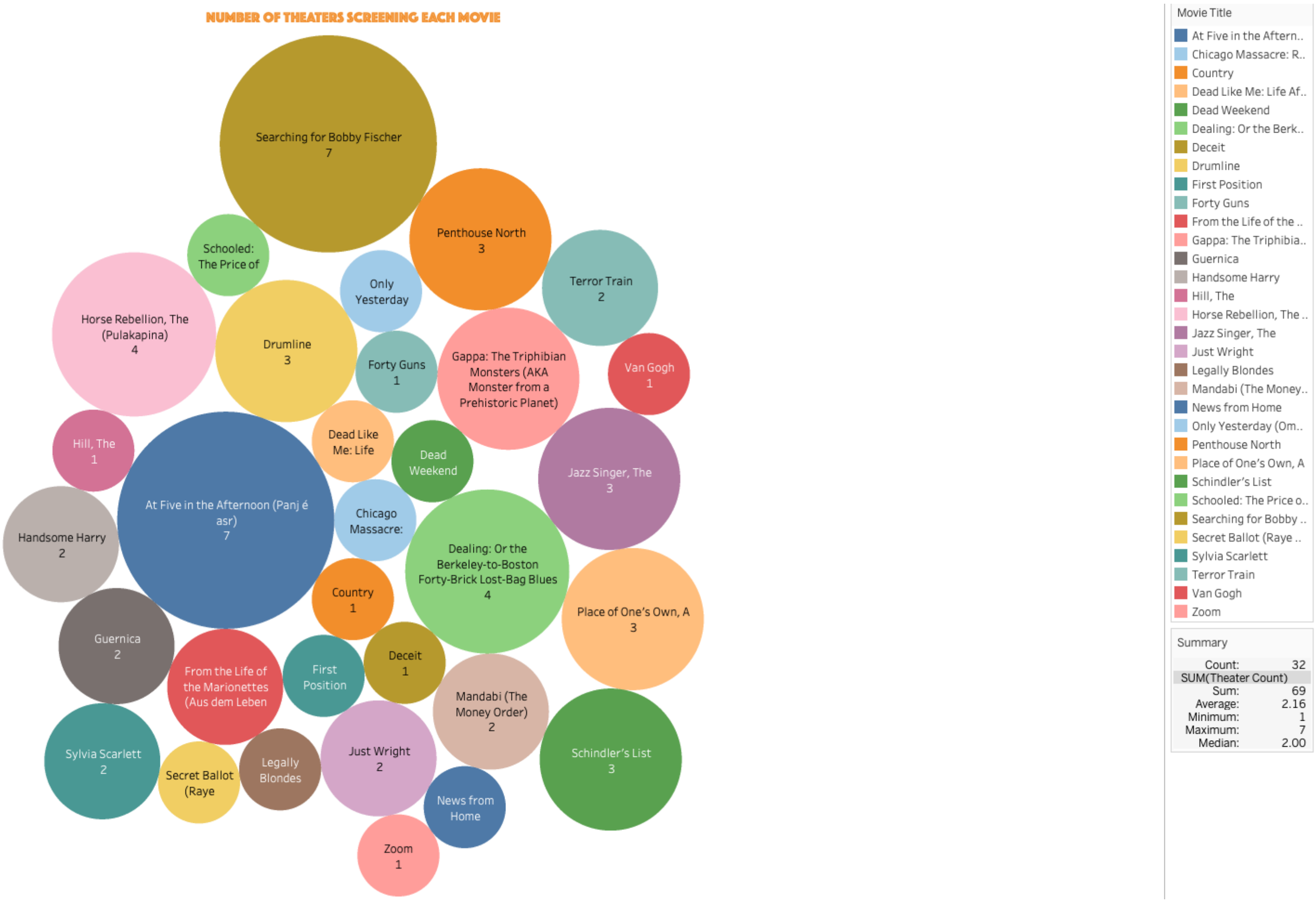
```
FOR m IN Movie
  FOR t IN Theater
    FILTER m.Movie_ID == t.Movie_ID
    COLLECT movie_title = m.Title INTO movie_group
    RETURN { movie_title: movie_title, theater_count: LENGTH(movie_group) }
```

Output in Arango DB:



movie_title	theater_count
At Five in the Afternoon (Panj é asr)	7
Chicago Massacre: Richard Speck	1
Country	1
Dead Like Me: Life After Death	1
Dead Weekend	1
Dealing: Or the Berkeley-to-Boston Forty-Brick Lost-Bag Blues	4
Deceit	1
Drumline	3

Visualization (packed bubbles):

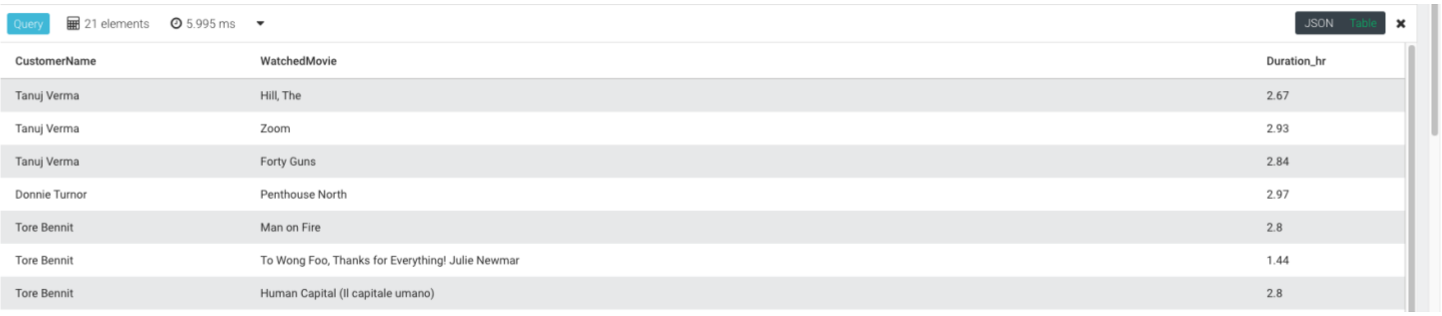


3.CUSTOMERS' MOVIE WATCHLIST AND DURATION

Query:

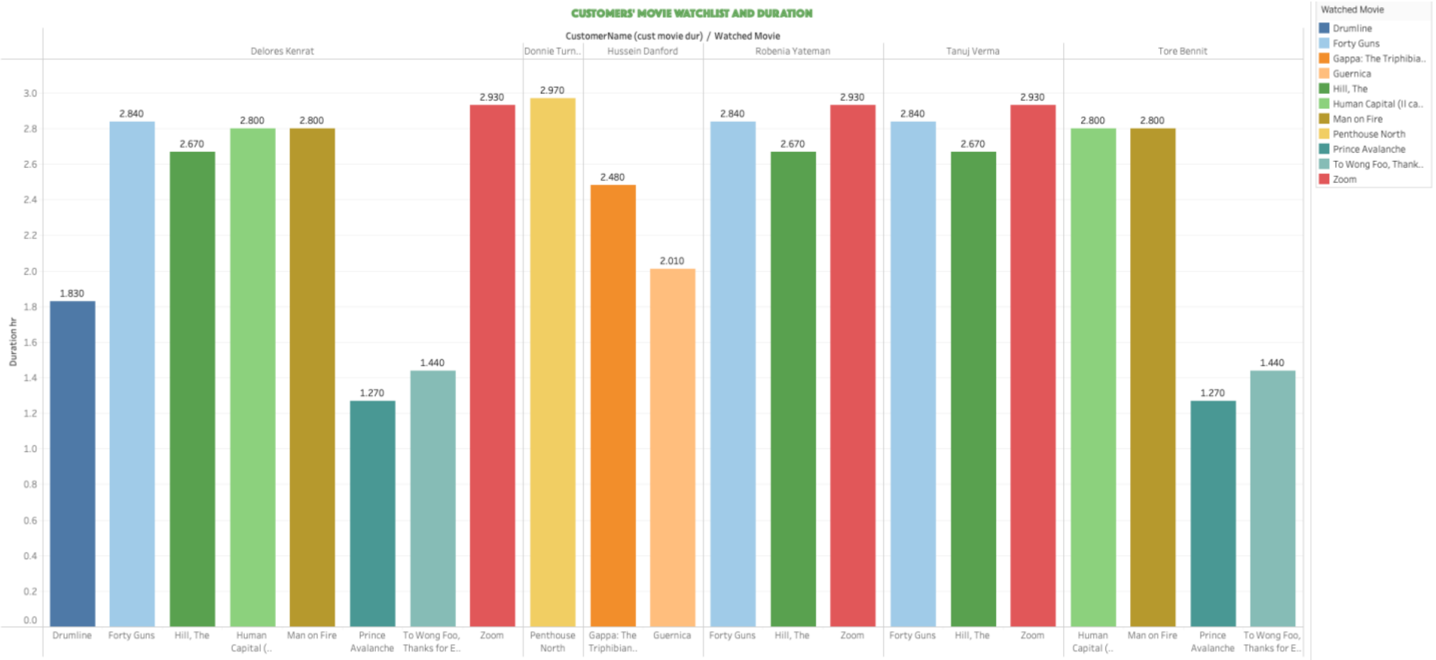
```
FOR c IN Customer FOR cr IN cust_resv
FILTER cr._from == c._id FOR r IN Reservation
FILTER r._id == cr._to FOR s IN ShowTime
FILTER s.Show_ID == r.Show_ID FOR sm IN show_movie
FILTER sm._from == s._id FOR m IN Movie
FILTER m._id == sm._to AND m.Duration_hr > 1
RETURN {"CustomerName": c.Name, "WatchedMovie": m.Title, "Duration_hr": m.Duration_hr}
```

Output in Arango DB:



CustomerName	WatchedMovie	Duration_hr
Tanuj Verma	Hill, The	2.67
Tanuj Verma	Zoom	2.93
Tanuj Verma	Forty Guns	2.84
Donnie Turnor	Penthouse North	2.97
Tore Bennit	Man on Fire	2.8
Tore Bennit	To Wong Foo, Thanks for Everything! Julie Newmar	1.44
Tore Bennit	Human Capital (Il capitale umano)	2.8

Visualization (Side-by-Side Bar plot with 3 entities):

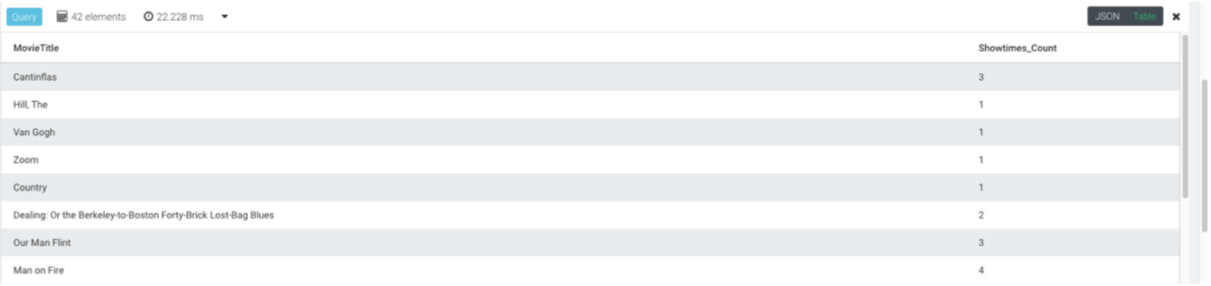


4. MOVIES WITH COUNT OF SHOWTIMES

Query :

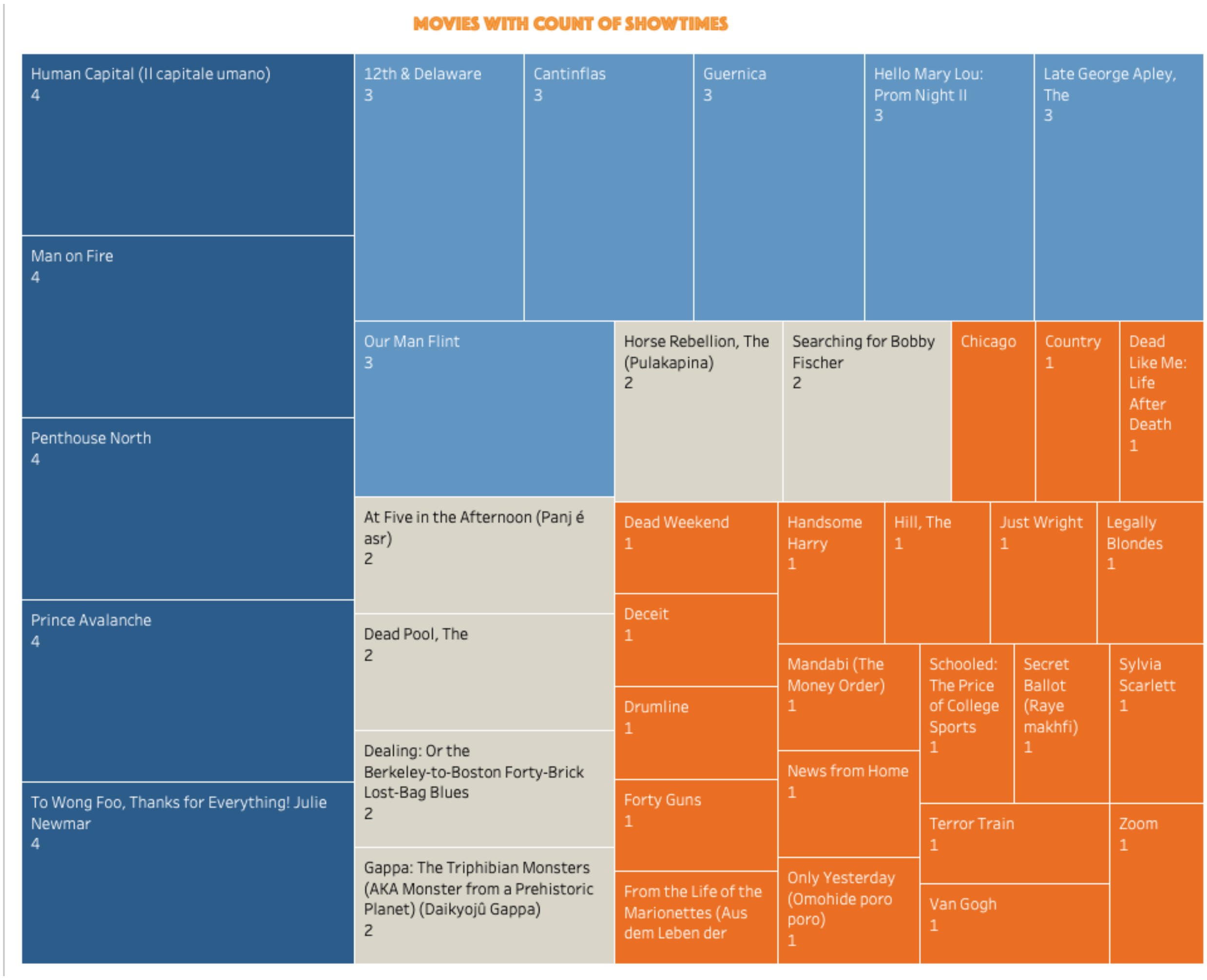
```
FOR m IN Movie
LET showtimeCount = (
FOR sm IN show_movie FILTER sm._to == m._id FOR s IN ShowTime
FILTER s._id == sm._from
RETURN 1
)RETURN {"MovieTitle": m.Title, "Showtimes_Count": SUM(showtimeCount)}
```

Output in Arango DB:



MovieTitle	Showtimes_Count
Cantinflas	3
Hill, The	1
Van Gogh	1
Zoom	1
Country	1
Dealing: Or the Berkeley-to-Boston Forty-Brick Lost-Bag Blues	2
Our Man Flint	3
Man on Fire	4

Visualization (Square Plot):

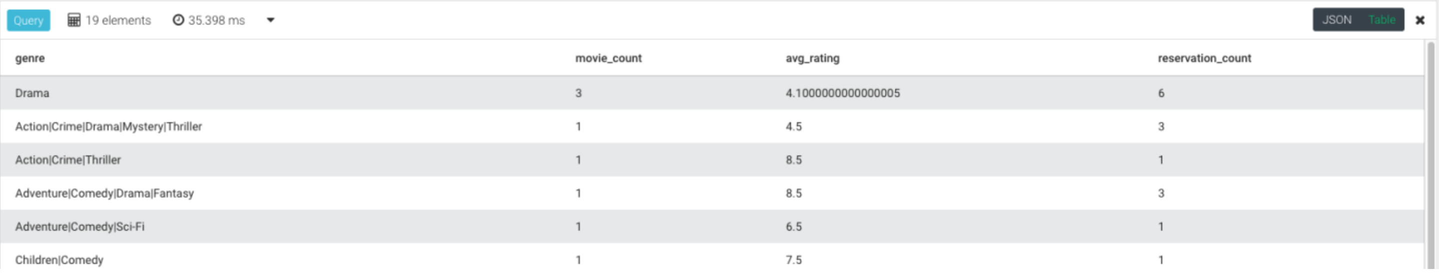


5. GENRE SUMMARY: RESERVATIONS, MOVIE COUNT, AND AVERAGE RATING

Query:

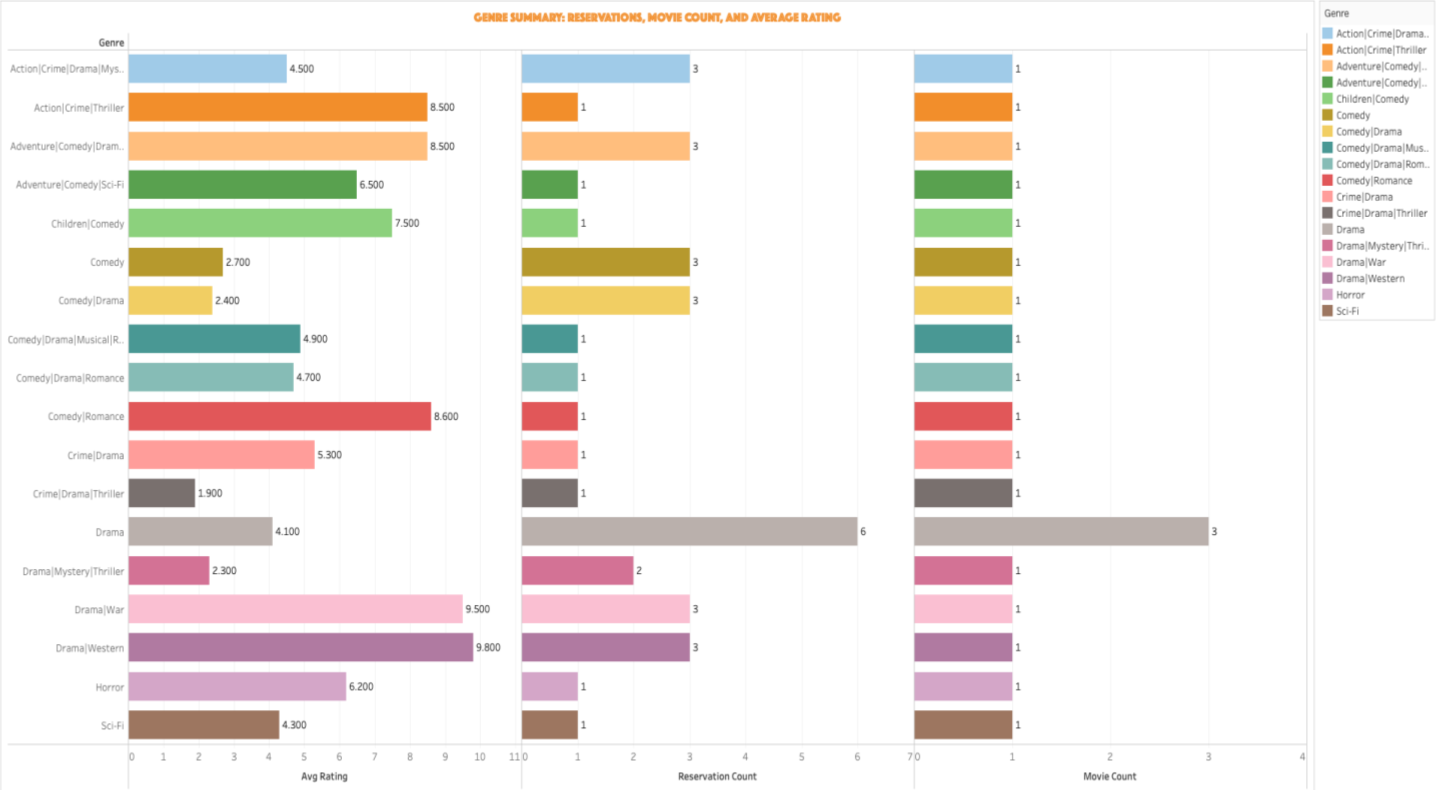
```
LET genre_summary = (
  FOR m IN Movie
    FOR r IN Reservation
      FOR s IN ShowTime
        FILTER r.Show_ID == s.Show_ID
        FILTER m.Movie_ID == s.Movie_ID
        COLLECT genre = m.Genre INTO genre_group
        RETURN {genre: genre,movie_count: LENGTH(UNIQUE(genre_group[*].m.Movie_ID)),avg_rating:
AVG(genre_group[*].m.Rating),
          reservation_count: LENGTH(genre_group)})
FOR summary IN genre_summary
  SORT summary.movie_count DESC
  RETURN summary
```

Output in Arango DB:



genre	movie_count	avg_rating	reservation_count
Drama	3	4.1000000000000005	6
Action|Crime|Drama|Mystery|Thriller	1	4.5	3
Action|Crime|Thriller	1	8.5	1
Adventure|Comedy|Drama|Fantasy	1	8.5	3
Adventure|Comedy|Sci-Fi	1	6.5	1
Children|Comedy	1	7.5	1

Visualization (horizontal Side-by-Side bar plot):

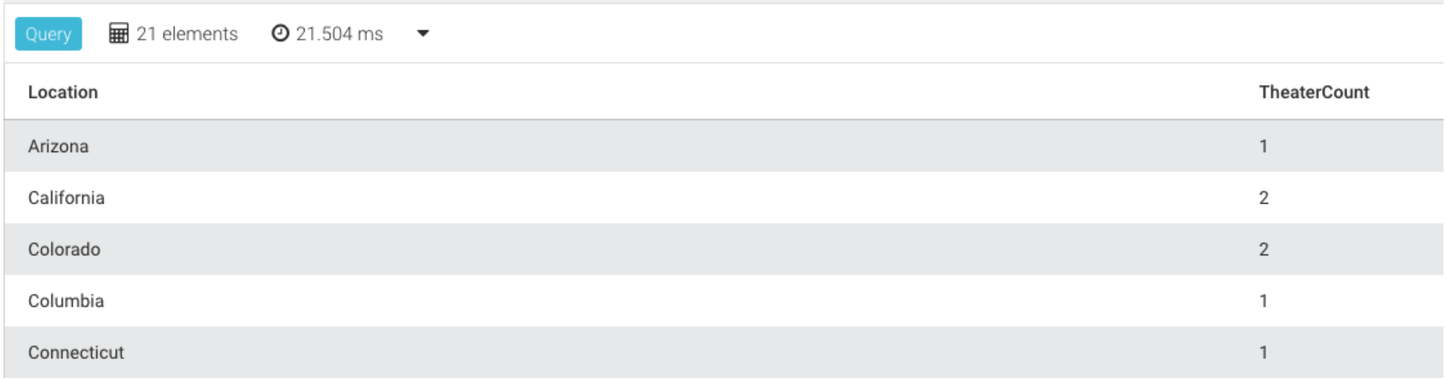


6. THEATER COUNT BY LOCATION

Query:

```
FOR t IN Theater
  COLLECT location = t.Location INTO theatersGroup
  LET theaterCount = LENGTH(theatersGroup)
  RETURN {"Location": location, "TheaterCount": theaterCount}
```

Output in Arango DB:

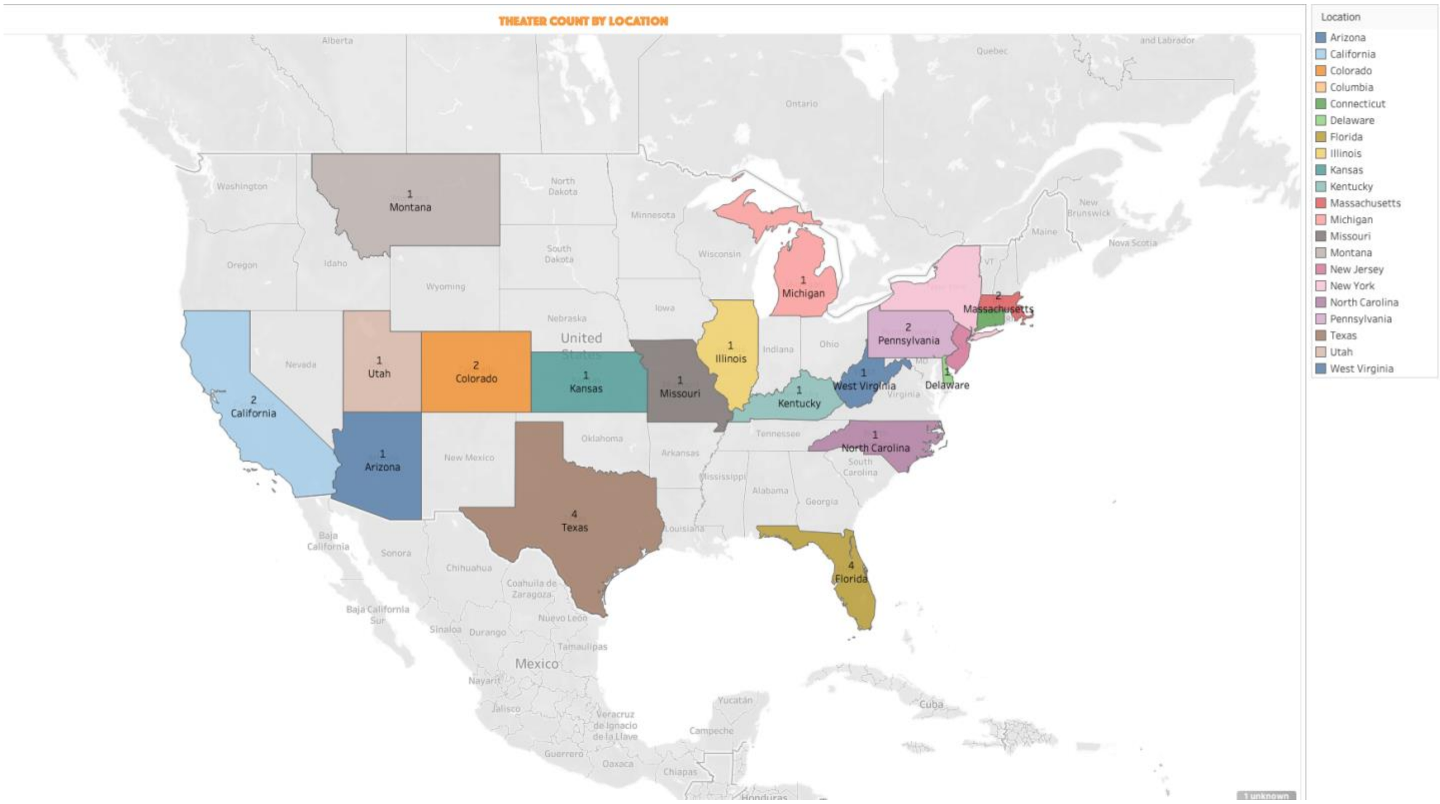


21 elements

21.504 ms

Location	TheaterCount
Arizona	1
California	2
Colorado	2
Columbia	1
Connecticut	1

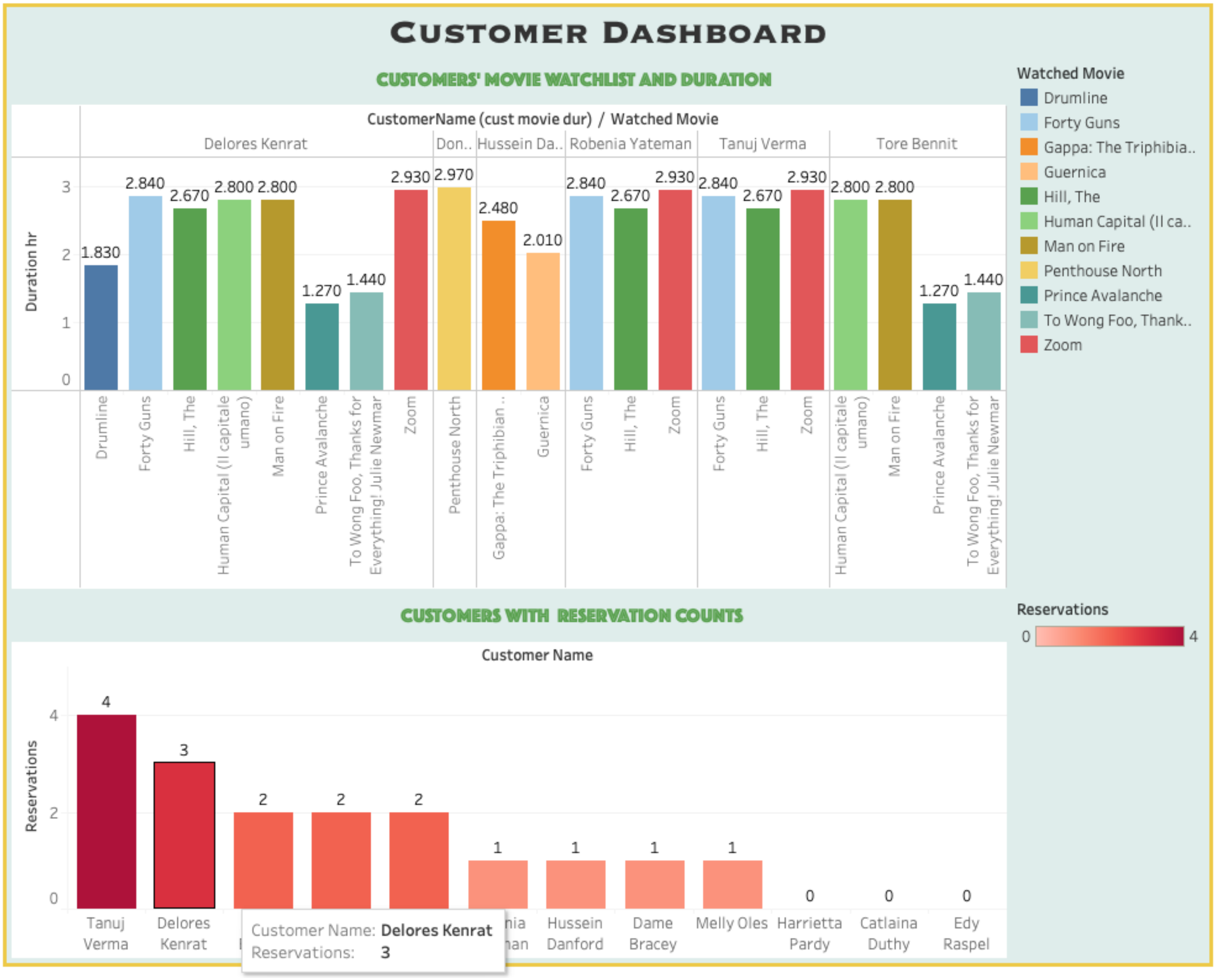
Visualization (Maps):



DASHBOARDS

1.CUSTOMER DASHBOARD

This dashboard contains Customer Analysis for our Arango DB data.

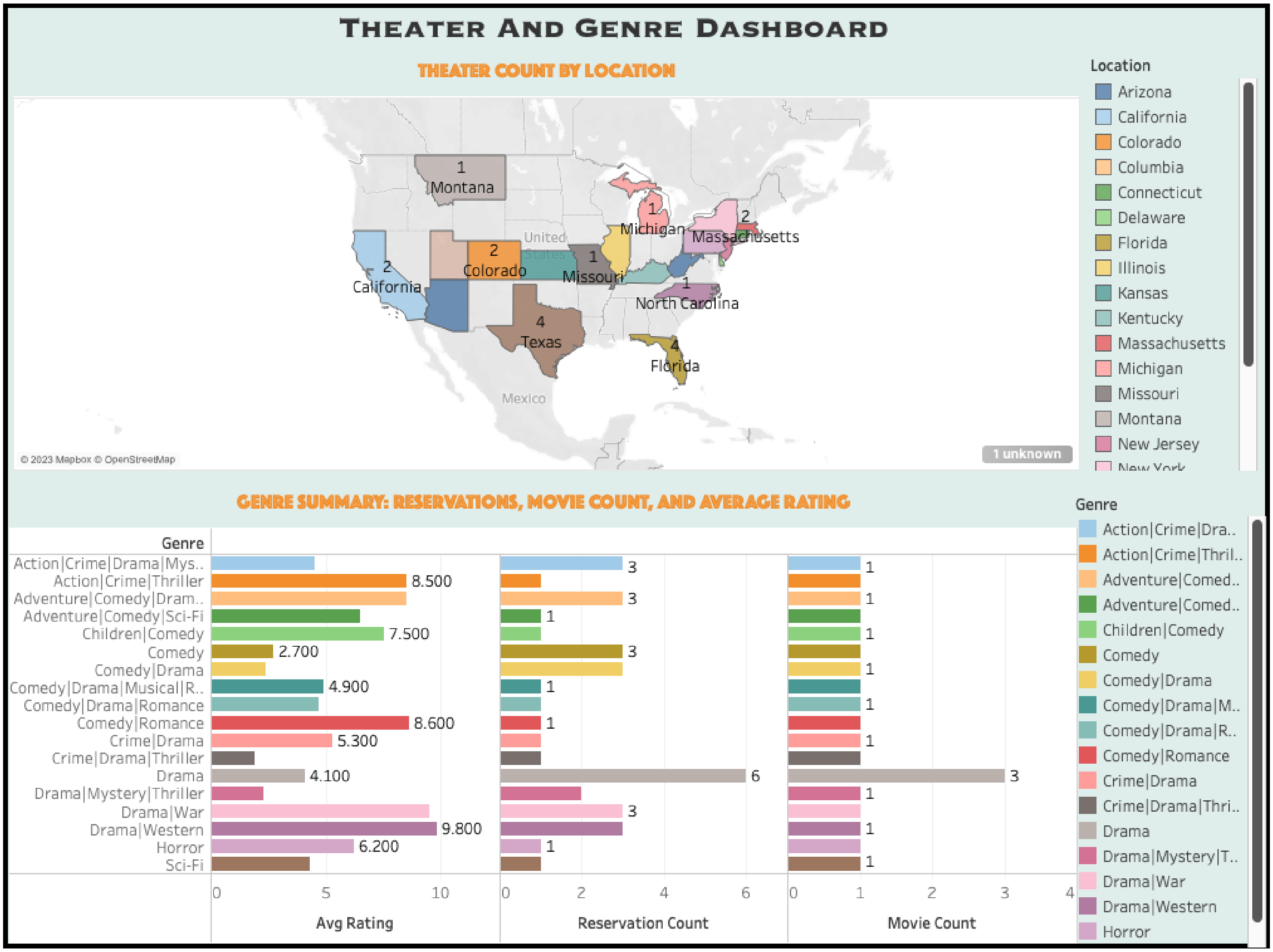


Key takeaways:

1. By analyzing the duration of the movies watched by the customers, we can identify what type of duration preference the customer has, and we can target those customers with the advertisements of their preferred duration movies, which will help increase the return rate of the customer and help increase revenue.
2. By analyzing the data of customers with high reservation count, we can identify the high revenue customers, and we can target the customers with discounts and offers based on the reservation counts, i.e., customers with higher reservation count will get more offers and discounts to keep them encouraged to keep coming back.

2.THEATER AND GENRE DASHBOARD

This dashboard contains analysis of Theater count by location and Genre Summary



Key takeaways:

1. Analyzing the theater count by location can help identify potential market expansion opportunities. States with lower or no theater counts may present untapped markets where the demand for movies might be high.
2. The genre summary visualization offers insights into moviegoers' preferences based on genre. This data can help identify which genres are the most popular and have the highest number of reservations.
3. By comparing the number of movie reservations per genre with the average rating, it is possible to identify which genres generate the most revenue while also maintaining high customer satisfaction. These genres can be targeted for future movie releases and marketing campaigns.

CONCLUSION

In this report, we present an in-depth overview of the ArangoDB project, including its architectural design, data modeling, data preprocessing, and data analysis. The project demonstrated the versatility and strength of ArangoDB as a multi-model database, which provides a robust and versatile solution for managing complex data structures and relationships.

We began by outlining the system's architecture and data flow between components, which enabled a clear comprehension of the structure of the project. Following this, we defined the collections, edges, and relationships between them using the Entity Relationship Diagram and Graph Diagram. These diagrams served as a firm basis for designing the database schema and comprehending the underlying data structure.

The Data Previews section provided insight into the dataset by providing a preview of the sample data, its format, and its arrangement within the collections. Then, we discussed the ETL procedure, which consisted of data extraction, transformation, and loading to ensure that the data was efficiently prepared and imported into ArangoDB. The sections that followed described the procedures involved in creating, defining, and importing data into the database, as well as administering data updates and refreshes. Furthermore, to automate the data refresh process and minimize manual intervention, we implemented a cron job that schedules and triggers the Python script at regular intervals.

Execution of queries and reports on the implemented database was a crucial aspect of this endeavor. We demonstrated the robust querying capabilities of ArangoDB by showcasing various query types and explaining their purpose and results. In addition, we investigated the visualizations produced by these queries, creating dashboards and interpreting the results to gain valuable insights.

This ArangoDB initiative effectively demonstrated the benefits of utilizing a multi-model database for managing and analyzing complex data. Through a well-structured design, efficient data preprocessing, and thorough analysis, we were able to extricate valuable insights and demonstrate the potential of ArangoDB for a variety of applications. To expand the project's scope, future work may involve further database schema optimization, investigation of advanced query techniques, and the incorporation of additional data sources.