

UC San Diego

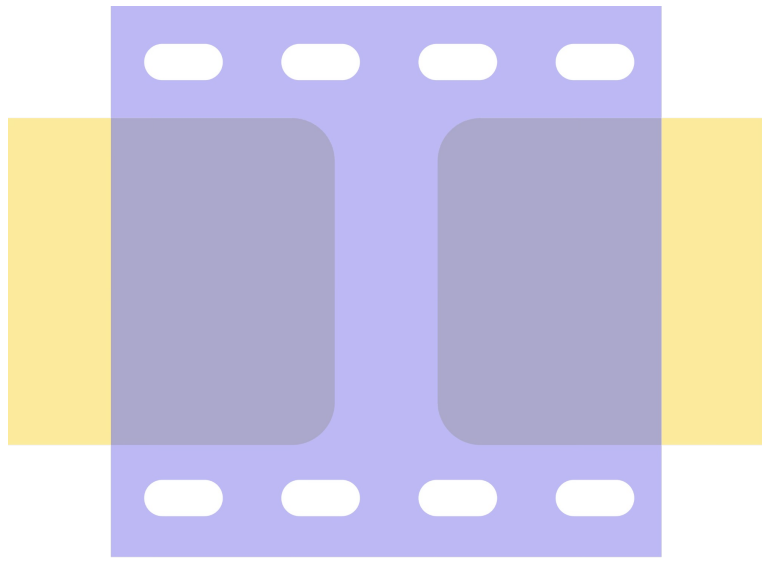
Cinescope:

Scalable Movie Genre Search and Analysis

Group #18

Joel Polizzi, Dongting Cai, Xuanwen Hua

March 18, 2025



Introduction

Project Goals:

- Create a developer to production pipeline on a platform that allows for scalability
- Use up-to-date TMDb (the movie database) datasets
- Build a relational Genre search tool with PostgreSQL
- Use Neo4J to query based on Genre to analyze graph structures in the data
- Integrate Redis as a database cache
- Build a flask application that is publicly available (cinescope.nrp-nautilus.io)

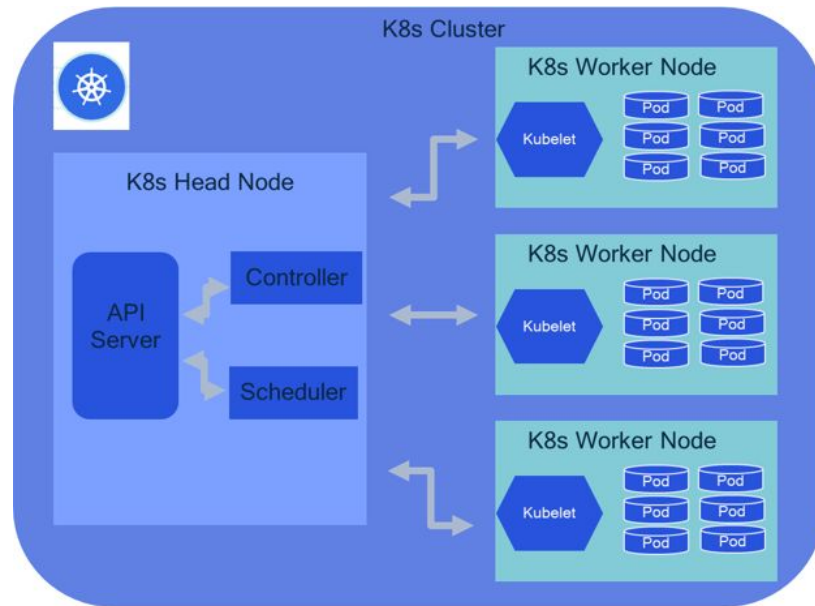
Infrastructure

- High throughput networking on CENIC (10gbps to ~400gbps)
- Microservice architecture
- Easy to customize for our application
- Containers allow for application replication
- Storage and compute resources are readily available

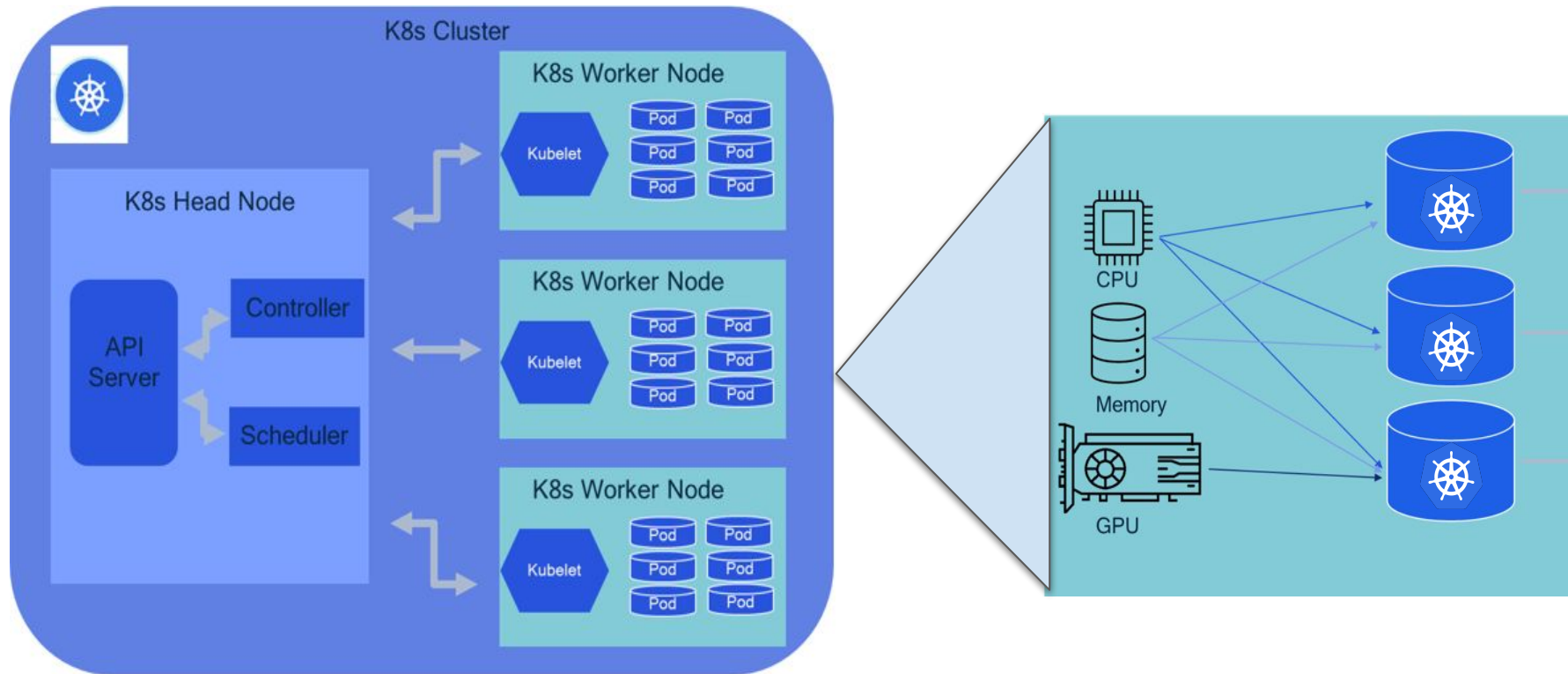


Infrastructure

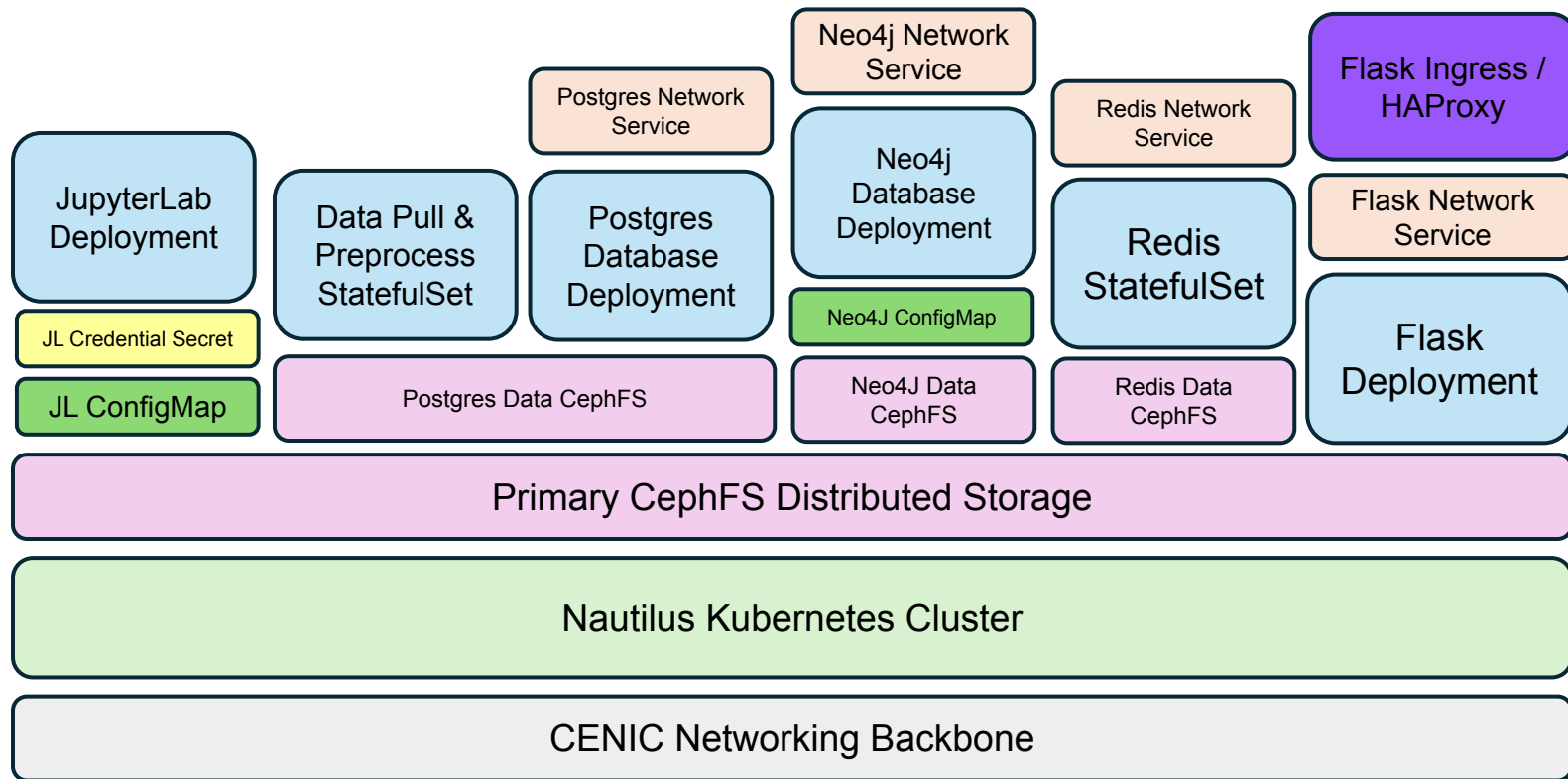
- High throughput networking on CENIC (10gbps to ~400gbps)
- Microservice architecture
- Easy to customize for our application
- Containers allow for application replication
- Storage and compute resources are readily available



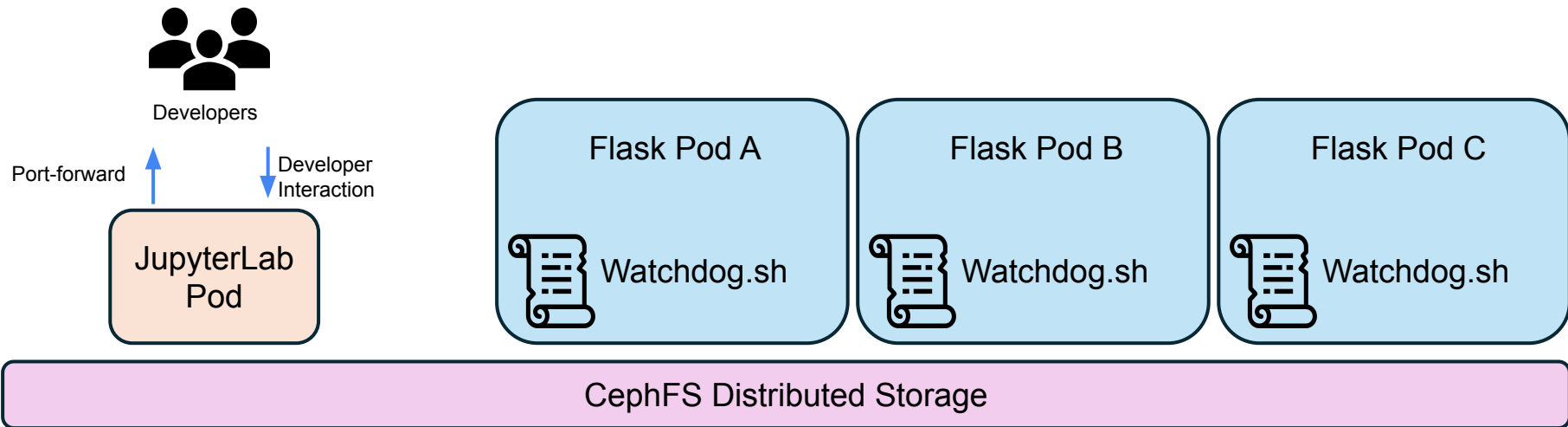
Infrastructure



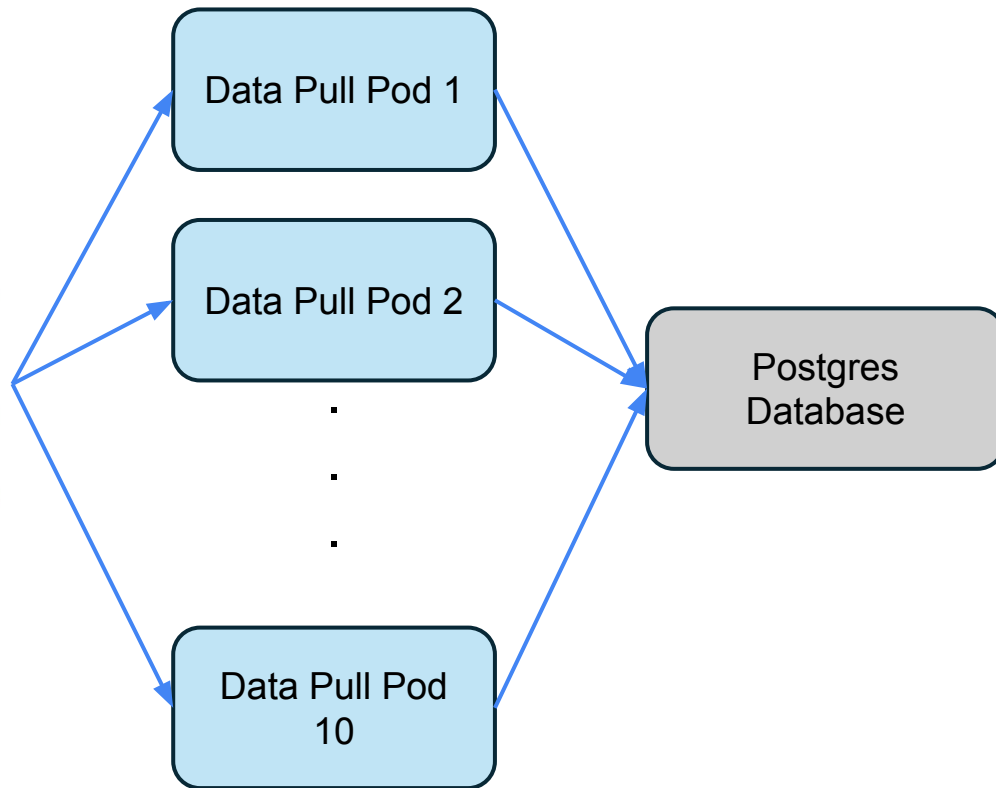
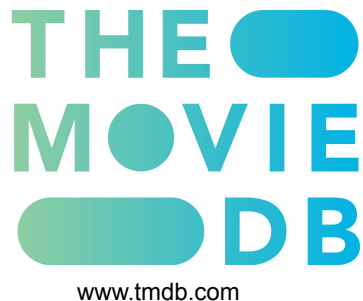
Cloud-Native Stack



From Development to Production

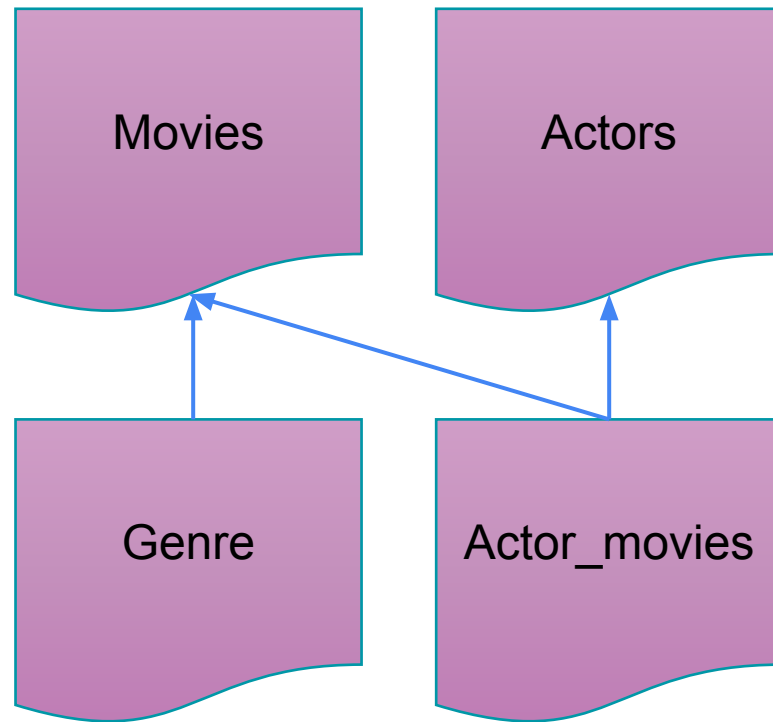


Data Collection



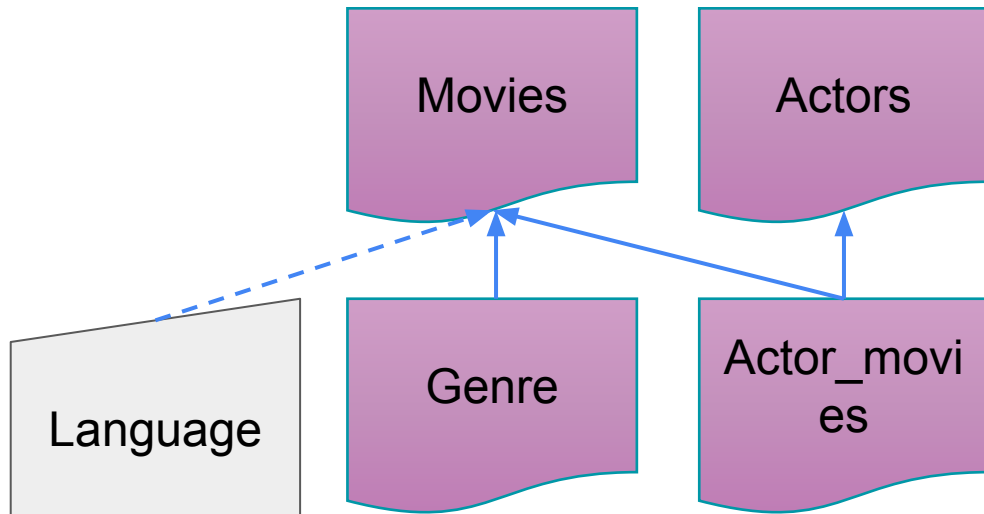
SQL Table Schema

- Movies table is the primary table referenced in the application.
- Actors table Contains an ID of an actor (or crew member) and associates the ID with their Name.
- Actor_movies bridges the Actors dataset and the Movies dataset.
- Genre table is used in conjunction with the Movies table



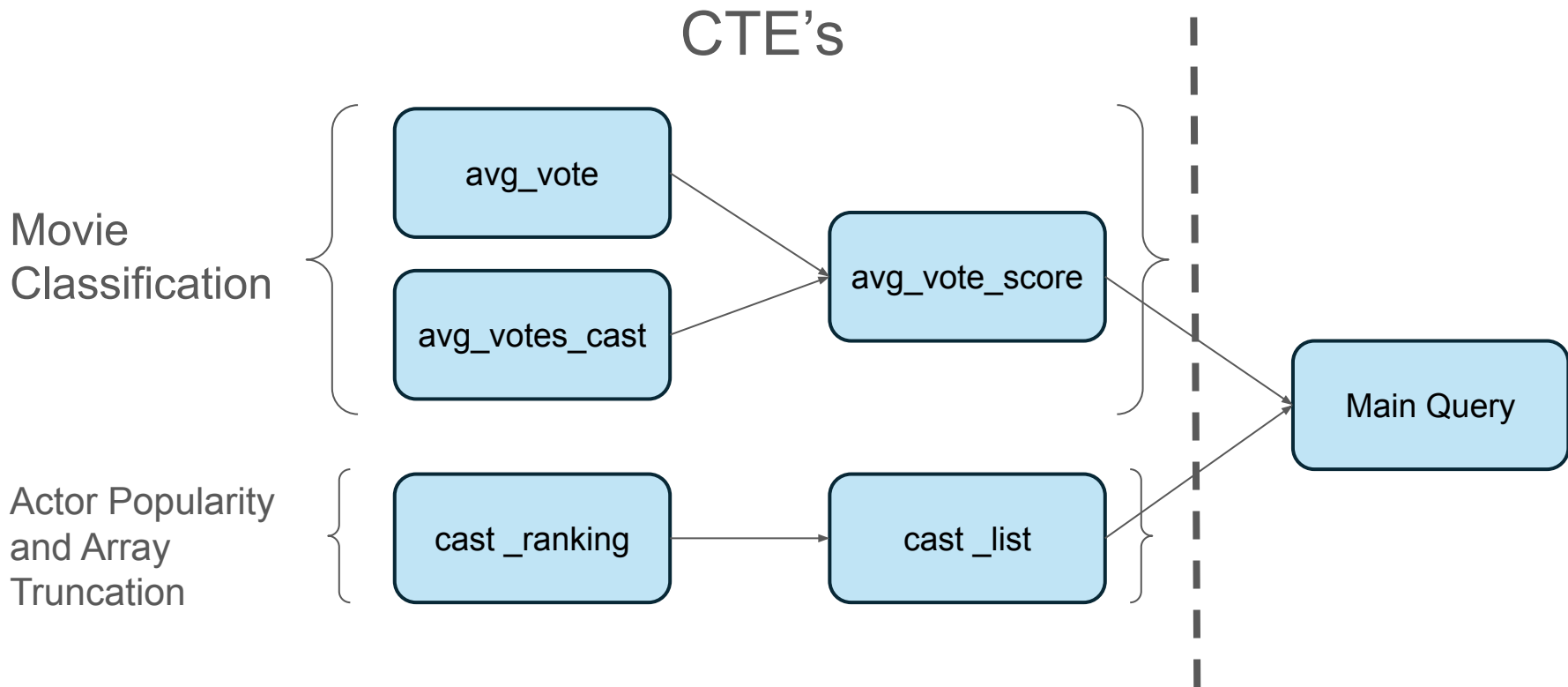
SQL Table Schema + Language

- Movies table is the primary table referenced in the application.
- Actors table Contains an ID of an actor (or crew member) and associates the ID with their Name.
- Actor_movies bridges the Actors dataset and the Movies dataset.
- Genre table is used in conjunction with the Movies table
- Language is introduced as a dictionary generated from querying the



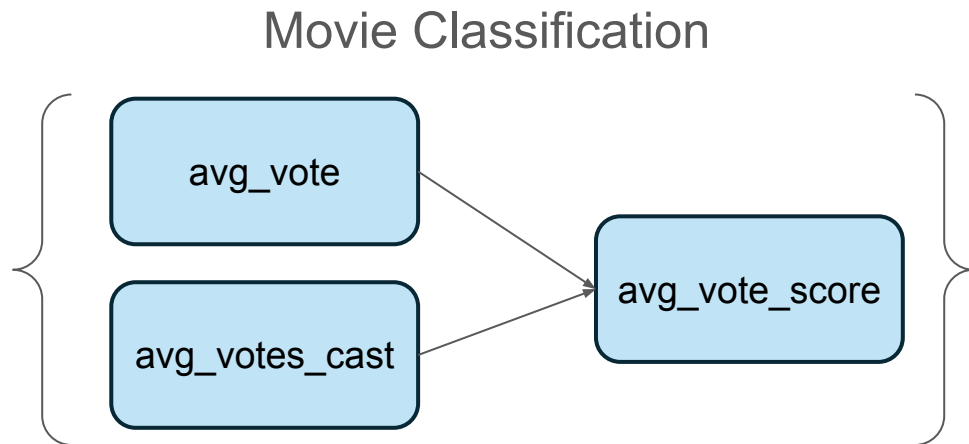
Pseudo-Star Schema

SQL Query Flow

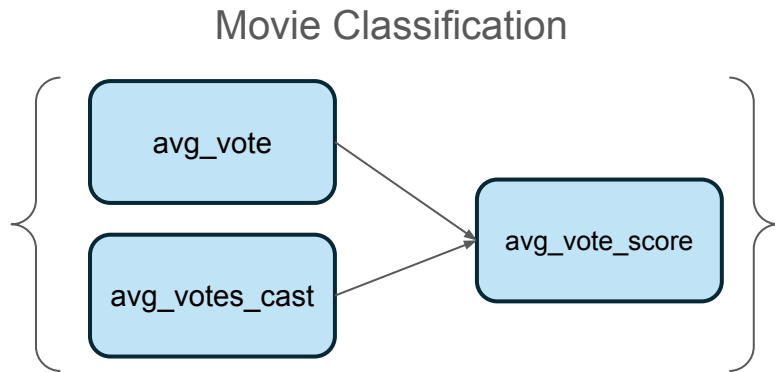
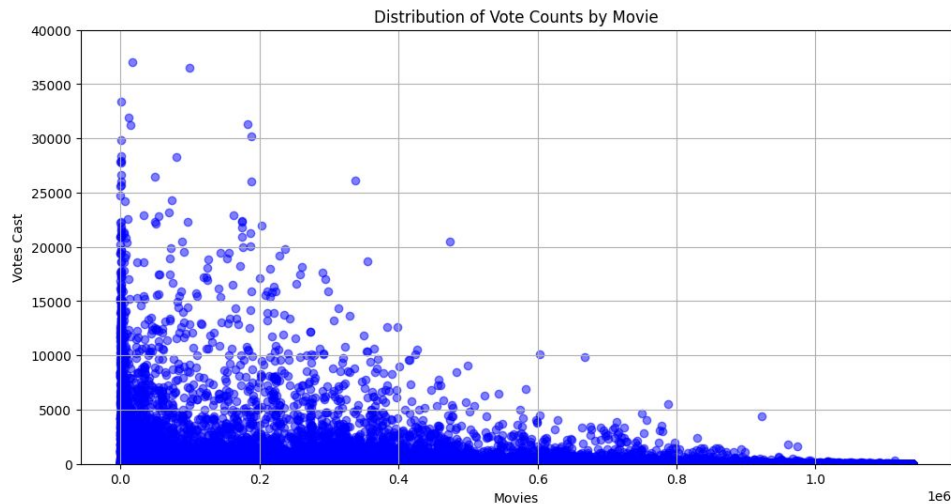


SQL Classification

- Average all vote averages
- Find the average of all votes cast
- Join the avg_vote CTE and the avg_votes_cast CTE into the avg_vote_score to design a classifier

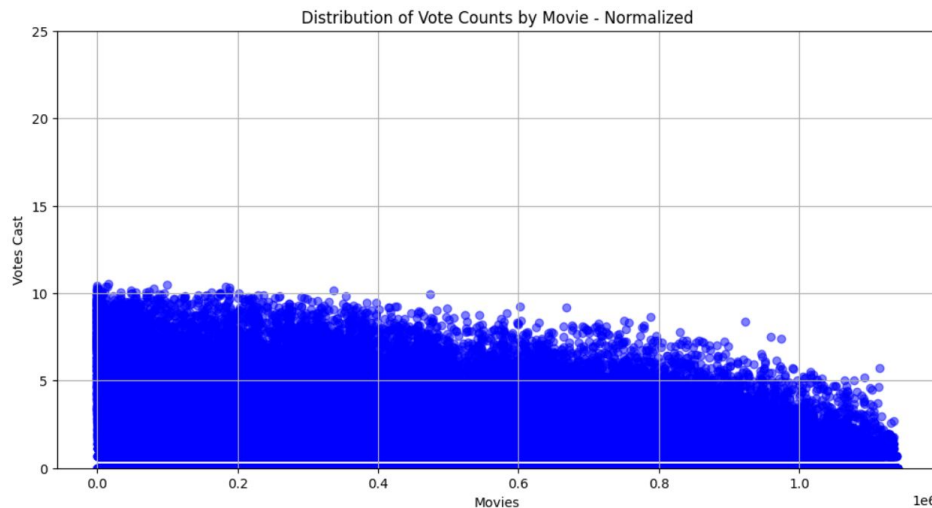
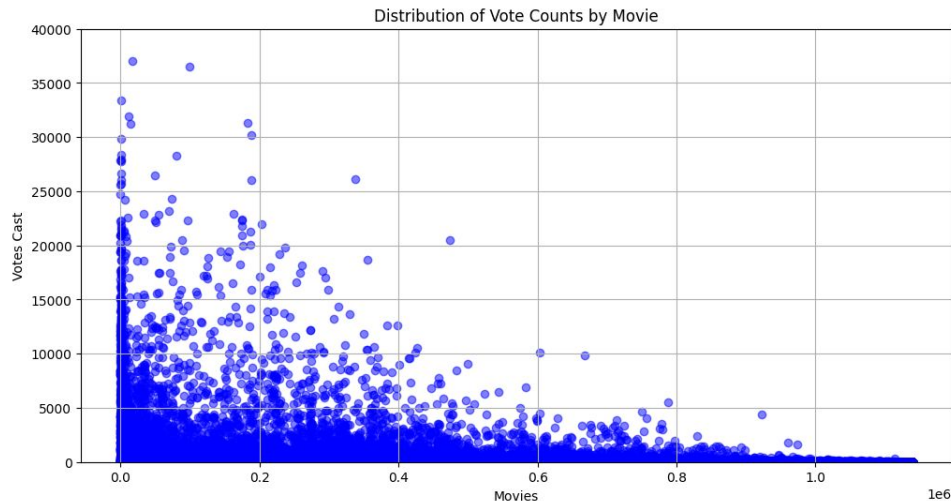


SQL Classification



SQL Classification

- Average of all votes ~21.6
- Wide gap in vote count distribution
- Normalize the data
- Decide what will be the floor of a High, Medium, and Low vote count

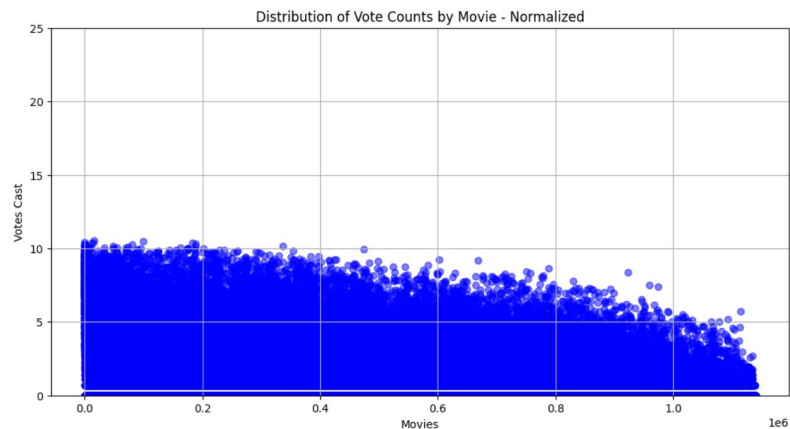


SQL Classification

- Find mean average of all vote counts
- Take the log of the mean avg. multiplied by 10 the get the floor of the “High” classification
- Take the log of the mean avg. multiplied by 2 to get the floor of the “Medium” classification

Let vote_count = x Let avg_count = y

$$\sum_{i=1}^n \begin{cases} \text{'High'} & \text{if } \ln(\text{NULLIF}(x_i, 0)) \geq \ln(y \times 10) \\ \text{'Medium'} & \text{if } \ln(\text{NULLIF}(x_i, 0)) \geq \ln(y \times 2) \\ \text{'Low'} & \text{else} \end{cases}$$



Avg. Vote
Counts

=

```
print(df['vote_count'].mean())
```

21.576217783652183

Conditionals (if)

High

\geq

```
print(np.log(mean_vc * 10))
```

5.374176772572347

Medium

\geq

```
print(np.log(mean_vc * 2))
```

3.7647388601382463

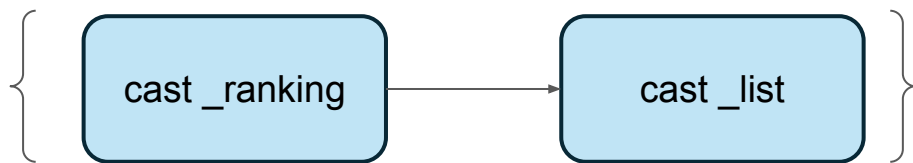
Low

$<$

SQL Ranking

- Cast Ranking will rank the cast members within each movie
- Cast List will create an array of the ranked cast members
- Filter the output to only include the top 5 most popular cast members

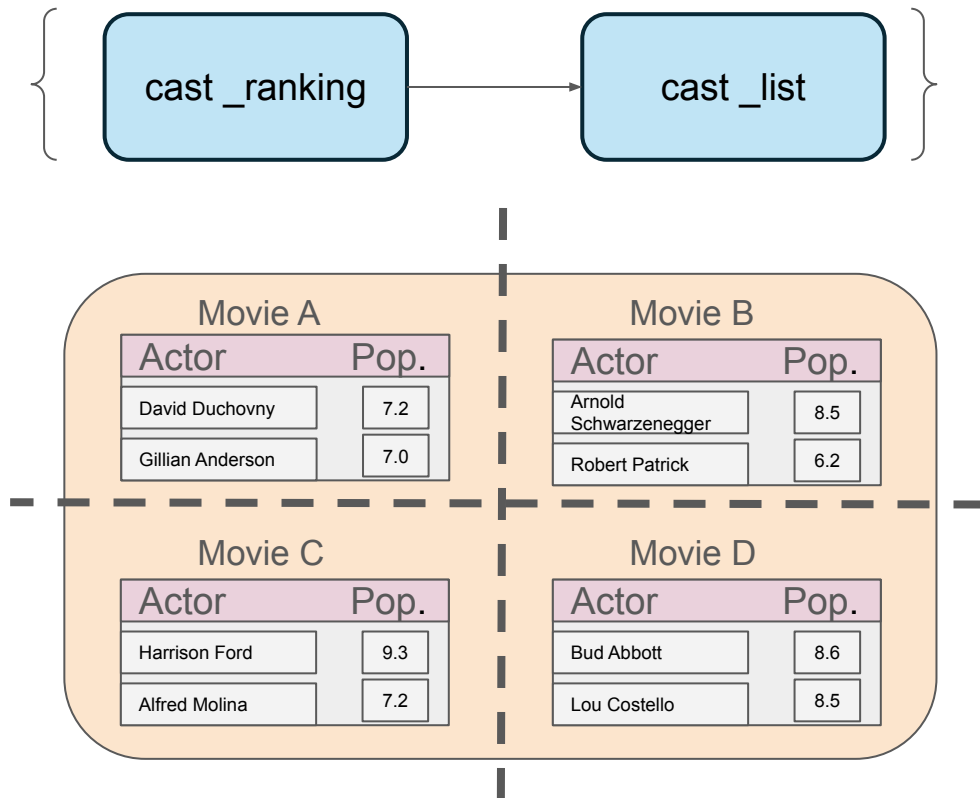
Actor Popularity and Array Truncation



SQL Ranking

- Partition the actors by each movie
- Within each movie order cast members by popularity

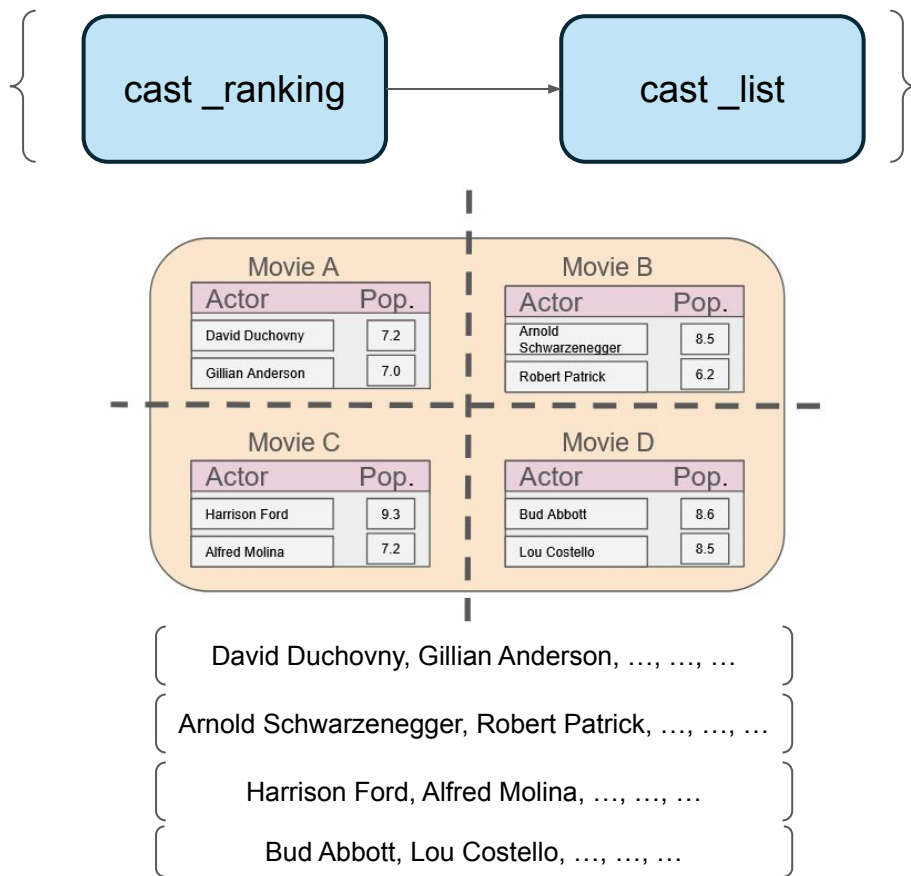
Actor Popularity and Array Truncation



SQL Ranking

- Partition the actors by each movie
- Within each movie order cast members by popularity
- Filter for the top 5 ranked actors for each movie
- Output an Array

Actor Popularity and Array Truncation



SQL Query Output

- Select title, classified viewer score, vote average, vote classification, normalized vote count, top 5 actors, and movie overview
- Subquery to return specified Genre and language
- Order movies by classification score and vote average.

| Movie | Viewer score classification | Vote Average | Vote classification | Vote_count | Movie_cast | Overview |
|---|-----------------------------|--------------|---------------------|--------------------|--|---|
| The Dark Knight | High | 8.5 | High | 10.41711923563258 | ['Gary Oldman', '陳冠希', 'Cillian Murphy', 'Morgan Freeman', 'Michael Caine'] | Batman raises the stakes... |
| The Lord of the Rings: The Return of the King | High | 8.5 | High | 10.114720452824503 | ['Cate Blanchett', 'Viggo Mortensen', 'Elijah Wood', 'Sean Bean', 'Ian McKellen'] | As armies mass for a final battle... |
| Inception | High | 8.4 | High | 10.52023953184236 | ['Leonardo DiCaprio', 'Tom Hardy', 'Cillian Murphy', 'Tom Berenger', 'Joseph Gordon-Levitt'] | Cobb, a skilled thief who commits corporate espionage... |
| The Lord of the Rings: The Fellowship of the Ring | High | 8.4 | High | 10.15085531407452 | ['Cate Blanchett', 'Viggo Mortensen', 'Elijah Wood', 'Sean Bean', 'Ian McKellen'] | Young hobbit Frodo Baggins, after inheriting a mysterious ring... |
| The Lord of the Rings: The Two Towers | High | 8.4 | High | 10.00946787607336 | ['Cate Blanchett', 'Viggo Mortensen', 'Elijah Wood', 'Ian McKellen', 'Liv Tyler'] | Frodo Baggins and the other members of the Fellowship... |

SQL Query Output

- Select title, classified viewer score, vote average, vote classification, normalized vote count, top 5 actors, and movie overview
- Subquery to return specified Genre and language
- Order movies by classification score and vote average.

| Movie | Viewer score classification | Vote Average | Vote classification | Vote_count | Movie_cast | Overview |
|---|-----------------------------|--------------|---------------------|--------------------|--|---|
| The Dark Knight | High | 8.5 | High | 10.41711923563258 | [Gary Oldman, 蝙蝠侠, 'Cillian Murphy', 'Morgan Freeman', 'Michael Caine'] | Batman raises the stakes... |
| The Lord of the Rings: The Return of the King | High | 8.5 | High | 10.114720452824503 | [Cate Blanchett, 'Viggo Mortensen', 'Elijah Wood', 'Sean Bean', 'Ian McKellen'] | As armies mass for a final battle... |
| Inception | High | 8.4 | High | 10.52023953184236 | [Leonardo DiCaprio, 'Tom Hardy', 'Cillian Murphy', 'Tom Berenger', 'Joseph Gordon-Levitt'] | Cobb, a skilled thief who commits corporate espionage... |
| The Lord of the Rings: The Fellowship of the Ring | High | 8.4 | High | 10.15085531407452 | [Cate Blanchett, 'Viggo Mortensen', 'Elijah Wood', 'Sean Bean', 'Ian McKellen'] | Young hobbit Frodo Baggins, after inheriting a mysterious ring... |
| The Lord of the Rings: The Two Towers | High | 8.4 | High | 10.00946787607336 | [Cate Blanchett, 'Viggo Mortensen', 'Elijah Wood', 'Ian McKellen', 'Liv Ullmann'] | Frodo Baggins and the other members of the Fellowship... |

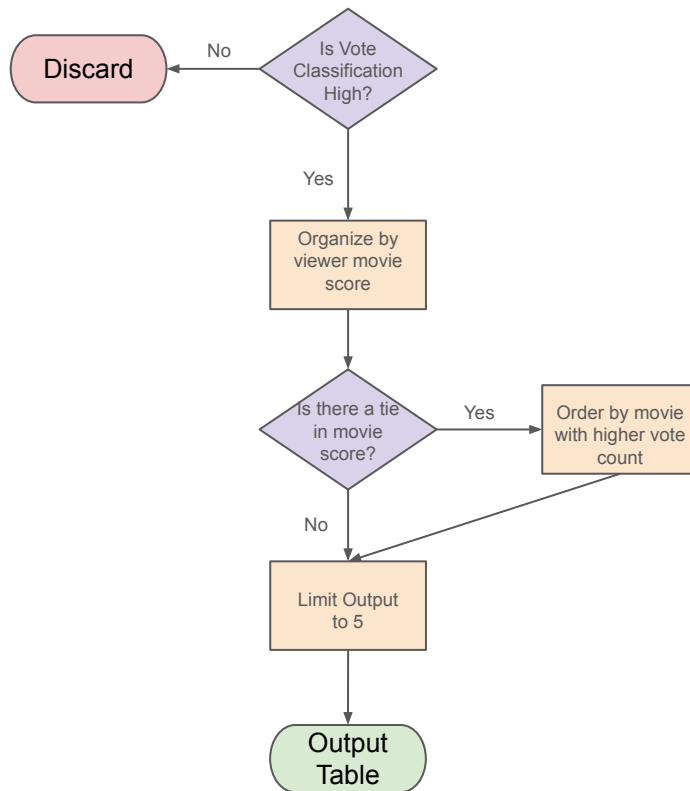
```

75 WHERE EXISTS (
76     SELECT 1
77     FROM JSONB_ARRAY_ELEMENTS(m.genres) AS genre
78     WHERE (genre ->> 'id')::INTEGER = {GENRE_ID} -- placeholder for genre id
79 )
80 AND m.original_language = '{LANGUAGE_PARAM}'

```

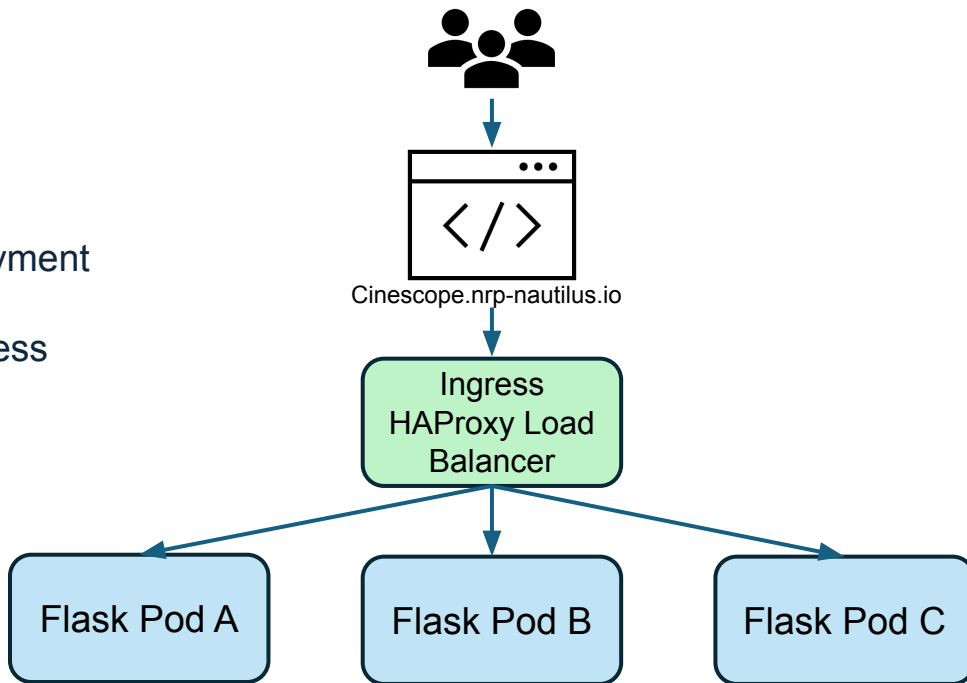
SQL Query Output

- Ordering The Output
 - Check that the movie has a “High” vote count classification.
 - Organize “High” movies by their average score
 - In cases where there is a tie, list the movie with the higher vote count in descending order.
 - Output the table.

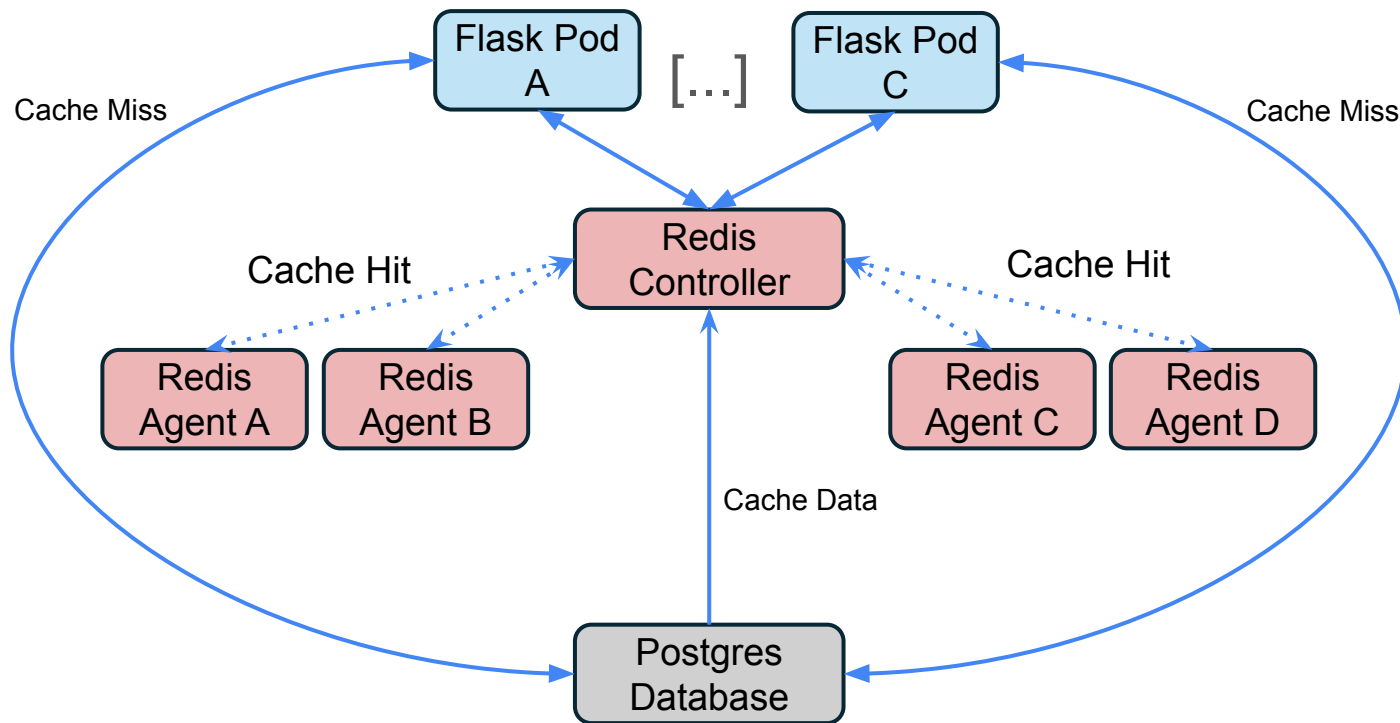


Horizontal Scaling

- Flask Pods have been created as a deployment with a 3 x replication
- As a End-user visits the webpage the ingress controller directs traffic

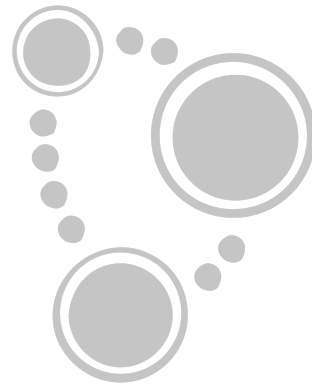


Horizontal Scaling



Neo4j in Cinescope: Enabling Graph-Based Analysis

- Leverages graph database for **relationship exploration**
- **Complements PostgreSQL** for efficient data analysis
- **Supports** fast traversal of movies, genres, and actors
- **Integrated** into a microservices architecture via **Flask**



Data Modeling & System Integration

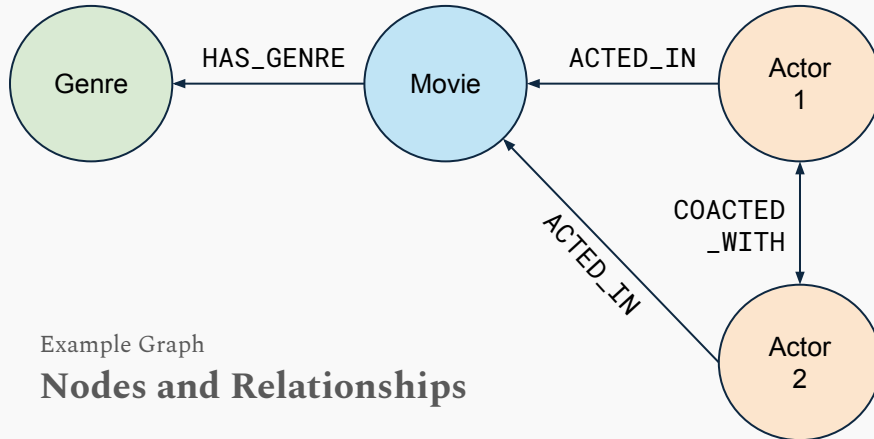
- **Nodes:**

Genre, Movie, Actor

- **Relationships:**

- ACTED_IN (Actor → Movie)
- HAS_GENRE (Movie → Genre)
- COACTED_WITH (Actor ↔ Actor)

- **PostgreSQL** supplies top movies based on SQL queries
- **Flask** routes coordinate data retrieval and graph construction



Implementation & Query Techniques

- Batch data import to CSV conversion
- Use of MERGE to prevent duplicate nodes
- Efficient Cypher queries:
 - Retrieve top movies per genre
 - Build actor co-acting networks
- Filtering on popularity to manage large datasets

Example Cypher Query

**Finds pairs of actors who
have co-starred in more
than one movie within the
specified genre**

```
MATCH (a1:Person)-[:ACTED_IN]->(m:Movie)<-[:ACTED_IN]-(a2:Person),  
      (m)-[:HAS_GENRE]->(g:Genre {name: $genre})  
WHERE a1 <> a2  
WITH a1, a2, COUNT(m) AS shared_movies  
WHERE shared_movies > 1  
RETURN a1.name AS Actor1, a2.name AS Actor2, shared_movies;
```

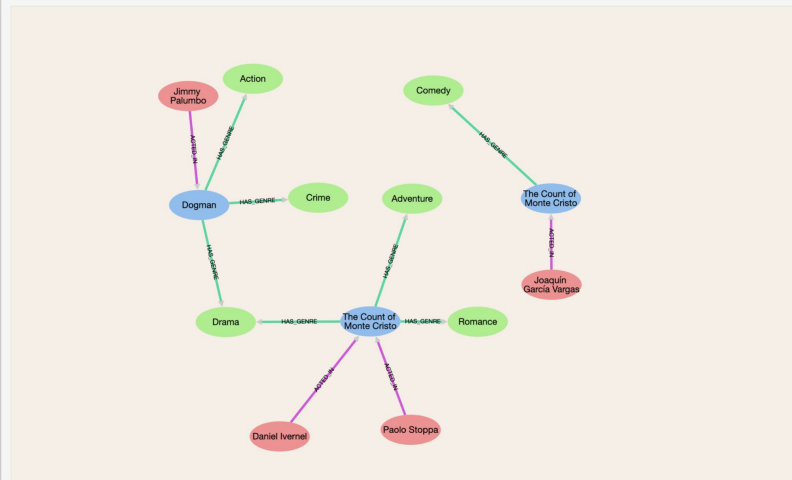
Interactive Graph Visualization

- Two **graph views**
- Rendered using Cytoscape.js
- Enhances data exploration and user engagement
- Opportunities for future interactive features

Graph View

Select a genre and click the button below to view an interactive graph of the top 5 movies and their actor relationships.

Load Graph

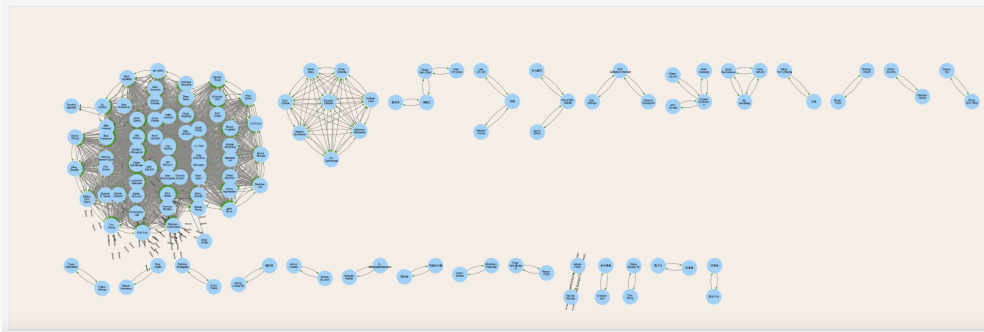


- ← **Graph 1: Top Movies & Their Actors**
- ↓ **Graph 2: Actor Co-acting Network**

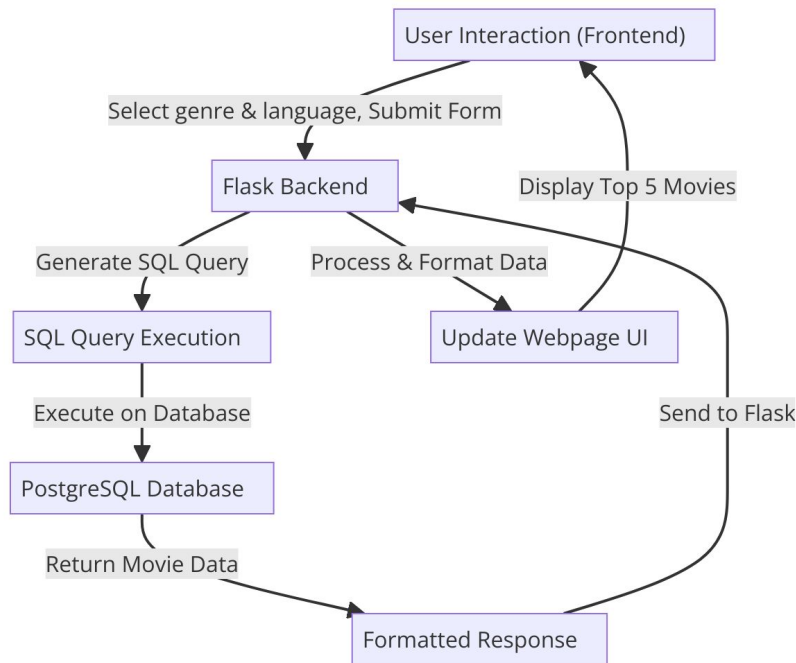
Graph View

View the relationships between actors who have played in multiple movies of the same genre.

Load Actor Graph



Front End development & PostgreSQL Integration



The **Front-End** captures user input and sends requests to the backend.

Flask (Back-End) processes the request, generates SQL queries, and interacts with the database.

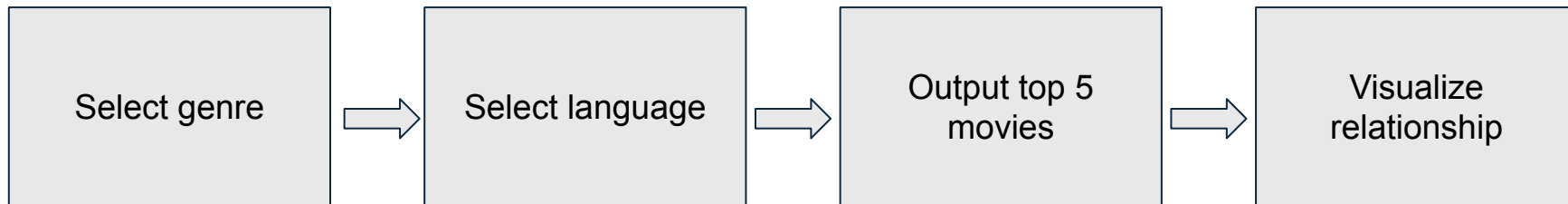
PostgreSQL executes queries and returns relevant data.

Flask formats the response and updates the **Front-End** dynamically.

The result is a **seamless user experience**, displaying movie recommendations in real time.

Front End development & PostgreSQL Integration

- 1 User selects genre & language (Dropdowns & UI Forms)
- 2 Flask retrieves corresponding genre ID from PostgreSQL
- 3 SQL Query fetches ****top 5**** most popular movies per selection
- 4 Results dynamically rendered in a structured HTML table



Front End development & PostgreSQL Integration

- Flask is used to tie together the back-end and front-end.
 - Handles GET requests to display genres and languages.
 - Processes POST requests to filter movies based on user selections.
 - Jinja2 templating integrates dynamic content into HTML pages.
-

- Flask acts as a bridge between user interactions & database queries.
- **Psycopg2** is used for secure SQL query execution.
- Queries dynamically filter ****movies by genre & language****.
- Results are sent back to Flask, formatted with Jinja2 templates.

Front End development & PostgreSQL Integration

- ✔ Flask (Backend Web Framework)
- ✔ HTML, CSS, Bootstrap (UI/UX Design)
- ✔ Jinja2 (Dynamic Content Rendering)
- ✔ Select2 & JavaScript (Enhanced User Interactions)
- ✔ PostgreSQL (Data Storage & Querying)

Select Language:

Afar--Ethiopia

Afar--Ethiopia

Abkhazian--Georgia

Afrikaans--South Africa

Akan--Ghana

Amharic--Ethiopia

Aragonese--Spain

Select Genre:

Action

Action

Adventure

Animation

Comedy

Crime

Documentary

Drama

Family

Fantasy

History

Horror

Music

Mystery

Romance

Science Fiction

TV Movie

Thriller

War

Western

Top 5 Movies

| Title | Viewer Score | Vote Average | Vote Classification | Vote Count | Top Cast | Plot Description |
|--|--------------|--------------|---------------------|-----------------------|---|--|
| Seven Samurai | High | 8.5 | High | 8,241,965,602,31802 | 〔三船敏郎, 小杉 龍男, 仲代 達夫, 津島 恵子, 藤里 進也〕 | A samurai answers a village's request for protection after he falls on hard times. The town needs protection from bandits, so the samurai gathers six others to help him teach the people how to defend themselves, and the villagers provide the soldiers with food. |
| Harakiri | High | 8.4 | High | 6,942,156,705,699,469 | 〔若下 忠雄, 仲代 達夫, 丹波 哲郎, 三国連太郎, 井川 比呂志〕 | Down on his luck veteran Tuganue Harakiri enters the courtyard of the prosperous House of Iyi. Unemployed, and with no family, he hopes to find a place to commit seppuku—and a worthy second to deliver the coup de grace in his suicide ritual. The senior counselor for the Iyi clan questions the ronin's resolve and integrity, suspecting Harakiri of seeking charity rather than an honorable end. What follows is a pair of interlocking stories which lay bare the difference between honor and respect, and promises to examine the legendary foundations of the Samurai code. |
| Attack on Titan: The Roar of Awakening | High | 8.4 | High | 5,703,782,474,656,201 | 〔小野 大輔, 津田 健次郎, 子安 武人, 神谷 浩史, 梶 裕貴〕 | Eren Yeager and others of the 104th Training Corps have just begun to become full members of the Survey Corps. As they ready themselves to face the Titans once again, their preparations are interrupted by the invasion of Wall Rose—but all is not as it seems as more mysteries are unveiled. As the Survey Corps races to save the wall, they uncover more about the invading Titans and the dark secrets of their own members. |
| Neon Genesis Evangelion: The End of Evangelion | High | 8.3 | High | 7,405,495,663,199,472 | 〔林 愿史 C.A., 子安 武人, 三石 琴乃, 宮村 優子, 緒方 恵美〕 | SEELE orders an all-out attack on NERV, aiming to destroy the Eva before Gendo can advance his own plans for the Human Instrumentality Project. Shinji is pushed to the limits of his sanity as he is forced to decide the fate of humanity. An alternate ending to the television series, "Neon Genesis Evangelion", which aired from 1995 to 1996 and whose final two episodes were controversial for their stylistically abstract direction. |
| Demon Slayer -Kimetsu no Yaiba- The Movie: Mugen Train | High | 8.2 | High | 8,296,297,112,642,508 | 〔早見 沙織, 鬼頭 幸枝, 花江 果梨, 佐藤 利幸, 松岡 雅也〕 | Tanjirō Kamado, joined with Inosuke Hashibira, a boy raised by boars who wears a boar's head, and Zenitsu Agatsuma, a scared boy who reveals his true power when he sleeps, boards the Infinity Train on a new mission with the Fire Hashira, Kyojuro Rengoku, to defeat a demon who has been tormenting the people and killing the demon slayers who oppose it. |

Thanks for your Listening!

DSC 202 Final Project
Cinescope: Scalable Movie Genre Search and Analysis

Group #18 Joel Polizzi, Dongting Cai, Xuanwen Hua
March 18, 2025



