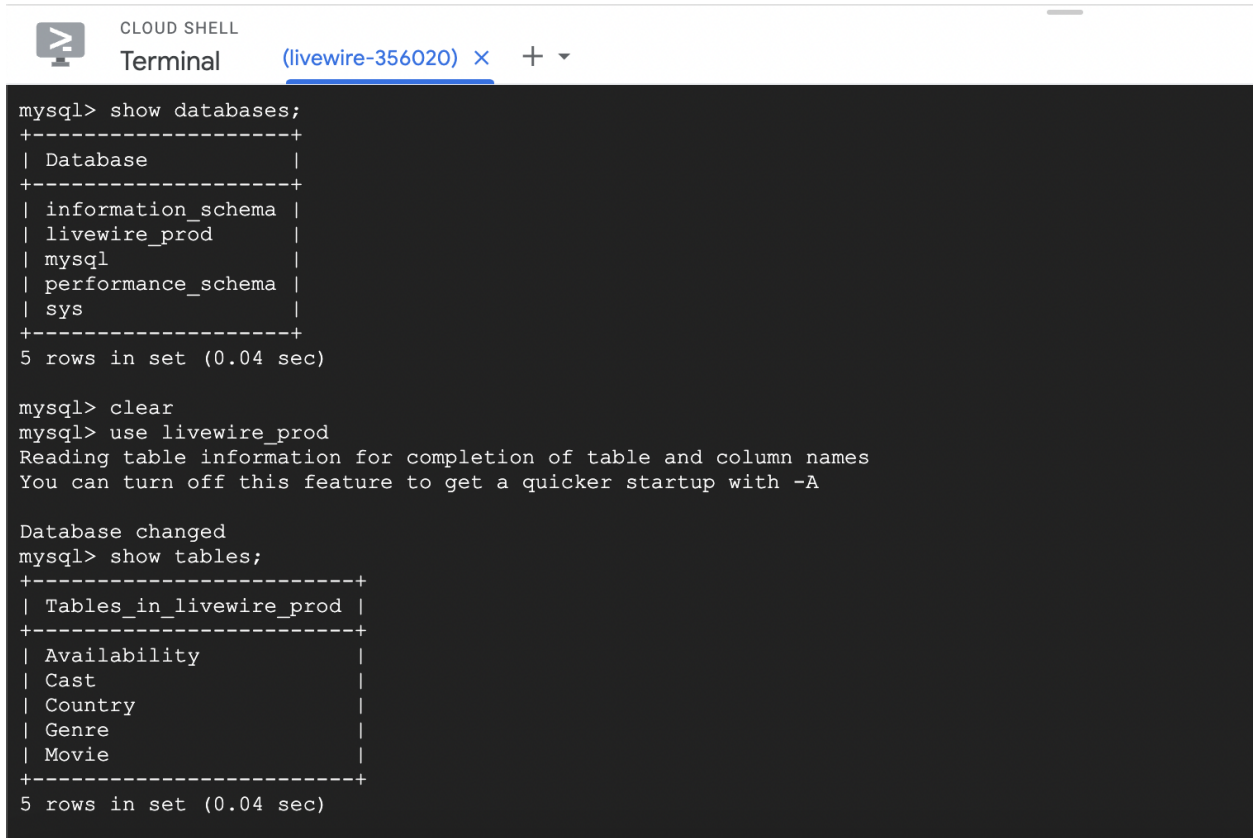


Stage 3: Database Implementation and Indexing

Database Implementation

1. Screenshot of the connection



The screenshot shows a Cloud Shell terminal window with the title "Terminal (livewire-356020)". The terminal displays the following MySQL commands and their outputs:

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| livewire_prod |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.04 sec)

mysql> clear
mysql> use livewire_prod
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_livewire_prod |
+-----+
| Availability |
| Cast |
| Country |
| Genre |
| Movie |
+-----+
5 rows in set (0.04 sec)
```

2. DDL Commands for the tables

```
CREATE TABLE Movie(movie_id INT PRIMARY KEY, name VARCHAR(255), media_type VARCHAR(255),
release_dt DATE, synopsis TEXT);
CREATE TABLE Country(country_id INT PRIMARY KEY, country_name VARCHAR(255));
CREATE TABLE Availability(availability_id INT PRIMARY KEY, movie_id INT, country_id INT, FOREIGN
KEY(movie_id) REFERENCES Movie(movie_id), FOREIGN KEY(country_id) REFERENCES
Country(country_id));
CREATE TABLE Cast (cast_id INT PRIMARY KEY, movie_id INT, role VARCHAR(255), name VARCHAR(255),
FOREIGN KEY(movie_id) REFERENCES Movie(movie_id));
CREATE TABLE Genre (genre_id INT PRIMARY KEY, movie_id INT, genre VARCHAR(255), FOREIGN
KEY(movie_id) REFERENCES Movie(movie_id));
```

3. Count Queries for each table

```
1 SELECT COUNT(*)
2 FROM Movie
```

100% 11:2

Result Grid Filter Rows: Search Export:

COUNT(*)
1000

```
1 SELECT COUNT(*)
2 FROM Cast
```

100% 10:2

Result Grid Filter Rows: Search Export:

COUNT(*)
3779

```
1 SELECT COUNT(*)
2 FROM Availability
```

100% 18:2

Result Grid Filter Rows: Search Export:

COUNT(*)
6253

```
1 SELECT COUNT(*)
2 FROM Genre
```

100% 11:2

Result Grid Filter Rows: Search Export:

COUNT(*)
3767

1	•	SELECT COUNT(*)
2		FROM Country
100% 7:2		
Result Grid Filter Rows: Search Export:		
		COUNT(*)
▶		60

Advanced Queries

Query 1

1	•	(SELECT M.name, M.synopsis
2		FROM Movie M LEFT JOIN Cast C ON M.movie_id = C.movie_id
3		WHERE C.role = "Director" AND C.name = "Vivek Athreya")
4		
5		UNION
6		
7	•	(SELECT name, synopsis
8		FROM Movie M NATURAL JOIN Genre G
9	•	WHERE genre IN (SELECT genre
10		FROM Movie M NATURAL JOIN Genre G
11		WHERE name = "Ante Sundaraniki"))
12		
13		LIMIT 15

	name	synopsis
▶	Ante Sundaraniki (Malayalam)	When sparks fly between childhood friends Sun...
▶	Ante Sundaraniki (Tamil)	When sparks fly between childhood friends Sun...
▶	Ante Sundaraniki	When sparks fly between childhood friends Sun...
▶	Swept Away	A shipwreck lands a fiery upper-class woman an...
▶	Funny Face	Fred Astaire's dancing feet come full circle...
▶	Rich and Strange	In this early film from Alfred Hitchcock, a marrie...
▶	Bollywood / Hollywood	Told that his sister can't marry until he is h...
▶	Fubar: The Movie	After receiving a heavy diagnosis, a metalhead...
▶	Jargo	After moving from Saudi Arabia to Germany, a t...
▶	Spaced Out	Aliens crash-land their cargo ship on Earth and...
▶	A Real Life	A petty thief falls for a respectable teacher and t...
▶	Fubar: Balls to the Wall	After one too many parties, dedicated headbang...
▶	Footy Legends	An unemployed young Australian tries to raise...
▶	The Pretty One	When she's mistaken for her deceased ide...
▶	Tiempo de valientes	When a psychoanalyst is assigned to treat a de...

Index Analysis

The indices that were already present on the datatables were those based the Primary and Foreign Key of the tables.

1. CREATE INDEX genreidx ON Genre(genre);
Creating this index brought the 1 row in set time down from 0.06 to 0.05 seconds. It increased the table scan on Cast time but reduced the table scan on Movie time which is joined to genre twice in the query.
2. CREATE INDEX movieidx ON Movie(name);
Creating this index brought the 1 row in set time down from 0.06 seconds to 0.05 seconds as well. Table scan times on Cast and Movie were both reduced by this index.
3. Both indices together (genreidx and movieidx)
Using both indices together increased the cost of table scan on union temporary, but reduced the actual time. There were improvements in the actual time for the rest of the table scans.

There was no significant difference due to the nature of the query.

Query 2

```
1 • SELECT name, synopsis, country_name
2   FROM (SELECT movie_id, country_name
3          FROM Movie M NATURAL JOIN Availability A NATURAL JOIN Country C
4          WHERE country_name = "France") AS T NATURAL JOIN Movie M
5  WHERE name LIKE "The%"
6  ORDER by name
7  LIMIT 15
8
```

	name	synopsis	country_name
▶	The 7 Lives of Lea	After finding a young man's remains, Léa...	France
●	The Adolf Eichmann Trial	This historical documentary follows the events o...	France
●	The Age Of Monsters	On his deathbed, a filmmaker gathers various c...	France
●	The Amber Light	This cozy documentary tours Scotland to explor...	France
●	The Amityville Theater	Following the loss of her parents, a teen inherits...	France
●	The Art of Incarceration	At the Fulham Correctional Centre, incarcerated...	France
●	The Beauty Queen of Jerusalem	In 1919 Jerusalem, housecleaner Rosa weds a...	France
●	The Beginning	This documentary follows fearless thrill seekers...	France
●	The Big Dream	Inspired by the story of Ghanaian footballer Ibra...	France
●	The Bird Can't Fly	A South African cook returns to her hometown t...	France
●	The Boss Baby: Back in the Crib	Framed for a corporate crime, an adult Ted Tem...	France
●	The Bubble	Sneaking out. Hooking up. Melting down. The c...	France
●	The Coming War on China	Award-winning journalist John Pilger explores h...	France
●	The Coolest Hair	To try and sell a suitcase full of drugs, two small...	France
●	The Creature Cases	Special agents Sam and Kit hop the globe with t...	France

Index analysis

The indices that were already present on the datatables were those based on the Primary and Foreign Key of the tables.

1. CREATE INDEX movieidx ON Movie(name);
This index reduced the 1 row in set time from 0.05 to 0.01s. We decided to try this again because it was effective in the previous query. We found that the actual time was slightly reduced and the sort time was increased. The table scan on Country actual time was reduced by half.
2. CREATE INDEX countryidx on Country(country_name);
This time countryidx was used during the search for 'France' and the time decreased from 0.044 to 0.013. Instead of scanning the entire joined table for a country name, the index greatly reduced the time spent searching.
3. Both indices together (countryidx and movieidx)
This had the country table benefits as well as the improvements in the actual time. The sort time returned to its original amount.