The below query is used to retrieve information about the different movies:

```
EXPLAIN SELECT distinct c.CatalogID, c.Title, c.Director, c.ReleaseDate, c.PosterUrl, m.Duration, g.Description as Genre
FROM movies AS m, catalog AS c, genre AS g
WHERE m.CatalogID = c.CatalogID AND c.CatalogID = g.CatalogID AND c.PosterUrl IS NOT NULL;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	g	NULL	ALL	CatalogID	NULL	NULL	NULL	3621	100.00	Using where; Using temporary
1	SIMPLE	c	NULL	eq_ref	PRIMARY,CatalogID	PRIMARY	4	prac_5.g.CatalogID	1	90.00	Using where
1	SIMPLE	m	NULL	ref	CatalogID	CatalogID	5	prac_5.g.CatalogID	1	100.00	NULL

Query Statistics

Timing (as measured at client side):
Execution time: 0:00:0.04700000

Timing (as measured by the server):
Execution time: 0:00:0.03245150
Table lock wait time: 0:00:0.00000400

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 669
Rows examined: 7911

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 1

Joins per Type:
Full table scans (Select_scan): 1
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
No Index used

Other Info:
Event Id: 1571
Thread Id: 141

According to the above query statistics analysis Execution time measured at client side is 0.047 seconds and the Execution time measured by the server is 0.03245150 seconds. One temporary table is created, meaning that the intermediate results are stored temporarily, which can slow down the performance. The temporary table is created due to the 'DISTINCT' keyword, which eliminates duplicate rows, causing additional overhead. The 'DISTINCT' keyword was used due to our initial dataset containing duplicate values; however, all our data must be unique therefore after deleting all duplicate values in our dataset we can see that optimization can be improved by removing the 'DISTINCT' keyword from the query.

```
EXPLAIN SELECT c.CatalogID, c.Title, c.Director, c.ReleaseDate, c.PosterUrl, m.Duration, g.Description as Genre
FROM movies AS m, catalog AS c, genre AS g
WHERE m.CatalogID = c.CatalogID AND c.CatalogID = g.CatalogID AND c.PosterUrl IS NOT NULL;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	g	NULL	ALL	CatalogID	NULL	NULL	NULL	3621	100.00	Using where
1	SIMPLE	c	NULL	eq_ref	PRIMARY,CatalogID	PRIMARY	4	prac_5.g.CatalogID	1	90.00	Using where
1	SIMPLE	m	NULL	ref	CatalogID	CatalogID	5	prac_5.g.CatalogID	1	100.00	NULL

Timing (as measured at client side):
Execution time: 0:00:0.01500000

Timing (as measured by the server):
Execution time: 0:00:0.02784640
Table lock wait time: 0:00:0.00000400

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 669
Rows examined: 7911

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 0

Joins per Type:
Full table scans (Select_scan): 1
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
No Index used

Other Info:
Event Id: 1510
Thread Id: 141

After optimising our query by removing the ‘DISTINCT’ keyword, we can observe the following performance gains achieved:

Execution time has decreased as the overhead (sorting and comparing rows) associated with removing duplicate values has been eliminated.

- Execution time measured at client side reduced from 0.047 seconds to 0.015 seconds.
- Execution time measured by the server reduced from 0.03245150 seconds to 0.02784640 seconds.

No temporary table was created as there is no need for a temporary table to store intermediate results associated with finding duplicate values.

- Leading to faster execution and better performance.

*side note: it is confirmed that there are no duplicate values contained in the dataset as the number of rows sent to the client remained the same, hence no need for the ‘DISTINCT’ keyword.