queries_with_exists

```
SELECT *
FROM renting
WHERE rating IS NOT NULL -- Exclude those with null ratings
AND customer_id = 115;
```

Explanation:

- This SQL query selects all columns (*) from the renting table where the rating is not NULL and the customer_id is 115. It filters the results to show only entries for customer 115 that have a rating assigned.

```
SELECT *
FROM renting
WHERE rating IS NOT NULL -- Exclude null ratings
AND customer_id = 1; -- Select all ratings from customer with ID 1
```

Explanation:

- This SQL query selects all columns (SELECT *) from the renting table where the rating column is not NULL and the customer_id is 1. It effectively retrieves all the rental information for customer with ID 1 that have associated ratings.

```
SELECT *
FROM customers AS c -- Select all customers with at least one rating
WHERE EXISTS
  (SELECT *
  FROM renting AS r
  WHERE rating IS NOT NULL
  AND r.customer_id = c.customer_id);
```

Explanation:

- This SQL query retrieves all customers who have at least one non-null rating in the renting table. It uses a correlated subquery with the EXISTS operator for efficiency. The outer query selects all columns from the customers table, while the inner query checks if a matching customer_id exists in the renting table with a non-null rating. If such a record exists, the customer is included in the result.

```
SELECT *  -- Select all columns
FROM actsin AS ai  -- From the 'actsin' table (aliased as 'ai')
LEFT JOIN movies AS m  -- Perform a left join with the 'movies' table (aliased as 'm')
```

```sql
ON ai.movie_id = m.movie_id  -- Join condition: matching 'movie_id' columns
WHERE m.genre = 'Comedy';  -- Filter results to include only movies of genre 'Comedy'
```

Explanation:

- This SQL query retrieves information about actors who have acted in comedy movies. It joins the actsin table (containing actor-movie relationships) with the movies table (containing movie details) to filter for movies with the genre ‘Comedy’. The LEFT JOIN ensures that all actors from actsin are included, even if they haven’t acted in a comedy. The WHERE clause filters the results to only show those related to comedy movies.

```sql
SELECT *
FROM actsin AS ai
LEFT JOIN movies AS m
ON m.movie_id = ai.movie_id
WHERE m.genre = 'Comedy'
AND ai.actor_id = 1; -- Select only the actor with ID 1
```

Explanation:

- This SQL query retrieves information about movies of the ‘Comedy’ genre in which a specific actor (with actor_id = 1) has acted. It uses a LEFT JOIN to include all entries from the actsin table (even if there’s no matching entry in movies), and filters the results based on the genre and actor ID. The SELECT * selects all columns from both tables.

```sql
SELECT *
FROM actors AS a
WHERE EXISTS
  (SELECT *
   FROM actsin AS ai
   LEFT JOIN movies AS m
   ON m.movie_id = ai.movie_id
   WHERE m.genre = 'Comedy'
   AND ai.actor_id = a.actor_id);
```

Explanation:

- This SQL query retrieves all information from the actors table for actors who have acted in at least one comedy movie. It uses an EXISTS subquery to efficiently check for the existence of a matching record in the actsin and movies tables, avoiding the need for a full join and improving performance. The subquery joins actsin (linking actors to movies) with movies to filter for movies of the ‘Comedy’ genre and matches actors based on actor_id.

```sql
SELECT a.nationality,
     COUNT(*) -- Report the nationality and the number of actors for each nationality
FROM actors AS a
WHERE EXISTS
   (SELECT ai.actor_id
    FROM actsin AS ai
    LEFT JOIN movies AS m
    ON m.movie_id = ai.movie_id
    WHERE m.genre = 'Comedy'
    AND ai.actor_id = a.actor_id)
GROUP BY a.nationality;
```

Explanation:

- This SQL query counts the number of actors of each nationality who have acted in comedy movies. It uses an EXISTS subquery to efficiently find actors who appear in at least one comedy movie, then groups the results by nationality to provide the count for each.