multiple_join_operations

```
SELECT *
FROM renting AS r
LEFT JOIN customers AS c   -- Add customer information
ON r.customer_id = c.customer_id
LEFT JOIN movies AS m   -- Add movie information
ON r.movie_id = m.movie_id;
```

Explanation:

- This SQL query retrieves all columns from the renting table and joins it with customers and movies tables to include customer and movie details in the result. It uses LEFT JOIN to ensure that all rows from the renting table are included, even if there's no matching entry in customers or movies. The joins are performed based on matching customer_id and movie_id respectively.

```
SELECT *
FROM renting AS r
LEFT JOIN customers AS c
ON c.customer_id = r.customer_id
LEFT JOIN movies AS m
ON m.movie_id = r.movie_id
WHERE c.date_of_birth BETWEEN '1970-01-01' AND '1979-12-31'; -- Select customers born in the 70s
```

Explanation:

- This SQL query retrieves all information from the renting table and joins it with the customers and movies tables using LEFT JOIN. It then filters the results to only include customers whose date of birth falls within the 1970s. The * selects all columns from all three tables. If a customer has no rentals or a movie has no rentals, the corresponding columns will have NULL values.

```
SELECT
  m.title,
  COUNT(*) AS total_views, -- Report number of views per movie
  AVG(r.rating) AS average_rating -- Report the average rating per movie
FROM
  renting AS r
LEFT JOIN
  customers AS c ON c.customer_id = r.customer_id
LEFT JOIN
  movies AS m ON m.movie_id = r.movie_id
```

```
WHERE
  c.date_of_birth BETWEEN '1970-01-01' AND '1979-12-31'
GROUP BY
  m.title;
```

Explanation:

- This SQL query calculates the total views and average rating for each movie rented by customers born between 1970 and 1979. It joins three tables (renting, customers, movies) to gather the necessary information and uses GROUP BY to aggregate results per movie title. COUNT(*) counts rentals (views), and AVG(r.rating) computes the average rating.

```
SELECT
  m.title,
  COUNT(*) AS total_rentals,
  AVG(r.rating) AS average_rating
FROM
  renting AS r
LEFT JOIN
  customers AS c ON c.customer_id = r.customer_id
LEFT JOIN
  movies AS m ON m.movie_id = r.movie_id
WHERE
  c.date_of_birth BETWEEN '1970-01-01' AND '1979-12-31'
GROUP BY
  m.title
HAVING
  COUNT(m.title) > 1
ORDER BY
  average_rating DESC;
```

Explanation:

- This SQL query analyzes movie rentals by customers born between 1970 and 1979. It joins three tables (renting, customers, movies) to aggregate data. The query counts the rentals of each movie, calculates the average rating for each movie, filters out movies rented only once, and then orders the results by average rating in descending order (highest rated first). The LEFT JOIN ensures that even movies with no rentals are included, albeit with NULL values for the rating and count. The HAVING clause filters out movies with only a single rental.

```
SELECT *
FROM renting as r
LEFT JOIN customers AS c  -- Augment table renting with information about customers
ON r.customer_id = c.customer_id
LEFT JOIN actsin AS ai -- Augment the table renting with the table actsin
```

```
ON r.movie_id = ai.movie_id
LEFT JOIN actors AS a -- Augment table renting with information about actors
ON ai.actor_id = a.actor_id;
```

Explanation:

- This SQL query performs a series of LEFT JOIN operations to combine data from four tables: renting, customers, actsin, and actors. The goal is to retrieve all information from the renting table and augment it with related data from the other tables based on matching customer_id, movie_id, and actor_id. A LEFT JOIN ensures that all rows from the renting table are included in the result, even if there's no match in the other tables. If there's no match, the columns from the joined table will have NULL values.

```
SELECT
  a.name,
  c.gender,
  COUNT(*) AS number_views,
  AVG(r.rating) AS avg_rating
FROM
  renting as r
LEFT JOIN
  customers AS c ON r.customer_id = c.customer_id
LEFT JOIN
  actsin as ai ON r.movie_id = ai.movie_id
LEFT JOIN
  actors as a ON ai.actor_id = a.actor_id
GROUP BY
  a.name, c.gender -- For each actor, separately for male and female customers
HAVING
  AVG(r.rating) IS NOT NULL AND COUNT(*) > 5 -- Report only actors with more than 5 movie
rentals
ORDER BY
  avg_rating DESC, number_views DESC;
```

Explanation:

- This SQL query calculates the average rating and the number of views for each actor, broken down by the gender of the customer who rented the movies. It uses joins to combine data from four tables (renting, customers, actsin, actors), then groups the results by actor name and customer gender. The HAVING clause filters out actors with fewer than 5 rentals or missing average ratings. Finally, it orders the results by average rating (descending) and then by number of views (descending).

```
SELECT
  a.name,
  c.gender,
  COUNT(*) AS number_views,
```

```
  AVG(r.rating) AS avg_rating
FROM
  renting as r
LEFT JOIN
  customers AS c ON r.customer_id = c.customer_id
LEFT JOIN
  actsin as ai ON r.movie_id = ai.movie_id
LEFT JOIN
  actors as a ON ai.actor_id = a.actor_id
WHERE
  c.country = 'Spain' -- Select only customers from Spain
GROUP BY
  a.name, c.gender
HAVING
  AVG(r.rating) IS NOT NULL AND COUNT(*) > 5
ORDER BY
  avg_rating DESC, number_views DESC;
```

Explanation:

- This SQL query analyzes movie rental data to find the average rating and view count for actors in movies rented by Spanish customers. It filters out actors with fewer than 5 views and those without average ratings, then orders the results by average rating and view count. The query uses multiple LEFT JOINs to combine data from different tables (renting, customers, actsin, actors).

```
SELECT *
FROM renting AS r
LEFT JOIN customers AS c
ON r.customer_id = c.customer_id
LEFT JOIN movies AS m
ON r.movie_id = m.movie_id
WHERE date_renting >= '2019-01-01';
```

Explanation:

- This SQL query retrieves all columns from the renting table and joins it with customers and movies tables to include customer and movie information in the results. It uses LEFT JOIN to ensure that all rentals are included, even if there's missing data in the customers or movies tables. The WHERE clause filters the results to show only rentals made since January 1st, 2019.

```
SELECT
  c.country,              -- For each country report
  COUNT(*) AS number_renting, -- The number of movie rentals
  AVG(r.rating) AS average_rating, -- The average rating
  SUM(m.renting_price) AS revenue      -- The revenue from movie rentals
```

```
FROM renting AS r
LEFT JOIN customers AS c
ON c.customer_id = r.customer_id
LEFT JOIN movies AS m
ON m.movie_id = r.movie_id
WHERE date_renting >= '2019-01-01'
GROUP BY c.country;
```

Explanation:

This SQL query calculates the number of movie rentals, average rating, and total revenue for each country since January 1st, 2019. It joins three tables (renting, customers, and movies) to aggregate data based on the country of the customer. The LEFT JOIN ensures that all countries are included in the results, even if they have no rentals. The WHERE clause filters the data to include only rentals from 2019 onwards. The GROUP BY clause groups the results by country.