OLAP_SQL_queries

```sql
SELECT *
FROM renting AS r
LEFT JOIN movies AS m
ON m.movie_id = r.movie_id
WHERE r.movie_id IN ( -- Select records of movies with at least 4 ratings
  SELECT movie_id
  FROM renting
  GROUP BY movie_id
  HAVING COUNT(rating) >= 4
)
AND r.date_renting >= '2018-04-01'; -- Select records of movie rentals since 2018-04-01
```

Explanation:

- This SQL query retrieves information about movie rentals. It first filters for movies that have at least 4 ratings using a subquery. Then, it joins the renting and movies tables and filters the results to include only rentals made on or after April 1st, 2018. The LEFT JOIN ensures that all rental records are included, even if a corresponding movie record is missing.

```sql
SELECT
  m.genre,
  AVG(r.rating) AS avg_rating,
  COUNT(r.rating) AS n_rating,
  COUNT(r.renting_id) AS n_rentals,
  COUNT(DISTINCT m.movie_id) AS n_movies
FROM
  renting AS r
LEFT JOIN
  movies AS m ON m.movie_id = r.movie_id
WHERE
  r.movie_id IN (
    SELECT movie_id
    FROM renting
    GROUP BY movie_id
    HAVING COUNT(rating) >= 3
  )
  AND r.date_renting >= '2018-01-01'
GROUP BY
  m.genre;
```

Explanation:

- This SQL query calculates aggregate statistics (average rating, number of ratings, rentals, and distinct movies) for each movie genre, considering only movies with at least 3 ratings and rentals after January 1st, 2018. It joins the renting and movies tables to link rental information with movie genre. The subquery ensures that only movies with 3 or more ratings are included.

```
SELECT
  genre,
  AVG(rating) AS avg_rating,
  COUNT(rating) AS n_rating,
  COUNT(*) AS n_rentals,
  COUNT(DISTINCT m.movie_id) AS n_movies
FROM
  renting AS r
LEFT JOIN
  movies AS m ON m.movie_id = r.movie_id
WHERE
  r.movie_id IN (
    SELECT movie_id
    FROM renting
    GROUP BY movie_id
    HAVING COUNT(rating) >= 3
  )
  AND r.date_renting >= '2018-01-01'
GROUP BY
  genre
ORDER BY
  avg_rating DESC; -- Order the table by decreasing average rating
```

Explanation:

- This SQL query calculates the average rating, number of ratings, number of rentals, and number of distinct movies for each genre, considering only movies with at least 3 ratings and rentals after January 1st, 2018. It joins the renting and movies tables, filters the data based on the conditions, groups the results by genre, and orders them by average rating in descending order.

```
SELECT
  a.nationality,
  a.gender,
  AVG(r.rating) AS avg_rating, -- The average rating
  COUNT(r.rating) AS n_rating, -- The number of ratings
  COUNT(*) AS n_rentals, -- The number of movie rentals
  COUNT(DISTINCT a.name) AS n_actors -- The number of actors
FROM
  renting AS r
LEFT JOIN
```

```
  actsin AS ai ON ai.movie_id = r.movie_id
LEFT JOIN
  actors AS a ON ai.actor_id = a.actor_id
WHERE
  r.movie_id IN (
    SELECT movie_id
    FROM renting
    GROUP BY movie_id
    HAVING COUNT(rating) >= 4
  )
  AND r.date_renting >= '2018-04-01'
GROUP BY
  a.nationality, a.gender; -- Report results for each combination of the actors' nationality and gender
```

Explanation:

- This SQL query calculates statistics on movie rentals, focusing on actors' nationalities and genders. It filters rentals after April 1st, 2018, and considers only movies with at least 4 ratings. The results are grouped by actor nationality and gender, showing the average rating, number of ratings, total rentals, and the number of unique actors for each group. The query uses joins to combine data from three tables: renting, actsin (linking actors to movies), and actors.

```
SELECT a.nationality,
    a.gender,
    AVG(r.rating) AS avg_rating,
    COUNT(r.rating) AS n_rating,
    COUNT(*) AS n_rentals,
    COUNT(DISTINCT a.actor_id) AS n_actors
FROM renting AS r
LEFT JOIN actsin AS ai
ON ai.movie_id = r.movie_id
LEFT JOIN actors AS a
ON ai.actor_id = a.actor_id
WHERE r.movie_id IN (
  SELECT movie_id
  FROM renting
  GROUP BY movie_id
  HAVING COUNT(rating) >= 4)
AND r.date_renting >= '2018-04-01'
GROUP BY CUBE (a.nationality, a.gender); -- Provide results for all aggregation levels represented in a pivot table
```

Explanation:

- This SQL query analyzes movie rental data to calculate average ratings, the number of ratings, rentals, and distinct actors for different combinations of actor

nationalities and genders. It filters for movies with at least 4 ratings and rentals after April 1st, 2018. The CUBE function generates aggregations at all levels (nationality, gender, and their combinations, including a grand total). The joins connect rental data with actor information.