basic_nested_queries

```sql
SELECT movie_id -- Select movie IDs with more than 5 views
FROM renting
GROUP BY movie_id
HAVING COUNT(*) > 5;
```

Explanation:

- This SQL query retrieves the IDs of movies that have been rented more than 5 times. It groups the renting table by movie_id and then uses the HAVING clause to filter out groups (movies) with less than or equal to 5 rentals.

```sql
SELECT *
FROM movies
WHERE movie_id IN -- Select movie IDs from the inner query
  (SELECT movie_id
  FROM renting
  GROUP BY movie_id
  HAVING COUNT(*) > 5);
```

Explanation:

- This SQL query retrieves all information from the movies table for movies that have been rented more than 5 times. It does this by using a subquery to find the movie_ids that appear more than 5 times in the renting table and then selecting the corresponding rows from the movies table. The IN operator checks if a movie_id from the movies table exists in the result set of the subquery.

```sql
SELECT *
FROM customers
WHERE customer_id IN -- Select all customers with more than 10 movie rentals
  (SELECT customer_id
  FROM renting
  GROUP BY customer_id
  HAVING COUNT(*) > 10);
```

Explanation:

- This SQL query retrieves all customer information from the customers table where the customer_id exists in a subquery. The subquery identifies customer_ids from the renting table that have more than 10 rental entries (meaning those customers rented more than 10 movies). It uses GROUP BY and HAVING to filter the results of the subquery. The main query then uses IN to select only those customers found in the subquery's result set.

```
SELECT movie_id, -- Select movie IDs and calculate the average rating
    AVG(rating) AS average_rating
FROM renting
GROUP BY movie_id
HAVING AVG(rating) >        -- Of movies with rating above average
  (SELECT AVG(rating)
  FROM renting);
```

Explanation:

This SQL query finds movies with an average rating higher than the overall average rating of all movies. It does this by:

1. **Grouping:** Grouping the renting table by movie_id to calculate the average rating for each movie.
2. **Averaging:** Using AVG(rating) to compute the average rating for each movie.
3. **Filtering:** Using a HAVING clause to filter out movies whose average rating is not greater than the overall average rating (calculated in the subquery).
4. **Subquery:** A subquery is used to calculate the overall average rating of all movies.
5. **Selecting:** Selecting the movie_id and its average_rating. I've added AS average_rating for better readability.

In essence, it identifies movies that are rated better than the average.

```
SELECT title -- Report the movie titles of all movies with average rating higher than the total average
FROM movies
WHERE movie_id IN
  (SELECT movie_id
   FROM renting
   GROUP BY movie_id
   HAVING AVG(rating) >
     (SELECT AVG(rating)
      FROM renting));
```

Explanation:

- This SQL query retrieves the titles of movies whose average rating is higher than the overall average rating of all movies. It uses subqueries to first calculate the average rating for each movie and then compare it to the global average rating. The inner subquery finds the average rating for each movie, and the outer subquery filters to only include movies whose average exceeds the global average.