# basic_aggregation_operations

```sql
SELECT
  country,  -- Selects the country column
  MIN(date_account_start) AS first_account -- Calculates the minimum (earliest) date_account_start for each country and names it first_account
FROM
  customers  -- Specifies the table to query
GROUP BY
  country  -- Groups the results by country so MIN() operates within each country
ORDER BY
  first_account ASC; -- Orders the results in ascending order based on the earliest account creation date
```

**Explanation:**

- This SQL query finds the earliest account creation date (date_account_start) for each country in the customers table. It uses the MIN() aggregate function to find the minimum date within each country group and the GROUP BY clause to group the data by country. The results are then ordered by the earliest account creation date (first_account) in ascending order.

```sql
SELECT movie_id,
    AVG(rating) AS average_rating -- Calculate average rating per movie and name the column 'average_rating'
FROM renting
GROUP BY movie_id;
```

**Explanation:**

- This SQL query calculates the average rating for each movie. It selects the movie_id and computes the average of the rating column using the AVG() aggregate function. The GROUP BY clause groups the rows by movie_id, ensuring that the average rating is calculated separately for each movie. The AS average_rating renames the resulting average column for clarity.

```sql
SELECT
  movie_id,
  AVG(rating) AS avg_rating, -- Average rating for each movie
  COUNT(rating) AS number_rating,          -- Number of ratings for each movie
  COUNT(movie_id) AS number_renting          -- Number of times each movie was rented
FROM
  renting
```

```
GROUP BY
  movie_id;
```

**Explanation:**

- This SQL query calculates the average rating, total number of ratings, and total number of rentals for each movie from a table named renting. It uses aggregate functions (AVG, COUNT) and the GROUP BY clause to group the results by movie_id. The AS keyword creates aliases for the calculated columns, making the output more readable.

```
SELECT
  movie_id,
  AVG(rating) AS avg_rating,
  COUNT(rating) AS number_ratings,
  COUNT(*) AS number_renting
FROM
  renting
GROUP BY
  movie_id
ORDER BY
  avg_rating DESC; -- Order by average rating in decreasing order
```

**Explanation:**

- This SQL query calculates the average rating and the number of ratings for each movie in a table named renting. It then orders the results by average rating in descending order, showing the highest-rated movies first. COUNT(*) counts all rows for each movie (including those without ratings), while COUNT(rating) only counts rows with non-null ratings.

```
SELECT
  customer_id,  -- Report the customer_id
  AVG(rating),   -- Report the average rating per customer
  COUNT(rating),  -- Report the number of ratings per customer
  COUNT(movie_id) -- Report the number of movie rentals per customer
FROM
  renting
GROUP BY
  customer_id
HAVING
  COUNT(movie_id) > 7 -- Select only customers with more than 7 movie rentals
ORDER BY
  AVG(rating) ASC; -- Order by the average rating in ascending order
```

**Explanation:**

- This SQL query analyzes a renting table to find customers who have rented more than 7 movies. It then calculates and reports their average rating, the total number of ratings they've given, and the total number of movies they've rented. The results are ordered by average rating in ascending order (lowest average rating first).