data_analysis_basic_operations

```sql
SELECT *
FROM renting
WHERE date_renting = '2018-10-09';
```

Explanation:

- This SQL query selects all columns (SELECT *) from the renting table where the date_renting column is equal to '2018-10-09'. It retrieves all records related to rentals on that specific date.

```sql
SELECT *
FROM renting
WHERE date_renting BETWEEN '2018-04-01' AND '2018-08-31';
```

Explanation:

- This SQL query selects all columns (*) from the renting table where the date_renting column falls within the range of April 1st, 2018, and August 31st, 2018 (inclusive). It's a simple date range query.

```sql
SELECT *
FROM renting
WHERE date_renting BETWEEN '2018-04-01' AND '2018-08-31'
ORDER BY date_renting DESC;
```

Explanation:

- This SQL query selects all columns (*) from the renting table. It filters the results to include only rows where the date_renting column falls between April 1st, 2018, and August 31st, 2018 (inclusive). Finally, it sorts the results in descending order based on the date_renting column (most recent dates first).

```sql
SELECT *
FROM movies
WHERE genre <> 'Drama';
```

Explanation:

- This SQL query selects all columns (SELECT *) from the movies table where the genre is not equal to 'Drama' (WHERE genre <> 'Drama'). The <> operator means "not equal to".

```sql
SELECT *
FROM movies
WHERE title IN ('Showtime','Love Actually','The Fighter');
```

Explanation:

- This SQL query selects all columns (SELECT *) from the movies table where the title column is equal to one of the values in the list: 'Showtime', 'Love Actually', or 'The Fighter'. The IN operator makes this more concise than using multiple OR conditions.

```sql
SELECT *
FROM movies
ORDER BY renting_price ASC;
```

Explanation:

- This SQL query selects all columns (*) from the movies table and orders the results in ascending order (ASC) based on the renting_price column. This means the movies with the lowest rental prices will appear first in the result set.

```sql
SELECT *
FROM renting
WHERE date_renting BETWEEN '2018-01-01' AND '2018-12-31' -- Renting in 2018
AND rating IS NOT NULL; -- Rating exists
```

Explanation:

- This SQL query selects all columns (SELECT *) from the renting table where the date_renting falls within the year 2018 and a rating is provided (i.e., rating is not NULL). It filters the results to only include rentals with complete rating information from the specified year.

```sql
SELECT COUNT(name) -- Count the total number of customers
FROM customers
WHERE date_of_birth BETWEEN '1980-01-01' AND '1989-12-31'; -- Select customers born between 1980-01-01 and 1989-12-31
```

Explanation:

- This SQL query counts the number of customers in the customers table who were born between January 1, 1980, and December 31, 1989. It uses the COUNT() function to count the number of names and the BETWEEN operator to filter the results based on the date_of_birth column.

```sql
SELECT COUNT(name)  -- Count the total number of customers
FROM customers
WHERE country = 'Germany'; -- Select all customers from Germany
```

Explanation:

- This SQL query counts the number of customers from Germany in the customers table. It uses the COUNT() function to count the number of names (representing customers) and the WHERE clause to filter for customers where the country is 'Germany'.

```sql
SELECT COUNT(DISTINCT(country))  -- Count the number of unique countries
FROM customers;
```

Explanation:

- This SQL query counts the number of unique countries listed in the customers table. COUNT(DISTINCT(country)) ensures that each country is counted only once, even if it appears multiple times in the table.

```sql
SELECT
  MIN(rating) AS min_rating, -- Calculate the minimum rating and use alias min_rating
  MAX(rating) AS max_rating, -- Calculate the maximum rating and use alias max_rating
  AVG(rating) AS avg_rating, -- Calculate the average rating and use alias avg_rating
  COUNT(rating) AS number_ratings -- Count the number of ratings and use alias number_ratings
FROM
  renting
WHERE
  movie_id = '25'; -- Select all records of the movie with ID 25
```

Explanation:

- This SQL query calculates the minimum, maximum, average rating, and the total number of ratings for movie with ID 25 from the renting table. The results are aliased for better readability.

```sql
SELECT * -- Select all records of movie rentals since January 1st 2019
FROM renting
WHERE date_renting >= '2019-01-01';
```

Explanation:

- This SQL query retrieves all columns (SELECT *) from the renting table where the date_renting column is greater than or equal to January 1st, 2019. This effectively filters the results to show only movie rentals from that date onwards.

```sql
SELECT
  COUNT(renting_id) AS TotalRentedMovies, -- Count the total number of rented movies
  AVG(rating) AS AverageRating -- Add the average rating
FROM renting
WHERE date_renting >= '2019-01-01';
```

Explanation:

- This SQL query calculates the total number of movies rented and the average rating of those movies from the 'renting' table, considering only rentals made from January 1st, 2019, onwards. COUNT(renting_id) counts the number of rentals, and AVG(rating) calculates the average rating. The WHERE clause filters the data to include only rentals after the specified date. AS TotalRentedMovies and AS AverageRating provide more descriptive names for the resulting columns.

```sql
SELECT
  COUNT(*) AS number_renting, -- Give it the column name number_renting
  AVG(rating) AS average_rating  -- Give it the column name average_rating
FROM renting
WHERE date_renting >= '2019-01-01';
```

Explanation:

- This SQL query calculates the total number of rentals and the average rating for rentals made on or after January 1st, 2019. It selects from a table named renting and filters the results using a WHERE clause. COUNT(*) counts all rows meeting the condition, and AVG(rating) calculates the average of the rating column. The results are aliased as number_renting and average_rating respectively.

```sql
SELECT
  COUNT(*) AS number_renting,
  AVG(rating) AS average_rating,
  COUNT(rating) AS number_ratings -- Add the total number of ratings here.
FROM renting
WHERE date_renting >= '2019-01-01';
```

Explanation:

This SQL query calculates the total number of rentals, the average rating of those rentals, and the total number of ratings received since January 1st, 2019 from the renting table. COUNT(*) counts all rows, AVG(rating) calculates the average of the rating column, and COUNT(rating) counts non-NULL values in the rating column. The WHERE clause filters the data to include only rentals on or after 2019-01-01.