basic_join_operations

```sql
SELECT * -- Join renting with customers
FROM renting AS r
LEFT JOIN customers AS c
ON c.customer_id = r.customer_id;
```

Explanation:

- This SQL query performs a LEFT JOIN between two tables: renting and customers. It retrieves all columns (SELECT *) from both tables. The join condition (ON c.customer_id = r.customer_id) links rows where the customer_id matches in both tables. A LEFT JOIN ensures that all rows from the renting table are included in the result, even if there's no matching row in the customers table (in which case, the columns from customers will have NULL values).

```sql
SELECT *
FROM renting AS r
LEFT JOIN customers AS c
ON r.customer_id = c.customer_id
WHERE c.country = 'Belgium'; -- Select only records from customers coming from Belgium
```

Explanation:

- This SQL query performs a LEFT JOIN operation between the renting and customers tables. It retrieves all columns from both tables, matching rows where the customer_id in both tables are equal. The WHERE clause filters the results to include only customers from Belgium. A LEFT JOIN ensures that all rows from the renting table are included in the result, even if there's no matching row in the customers table (in those cases, customer-related columns will have NULL values).

```sql
SELECT AVG(r.rating) -- Average ratings of customers from Belgium
FROM renting AS r
LEFT JOIN customers AS c
ON r.customer_id = c.customer_id
WHERE c.country='Belgium';
```

Explanation:

- This SQL query calculates the average rating of customers from Belgium. It joins the renting table (containing ratings) with the customers table (containing customer information) using a LEFT JOIN to include all renters, even those without a matching entry in the customers table. The WHERE clause filters the results to include only

customers from Belgium, and the AVG() function computes the average rating for this subset.

```
SELECT *
FROM renting AS r
LEFT JOIN movies AS m -- Choose the correct join statment
ON r.movie_id = m.movie_id;
```

Explanation:

- This SQL query performs a LEFT JOIN between two tables: renting and movies. It retrieves all columns from both tables. The join condition r.movie_id = m.movie_id links rows where the movie_id column matches in both tables. A LEFT JOIN ensures that all rows from the renting table are included in the result, even if there's no matching row in the movies table (in which case, the columns from the movies table will have NULL values).

```
SELECT
  SUM(m.renting_price) AS total_revenue, -- Get the revenue from movie rentals
  COUNT(*) AS total_rentals, -- Count the number of rentals
  COUNT(DISTINCT(r.customer_id)) AS unique_customers  -- Count the number of unique customers
FROM renting AS r
LEFT JOIN movies AS m
ON r.movie_id = m.movie_id;
```

Explanation:

- This SQL query calculates the total revenue from movie rentals, the total number of rentals, and the number of unique customers who rented movies. It does this by joining the renting and movies tables and using aggregate functions (SUM, COUNT). A LEFT JOIN ensures that all rentals are included, even if a movie's information is missing from the movies table. The AS keyword assigns aliases to the calculated columns for better readability.

```
SELECT
  SUM(m.renting_price),
  COUNT(*),
  COUNT(DISTINCT r.customer_id)
FROM renting AS r
LEFT JOIN movies AS m
ON r.movie_id = m.movie_id
-- Only look at movie rentals in 2018
WHERE date_renting BETWEEN '2018-01-01' AND '2018-12-31'
```

Explanation:

- This SQL query calculates the total renting price, the total number of rentals, and the total number of unique customers who rented movies in 2018. It joins the renting and movies tables to access both renting information and movie prices. The WHERE clause filters the results to include only rentals within the year 2018.

```
SELECT DISTINCT a.name, -- Create a list of movie titles and actor names
    m.title
FROM actsin AS ai
LEFT JOIN movies AS m
ON m.movie_id = ai.movie_id
LEFT JOIN actors AS a
ON a.actor_id = ai.actor_id;
```

Explanation:

- This SQL query retrieves a list of unique movie titles and the names of actors who appeared in those movies. It uses three tables: actsin (linking actors to movies), movies, and actors. The LEFT JOIN ensures that all movies are included, even if an actor's name is missing from the actsin table for that movie. DISTINCT removes duplicate rows, ensuring each movie-actor pair is listed only once.