

Rockbuster Analysis:

SQL query documentation

Index

- **Slides 1&2: Maps of Customer Distribution**
 - granular_rental_table_2
- **Slide 2: Histograms of Customer Distribution**
 - district_histogram
 - Script Note
 - country_histogram
 - Correction (subquery → CTE)
- **Slide 3: Scatterplots of Avg. Rental Frequency & Revenue Per Rental**
 - country_engagement_TABLE
 - country_analysis
 - country_engagement_analysis
 - Correction
 - stddev_absolutes
- **Slide 4: Growth % Calculator**
- **Slide 5: Bar Graph - Genre Popularity (normalized by Inventory)**
 - genre_popularity_norm
- **Slide 5: Heatmap - Outlier Countries Genre Analysis**
 - avg_rentals_outliers
 - Correction
 - genre_popularity_country
- **Slide 6: Scatterplot & Box-Whisker Graph - Tenure Length Analysis**
 - customer_engagement_district
 - Correction
- **Slide 7: Pie Chart - Revenue Breakdown**
 - late_charge_noneg
 - CASE function explanation
- **Slide 7: Rental Rate Mode Graph**
 - country_engagement_TABLE

Slides 1 & 2: Maps of Customer Distribution

granular_rental_table_2

This table records individual rental transactions for a single physical copy of a movie at a time. It captures customer details (name, unique rental ID, city, district, and country), rental specifics (rental and return dates, amount paid, rental rate, rental period), and movie attributes (movie title, release year, rating, length, and genre). Each row represents a unique transaction involving a customer renting a specific movie.

```
SELECT r.rental_id,
       p.payment_id,
       c.customer_id,
       c.first_name,
       c.last_name,
       b.city,
       a.district,
       d.country,
       p.amount AS payment_amount,
       r.rental_date,
       r.return_date,
       f.title,
       f.release_year,
       f.rental_duration,
       f.rental_rate,
       f.length,
       f.rating,
       l.name AS language,
       g.name AS genre
FROM rental r
     LEFT JOIN payment p ON r.rental_id = p.rental_id
     LEFT JOIN customer c ON p.customer_id = c.customer_id
     LEFT JOIN address a ON c.address_id = a.address_id
     LEFT JOIN city b ON a.city_id = b.city_id
     LEFT JOIN country d ON b.country_id = d.country_id
     LEFT JOIN inventory i ON r.inventory_id = i.inventory_id
     LEFT JOIN film f ON i.film_id = f.film_id
     LEFT JOIN language l ON f.language_id = l.language_id
     JOIN film_category fc ON f.film_id = fc.film_id
     LEFT JOIN category g ON fc.category_id = g.category_id;
```

| | rental_id integer | payment_id integer | customer_id integer | first_name character varying (45) | last_name character varying (45) | city character varying (50) | district character varying (20) | country character varying (20) |
|---|----------------------|-----------------------|------------------------|--------------------------------------|-------------------------------------|--------------------------------|------------------------------------|-----------------------------------|
| 1 | 1520 | 17503 | 341 | Peter | Menard | Ede | Gelderland | Netherlands |
| 2 | 1778 | 17504 | 341 | Peter | Menard | Ede | Gelderland | Netherlands |
| 3 | 1849 | 17505 | 341 | Peter | Menard | Ede | Gelderland | Netherlands |
| 4 | 2829 | 17506 | 341 | Peter | Menard | Ede | Gelderland | Netherlands |
| 5 | 3130 | 17507 | 341 | Peter | Menard | Ede | Gelderland | Netherlands |
| 6 | 3382 | 17508 | 341 | Peter | Menard | Ede | Gelderland | Netherlands |
| 7 | 2190 | 17509 | 342 | Harold | Martino | Boa Vista | Roraima | Brazil |

Total rows: 16048 Query complete 00:00:00.543

LF Ln 1, Col 38

| | payment_amount numeric (5,2) | rental_date timestamp without time zone | return_date timestamp without time zone | title character varying (255) | release_year integer | rental_duration smallint |
|---|---------------------------------|--|--|----------------------------------|-------------------------|-----------------------------|
| 1 | 7.99 | 2005-06-15 23:57:20 | 2005-06-24 23:46:20 | Rules Human | 2006 | 6 |
| 2 | 1.99 | 2005-06-16 18:54:48 | 2005-06-22 16:08:48 | Majestic Floats | 2006 | 5 |
| 3 | 7.99 | 2005-06-17 00:13:19 | 2005-06-23 18:37:19 | Maiden Home | 2006 | 3 |
| 4 | 2.99 | 2005-06-19 21:11:30 | 2005-06-24 18:10:30 | Hyde Doctor | 2006 | 5 |
| 5 | 7.99 | 2005-06-20 19:03:22 | 2005-06-29 18:05:22 | Massacre Usual | 2006 | 6 |
| 6 | 5.99 | 2005-06-21 14:05:23 | 2005-06-29 19:13:23 | Annie Identity | 2006 | 3 |
| 7 | 5.99 | 2005-06-18 01:29:51 | 2005-06-26 04:31:51 | Wash Heavenly | 2006 | 7 |

Total rows: 16048 Query complete 00:00:00.543

LF

| rental_rate numeric (4,2) | length smallint | rating mpaa_rating | language character (20) | genre character varying (25) |
|------------------------------|--------------------|-----------------------|----------------------------|---------------------------------|
| 4.99 | 153 | R | English | Horror |
| 0.99 | 130 | PG | English | Documentary |
| 4.99 | 138 | PG | English | New |
| 2.99 | 100 | G | English | Classics |
| 4.99 | 165 | R | English | Games |
| 0.99 | 86 | G | English | Sci-Fi |
| 4.99 | 161 | R | English | Foreign |

LF Ln 1, Col 38

Slide 2: Histograms of Customer Distribution

district_histogram

This table records the count of districts, grouped by the number of customers in each district, ranging from 1 to 10 customers.

```

SELECT subquery.customer_population_size,
       count(DISTINCT subquery.district) AS number_of_districts
FROM ( SELECT granular_rental_table_2.district,
              count(DISTINCT granular_rental_table_2.customer_id) AS
customer_population_size
      FROM granular_rental_table_2
      GROUP BY granular_rental_table_2.district) subquery
GROUP BY subquery.customer_population_size
HAVING subquery.customer_population_size > 0
ORDER BY subquery.customer_population_size;

```

| | customer_population_size bigint | number_of_districts bigint |
|----------------|------------------------------------|-------------------------------|
| 1 | 1 | 273 |
| 2 | 2 | 52 |
| 3 | 3 | 25 |
| 4 | 4 | 11 |
| 5 | 5 | 6 |
| 6 | 6 | 1 |
| 7 | 7 | 2 |
| | | |
| Total rows: 10 | | Query complete 00:00:00.153 |

Script Note

The subquery isolates the logic of grouping by **district** and calculating customer populations. The outer query **then** analyzes these results (grouping them by population size and counting districts). (**Subquery runs first.**)

country_histogram

This table records the count of countries, grouped by the number of customers in each country, organized into intervals of 10 (e.g., 1-10, 11-20, up to 51-60).

/* Main Select Query: CASE statement creates population size bins. At the end, it counts the distinct number of countries sorted into each bin.*/

```
SELECT
    CASE
        WHEN subquery.customer_population_size >= 1 AND
subquery.customer_population_size <= 10 THEN '1-10'::text
        WHEN subquery.customer_population_size >= 11 AND
subquery.customer_population_size <= 20 THEN '11-20'::text
        WHEN subquery.customer_population_size >= 21 AND
subquery.customer_population_size <= 30 THEN '21-30'::text
        WHEN subquery.customer_population_size >= 31 AND
subquery.customer_population_size <= 40 THEN '31-40'::text
        WHEN subquery.customer_population_size >= 41 AND
subquery.customer_population_size <= 50 THEN '41-50'::text
        WHEN subquery.customer_population_size >= 51 AND
subquery.customer_population_size <= 60 THEN '51-60'::text
        ELSE NULL::text
    END AS population_bin,
    count(DISTINCT subquery.country) AS number_of_countries
```

/*SUBQUERY: The subquery is nested within the main query. It gathers the necessary data from the `granular_rental_table_2`, where each row contains a rental transaction with customer and country information. The subquery does two main things: 1) It groups by country. 2) It counts the distinct number of customers (represented as `customer_id`) for each country.)

```
FROM ( SELECT granular_rental_table_2.country,
            count(DISTINCT granular_rental_table_2.customer_id) AS
customer_population_size
      FROM granular_rental_table_2
      GROUP BY granular_rental_table_2.country -- Groups by country.
      HAVING count(DISTINCT granular_rental_table_2.customer_id) > 0) subquery
-- Counts up the distinct number of customers.
```

/*GROUPING - part of main query*/

```
GROUP BY (
  CASE
    WHEN subquery.customer_population_size >= 1 AND
subquery.customer_population_size <= 10 THEN '1-10'::text
    WHEN subquery.customer_population_size >= 11 AND
subquery.customer_population_size <= 20 THEN '11-20'::text
    WHEN subquery.customer_population_size >= 21 AND
subquery.customer_population_size <= 30 THEN '21-30'::text
    WHEN subquery.customer_population_size >= 31 AND
subquery.customer_population_size <= 40 THEN '31-40'::text
    WHEN subquery.customer_population_size >= 41 AND
subquery.customer_population_size <= 50 THEN '41-50'::text
    WHEN subquery.customer_population_size >= 51 AND
subquery.customer_population_size <= 60 THEN '51-60'::text
    ELSE NULL::text
  END)
```

/*ORDERING - part of main query*/

```
ORDER BY (
  CASE
    WHEN subquery.customer_population_size >= 1 AND
subquery.customer_population_size <= 10 THEN '1-10'::text
    WHEN subquery.customer_population_size >= 11 AND
subquery.customer_population_size <= 20 THEN '11-20'::text
    WHEN subquery.customer_population_size >= 21 AND
subquery.customer_population_size <= 30 THEN '21-30'::text
    WHEN subquery.customer_population_size >= 31 AND
subquery.customer_population_size <= 40 THEN '31-40'::text
    WHEN subquery.customer_population_size >= 41 AND
subquery.customer_population_size <= 50 THEN '41-50'::text
    WHEN subquery.customer_population_size >= 51 AND
subquery.customer_population_size <= 60 THEN '51-60'::text
    ELSE NULL::text
  END);
```

| | population_bin text | number_of_countries bigint |
|---|------------------------|-------------------------------|
| 1 | 1-10 | 95 |
| 2 | 11-20 | 6 |
| 3 | 21-30 | 3 |
| 4 | 31-40 | 2 |
| 5 | 51-60 | 2 |

Correction

CTE version - would have removed inefficiency of repeating the CASE statement in multiple places.

```

/* Subquery/CTE: Generate customer population size for each country */
WITH customer_population AS (
  SELECT
    country,
    COUNT(DISTINCT customer_id) AS customer_population_size
  FROM granular_rental_table_2
  GROUP BY country
  HAVING COUNT(DISTINCT customer_id) > 0
),
/* CTE: Create bins for customer population size */
binned_data AS (
  SELECT
    country,
    CASE
      WHEN customer_population_size BETWEEN 1 AND 10 THEN '1-10'
      WHEN customer_population_size BETWEEN 11 AND 20 THEN '11-20'
      WHEN customer_population_size BETWEEN 21 AND 30 THEN '21-30'
      WHEN customer_population_size BETWEEN 31 AND 40 THEN '31-40'
      WHEN customer_population_size BETWEEN 41 AND 50 THEN '41-50'
      WHEN customer_population_size BETWEEN 51 AND 60 THEN '51-60'
      ELSE NULL
    END AS population_bin
  FROM customer_population
)
/* Main Query: Count distinct countries in each bin */
SELECT
  population_bin,
  COUNT(DISTINCT country) AS number_of_countries
FROM binned_data
GROUP BY population_bin
ORDER BY population_bin;

```

Slide 3: Scatterplots of Avg Rental Frequency & Revenue per Rental

country_engagement_TABLE

* A combination of two views: country_analysis and country_engagement_analysis;

This table aggregates performance and customer behavior by country, including the number of customers, total rentals, total revenue, and key metrics such as average revenue per customer, average revenue per rental, average monthly spending (normalized by the average tenure of customers in the country), average rentals per month (also normalized), the most popular rental rate, and average customer tenure length.

```
SELECT e.country,
       e.number_of_customers,
       c.total_rentals,
       c.total_revenue,
       c.avg_revenue_per_customer,
       e.avg_revenue_per_rental,
       e.avg_spending_per_month_per_customer,
       e.avg_rentals_per_month_per_customer,
       e.mode_rental_rate,
       e.avg_active_days_per_country
FROM country_engagement_analysis e
     LEFT JOIN country_analysis c ON e.country::text = c.country::text;
```

| | country character varying (50) | number_of_customers bigint | total_rentals bigint | total_revenue numeric | avg_revenue_per_customer numeric | avg_revenue_per_rental numeric | avg_spending_per_month_per_customer double precision |
|--|-----------------------------------|-------------------------------|-------------------------|--------------------------|-------------------------------------|-----------------------------------|---|
| 1 | India | 60 | 1572 | 6032.79 | 100.546500000000000000 | 3.8376526717557252 | |
| 2 | China | 53 | 1426 | 5247.04 | 99.0007547169811321 | 3.6795511921458626 | |
| 3 | United States | 36 | 972 | 3694.27 | 102.618611111111111111 | 3.8006893004115226 | |
| 4 | Japan | 31 | 825 | 3121.52 | 100.6941935483870968 | 3.7836606060606061 | |
| 5 | Mexico | 30 | 796 | 2984.82 | 99.494000000000000000 | 3.7497738693467337 | |
| 6 | Brazil | 28 | 748 | 2919.19 | 104.2567857142857143 | 3.9026604278074866 | |
| 7 | Russian Federation | 28 | 713 | 2765.62 | 98.7721428571428571 | 3.8788499298737728 | |
| Total rows: 108 Query complete 00:00:00.826 LF Ln 1, Col 39 | | | | | | | |

| avg_spending_per_month_per_customer double precision | avg_rentals_per_month_per_customer double precision | mode_rental_rate numeric (4,2) | avg_active_days_per_country double precision |
|---|--|-----------------------------------|---|
| 29.879040390637524 | 7.671215630329574 | 0.99 | 141.55089058524172 |
| 30.061761196001157 | 8.283221498331088 | 0.99 | 128.7328190743338 |
| 31.590523569042645 | 8.106391990297544 | 0.99 | 132.2314814814815 |
| 31.001195437096023 | 8.207881563575139 | 4.99 | 125.38181818181818 |
| 31.347137224272206 | 8.32435820333583 | 0.99 | 121.8605527638191 |
| 34.162443448782895 | 8.679666732168924 | 0.99 | 117.69251336898395 |
| 33.553489215424214 | 8.526465790665224 | 4.99 | 111.44039270687237 |
| -----> 00:00:00.826 LF Ln 1, Col 39 | | | |

country_analysis

*The earlier version of the table included an 'avg_rentals_per_month' column that did not normalize rental frequency properly. It calculated rental frequency based on a country-level tenure length—defined as the difference in days between the most recent and earliest rental in a country—which inaccurately assumes that a country itself can have a tenure. A subsequent version of the table corrected this by including a column for the average tenure length of customers in each country, which was then used to calculate a more accurate 'avg_rental_frequency_per_month_per_customer.' However, this updated table (country_engagement_analysis) unintentionally over-aggregated total rentals, total revenue, throwing off the calculation for avg_revenue_per_customer, necessitating a join with the earlier version to preserve accurate aggregated metrics.

```

/* First CTE */
WITH rental_data AS (
    SELECT d_1.country,
           f_1.rental_rate
    FROM rental r_1
         LEFT JOIN inventory i_1 ON r_1.inventory_id = i_1.inventory_id
         LEFT JOIN film f_1 ON i_1.film_id = f_1.film_id
         LEFT JOIN customer c_1 ON r_1.customer_id = c_1.customer_id
         LEFT JOIN address a_1 ON c_1.address_id = a_1.address_id
         LEFT JOIN city b_1 ON a_1.city_id = b_1.city_id
         LEFT JOIN country d_1 ON b_1.country_id = d_1.country_id
),
/*Second CTE (within WITH clause)*/
customer_spending_per_month AS (
    SELECT c_1.customer_id,
           d_1.country,
           /*Calculate Total Spending per customer: COALESCE ensures that Nulls are
            treated as 0*/
           COALESCE(sum(p_1.amount), 0)::numeric)::double precision /
           /*Calculate Rental Duration in Months*/
           (date_part('day'::text, max(r_1.rental_date) - min(r_1.rental_date)) / 30.44)::double
           precision) AS spending_per_month
    FROM rental r_1
         LEFT JOIN customer c_1 ON r_1.customer_id = c_1.customer_id
         LEFT JOIN address a_1 ON c_1.address_id = a_1.address_id
         LEFT JOIN city b_1 ON a_1.city_id = b_1.city_id
         LEFT JOIN country d_1 ON b_1.country_id = d_1.country_id
         LEFT JOIN payment p_1 ON r_1.rental_id = p_1.rental_id
    GROUP BY c_1.customer_id, d_1.country
)
/*MAIN QUERY*/
SELECT d.country,
       count(DISTINCT c.customer_id) AS number_of_customers,
       count(r.rental_id) AS total_rentals,
       COALESCE(sum(p.amount), 0)::numeric AS total_revenue,
       COALESCE(sum(p.amount), 0)::numeric / count(DISTINCT c.customer_id)::numeric AS
avg_revenue_per_customer,
       /*These two CASE statements are calculating different metrics for the rentals
and their associated revenue; conditional logic statement, based on whether rentals
have been made.*/
       CASE

```



```

        WHEN count(r.rental_id) > 0 THEN COALESCE(sum(p.amount), 0::numeric) /
count(r.rental_id)::numeric
        ELSE 0::numeric
    END AS avg_revenue_per_rental,
    CASE
        WHEN count(r.rental_id) > 0 THEN count(r.rental_id)::double precision /
(date_part('day'::text, max(r.rental_date) - min(r.rental_date)) / 30.44::double
precision)
        ELSE 0::double precision
    END AS avg_rentals_per_month,

/*Refers to CTE "customer_spending_per_month"*/
    COALESCE(avg(csm.spending_per_month), 0::double precision) AS
avg_spending_per_month_per_customer,
/*This is a scalar subquery that is used to determine the mode (the most frequent
value) of the rental_rate for each country; refers to CTE "rental_data"*/
    ( SELECT rd.rental_rate
      FROM rental_data rd
     WHERE rd.country::text = d.country::text
     GROUP BY rd.rental_rate
     ORDER BY (count(*)) DESC, rd.rental_rate
     LIMIT 1) AS mode_rental_rate

FROM rental r
LEFT JOIN customer c ON r.customer_id = c.customer_id
LEFT JOIN address a ON c.address_id = a.address_id
LEFT JOIN city b ON a.city_id = b.city_id
LEFT JOIN country d ON b.country_id = d.country_id
LEFT JOIN payment p ON r.rental_id = p.rental_id
LEFT JOIN inventory i ON r.inventory_id = i.inventory_id
LEFT JOIN film f ON i.film_id = f.film_id
LEFT JOIN customer_spending_per_month csm ON csm.customer_id = c.customer_id
GROUP BY d.country
ORDER BY (count(DISTINCT c.customer_id)) DESC;

```

country_engagement_analysis

```

WITH rental_data AS (
    SELECT d_1.country,
           f_1.rental_rate
    FROM rental r_1
    LEFT JOIN inventory i_1 ON r_1.inventory_id = i_1.inventory_id
    LEFT JOIN film f_1 ON i_1.film_id = f_1.film_id
    LEFT JOIN customer c_1 ON r_1.customer_id = c_1.customer_id
    LEFT JOIN address a_1 ON c_1.address_id = a_1.address_id
    LEFT JOIN city b_1 ON a_1.city_id = b_1.city_id
    LEFT JOIN country d_1 ON b_1.country_id = d_1.country_id
), customer_spending_per_month AS (
    SELECT c_1.customer_id,
           d_1.country,
           COALESCE(sum(p_1.amount), 0::numeric)::double precision /
(date_part('day'::text, max(r_1.rental_date) - min(r_1.rental_date)) / 30.44::double
precision) AS spending_per_month

```

```

FROM rental r_1
  LEFT JOIN customer c_1 ON r_1.customer_id = c_1.customer_id
  LEFT JOIN address a_1 ON c_1.address_id = a_1.address_id
  LEFT JOIN city b_1 ON a_1.city_id = b_1.city_id
  LEFT JOIN country d_1 ON b_1.country_id = d_1.country_id
  LEFT JOIN payment p_1 ON r_1.rental_id = p_1.rental_id
GROUP BY c_1.customer_id, d_1.country
), customer_active_days AS (
  SELECT c_1.customer_id,
    date_part('day'::text, max(r_1.rental_date) - min(r_1.rental_date)) AS
active_days
  FROM rental r_1
  LEFT JOIN customer c_1 ON r_1.customer_id = c_1.customer_id
GROUP BY c_1.customer_id
), customer_rentals_per_customer AS (
  SELECT c_1.customer_id,
    count(r_1.rental_id) AS rentals_per_customer
  FROM rental r_1
  LEFT JOIN customer c_1 ON r_1.customer_id = c_1.customer_id
GROUP BY c_1.customer_id
), customer_active_days_per_country AS (
  SELECT c_1.customer_id,
    d_1.country,
    date_part('day'::text, max(r_1.rental_date) - min(r_1.rental_date)) AS
active_days
  FROM rental r_1
  LEFT JOIN customer c_1 ON r_1.customer_id = c_1.customer_id
  LEFT JOIN address a_1 ON c_1.address_id = a_1.address_id
  LEFT JOIN city b_1 ON a_1.city_id = b_1.city_id
  LEFT JOIN country d_1 ON b_1.country_id = d_1.country_id
GROUP BY c_1.customer_id, d_1.country
)
SELECT d.country,
  count(DISTINCT c.customer_id) AS number_of_customers,
  count(r.rental_id) AS total_rentals,
  COALESCE(sum(p.amount), 0::numeric) AS total_revenue,
  COALESCE(sum(p.amount), 0::numeric) / count(DISTINCT c.customer_id)::numeric AS
avg_revenue_per_customer,
  CASE
    WHEN count(r.rental_id) > 0 THEN COALESCE(sum(p.amount), 0::numeric) /
count(r.rental_id)::numeric
    ELSE 0::numeric
  END AS avg_revenue_per_rental,
  COALESCE(avg(csm.spending_per_month), 0::double precision) AS
avg_spending_per_month_per_customer,
  COALESCE(avg(
    CASE
      WHEN cad.active_days > 0::double precision THEN
crp.rentals_per_customer::double precision / (cad.active_days / 30.44::double
precision)
      ELSE 0::double precision
    END), 0::double precision) AS avg_rentals_per_month_per_customer,
  ( SELECT rd.rental_rate
    FROM rental_data rd
    WHERE rd.country::text = d.country::text

```

```

        GROUP BY rd.rental_rate
        ORDER BY (count(*)) DESC, rd.rental_rate
    LIMIT 1) AS mode_rental_rate,
    COALESCE(avg(cad.active_days), 0::double precision) AS avg_active_days_per_country
FROM rental r
    LEFT JOIN customer c ON r.customer_id = c.customer_id
    LEFT JOIN address a ON c.address_id = a.address_id
    LEFT JOIN city b ON a.city_id = b.city_id
    LEFT JOIN country d ON b.country_id = d.country_id
    LEFT JOIN payment p ON r.rental_id = p.rental_id
    LEFT JOIN inventory i ON r.inventory_id = i.inventory_id
    LEFT JOIN film f ON i.film_id = f.film_id
    LEFT JOIN customer_spending_per_month csm ON csm.customer_id = c.customer_id
    LEFT JOIN customer_active_days cad ON cad.customer_id = c.customer_id
    LEFT JOIN customer_rentals_per_customer crp ON crp.customer_id = c.customer_id
    LEFT JOIN customer_active_days_per_country cadc ON cadc.country::text =
d.country::text
    GROUP BY d.country
    ORDER BY (count(DISTINCT c.customer_id)) DESC;

```

***NOTE: inflated values (total_rentals, total_revenue, avg. revenue per customer)**

| | country character varying (50) | number_of_customers bigint | total_rentals bigint | total_revenue numeric | avg_revenue_per_customer numeric | avg_nur |
|--|-----------------------------------|-------------------------------|-------------------------|--------------------------|-------------------------------------|---------|
| 1 | India | 60 | 94320 | 361967.40 | 6032.7900000000000000 | |
| 2 | China | 53 | 75578 | 278093.12 | 5247.0400000000000000 | |
| 3 | United States | 36 | 34992 | 132993.72 | 3694.2700000000000000 | |
| 4 | Japan | 31 | 25575 | 96767.12 | 3121.5200000000000000 | |
| 5 | Mexico | 30 | 23880 | 89544.60 | 2984.8200000000000000 | |
| 6 | Brazil | 28 | 20944 | 81737.32 | 2919.1900000000000000 | |
| 7 | Russian Federation | 28 | 19964 | 77437.36 | 2765.6200000000000000 | |
| Total rows: 108 Query complete 00:00:00.707 | | | | | | |

Correction - would not have needed hybrid 'country_engagement_TABLE'

/*This CTE gathers **country** and **rental_rate** from several joined tables (rental, inventory, film, customer, address, city, and country). **We use this to pull rental rate data and relate it to countries, so we can calculate the most common rental rate per country later.***/

```

WITH rental_data AS (
    SELECT d_1.country,
           f_1.rental_rate
    FROM rental r_1
    LEFT JOIN inventory i_1 ON r_1.inventory_id = i_1.inventory_id
    LEFT JOIN film f_1 ON i_1.film_id = f_1.film_id
    LEFT JOIN customer c_1 ON r_1.customer_id = c_1.customer_id
    LEFT JOIN address a_1 ON c_1.address_id = a_1.address_id
    LEFT JOIN city b_1 ON a_1.city_id = b_1.city_id
    LEFT JOIN country d_1 ON b_1.country_id = d_1.country_id
),

```

/*This CTE calculates the spending per month for each customer, considering the total amount spent (SUM(p_1.amount)) divided by the number of months (tenure-length normalized). It allows us to calculate average spending per month for each customer across different countries.*/

```
customer_spending_per_month AS (
  SELECT c_1.customer_id,
         d_1.country,
         COALESCE(SUM(p_1.amount), 0::numeric) /
         (DATE_PART('day', MAX(r_1.rental_date) - MIN(r_1.rental_date)) / 30.44) AS spending_per_month
  FROM rental r_1
  LEFT JOIN customer c_1 ON r_1.customer_id = c_1.customer_id
  LEFT JOIN address a_1 ON c_1.address_id = a_1.address_id
  LEFT JOIN city b_1 ON a_1.city_id = b_1.city_id
  LEFT JOIN country d_1 ON b_1.country_id = d_1.country_id
  LEFT JOIN payment p_1 ON r_1.rental_id = p_1.rental_id
  GROUP BY c_1.customer_id, d_1.country
),
```

/*This CTE calculates the total tenure length in days for each customer. The number of days can be used to measure customer activity over time. This is later used to adjust rental behavior by time.*/

```
customer_active_days AS (
  SELECT c_1.customer_id,
         DATE_PART('day', MAX(r_1.rental_date) - MIN(r_1.rental_date)) AS active_days
  FROM rental r_1
  LEFT JOIN customer c_1 ON r_1.customer_id = c_1.customer_id
  GROUP BY c_1.customer_id
),
```

/*This CTE calculates the total number of rentals for each customer by counting the rental IDs. It's useful for measuring customer behavior in terms of rental volume.*/

```
customer_rentals_per_customer AS (
  SELECT c_1.customer_id,
         COUNT(r_1.rental_id) AS rentals_per_customer
  FROM rental r_1
  LEFT JOIN customer c_1 ON r_1.customer_id = c_1.customer_id
  GROUP BY c_1.customer_id
)
```

/*MAIN QUERY*/

```
SELECT d.country,
       COUNT(DISTINCT c.customer_id) AS number_of_customers,
       COUNT(r.rental_id) AS total_rentals,
       COALESCE(SUM(p.amount), 0::numeric) AS total_revenue,
       COALESCE(SUM(p.amount), 0::numeric) / NULLIF(COUNT(DISTINCT c.customer_id), 0) AS
avg_revenue_per_customer,
       CASE
         WHEN COUNT(r.rental_id) > 0 THEN COALESCE(SUM(p.amount), 0::numeric) / COUNT(r.rental_id)
         ELSE 0::numeric
       END AS avg_revenue_per_rental,
       COALESCE(AVG(csm.spending_per_month), 0::double precision) AS
avg_spending_per_month_per_customer,
       COALESCE(AVG(
         CASE
```

```

        WHEN cad.active_days > 0 THEN crp.rentals_per_customer / (cad.active_days / 30.44)
        ELSE 0
    END), 0) AS avg_rentals_per_month_per_customer,

/*subquery for mode rental rate*/
    (SELECT rd.rental_rate
     FROM rental_data rd
     WHERE rd.country = d.country
     GROUP BY rd.rental_rate
     ORDER BY COUNT(*) DESC, rd.rental_rate
     LIMIT 1) AS mode_rental_rate,

    COALESCE(AVG(cad.active_days), 0::double precision) AS avg_active_days_per_country

FROM rental r
LEFT JOIN customer c ON r.customer_id = c.customer_id
LEFT JOIN address a ON c.address_id = a.address_id
LEFT JOIN city b ON a.city_id = b.city_id
LEFT JOIN country d ON b.country_id = d.country_id
LEFT JOIN payment p ON r.rental_id = p.rental_id
LEFT JOIN customer_spending_per_month csm ON csm.customer_id = c.customer_id
LEFT JOIN customer_active_days cad ON cad.customer_id = c.customer_id
LEFT JOIN customer_rentals_per_customer crp ON crp.customer_id = c.customer_id
GROUP BY d.country
ORDER BY COUNT(DISTINCT c.customer_id) DESC;

```

While both queries use CTEs, the key differences lie in:

1. **Function Name Casing:** The corrected query uses standard uppercase for SQL functions (**SUM**, **COUNT**, **AVG**) while the original query uses lowercase.
2. **Handling of Division by Zero:** The corrected query ensures no division by zero errors occur, using **NULLIF**.
3. **Simplification of Expressions:** The corrected query removes unnecessary type casts (e.g., **::double precision**), simplifying calculations and improving readability.
4. **Formatting:** Overall, the corrected query adheres to cleaner SQL formatting, improving consistency and preventing potential errors.

stddev_absolutes

This table records statistical descriptions of key metrics aggregated by country, including the standard deviation and its absolute value (for visualization purposes). Metrics include average revenue per customer, average revenue per rental, average monthly spending (normalized by average tenure), and average rentals per month (normalized).

```

WITH country_rentals AS (
    SELECT country_engagement_table.country,
           country_engagement_table.avg_rentals_per_month_per_customer,
           country_engagement_table.avg_spending_per_month_per_customer,
           country_engagement_table.avg_revenue_per_rental,
           country_engagement_table.avg_revenue_per_customer
    FROM country_engagement_table
    GROUP BY country_engagement_table.country,
             country_engagement_table.avg_rentals_per_month_per_customer,
             country_engagement_table.avg_spending_per_month_per_customer,
             country_engagement_table.avg_revenue_per_rental,
             country_engagement_table.avg_revenue_per_customer
),
stats AS (
    SELECT avg(country_engagement_table.avg_rentals_per_month_per_customer) AS
mean_rentals,
           stddev(country_engagement_table.avg_rentals_per_month_per_customer) AS
stddev_rentals,
           avg(country_engagement_table.avg_spending_per_month_per_customer) AS
mean_spending,
           stddev(country_engagement_table.avg_spending_per_month_per_customer) AS
stddev_spending,
           avg(country_engagement_table.avg_revenue_per_rental) AS
mean_revenue_per_rental,
           stddev(country_engagement_table.avg_revenue_per_rental) AS
stddev_revenue_per_rental,
           avg(country_engagement_table.avg_revenue_per_customer) AS
mean_revenue_per_customer,
           stddev(country_engagement_table.avg_revenue_per_customer) AS
stddev_revenue_per_customer
    FROM country_engagement_table
)
SELECT cr.country, /*These calculations possible by using Cartesian Join at end of
script, to join the two CTEs*?
       cr.avg_rentals_per_month_per_customer,
       cr.avg_spending_per_month_per_customer,
       cr.avg_revenue_per_rental,
       cr.avg_revenue_per_customer,
       (cr.avg_rentals_per_month_per_customer - s.mean_rentals) / s.stddev_rentals AS
rentals_standard_deviation,
       abs((cr.avg_rentals_per_month_per_customer - s.mean_rentals) / s.stddev_rentals)
AS rentals_absolute_stddev,
       (cr.avg_spending_per_month_per_customer - s.mean_spending) / s.stddev_spending AS
spending_standard_deviation,
       abs((cr.avg_spending_per_month_per_customer - s.mean_spending) /
s.stddev_spending) AS spending_absolute_stddev,
       (cr.avg_revenue_per_rental - s.mean_revenue_per_rental) /
s.stddev_revenue_per_rental AS revenue_per_rental_standard_deviation,
       abs((cr.avg_revenue_per_rental - s.mean_revenue_per_rental) /
s.stddev_revenue_per_rental) AS revenue_per_rental_absolute_stddev,
       (cr.avg_revenue_per_customer - s.mean_revenue_per_customer) /
s.stddev_revenue_per_customer AS revenue_per_customer_standard_deviation,

```

```
abs((cr.avg_revenue_per_customer - s.mean_revenue_per_customer) /
s.stddev_revenue_per_customer) AS revenue_per_customer_absolute_stddev
FROM country_rentals cr
JOIN stats s ON 1 = 1;
```

| | country character varying (50) | avg_rentals_per_month_per_customer double precision | avg_spending_per_month_per_customer double precision | avg_revenue_per_rental numeric | avg_revenue_per_customer numeric |
|--|-----------------------------------|--|---|-----------------------------------|-------------------------------------|
| 1 | Afghanistan | 8.05764705882353 | 30.359423529411757 | 3.767777777777778 | 67.8200000000 |
| 2 | Algeria | 7.029239728378271 | 28.148716778961838 | 3.879777777777778 | 116.393333333 |
| 3 | American Samoa | 8.952941176470585 | 21.419911764705883 | 2.392500000000000 | 47.8500000000 |
| 4 | Angola | 8.852462402765763 | 31.88784031979257 | 3.6067307692307692 | 93.7750000000 |
| 5 | Anguilla | 11.970786516853932 | 34.0928 | 2.848000000000000 | 99.6800000000 |
| 6 | Argentina | 6.093668752758295 | 22.084310428067223 | 3.6897727272727273 | 99.907692307 |
| 7 | Armenia | 3.170833333333334 | 15.061458333333343 | 4.750000000000000 | 118.750000000 |
| Total rows: 108 Query complete 00:00:00.850 | | | | | |
| LF Ln 1, Col 31 | | | | | |

----->

| avg_revenue_per_customer numeric | rentals_standard_deviation double precision | rentals_absolute_stddev double precision | spending_standard_deviation double precision | spending_absolute_stddev double precision | revenue_per_rental numeric |
|--|--|---|---|--|-------------------------------|
| 67.8200000000000000 | -0.04812005140556546 | 0.04812005140556546 | -0.09819562837420331 | 0.09819562837420331 | -0.0993948804161 |
| 116.3933333333333333 | -0.43672998758022263 | 0.43672998758022263 | -0.298470102655402 | 0.298470102655402 | 0.1715650599719 |
| 47.8500000000000000 | 0.2901896626702107 | 0.2901896626702107 | -0.908052484622862 | 0.908052484622862 | -3.4265840293781 |
| 93.7750000000000000 | 0.2522211956237605 | 0.2522211956237605 | 0.04026817697391756 | 0.04026817697391756 | -0.4890135218052 |
| 99.6800000000000000 | 1.4305594853975527 | 1.4305594853975527 | 0.24002200713475158 | 0.24002200713475158 | -2.3245996289604 |
| 99.9076923076923077 | -0.7902593395784513 | 0.7902593395784513 | -0.847862641265871 | 0.847862641265871 | -0.2881113432270 |
| 118.7500000000000000 | -1.8947272407358524 | 1.8947272407358524 | -1.4840836194400207 | 1.4840836194400207 | 2.2768807872731 |
| rows: 108 Query complete 00:00:00.850 | | | | | |
| LF Ln 1, Col 31 | | | | | |

----->

| revenue_per_rental_standard_deviation numeric | revenue_per_rental_absolute_stddev numeric | revenue_per_customer_standard_deviation numeric | revenue_per_customer_absolute_stddev numeric |
|--|---|--|---|
| -0.09939488041611529944372408063616 | 0.09939488041611529944372408063616 | -1.85245409960338585985061416110565 | 1.85245409960338585985061416110565 |
| 0.17156505997198437900055818341272 | 0.17156505997198437900055818341272 | 0.62841677765805988353399092017862 | 0.62841677765805988353399092017862 |
| -3.42658402937812306148349769262525 | 3.42658402937812306148349769262525 | -2.87241686557053665993411654249774 | 2.87241686557053665993411654249774 |
| -0.48901352180521556432757778687541 | 0.48901352180521556432757778687541 | -0.52680895234863378092385881263135 | 0.52680895234863378092385881263135 |
| -2.32459962896044981566768902053358 | 2.32459962896044981566768902053358 | -0.22521255109495198450457581002142 | 0.22521255109495198450457581002142 |
| -0.28811134322700090832774280268411 | 0.28811134322700090832774280268411 | -0.21358322330610422644347090787730 | 0.21358322330610422644347090787730 |
| 2.27688078727317150934146157072515 | 2.27688078727317150934146157072515 | 0.74878293949060463830926447105755 | 0.74878293949060463830926447105755 |
| rows: 108 Query complete 00:00:00.850 | | | |
| LF Ln 1, Col 31 | | | |

Calculator: Slide 4

[no View]

This one-row table reports the baseline metrics required for the growth percentage calculator, including total revenue, total rentals, average revenue per rental (to convert additional revenue targets into rental numbers), and average rentals per month for the average Rockbuster customer (to convert rental numbers into customer counts).

```
SELECT
    SUM(total_rentals) AS total_rentals,
    SUM(total_revenue) AS total_revenue,
    AVG(avg_revenue_per_rental) AS avg_revenue_per_rental,
    AVG(avg_rentals_per_month_per_customer) AS avg_rentals_per_month_per_customer
```

```
FROM country_engagement_TABLE;
```

| | total_rentals numeric | total_revenue numeric | avg_revenue_per_rental numeric | avg_rentals_per_month_per_customer double precision |
|---|--------------------------|--------------------------|-----------------------------------|--|
| 1 | 16048 | 61312.04 | 3.8088621778948581 | 8.184990724026811 |

Tableau calculated fields:

- **Additional Revenue:**

$$[\text{Total Revenue}] * (1 + [\text{Growth \%}] / 100) - [\text{Total Revenue}]$$
- **Additional Rentals:**

$$([\text{Total Revenue}] * (1 + [\text{Growth \%}] / 100) - [\text{Total Revenue}]) / [\text{Avg Revenue Per Rental}]$$
- **Additional Customers:**

$$([\text{Total Rentals}] * (1 + [\text{Growth \%}] / 100) - [\text{Total Rentals}]) / [\text{Avg Rentals Per Month Per Customer}]$$

Bar Graph - Genre Popularity: Slide 5

genre_popularity_norm

This table records the total number of rentals and total movie titles for each genre, and calculates the number of rentals per title within each genre.

```
SELECT g.genre,
       sum(g.number_of_rentals) AS number_of_rentals,
       t.number_of_titles,
       sum(g.number_of_rentals) / t.number_of_titles::numeric AS
average_rentals_per_title
FROM genre_popularity_country g
     LEFT JOIN title_count_genre t ON g.genre::text = t.genre::text
GROUP BY g.genre, t.number_of_titles
```



```
ORDER BY (sum(g.number_of_rentals) / t.number_of_titles::numeric) DESC;
```

| | genre character varying (25) | number_of_rentals numeric | number_of_titles bigint | average_rentals_per_title numeric |
|----------------|---------------------------------|------------------------------|----------------------------|--------------------------------------|
| 1 | Sci-Fi | 1101 | 59 | 18.6610169491525424 |
| 2 | Action | 1112 | 61 | 18.2295081967213115 |
| 3 | Animation | 1166 | 64 | 18.2187500000000000 |
| 4 | Classics | 939 | 54 | 17.3888888888888889 |
| 5 | Drama | 1060 | 61 | 17.3770491803278689 |
| 6 | Comedy | 941 | 56 | 16.8035714285714286 |
| 7 | Games | 969 | 58 | 16.7068965517241379 |
| Total rows: 17 | | Query complete 00:00:00.479 | | |

Heatmap - Outlier Countries Genre Analysis: Slide 5

avg_rentals_outliers

This table records key performance and customer behavior metrics for the 19 outlier countries (those with above or below average rental frequency per customer). It includes the standard deviation and absolute value of these standard deviations for each metric.

```
/*CTE 1: country_rentals - aggregated by country*/
```

```
WITH country_rentals AS (
```

```
    SELECT country_engagement_table.country,
           country_engagement_table.avg_rentals_per_month_per_customer,
           country_engagement_table.avg_spending_per_month_per_customer,
           country_engagement_table.avg_revenue_per_rental,
           country_engagement_table.avg_revenue_per_customer
    FROM country_engagement_table
```

```
    GROUP BY country_engagement_table.country,
             country_engagement_table.avg_rentals_per_month_per_customer,
             country_engagement_table.avg_spending_per_month_per_customer,
             country_engagement_table.avg_revenue_per_rental,
             country_engagement_table.avg_revenue_per_customer
),
```

```
/*CTE 2: stats - These statistics will be used in the final CTE to standardize the data.*/
```

```
stats AS (
```

```
    SELECT avg(country_engagement_table.avg_rentals_per_month_per_customer) AS
mean_rentals,
           stddev(country_engagement_table.avg_rentals_per_month_per_customer) AS
stddev_rentals,
           avg(country_engagement_table.avg_spending_per_month_per_customer) AS
mean_spending,
           stddev(country_engagement_table.avg_spending_per_month_per_customer) AS
stddev_spending,
```

```

        avg(country_engagement_table.avg_revenue_per_rental) AS
mean_revenue_per_rental,
        stddev(country_engagement_table.avg_revenue_per_rental) AS
stddev_revenue_per_rental,
        avg(country_engagement_table.avg_revenue_per_customer) AS
mean_revenue_per_customer,
        stddev(country_engagement_table.avg_revenue_per_customer) AS
stddev_revenue_per_customer
    FROM country_engagement_table
),

```

/*CTE 3: final - This is where the actual "work" happens:

-The **country_rentals** CTE is **joined** to the **stats** CTE using **JOIN ON 1 = 1: cross-joins** the statistics to every row in **country_rentals**.

-Then, it calculates **z-scores** for each of the metrics by subtracting the mean from each value and dividing by the standard deviation **((value - mean) / stddev)**.

-The **absolute value** of these z-scores is also calculated.*/

```

final AS (
    SELECT cr.country,
           cr.avg_rentals_per_month_per_customer,
           cr.avg_spending_per_month_per_customer,
           cr.avg_revenue_per_rental,
           cr.avg_revenue_per_customer,
           (cr.avg_rentals_per_month_per_customer - s.mean_rentals) /
s.stddev_rentals AS rentals_standard_deviation,
           abs((cr.avg_rentals_per_month_per_customer - s.mean_rentals) /
s.stddev_rentals) AS rentals_absolute_stddev,
           (cr.avg_spending_per_month_per_customer - s.mean_spending) /
s.stddev_spending AS spending_standard_deviation,
           abs((cr.avg_spending_per_month_per_customer - s.mean_spending) /
s.stddev_spending) AS spending_absolute_stddev,
           (cr.avg_revenue_per_rental - s.mean_revenue_per_rental) /
s.stddev_revenue_per_rental AS revenue_per_rental_standard_deviation,
           abs((cr.avg_revenue_per_rental - s.mean_revenue_per_rental) /
s.stddev_revenue_per_rental) AS revenue_per_rental_absolute_stddev,
           (cr.avg_revenue_per_customer - s.mean_revenue_per_customer) /
s.stddev_revenue_per_customer AS revenue_per_customer_standard_deviation,
           abs((cr.avg_revenue_per_customer - s.mean_revenue_per_customer) /
s.stddev_revenue_per_customer) AS revenue_per_customer_absolute_stddev
    FROM country_rentals cr
    JOIN stats s ON 1 = 1
)

```

/*Last part picks the results from the **final CTE and filters the countries where the **absolute rental standard deviation** is greater than or equal to 1.5. It also orders the countries by their **rental standard deviation***/**

```

SELECT final.country,
       final.avg_rentals_per_month_per_customer,
       final.avg_spending_per_month_per_customer,
       final.avg_revenue_per_rental,
       final.avg_revenue_per_customer,
       final.rentals_standard_deviation,
       final.rentals_absolute_stddev,
       final.spending_standard_deviation,
       final.spending_absolute_stddev,
       final.revenue_per_rental_standard_deviation,

```

```

    final.revenue_per_rental_absolute_stddev,
    final.revenue_per_customer_standard_deviation,
    final.revenue_per_customer_absolute_stddev
FROM final
WHERE final.rentals_absolute_stddev >= 1.5::double precision
ORDER BY final.rentals_standard_deviation DESC;

```

| | country character varying (50) | avg_rentals_per_month_per_customer double precision | avg_spending_per_month_per_customer double precision | avg_revenue_per_rental numeric | avg_revenue_per_customer numeric |
|---|-----------------------------------|--|---|-----------------------------------|-------------------------------------|
| 1 | Reunion | 16.28186046511629 | 74.87886046511629 | 4.5989130434782609 | 211.5500000000000000 |
| 2 | Moldova | 14.999420289855081 | 56.31841159420288 | 3.7547058823529412 | 127.6600000000000000 |
| 3 | Hong Kong | 13.836363636363632 | 48.31658181818184 | 3.4920000000000000 | 104.7600000000000000 |
| 4 | Sweden | 12.32095238095239 | 50.61374761904765 | 4.1079411764705882 | 139.6700000000000000 |
| 5 | Holy See (Vatican City State) | 12.175999999999993 | 52.52869647058827 | 4.3141176470588235 | 146.6800000000000000 |
| 6 | Brunei | 4.0356060606060575 | 12.413524242424232 | 3.0760000000000000 | 107.6600000000000000 |
| 7 | Zambia | 3.7906415094339643 | 13.97942641509434 | 3.6878787878787879 | 121.7000000000000000 |
| Total rows: 19 Query complete 00:00:00.991 LF Ln 1, Col 35 | | | | | |

----->

| rentals_standard_deviation double precision | rentals_absolute_stddev double precision | spending_standard_deviation double precision | spending_absolute_stddev double precision | revenue_per_rental_standard_deviation numeric | revenue_per_customer_standard_deviation numeric |
|---|---|---|--|--|--|
| 3.059608717430071 | 3.059608717430071 | 3.9349520633215023 | 3.9349520633215023 | 1.91135835216111941581587050088815 | 1.91135835216111941581587050088815 |
| 2.575005992515323 | 2.575005992515323 | 2.2535060102487745 | 2.2535060102487745 | -0.13101952331855456780837023204126 | -0.13101952331855456780837023204126 |
| 2.1355153758108165 | 2.1355153758108165 | 1.5285965434666675 | 1.5285965434666675 | -0.76657997172887666461306600225250 | -0.76657997172887666461306600225250 |
| 1.5628785870505222 | 1.5628785870505222 | 1.7367035992810662 | 1.7367035992810662 | 0.72355738951261050472207861644291 | 0.72355738951261050472207861644291 |
| 1.5081046333430093 | 1.5081046333430093 | 1.9101844892451607 | 1.9101844892451607 | 1.22235706969133392911451736344990 | 1.22235706969133392911451736344990 |
| -1.5679508123672048 | 1.5679508123672048 | -1.7239678138860535 | 1.7239678138860535 | -1.77300260745610404169182869729121 | -1.77300260745610404169182869729121 |
| -1.6605169150950265 | 1.6605169150950265 | -1.5821080966967578 | 1.5821080966967578 | -0.29269332273843388562610744530397 | -0.29269332273843388562610744530397 |
| Total rows: 19 Query complete 00:00:00.991 LF Ln 1, Col 35 | | | | | |

| revenue_per_rental_absolute_stddev numeric | revenue_per_customer_standard_deviation numeric | revenue_per_customer_absolute_stddev numeric |
|---|--|---|
| 1.91135835216111941581587050088815 | 5.48851977883720425011442325889862 | 5.48851977883720425011442325889862 |
| 0.13101952331855456780837023204126 | 1.20385896576838699327891926415738 | 1.20385896576838699327891926415738 |
| 0.76657997172887666461306600225250 | 0.03424718105893514941751693396815 | 0.03424718105893514941751693396815 |
| 0.72355738951261050472207861644291 | 1.81726671836054929217741018055793 | 1.81726671836054929217741018055793 |
| 1.22235706969133392911451736344990 | 2.17530071883274393957777831744119 | 2.17530071883274393957777831744119 |
| 1.77300260745610404169182869729121 | 0.18236395728851638728642814608819 | 0.18236395728851638728642814608819 |
| 0.29269332273843388562610744530397 | 0.89945345324138555269315691097276 | 0.89945345324138555269315691097276 |
| Total rows: 19 Query complete 00:00:00.991 LF Ln 1, Col 35 | | |

Correction:

No need for a second CTE (**stats**):

The need for the second CTE (**stats**) to calculate global averages and standard deviations is eliminated, simplifying the query.

```

WITH country_rentals AS (
  SELECT
    country_engagement_table.country,
    country_engagement_table.avg_rentals_per_month_per_customer,
    country_engagement_table.avg_spending_per_month_per_customer,
    country_engagement_table.avg_revenue_per_rental,

```

```

country_engagement_table.avg_revenue_per_customer
FROM country_engagement_table
),
standard_devs AS (
  SELECT
    cr.country,
    cr.avg_rentals_per_month_per_customer,
    cr.avg_spending_per_month_per_customer,
    cr.avg_revenue_per_rental,
    cr.avg_revenue_per_customer,
    (cr.avg_rentals_per_month_per_customer - AVG(cr.avg_rentals_per_month_per_customer) OVER ()) /
    STDDEV(cr.avg_rentals_per_month_per_customer) OVER () AS rentals_standard_deviation,
    ABS((cr.avg_rentals_per_month_per_customer - AVG(cr.avg_rentals_per_month_per_customer) OVER ()) /
    STDDEV(cr.avg_rentals_per_month_per_customer) OVER ()) AS rentals_absolute_stddev,
    (cr.avg_spending_per_month_per_customer - AVG(cr.avg_spending_per_month_per_customer) OVER ()) /
    STDDEV(cr.avg_spending_per_month_per_customer) OVER () AS spending_standard_deviation,
    ABS((cr.avg_spending_per_month_per_customer - AVG(cr.avg_spending_per_month_per_customer) OVER
    ()) / STDDEV(cr.avg_spending_per_month_per_customer) OVER ()) AS spending_absolute_stddev,
    (cr.avg_revenue_per_rental - AVG(cr.avg_revenue_per_rental) OVER ()) /
    STDDEV(cr.avg_revenue_per_rental) OVER () AS revenue_per_rental_standard_deviation,
    ABS((cr.avg_revenue_per_rental - AVG(cr.avg_revenue_per_rental) OVER ()) /
    STDDEV(cr.avg_revenue_per_rental) OVER ()) AS revenue_per_rental_absolute_stddev,
    (cr.avg_revenue_per_customer - AVG(cr.avg_revenue_per_customer) OVER ()) /
    STDDEV(cr.avg_revenue_per_customer) OVER () AS revenue_per_customer_standard_deviation,
    ABS((cr.avg_revenue_per_customer - AVG(cr.avg_revenue_per_customer) OVER ()) /
    STDDEV(cr.avg_revenue_per_customer) OVER ()) AS revenue_per_customer_absolute_stddev
  FROM country_rentals cr
)
SELECT
  sd.country,
  sd.avg_rentals_per_month_per_customer,
  sd.avg_spending_per_month_per_customer,
  sd.avg_revenue_per_rental,
  sd.avg_revenue_per_customer,
  sd.rentals_standard_deviation,
  sd.rentals_absolute_stddev,
  sd.spending_standard_deviation,
  sd.spending_absolute_stddev,
  sd.revenue_per_rental_standard_deviation,
  sd.revenue_per_rental_absolute_stddev,
  sd.revenue_per_customer_standard_deviation,
  sd.revenue_per_customer_absolute_stddev
FROM standard_devs sd
WHERE sd.rentals_absolute_stddev >= 1.5
ORDER BY sd.rentals_standard_deviation DESC;

```

genre_popularity_country

This table records the performance of each movie genre in each country, including the number of rentals, distinct titles rented, and the rentals per title for each genre in each country.

```
SELECT g.name AS genre,
       k.country,
       count(DISTINCT r.rental_id) AS number_of_rentals,
       count(DISTINCT f.film_id) AS title_count,
       count(DISTINCT r.rental_id)::numeric * 1.0 / count(DISTINCT f.film_id)::numeric AS
rentals_per_title
FROM rental r
     LEFT JOIN granular_rental_table_2 k ON r.rental_id = k.rental_id
     LEFT JOIN inventory i ON r.inventory_id = i.inventory_id
     LEFT JOIN film f ON f.film_id = i.film_id
     LEFT JOIN film_category q ON f.film_id = q.film_id
     LEFT JOIN category g ON q.category_id = g.category_id
GROUP BY g.name, k.country
ORDER BY (count(DISTINCT r.rental_id)::numeric * 1.0 / count(DISTINCT
f.film_id)::numeric) DESC;
```

| | genre character varying (25) 🔒 | country character varying (50) 🔒 | number_of_rentals bigint 🔒 | title_count bigint 🔒 | rentals_per_title numeric 🔒 |
|------------------|-----------------------------------|-------------------------------------|-------------------------------|-------------------------|--------------------------------|
| 1 | Comedy | [null] | 90 | 40 | 2.2500000000000000 |
| 2 | Horror | [null] | 73 | 33 | 2.2121212121212121 |
| 3 | Classics | India | 96 | 44 | 2.1818181818181818 |
| 4 | Games | [null] | 85 | 39 | 2.1794871794871795 |
| 5 | Foreign | India | 99 | 46 | 2.1521739130434783 |
| 6 | Documentary | India | 103 | 48 | 2.1458333333333333 |
| 7 | Children | India | 98 | 46 | 2.1304347826086957 |
| Total rows: 1593 | | Query complete 00:00:00.235 | | | |

Scatterplot & Box-Whisker Graph - Tenure Length Analysis: Slide 6

customer_engagement_district

This table records customer behavior metrics, including location (city, district, country), total rentals, tenure length (calculated from earliest and most recent rentals), total amount paid, and average spending and rental frequency per week and month during their tenure.

```
SELECT c.customer_id,
       c.first_name,
       c.last_name,
```

```

b.city,
a.district,
d.country,
count(r.rental_id) AS number_of_rentals,
min(r.rental_date) AS earliest_rental_date,
max(r.rental_date) AS most_recent_rental_date,
date_part('day'::text, max(r.rental_date) - min(r.rental_date)) AS active_days,
COALESCE(sum(p.amount), 0::numeric) AS total_amount_paid,
/* Could have just used the Alias 'active_days' instead of redoing the calculation
each time.*/
CASE
    WHEN date_part('day'::text, max(r.rental_date) - min(r.rental_date)) >=
7::double precision THEN COALESCE(sum(p.amount), 0::numeric)::double precision /
(date_part('day'::text, max(r.rental_date) - min(r.rental_date)) / 7::double
precision)
    ELSE 0::double precision
END AS avg_spending_per_week,
CASE
    WHEN date_part('day'::text, max(r.rental_date) - min(r.rental_date)) >=
30::double precision THEN COALESCE(sum(p.amount), 0::numeric)::double precision /
(date_part('day'::text, max(r.rental_date) - min(r.rental_date)) / 30.44::double
precision)
    ELSE 0::double precision
END AS avg_spending_per_month,
CASE
    WHEN date_part('day'::text, max(r.rental_date) - min(r.rental_date)) >=
7::double precision THEN count(r.rental_id)::double precision /
(date_part('day'::text, max(r.rental_date) - min(r.rental_date)) / 7::double
precision)
    ELSE 0::double precision
END AS rental_frequency_per_week,
CASE
    WHEN date_part('day'::text, max(r.rental_date) - min(r.rental_date)) >=
30::double precision THEN count(r.rental_id)::double precision /
(date_part('day'::text, max(r.rental_date) - min(r.rental_date)) / 30.44::double
precision)
    ELSE 0::double precision
END AS rental_frequency_per_month
FROM rental r
LEFT JOIN customer c ON r.customer_id = c.customer_id
LEFT JOIN address a ON c.address_id = a.address_id
LEFT JOIN city b ON a.city_id = b.city_id
LEFT JOIN country d ON b.country_id = d.country_id
LEFT JOIN payment p ON r.rental_id = p.rental_id
GROUP BY c.customer_id, c.first_name, c.last_name, b.city, a.district, d.country
ORDER BY (count(r.rental_id)) DESC;

```

| | customer_id integer | first_name character varying (45) | last_name character varying (45) | city character varying (50) | district character varying (20) | country character varying (50) |
|---|------------------------|--------------------------------------|-------------------------------------|--------------------------------|------------------------------------|-----------------------------------|
| 1 | 148 | Eleanor | Hunt | Saint-Denis | Saint-Denis | Reunion |
| 2 | 526 | Karl | Seal | Cape Coral | Florida | United States |
| 3 | 236 | Marcia | Dean | Tanza | Southern Tagalog | Philippines |
| 4 | 144 | Clara | Shaw | Molodetno | Minsk | Belarus |
| 5 | 75 | Tammy | Sanders | Changhwa | Changhwa | Taiwan |
| 6 | 469 | Wesley | Bull | Ourense (Orense) | Galicia | Spain |
| 7 | 197 | Sue | Peters | Changzhou | Jiangsu | China |
| Total rows: 599 Query complete 00:00:00.202 LF Ln 1, C | | | | | | |

----->

| number_of_rentals bigint | earliest_rental_date timestamp without time zone | most_recent_rental_date timestamp without time zone | active_days double precision | total_amount_paid numeric | avg_spending_per_week double precision | avg_spe double p |
|--|---|--|---------------------------------|------------------------------|---|---------------------|
| 46 | 2005-05-28 23:53:18 | 2005-08-23 05:57:04 | 86 | 211.55 | 17.219186046511627 | 7 |
| 45 | 2005-05-28 00:40:48 | 2005-08-23 22:21:03 | 87 | 208.58 | 16.782298850574712 | 7 |
| 42 | 2005-05-26 15:46:56 | 2006-02-14 15:16:03 | 263 | 166.61 | 4.434486692015209 | 1 |
| 42 | 2005-05-27 00:49:27 | 2005-08-23 12:43:30 | 88 | 189.60 | 15.081818181818182 | 6 |
| 41 | 2005-05-26 04:46:23 | 2006-02-14 15:16:03 | 264 | 149.61 | 3.9669318181818185 | 17 |
| 40 | 2005-05-26 03:07:43 | 2005-08-23 15:46:33 | 89 | 158.65 | 12.47808988764045 | 5 |
| 40 | 2005-05-25 16:03:42 | 2005-08-23 04:12:52 | 89 | 133.68 | 10.514157303370787 | 45 |
| rows: 599 Query complete 00:00:00.202 LF Ln 1, Col 43 | | | | | | |

| avg_spending_per_month double precision | rental_frequency_per_week double precision | rental_frequency_per_month double precision |
|--|---|--|
| 74.87886046511629 | 3.744186046511628 | 16.281860465116278 |
| 72.97902528735632 | 3.6206896551724137 | 15.744827586206897 |
| 19.28368212927757 | 1.11787072243346 | 4.861140684410647 |
| 65.58436363636363 | 3.340909090909091 | 14.52818181818182 |
| 17.250486363636366 | 1.0871212121212122 | 4.7274242424242425 |
| 54.26186516853933 | 3.146067415730337 | 13.680898876404495 |
| 45.721564044943825 | 3.146067415730337 | 13.680898876404495 |

----->

LF Ln 1, Col 43

Correction

Some improvements that could be considered:

1. Reducing Repetition in CASE Statements

There are several CASE statements that calculate averages for weekly and monthly spending, frequency, etc. These formulas are repeated with very minor differences (for weekly vs monthly metrics). To avoid redundancy and improve maintainability, you could calculate the number of days in a separate expression and reuse it for all these metrics.

2. Use Subquery or CTE to Calculate Date Differences

Instead of repeatedly calculating the date difference between `max(r.rental_date) - min(r.rental_date)` for each CASE statement, you could perform this calculation once in a subquery or CTE, which would improve clarity and possibly performance.

3. Remove ::double precision Casts

If the data types you're working with (e.g., `numeric`, `integer`, `date`) already match the needed

precision, you could omit the explicit casting (e.g., `::double precision`), unless it's absolutely necessary for certain operations.

4. Using **COALESCE** More Effectively

Instead of `COALESCE(sum(p.amount), 0::numeric)` in multiple places, you could handle the **NULL** values once at the outer level, which reduces redundant calls to **COALESCE**.

```
SELECT
  c.customer_id,
  c.first_name,
  c.last_name,
  b.city,
  a.district,
  d.country,
  count(r.rental_id) AS number_of_rentals,
  min(r.rental_date) AS earliest_rental_date,
  max(r.rental_date) AS most_recent_rental_date,
  date_part('day', max(r.rental_date) - min(r.rental_date)) AS active_days,
  COALESCE(sum(p.amount), 0) AS total_amount_paid,
  -- Calculate active days once
  CASE
    WHEN active_days >= 7 THEN COALESCE(sum(p.amount), 0) / (active_days / 7)
    ELSE 0
  END AS avg_spending_per_week,

  CASE
    WHEN active_days >= 30 THEN COALESCE(sum(p.amount), 0) / (active_days / 30.44)
    ELSE 0
  END AS avg_spending_per_month,

  CASE
    WHEN active_days >= 7 THEN count(r.rental_id) / (active_days / 7)
    ELSE 0
  END AS rental_frequency_per_week,

  CASE
    WHEN active_days >= 30 THEN count(r.rental_id) / (active_days / 30.44)
    ELSE 0
  END AS rental_frequency_per_month
FROM
  rental r
  LEFT JOIN customer c ON r.customer_id = c.customer_id
  LEFT JOIN address a ON c.address_id = a.address_id
  LEFT JOIN city b ON a.city_id = b.city_id
  LEFT JOIN country d ON b.country_id = d.country_id
  LEFT JOIN payment p ON r.rental_id = p.rental_id
GROUP BY
  c.customer_id, c.first_name, c.last_name, b.city, a.district, d.country
ORDER BY
  number_of_rentals DESC;
```


Pie Chart: Slide 7

late_charge_noneg

This table records the number of days a transaction was overdue and calculates the corresponding late fees for each transaction (represented in the last two columns).

```

SELECT r.rental_id,
       p.payment_id,
       c.customer_id,
       c.first_name,
       c.last_name,
       b.city,
       d.country,
       a.district,
       p.amount AS payment_amount,
       f.title,
       f.rental_duration,
       f.rental_rate,
       f.length,
       f.rating,
       r.rental_date,
       r.return_date,
       CASE
         WHEN r.return_date > (r.rental_date + '1 day'::interval) THEN
GREATEST(0::numeric, EXTRACT(day FROM r.return_date - r.rental_date) -
f.rental_duration::numeric)
         ELSE 0::numeric
       END AS overdue_days,
       CASE
         WHEN r.return_date > (r.rental_date + '1 day'::interval) THEN
GREATEST(0::numeric, EXTRACT(day FROM r.return_date - r.rental_date) -
f.rental_duration::numeric)
         ELSE 0::numeric
       END * 1.00 AS late_charge
FROM rental r
  LEFT JOIN payment p ON r.rental_id = p.rental_id
  LEFT JOIN customer c ON p.customer_id = c.customer_id
  LEFT JOIN address a ON c.address_id = a.address_id
  LEFT JOIN city b ON a.city_id = b.city_id
  LEFT JOIN country d ON b.country_id = d.country_id
  LEFT JOIN inventory i ON r.inventory_id = i.inventory_id
  LEFT JOIN film f ON i.film_id = f.film_id
  JOIN film_category fc ON f.film_id = fc.film_id
  LEFT JOIN category g ON fc.category_id = g.category_id;

```

| | rental_id integer | payment_id integer | customer_id integer | first_name character varying (45) | last_name character varying (45) | city character varying (50) | country character varying (50) | d |
|--|----------------------|-----------------------|------------------------|--------------------------------------|-------------------------------------|--------------------------------|-----------------------------------|--------------------|
| 1 | 1520 | 17503 | 341 | Peter | Menard | Ede | Netherlands | (|
| 2 | 1778 | 17504 | 341 | Peter | Menard | Ede | Netherlands | (|
| 3 | 1849 | 17505 | 341 | Peter | Menard | Ede | Netherlands | (|
| 4 | 2829 | 17506 | 341 | Peter | Menard | Ede | Netherlands | (|
| 5 | 3130 | 17507 | 341 | Peter | Menard | Ede | Netherlands | (|
| 6 | 3382 | 17508 | 341 | Peter | Menard | Ede | Netherlands | (|
| 7 | 2190 | 17509 | 342 | Harold | Martino | Boa Vista | Brazil | f |
| Total rows: 16048 Query complete 00:00:00.298 | | | | | | | | LF Ln 1, Col 32 |

----->

| district character varying (20) | payment_amount numeric (5,2) | title character varying (255) | rental_duration smallint | rental_rate numeric (4,2) | length smallint | rating mpaa_rating | rental_date timestamp without time zone |
|------------------------------------|---------------------------------|----------------------------------|-----------------------------|------------------------------|--------------------|-----------------------|--|
| Gelderland | 7.99 | Rules Human | 6 | 4.99 | 153 | R | 2005-06-15 23:57:20 |
| Gelderland | 1.99 | Majestic Floats | 5 | 0.99 | 130 | PG | 2005-06-16 18:54:48 |
| Gelderland | 7.99 | Maiden Home | 3 | 4.99 | 138 | PG | 2005-06-17 00:13:19 |
| Gelderland | 2.99 | Hyde Doctor | 5 | 2.99 | 100 | G | 2005-06-19 21:11:30 |
| Gelderland | 7.99 | Massacre Usual | 6 | 4.99 | 165 | R | 2005-06-20 19:03:22 |
| Gelderland | 5.99 | Annie Identity | 3 | 0.99 | 86 | G | 2005-06-21 14:05:23 |
| Roraima | 5.99 | Wash Heavenly | 7 | 4.99 | 161 | R | 2005-06-18 01:29:51 |
| | | | | | | | |
| rows: 16048 | Query complete 00:00:00.298 | | LF Ln 1, Col 32 | | | | |

----->

| return_date timestamp without time zone | overdue_days numeric | late_charge numeric |
|--|-------------------------|------------------------|
| 2005-06-24 23:46:20 | 2 | 2.00 |
| 2005-06-22 16:08:48 | 0 | 0.00 |
| 2005-06-23 18:37:19 | 3 | 3.00 |
| 2005-06-24 18:10:30 | 0 | 0.00 |
| 2005-06-29 18:05:22 | 2 | 2.00 |
| 2005-06-29 19:13:23 | 5 | 5.00 |
| 2005-06-26 04:31:51 | 1 | 1.00 |
| LF Ln 1, Col 32 | | |

CASE function explanations

overdue_days

- **Condition:**
 - The **CASE** checks if the **r.return_date** is greater than the **r.rental_date** plus one day (i.e., **r.rental_date + '1 day'::interval**).
 - This checks if the item was returned after the rental was due. The **+ '1 day'::interval** part ensures that even if it was returned the day after the rental period ends, it is considered overdue.
- **When the condition is true** (**r.return_date > r.rental_date + '1 day'**):
 - The **EXTRACT(day FROM r.return_date - r.rental_date)** part calculates the number of days between the rental and return date.
 - Then, the rental duration (**f.rental_duration**) is subtracted from this difference to find the **excess days**.
 - The **GREATEST(0::numeric, ...)** ensures that if the result is negative (i.e., if the rental was returned within the expected duration), it returns 0 instead of a negative value. This means the item isn't considered overdue if it was returned on time.

- When the condition is false (`r.return_date <= r.rental_date + '1 day'`):
 - It simply returns 0, indicating no overdue days.

So, **overdue days** represents how many days the rental is overdue after considering the expected rental duration, but not going below zero.

late_charge

How this works:

- The structure is **identical to the overdue days calculation**, but the result is then multiplied by `1.00` to return it as a numeric value with a decimal. This is typically done when the late charge calculation needs to be a floating-point number, possibly for further calculations.

Rental Rate Mode Graph: Slide 7

country_engagement_TABLE

(Same as Scatterplots: Slide 3)