Customer churn analytics banking industry

1. Churn Rate Calculation

This query calculates the overall churn rate by counting the number of churned customers (where Exited = 1) and comparing it to the total number of customers.

```
SELECT
    COUNT(*) FILTER (WHERE "Exited" = 1) * 100.0 / COUNT(*) AS
churn_rate_percentage,
    COUNT(*) FILTER (WHERE "Exited" = 0) * 100.0 / COUNT(*) AS
non_churn_rate_percentage
FROM
    bank;
```

2. Average Age of Churned vs. Non-Churned Customers

To identify if there is an age difference between churned and non-churned customers, we can calculate the average age for each group.

```
SELECT
    AVG("Age") FILTER (WHERE "Exited" = 1) AS avg_age_churned,
    AVG("Age") FILTER (WHERE "Exited" = 0) AS avg_age_non_churned
FROM
    bank;
```

3. Average Credit Score of Churned vs. Non-Churned Customers

This query calculates the average credit score for churned and non-churned customers to see if there's a difference.

```
SELECT
```

```
   AVG("CreditScore") FILTER (WHERE "Exited" = 1) AS
avg_credit_score_churned,
   AVG("CreditScore") FILTER (WHERE "Exited" = 0) AS
avg_credit_score_non_churned
FROM
   bank;
```

4. Average Balance of Churned vs. Non-Churned Customers

To check if balance has an impact on churn, this query provides the average balance for both churned and non-churned customers.

```
SELECT
   AVG("Balance") FILTER (WHERE "Exited" = 1) AS avg_balance_churned,
   AVG("Balance") FILTER (WHERE "Exited" = 0) AS avg_balance_non_churned
FROM
   bank;
```

5. Churn Rate by Geography

This query calculates the churn rate for each country in the Geography column.

```
SELECT
   "Geography",
   COUNT(*) FILTER (WHERE "Exited" = 1) * 100.0 / COUNT(*) AS
churn_rate_percentage
FROM
   bank
GROUP BY
   "Geography";
```

6. Churn Rate by Gender

This query determines the churn rate for male and female customers.

```
SELECT
   "Gender",
   COUNT(*) FILTER (WHERE "Exited" = 1) * 100.0 / COUNT(*) AS
churn_rate_percentage
FROM
   bank
GROUP BY
   "Gender";
```

7. Churn Rate by Number of Products

This query calculates the churn rate for each category of product usage (NumOfProducts), to determine if certain product usage levels are linked to higher churn rates.

```
SELECT
   "NumOfProducts",
   COUNT(*) FILTER (WHERE "Exited" = 1) * 100.0 / COUNT(*) AS
churn_rate_percentage
FROM
   bank
GROUP BY
   "NumOfProducts";
```

8. Churn Rate by Credit Card Ownership

To see if having a credit card impacts churn, this query provides the churn rate for customers with and without credit cards.

```
SELECT
   "HasCrCard",
   COUNT(*) FILTER (WHERE "Exited" = 1) * 100.0 / COUNT(*) AS
churn_rate_percentage
FROM
   bank
GROUP BY
   "HasCrCard";
```

9. Churn Rate by Active Member Status

This query shows the churn rate for active and inactive members, as engagement may influence churn likelihood.

```
SELECT
   "IsActiveMember",
   COUNT(*) FILTER (WHERE "Exited" = 1) * 100.0 / COUNT(*) AS
churn_rate_percentage
FROM
   bank
GROUP BY
   "IsActiveMember";
```

10. Summary Statistics for Credit Score and Balance

- **Mean: Provides the average value.**
- **Median: Shows the middle value, less sensitive to outliers than the mean.**
- **Min and Max: Define the range of the data.**
- **Quartiles (Q1 and Q3): Help identify the spread and potential outliers in the dataset.**
- **Standard Deviation: Measures variability and helps understand how dispersed the values are from the mean.**

11. Query to generate these statistics for CreditScore and Balance:

```
SELECT
   'CreditScore' AS metric,
   AVG("CreditScore") AS mean,
   PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY "CreditScore") AS
median,
   MIN("CreditScore") AS min,
   MAX("CreditScore") AS max,
   PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY "CreditScore") AS q1,
```

```
  PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY "CreditScore") AS q3,
  STDDEV("CreditScore") AS std_dev
FROM public.bank

UNION ALL

SELECT
  'Balance' AS metric,
  AVG("Balance") AS mean,
  PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY "Balance") AS median,
  MIN("Balance") AS min,
  MAX("Balance") AS max,
  PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY "Balance") AS q1,
  PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY "Balance") AS q3,
  STDDEV("Balance") AS std_dev
FROM public.bank;
```

12. Top 5 Customers Based on Balance

```
WITH ranked_customers AS (
    SELECT
        "CustomerId",
        "Surname",
        "Age",
        "Gender",
        "Geography",
        "CreditScore",
        "Balance",
        ROW_NUMBER() OVER (ORDER BY "Balance" DESC) AS
rank_by_balance
    FROM
        bank
)
SELECT *
FROM ranked_customers
WHERE rank_by_balance <= 5;
```

13. Top 5 High-Value Customers for Reduced Interest Rate

Similarly, we'll use a CTE to rank customers based on both Balance and Credit Score and then filter the top 5 in the outer query.

```
WITH ranked_customers AS (
    SELECT
        "CustomerId",
        "Surname",
        "Age",
        "Gender",
        "Geography",
        "CreditScore",
        "Balance",
        "IsActiveMember",
        "Exited" AS Churn_Status,
        ROW_NUMBER() OVER (ORDER BY "Balance" DESC,
"CreditScore" DESC) AS rank_important_customers
    FROM
        bank
)
SELECT *
FROM ranked_customers
WHERE rank_important_customers <= 5;
```