

## Step 1: Problem Definition

### Problem:

- Telecommunication problems on customers frequency in Rwanda

### Business context

- A telecommunication company in Rwanda (MTN or Airtel) wants to understand customer usage behavior to improve loyalty and revenue.

### Data challenge

- The company collects large numbers of call and data records but struggles to identify how frequently customers use services. It doesn't always come easy to separate active customers from inactive ones and detect whether their services is covering all the areas.
- Not forgetting to mention that many customers subscribe to their services but use them rarely due to cost, device limitations, or poor network quality hence this may lead to users switching to competitors or abandoning their SIM cards.

### Expected Outcome

A PL/SQL approach can help by:

- Calculating customer frequency scores, segments users (frequent, occasional, inactive), and provides insights to guide promotions and retention strategies.

## Step 2: Success Criteria

### 5 Measurable Goals

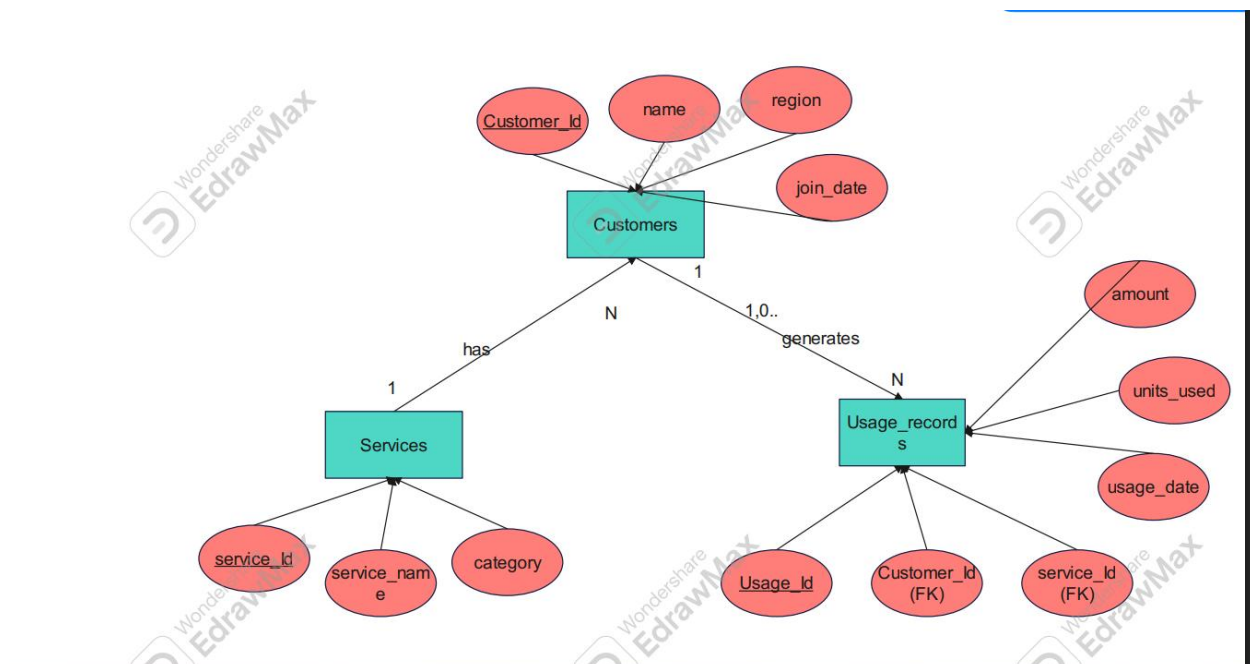
1. **Top 5 Services per Region/Quarter:** Identify the five most frequently used telecom services (Voice, SMS, data bundles, Mobile Money) by using RANK ()
2. **Running Monthly usage totals:** Track the cumulative number of customer interactions (calls, SMS,..etc) by using SUM () OVER (ORDER BY MONTH) to show growth in total.
3. **Month-over-Month Growth in Usage:** Measure the increase or decrease in customer service usage compared to the previous month, by using LAG () for comparison
4. **Customers Quartiles by Frequency:** Divide customers into four groups (frequent, regular, occasional, inactive) based on usage frequency by using NTILE (4) to segment customers.

5. **3-month Moving Average of Usage:** Running customers activity trends by averaging service usage over the current and last two months by using `AVG () OVER ()`.

### Step 3: Database Schema

Table	Purpose	Key Columns	Example Row
<b>Customers</b>	Stores customer info	Customer_id (PK), name, region, join_date	1001, Alice Uwase, Kigali, 2022-05-12
<b>Services</b>	Lists of telecommunication services	Service_id (PK), service_name, category (Voice, SMS, data, Mobile Money)	2001, Monthly Data Bundle, Data
<b>Usage records</b>	Tracks customer service usage	Usage_id (PK), customer_id (FK), service_id (FK), usage_date, units_used, amount	3001,1001, 2001, 2022-05-20, 2GB, 2000

### ER Diagram



✓ **Customers Table**

```
CREATE TABLE customers (  
    customer_id NUMBER PRIMARY KEY,  
    name VARCHAR2(100),  
    region VARCHAR2(50),  
    join_date DATE  
);
```

- **Inserting data**

```
INSERT INTO customers VALUES (1001, 'Alice Uwase', 'Kigali', TO_DATE('2023-06-12','YYYY-MM-DD'));
```

```
INSERT INTO customers VALUES (1002, 'Jean Bosco', 'Musanze', TO_DATE('2023-07-05','YYYY-MM-DD'));
```

```
INSERT INTO customers VALUES (1003, 'Aline Mukamana', 'Huye', TO_DATE('2023-05-20','YYYY-MM-DD'));
```

```
INSERT INTO customers VALUES (1004, 'Eric Nshimiyimana', 'Kigali', TO_DATE('2023-08-15','YYYY-MM-DD'));
```

```
INSERT INTO customers VALUES (1005, 'Sandra Uwitonze', 'Rubavu', TO_DATE('2023-04-10','YYYY-MM-DD'));
```

```
C:\WINDOWS\system32\cmd. x + v
Copyright (c) 1982, 2005, Oracle. All rights reserved.

Enter user-name: system
Enter password:

Connected to:
Oracle Database 10g Express Edition Release 10.2.0.1.0 - Production

SQL> INSERT INTO customers VALUES (1001, 'Alice Uwase', 'Kigali', TO_DATE('2023-06-12','YYYY-MM-DD'));

1 row created.

SQL> INSERT INTO customers VALUES (1002, 'Jean Bosco', 'Musanze', TO_DATE('2023-07-05','YYYY-MM-DD'));

1 row created.

SQL> INSERT INTO customers VALUES (1003, 'Aline Mukamana', 'Huye', TO_DATE('2023-05-20','YYYY-MM-DD'));

1 row created.

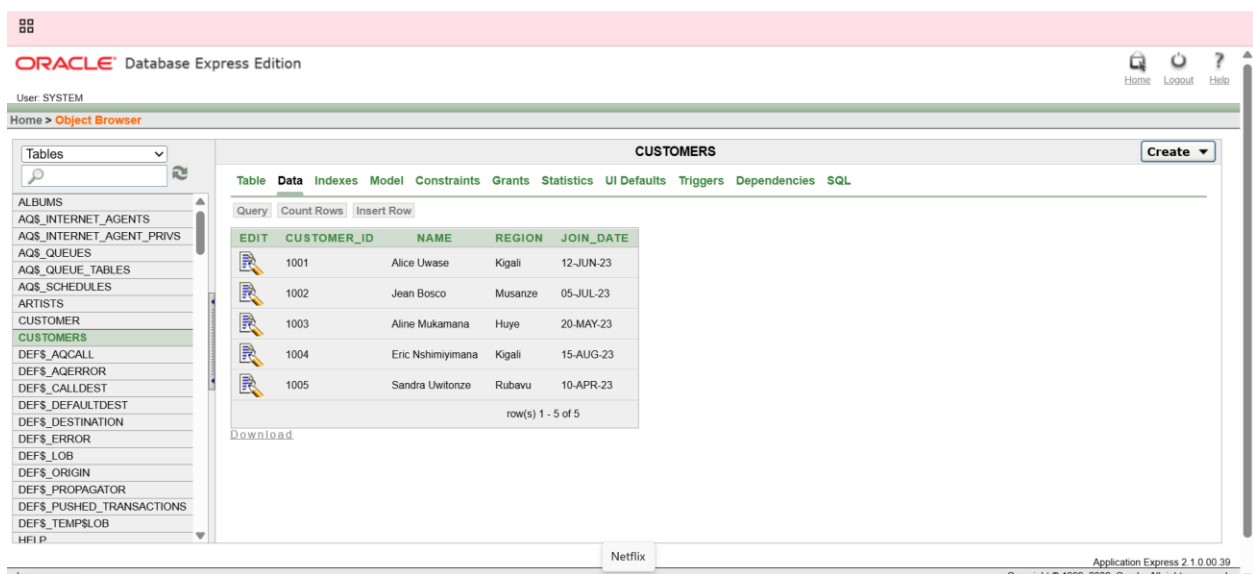
SQL> INSERT INTO customers VALUES (1004, 'Eric Nshimiyanana', 'Kigali', TO_DATE('2023-08-15','YYYY-MM-DD'));

1 row created.

SQL> INSERT INTO customers VALUES (1005, 'Sandra Uwitonze', 'Rubavu', TO_DATE('2023-04-10','YYYY-MM-DD'));

1 row created.

SQL> |
```



## ✓ Services Table

## CREATE TABLE services (

service id NUMBER PRIMARY KEY,

```
service_name VARCHAR2(100),
```

category VARCHAR2(50)

);

- **Inserting data**

INSERT INTO services VALUES (2001, 'Monthly Data Bundle', 'Data');

INSERT INTO services VALUES (2002, 'Daily Voice Pack', 'Voice');

INSERT INTO services VALUES (2003, 'SMS Pack', 'SMS');

INSERT INTO services VALUES (2004, 'Mobile Money Transfer', 'Mobile Money');

INSERT INTO services VALUES (2005, 'Weekend Data Bundle', 'Data');

```
C:\WINDOWS\system32\cmd. x + v

SQL> CREATE TABLE services (
  2     service_id NUMBER PRIMARY KEY,
  3     service_name VARCHAR2(100),
  4     category VARCHAR2(50)
  5 );

Table created.

SQL> INSERT INTO services VALUES (2001, 'Monthly Data Bundle', 'Data');

1 row created.

SQL> INSERT INTO services VALUES (2002, 'Daily Voice Pack', 'Voice');

1 row created.

SQL> INSERT INTO services VALUES (2003, 'SMS Pack', 'SMS');

1 row created.

SQL> INSERT INTO services VALUES (2004, 'Mobile Money Transfer', 'Mobile Money');

1 row created.

SQL> INSERT INTO services VALUES (2005, 'Weekend Data Bundle', 'Data');

1 row created.

SQL>
```

ORACLE Database Express Edition

User: SYSTEM

Home > Object Browser

Tables

ALBUMS  
AQ\$\_INTERNET\_AGENTS  
AQ\$\_INTERNET\_AGENT\_PRIVS  
AQ\$\_QUEUES  
AQ\$\_QUEUE\_TABLES  
AQ\$\_SCHEDULES  
ARTISTS  
CUSTOMER  
**CUSTOMERS**  
DEF\$\_AQCALL  
DEF\$\_AQERROR  
DEF\$\_CALLDEST  
DEF\$\_DEFAULTDEST  
DEF\$\_DESTINATION  
DEF\$\_ERROR  
DEF\$\_LOB  
DEF\$\_ORIGIN  
DEF\$\_PROPAGATOR  
DEF\$\_PUSHED\_TRANSACTIONS  
DEF\$\_TEMP\$LOB  
HFIP

**CUSTOMERS**

Table Data Indexes Model Constraints Grants Statistics UI Defaults Triggers Dependencies SQL

Query Count Rows Insert Row

EDIT	CUSTOMER_ID	NAME	REGION	JOIN_DATE
	1001	Alice Uwase	Kigali	12-JUN-23
	1002	Jean Bosco	Musanze	05-JUL-23
	1003	Aline Mukamana	Huye	20-MAY-23
	1004	Eric Nshimiimana	Kigali	15-AUG-23
	1005	Sandra Uwitonze	Rubavu	10-APR-23

row(s) 1 - 5 of 5

Download

Netfix

Application Express 2.1.0.00.39

## ✓ Usage Records Table

```
CREATE TABLE usage_records (  
    usage_id NUMBER PRIMARY KEY,  
    customer_id NUMBER REFERENCES customers(customer_id),  
    service_id NUMBER REFERENCES services(service_id),  
    usage_date DATE,  
    units_used VARCHAR2(50),  
    amount NUMBER  
);
```

- **Inserting data**

```
INSERT INTO usage_records VALUES (3001, 1001, 2001, TO_DATE('2024-01-15','YYYY-MM-DD'), '2GB', 2500);  
INSERT INTO usage_records VALUES (3002, 1002, 2002, TO_DATE('2024-01-16','YYYY-MM-DD'), '60 minutes', 1500);  
INSERT INTO usage_records VALUES (3003, 1003, 2003, TO_DATE('2024-01-17','YYYY-MM-DD'), '100 SMS', 800);  
INSERT INTO usage_records VALUES (3004, 1004, 2004, TO_DATE('2024-01-18','YYYY-MM-DD'), 'RWF 5000', 5000);  
INSERT INTO usage_records VALUES (3005, 1005, 2005, TO_DATE('2024-01-19','YYYY-MM-DD'), '5GB', 6000);
```

C:\WINDOWS\system32\cmd. X + v

```

SQL> CREATE TABLE usage_records (
2     usage_id NUMBER PRIMARY KEY,
3     customer_id NUMBER REFERENCES customers(customer_id),
4     service_id NUMBER REFERENCES services(service_id),
5     usage_date DATE,
6     units_used VARCHAR2(50),
7     amount NUMBER
8 );

Table created.

SQL> INSERT INTO usage_records VALUES (3001, 1001, 2001, TO_DATE('2024-01-15','YYYY-MM-DD'), '2GB', 2500);

1 row created.

SQL> INSERT INTO usage_records VALUES (3002, 1002, 2002, TO_DATE('2024-01-16','YYYY-MM-DD'), '60 minutes', 1500);

1 row created.

SQL> INSERT INTO usage_records VALUES (3003, 1003, 2003, TO_DATE('2024-01-17','YYYY-MM-DD'), '100 SMS', 800);

1 row created.

SQL> INSERT INTO usage_records VALUES (3004, 1004, 2004, TO_DATE('2024-01-18','YYYY-MM-DD'), 'RWF 5000', 5000);

1 row created.

SQL> INSERT INTO usage_records VALUES (3005, 1005, 2005, TO_DATE('2024-01-19','YYYY-MM-DD'), '5GB', 6000);

```

127.0.0.1:8080/apex/?p=4500:1001:3554010114197700:NO::

ORACLE Database Express Edition

User SYSTEM

Home > Object Browser

Tables

REPCATS\_REPCOLUMN  
REPCATS\_REPGROUP\_PRIVS  
REPCATS\_REPOBJECT  
REPCATS\_REPPROP  
REPCATS\_REPSHEMA  
REPCATS\_RESOLUTION  
REPCATS\_RESOLUTION\_METHOD  
REPCATS\_RESOLUTION\_STATISTI  
REPCATS\_RESOL\_STATS\_CONTR  
REPCATS\_RUNTIME\_PARM  
REPCATS\_SITES\_NEW  
REPCATS\_SITE\_OBJECTS  
REPCATS\_SNAPGROUP  
REPCATS\_TEMPLATE\_OBJECTS  
REPCATS\_TEMPLATE\_PARM  
REPCATS\_TEMPLATE\_REFGROUP  
REPCATS\_TEMPLATE\_SITES  
REPCATS\_TEMPLATE\_STATUS  
REPCATS\_TEMPLATE\_TARGETS  
REPCATS\_TEMPLATE\_TYPES  
REPCATS\_USER\_AUTHORIZATION

USAGE\_RECORDS

Create

Table Data Indexes Model Constraints Grants Statistics UI Defaults Triggers Dependencies SQL

Query Count Rows Insert Row

EDIT	USAGE_ID	CUSTOMER_ID	SERVICE_ID	USAGE_DATE	UNITS_USED	AMOUNT
	3001	1001	2001	15-JAN-24	2GB	2500
	3002	1002	2002	16-JAN-24	60 minutes	1500
	3003	1003	2003	17-JAN-24	100 SMS	800
	3004	1004	2004	18-JAN-24	RWF 5000	5000
	3005	1005	2005	19-JAN-24	5GB	6000

Download

row(s) 1 - 5 of 5

Application Express 2.1.0.00.39

## Step 4: Windows Functions Implementation

### 1. Ranking

SELECT

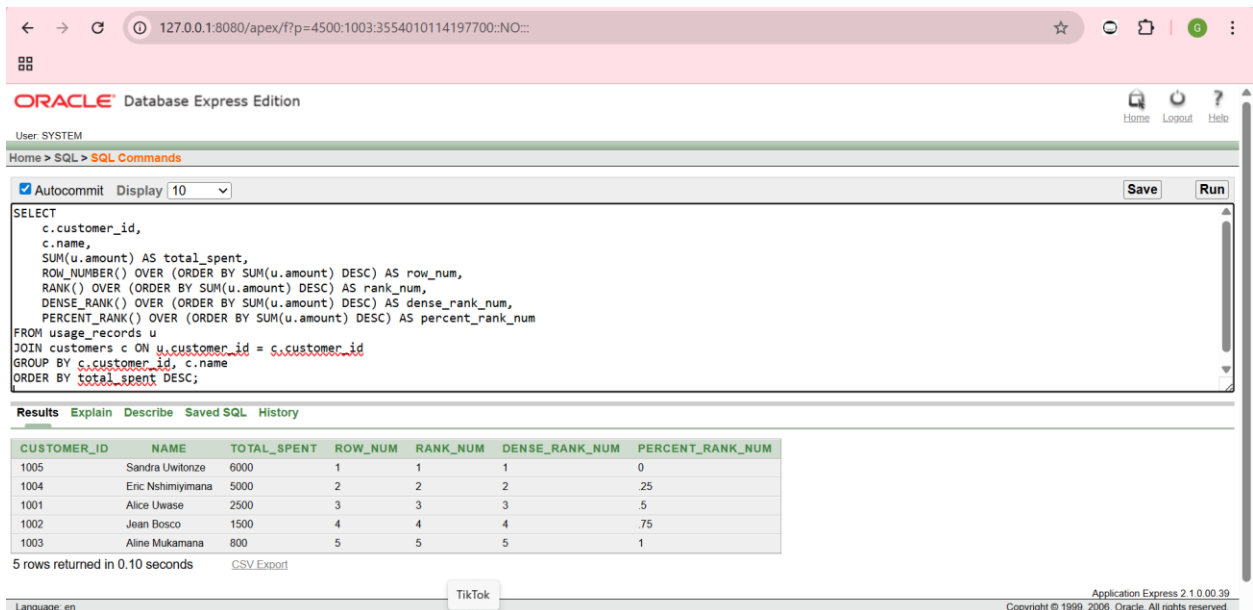
c.customer\_id,

c.name,

```

SUM(u.amount) AS total_spent,
ROW_NUMBER() OVER (ORDER BY SUM(u.amount) DESC) AS row_num,
RANK() OVER (ORDER BY SUM(u.amount) DESC) AS rank_num,
DENSE_RANK() OVER (ORDER BY SUM(u.amount) DESC) AS dense_rank_num,
PERCENT_RANK() OVER (ORDER BY SUM(u.amount) DESC) AS percent_rank_num
FROM usage_records u
JOIN customers c ON u.customer_id = c.customer_id
GROUP BY c.customer_id, c.name
ORDER BY total_spent DESC;

```



The screenshot shows the Oracle Database Express Edition interface. The SQL command window contains the following query:

```

SELECT
  c.customer_id,
  c.name,
  SUM(u.amount) AS total_spent,
  ROW_NUMBER() OVER (ORDER BY SUM(u.amount) DESC) AS row_num,
  RANK() OVER (ORDER BY SUM(u.amount) DESC) AS rank_num,
  DENSE_RANK() OVER (ORDER BY SUM(u.amount) DESC) AS dense_rank_num,
  PERCENT_RANK() OVER (ORDER BY SUM(u.amount) DESC) AS percent_rank_num
FROM usage_records u
JOIN customers c ON u.customer_id = c.customer_id
GROUP BY c.customer_id, c.name
ORDER BY total_spent DESC;

```

The query was executed successfully, returning 5 rows in 0.10 seconds. The results are displayed in a table with the following columns: CUSTOMER\_ID, NAME, TOTAL\_SPENT, ROW\_NUM, RANK\_NUM, DENSE\_RANK\_NUM, and PERCENT\_RANK\_NUM.

CUSTOMER_ID	NAME	TOTAL_SPENT	ROW_NUM	RANK_NUM	DENSE_RANK_NUM	PERCENT_RANK_NUM
1005	Sandra Uwilonze	6000	1	1	1	0
1004	Eric Nshimiyimana	5000	2	2	2	.25
1001	Alice Uwase	2500	3	3	3	.5
1002	Jean Bosco	1500	4	4	4	.75
1003	Aline Mukamana	800	5	5	5	1

- The ranking query shows which customers contribute the most revenue allowing the company to identify the top spenders and compare them even when the totals are tied.

## 2. Aggregate

```

SELECT
TO_CHAR(u.usage_date, 'YYYY-MM') AS month,
SUM(u.amount) AS monthly_total,
SUM(SUM(u.amount)) OVER (

```



```
        ORDER BY TO_CHAR(u.usage_date, 'YYYY-MM')
        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
    ) AS running_total,
    AVG(SUM(u.amount)) OVER (
        ORDER BY TO_CHAR(u.usage_date, 'YYYY-MM')
        ROWS BETWEEN 2 PRECEDING AND CURRENT ROW
    ) AS three_month_avg,
    MIN(SUM(u.amount)) OVER (
        ORDER BY TO_CHAR(u.usage_date, 'YYYY-MM')
        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
    ) AS min_total,
    MAX(SUM(u.amount)) OVER (
        ORDER BY TO_CHAR(u.usage_date, 'YYYY-MM')
        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
    ) AS max_total
FROM usage_records u
GROUP BY TO_CHAR(u.usage_date, 'YYYY-MM')
ORDER BY month;
```

The screenshot shows the Oracle Database Express Edition interface. The SQL Command window contains the following query:

```
SELECT
  TO_CHAR(u.usage_date, 'YYYY-MM') AS month,
  SUM(u.amount) AS monthly_total,
  SUM(SUM(u.amount)) OVER (
    ORDER BY TO_CHAR(u.usage_date, 'YYYY-MM')
    ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
  ) AS running_total,
  AVG(SUM(u.amount)) OVER (
    ORDER BY TO_CHAR(u.usage_date, 'YYYY-MM')
    ROWS BETWEEN 2 PRECEDING AND CURRENT ROW
  ) AS three_month_avg,
  MIN(SUM(u.amount)) OVER (
    ORDER BY TO_CHAR(u.usage_date, 'YYYY-MM')
    ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
  ) AS min_total,
  MAX(SUM(u.amount)) OVER (
    ORDER BY TO_CHAR(u.usage_date, 'YYYY-MM')
    ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
  ) AS max_total
FROM user_transactions u
```

The Results tab shows the following data:

MONTH	MONTHLY_TOTAL	RUNNING_TOTAL	THREE_MONTH_AVG	MIN_TOTAL	MAX_TOTAL
2024-01	15800	15800	15800	15800	15800

1 rows returned in 0.05 seconds. CSV Export

- The aggregate help track spending trends over time, giving insights into whether overall customer usage is increasing, decreasing or stable.

### 3. Navigation

SELECT

TO\_CHAR(u.usage\_date, 'YYYY-MM') AS month,

SUM(u.amount) AS monthly\_total,

LAG(SUM(u.amount)) OVER (ORDER BY TO\_CHAR(u.usage\_date, 'YYYY-MM')) AS  
prev\_month\_total,

CASE

WHEN LAG(SUM(u.amount)) OVER (ORDER BY TO\_CHAR(u.usage\_date, 'YYYY-MM')) IS NULL

THEN NULL

ELSE ROUND(

(SUM(u.amount) - LAG(SUM(u.amount)) OVER (ORDER BY  
TO\_CHAR(u.usage\_date, 'YYYY-MM'))

/ LAG(SUM(u.amount)) OVER (ORDER BY TO\_CHAR(u.usage\_date, 'YYYY-MM')) \*  
100, 2

)

```

END AS growth_percent

FROM usage_records u

GROUP BY TO_CHAR(u.usage_date, 'YYYY-MM')

ORDER BY month;

```

The screenshot shows the Oracle Database Express Edition interface. The SQL Command window contains the following query:

```

SELECT
  TO_CHAR(u.usage_date, 'YYYY-MM') AS month,
  SUM(u.amount) AS monthly_total,
  LAG(SUM(u.amount)) OVER (ORDER BY TO_CHAR(u.usage_date, 'YYYY-MM')) AS prev_month_total,
  CASE
    WHEN LAG(SUM(u.amount)) OVER (ORDER BY TO_CHAR(u.usage_date, 'YYYY-MM')) IS NULL
    THEN NULL
    ELSE ROUND(
      (SUM(u.amount) - LAG(SUM(u.amount)) OVER (ORDER BY TO_CHAR(u.usage_date, 'YYYY-MM')))
      / LAG(SUM(u.amount)) OVER (ORDER BY TO_CHAR(u.usage_date, 'YYYY-MM')) * 100, 2
    )
  END AS growth_percent

```

The Results tab shows the following data:

MONTH	MONTHLY_TOTAL	PREV_MONTH_TOTAL	GROWTH_PERCENT
2024-01	15800	-	-

1 rows returned in 0.00 seconds. CSV Export

- Measuring growth or decline in usage, helping to identify seasonal changes or sudden changes in customer activity.

## 4. Distribution

```

SELECT

  c.customer_id,

  c.name,

  SUM(u.amount) AS total_spent,

  NTILE(4) OVER (ORDER BY SUM(u.amount) DESC) AS spending_quartile,

  CUME_DIST() OVER (ORDER BY SUM(u.amount) DESC) AS cume_distribution

FROM usage_records u

JOIN customers c ON u.customer_id = c.customer_id

GROUP BY c.customer_id, c.name

```

ORDER BY total\_spent DESC;

The screenshot shows the Oracle Database Express Edition interface. The browser address bar displays the URL: 127.0.0.1:8080/apex/?p=4500:1003:3554010114197700:NO::: The page title is "ORACLE Database Express Edition". The user is logged in as "SYSTEM". The navigation bar shows "Home > SQL > SQL Commands". The "Autocommit" checkbox is checked, and the "Display" dropdown is set to "10". The "Save" and "Run" buttons are visible. The SQL query is as follows:

```
SELECT
  c.customer_id,
  c.name,
  SUM(u.amount) AS total_spent,
  NTILE(4) OVER (ORDER BY SUM(u.amount) DESC) AS spending_quartile,
  CUME_DIST() OVER (ORDER BY SUM(u.amount) DESC) AS cume_distribution
FROM usage_records u
JOIN customers c ON u.customer_id = c.customer_id
GROUP BY c.customer_id, c.name
ORDER BY total_spent DESC;
```

The results are displayed in a table with the following columns: CUSTOMER\_ID, NAME, TOTAL\_SPENT, SPENDING\_QUARTILE, and CUME\_DISTRIBUTION. The table contains 5 rows of data.

CUSTOMER_ID	NAME	TOTAL_SPENT	SPENDING_QUARTILE	CUME_DISTRIBUTION
1005	Sandra Uwitonze	6000	1	2
1004	Eric Nshimiyimana	5000	1	4
1001	Alice Uwase	2500	2	6
1002	Jean Bosco	1500	3	8
1003	Aline Mukamana	800	4	1

5 rows returned in 0.03 seconds. CSV Export

Language: en

Show hidden icons

Application Express 2.1.0.00.39  
© 2006, Oracle. All rights reserved.

- Customer spending is segmented into quartiles, showing which group spends the most versus the least, which supports targeted marketing and retention strategies.