Names: Samuel NIYONSHUTI

ID: 28329

VEHICLE CUSTOMISATION DATABASE DESIGN

Step 1: Problem Definition

Business Context: A leading automotive manufacturer's customization department wants to analyze customer preferences and buying patterns to improve their personalized vehicle offerings.

Data Challenge: The department needs to identify the most popular customization options by vehicle model(e.g., interior upgrades, exterior paint, performance enhancements), understand how frequently customers make additional modifications, and segment customers based on their customization behavior.

Expected Outcome: The analysis should provide insights to help the department optimize their customization packages, target marketing campaigns, and enhance the overall customer experience.

The analysis should provide actionable insights to:

1. Identify the top customization options by vehicle type and customer demographic.
2. Analyze customer return frequency for additional modifications.
3. Segment customers based on their customization preferences to create targeted marketing campaigns.

Step 2: Success Criteria

To measure the success of the vehicle customization business, the company has defined the following 5 key goals:

1. **Top Customization Products per Region/Quarter**: Rank the top most popular customization products:
   - high-performance tires
   - Carbon fiber trim
   - Custom paint
   - Premium interiors

2. **Running Monthly Sales Totals**: Track the sum of monthly sales for customized vehicles to monitor overall revenue and growth.

3. **Month-over-Month Growth**: Analyze the percentage change in sales from one month to the next to assess whether demand for customization services is increasing or decreasing over time.
4. **Customer Quartiles**: Segment customers into quartiles based on the number of customisation purchases to understand the distribution of high-value, repeat customers and target them with personalized marketing campaigns.
5. **3-Month Moving Averages**: Calculate the 3-month moving average of customization sales to smooth out any short-term fluctuations and identify longer-term trends in customization demand.

Step 3: Database Schema

Customer info

Column	Type	Example row
Customer_id	VARCHAR(20)	c001
Name	VARCHAR(50)	Darius
Region	VARCHAR(50)	Kigali

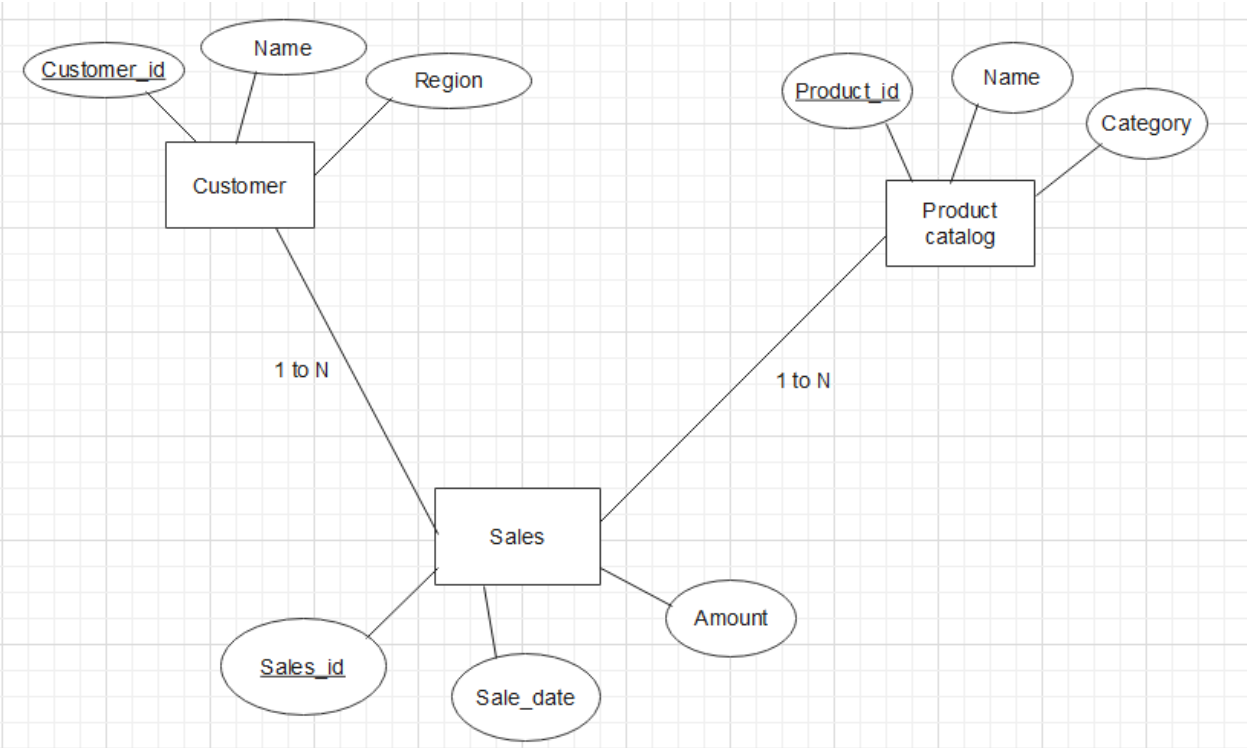
Product catalog

Column	Type	Example row
Product_id	VARCHAR(20)	001
Product_name	VARCHAR(50)	Exhaust systems
Category	VARCHAR(50)	Performance enhancement

Sales

Column	Type	Example row
Sales_id	VARCHAR(20)	010101
Customer_id	VARCHAR(20)	111
Product_id	VARCHAR(20)	001
Sales_date	VARCHAR(50)	21st Sept 2025
Amount	INTEGER	1,000$

ER DIAGRAM



The next step is to create tables, insert data and keep relation between them. The following commands have been written using Postgre.

- Creation of tables

```sql
--create customer info table
CREATE TABLE Customer (
Customer_id VARCHAR(20) PRIMARY KEY,
Name VARCHAR(50) NOT NULL,
Region VARCHAR(20)
);

--create Product catalog table
CREATE TABLE Product (
Product_id VARCHAR(20) PRIMARY KEY,
Product_name VARCHAR(50) NOT NULL,
Category VARCHAR(20)
);

CREATE TABLE Sales (
Sales_id VARCHAR(20) PRIMARY KEY,
Customer_id VARCHAR(20) REFERENCES Customer(Customer_id),
Product_id VARCHAR(20) REFERENCES Product(Product_id),
Sales_date VARCHAR(50) NOT NULL,
Amount INTEGER
);
```

- Insertion of data in tables

1. Customer table

```sql
INSERT INTO Customer (Customer_id, Name, Region)
VALUES ('c001', 'Darius', 'Gahanga');

INSERT INTO Customer (Customer_id, Name, Region)
VALUES ('c002', 'Musa', 'Kacyiru');

INSERT INTO Customer (Customer_id, Name, Region)
VALUES ('c003', 'Remy', 'Remera');

INSERT INTO Customer (Customer_id, Name, Region)
VALUES ('c004', 'Juste', 'Gisozi');
```

Table structure with data

```sql
SELECT*
From Customer;
```

	customer_id [PK] character varying (20)	name character varying (50)	region character varying (20)
1	c001	Darius	Gahanga
2	c002	Musa	Kacyiru
3	c003	Remy	Remera
4	c004	Juste	Gisozi

2. Product table

```
16  INSERT INTO Product (Product_id, Product_name, Category)
17  VALUES ('p111', 'wheels and tires', 'Exterior mods');
18
19  INSERT INTO Product (Product_id, Product_name, Category)
20  VALUES ('p222', 'Sound systems', 'Interior mods');
21
22  INSERT INTO Product (Product_id, Product_name, Category)
23  VALUES ('p333', 'Exhaust systems', 'Performance enhance');
24
25  INSERT INTO Product (Product_id, Product_name, Category)
26  VALUES ('p444', 'Towing equip', 'Utility mods');
27
```

Table structure with data

```
SELECT*
FROM Product
```

	product_id [PK] character varying (20)	product_name character varying (50)	category character varying (20)
1	p111	wheels and tires	Exterior mods
2	p222	Sound systems	Interior mods
3	p333	Exhaust systems	Performance enhance
4	p444	Towing equip	Utility mods

3. Sales table

```
INSERT INTO Sales (Sales_id, Customer_id, Product_id, Sales_date, Amount)
VALUES ('#1', 'c001', 'p444', '2025-07-20', 1000);

INSERT INTO Sales (Sales_id, Customer_id, Product_id, Sales_date, Amount)
VALUES ('#2', 'c002', 'p333', '2025-08-20', 4000);

INSERT INTO Sales (Sales_id, Customer_id, Product_id, Sales_date, Amount)
VALUES ('#3', 'c003', 'p222', '2025-09-10', 12000);

INSERT INTO Sales (Sales_id, Customer_id, Product_id, Sales_date, Amount)
VALUES ('#4', 'c004', 'p111', '2025-09-20', 7000);
```

Table structure with data

```
SELECT*
FROM Sales
```

	sales_id [PK] character varying (20)	customer_id character varying (20)	product_id character varying (20)	sales_date character varying (50)	amount integer
1	#1	c001	p444	2025-07-20	1000
2	#2	c002	p333	2025-08-20	4000
3	#3	c003	p222	2025-09-10	12000
4	#4	c004	p111	2025-09-20	7000

Step 4: Window functions implementation

Window functions are a type of SQL function that perform a calculation across a set of rows that are somehow related to the current row

1.Ranking

Ranking is a way to assign numerical position to each rwo within a result se based on a specified ordering.

i. ROW_NUMBER ()

Input

```
64  -- Rank Customers by total revenue using ROW_NUMBER() from high to lower
65  SELECT
66  Customer_id,
67  SUM(Amount) As total_revenue,
68  ROW_NUMBER() OVER (ORDER BY SUM(amount) DESC) AS row_num
69  FROM Sales
70  GROUP BY Customer_id;
```

Output

	customer_id character varying (20)	total_revenue bigint	row_num bigint
1	c003	12000	1
2	c004	7000	2
3	c002	4000	3
4	c001	1000	4

Interpretation

The table shows customer revenue data, with the customer 'c003' generating the highest total revenue of 12,000, followed by 'c004' at 7,000 and 'c002' at 4,000. The data could be used to understand the relative importance of customers and inform business decisions.

ii. RANK ()

Input

```
73  --Rank Customers by total revenue using RANK
74  SELECT
75  Customer_id,
76  SUM(Amount) As total_revenue,
77  RANK() OVER (ORDER BY SUM(amount) DESC) AS rank_pos
78  FROM Sales
79  GROUP BY Customer_id;
```

Output

	customer_id character varying (20)	total_revenue bigint	rank_pos bigint
1	c003	12000	1
2	c004	7000	2
3	c002	4000	3
4	c001	1000	4

Interpretation

This function assigns a unique, sequential number to each row based on the total revenue in descending order.

iii. PERCENT_RANK ()

Input

```
82  --Rank Customers by total revenue in PERCENT_RANK
83  SELECT
84  Customer_id,
85  SUM(Amount) As total_revenue,
86  PERCENT_RANK() OVER (ORDER BY SUM(amount) DESC) AS percent_rank
87  FROM Sales
88  GROUP BY Customer_id;
```

Output

	customer_id character varying (20)	total_revenue bigint	percent_rank double precision
1	c003	12000	0
2	c004	7000	0.3333333333333333
3	c002	4000	0.6666666666666666
4	c001	1000	1

Interpretation

The table shows the percent rank of each customer's total revenue, showing their relative positions in the customer revenue hierarchy.

2.Aggregate

Aggregation is the process of computing a single value from set of values

i. SUM ()

Input

```
--Running total of sales ordered by date
SELECT
sales_id,
customer_id,
sales_date,
amount,
SUM(amount) OVER (
ORDER BY sales_date
ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
) AS running_total
```

Output

	sales_id [PK] character varying (20)	customer_id character varying (20)	sales_date character varying (50)	amount integer	running_total bigint
1	#1	c001	2025-07-20	1000	1000
2	#2	c002	2025-08-20	4000	5000
3	#3	c003	2025-09-10	12000	17000
4	#4	c004	2025-09-20	7000	24000

Interpretation

The data indicates that customer 'c003' had the largest single sale of 12,000, while customer 'c004' had the highest running total sales of 24,000 by the end of the period shown.

ii. AVERAGE ()

Input

```
SELECT
  customer_id,
 FLOOR(AVG(amount)) AS avg_sale
FROM sales
GROUP BY customer_id;
```

Output

	customer_id character varying (20)	avg_sale numeric
1	c001	1000
2	c004	7000
3	c003	12000
4	c002	4000

Interpretation

The table shows the average sale value for each customer. Customer 'c003' has the

highest average sale of 12,000, followed by 'c004' at 7,000, 'c002' at 4,000, and 'c001' at 1,000. This data could be used to understand the relative importance and purchasing power of each customer.

iii. MIN () & MAX ()

Input

```
118  --Calculating the minimum and maximum sales
119  SELECT
120  MIN(amount) AS minimumsales,
121  MAX(amount) AS maximumsales
122  From sales;
```

Output

	minimumsales integer	maximumsales integer
1	1000	12000

Interpretation

This information can be useful for understanding the range of sales amounts and identifying any outliers or unusual transactions.

3.Navigation

Navigation is the process of traversing a database's structure to locate and retrieve specific data

i. LAG ()

```
125  --Compare each sale with the previous sale by periods
126  SELECT
127  sales_id,
128  customer_id,
129  sales_date,
130  amount,
131  LAG(amount, 1) OVER (ORDER BY sales_date) AS previous_sale,
132  amount - LAG(amount, 1) OVER (ORDER BY sales_date) AS difference
```

Output

	sales_id [PK] character varying (20)	customer_id character varying (20)	sales_date character varying (50)	amount integer	previous_sale integer	difference integer
1	#1	c001	2025-07-20	1000	[null]	[null]
2	#2	c002	2025-08-20	4000	1000	3000
3	#3	c003	2025-09-10	12000	4000	8000
4	#4	c004	2025-09-20	7000	12000	-5000

Interpretation

The data indicates that customer 'c003' had the largest single sale of 12,000, while customer 'c004' had the highest running total sales of 24,000 by the end of the period shown.

ii. LEAD ()

Input

```
136  --compare each sale with the next sale by period
137  SELECT
138    sales_id,
139    customer_id,
140    sales_date,
141    amount,
142    LEAD(amount, 1) OVER (ORDER BY sales_date) AS next_sale,
143    LEAD(amount, 1) OVER (ORDER BY sales_date) - amount AS difference
144  FROM sales
145  ORDER BY sales_date;
```

Output

	sales_id [PK] character varying (20)	customer_id character varying (20)	sales_date character varying (50)	amount integer	next_sale integer	difference integer
1	#1	c001	2025-07-20	1000	4000	3000
2	#2	c002	2025-08-20	4000	12000	8000
3	#3	c003	2025-09-10	12000	7000	-5000
4	#4	c004	2025-09-20	7000	[null]	[null]

Interpretation

his provides additional context on the sales trends, indicating that customer 'c002'

had the highest next sale of 12,000, while customer 'c003' had a negative difference between the current and next sale, suggesting a decrease in sales.

4.Distribution

involves splitting and storing data across multiple physical or logical locations to improve performance, scalability, and fault tolerance.

i.NTILE(4)

Input

```
148  --segment customers into 4 groups (1/4) by amount
149  SELECT
150  customer_id,
151  SUM(amount) AS total_revenue,
152  NTILE(4) OVER (ORDER BY SUM(amount) DESC) AS quartile
153  FROM sales
154  GROUP BY customer_id
155  ORDER BY total_revenue DESC;
```

Output

	customer_id character varying (20)	total_revenue bigint	quartile integer
1	c003	12000	1
2	c004	7000	2
3	c002	4000	3
4	c001	1000	4

Interpretation

Customer 'c003' had the highest total revenue of 12,000, placing them in the top quartile. Customers 'c004', 'c002', and 'c001' are in the 2nd, 3rd, and 4th quartiles respectively, based on their total revenue amounts.

ii. CUME_DIST ()

Input

```
158  --cumulative distribution of customers by revenue
159  SELECT
160  customer_id,
161  SUM(amount) AS total_revenue,
162  CUME_DIST() OVER (ORDER BY SUM(amount) DESC) AS cum_dist
163  FROM sales
164  GROUP BY customer_id
165  ORDER BY total_revenue DESC;
```

Output

	customer_id character varying (20)	total_revenue bigint	cum_dist double precision
1	c003	12000	0.25
2	c004	7000	0.5
3	c002	4000	0.75
4	c001	1000	1

Interpretation

Customer 'c001' has the highest cumulative distribution of 1.0, indicating they account for 100% of the total revenue. Customers 'c002' and 'c004' have cumulative distributions of 0.75 and 0.5 respectively, suggesting they account for 75% and 50% of the total revenue.

Step 6: Results analysis and interpretation

1.Descriptive Layer

- The data shows sales records for 4 customers (c001, c002, c003, c004) with different product IDs (p444, p333, p222, p111) and sales amounts ranging from 1000 to 12000.
- Customer c003 had the highest single sale of 12000 on 2025-09-10, while customer c004 had the second highest sale of 7000 on 2025-09-20.
- Customers c001 and c002 had lower sales amounts of 1000 and 4000 respectively.

2. Diagnostic Layer

- The varying sales amounts across customers suggest differences in their purchasing power, product preferences, or sales strategies.
- The timing of the sales indicates potential seasonal or promotional factors influencing the purchase behavior.
- The diverse product IDs imply the company may offer a wide range of products to cater to different customer needs.

3.Prescriptive Layer

- Analyze the customer purchasing patterns and preferences to optimize product offerings and marketing strategies.
- Investigate the factors driving the high sales for c003 and c004 to replicate the success with other customers.
- Continuously monitor sales data to identify emerging trends and adjust business plans accordingly.

References

- TechTFQ. (2023). Advanced SQL Window Functions Explained [Video]. YouTube. https://www.youtube.com/watch?v=Ww71knvhQ-s
- PostgreSQL.org. (2024). SQL Syntax: Window Functions. Retrieved from https://www.postgresql.org/about/news/postgresql-18-rc-1-released-3130/
- PostgreSQL.org. (2024). Window Functions Documentation. Retrieved from https://www.postgresql.org/docs/current/tutorial-window.html
- Oracle Corporation. (2024). PL/SQL Language Reference. Oracle Documentation.
- W3Schools. (2024). SQL Window Functions. Retrieved from https://www.w3schools.com/sql/sql_window%20functions.asp
- Maniraguha, E. (2025). Database Development with PL/SQL - Lecture 02: Introduction to GitHubs. AUCA.
- Maniraguha, E. (2025). Database Development with PL/SQL - Lecture 01: Introduction to SQL Command Basics (Recap). AUCA.
- Githubs tutorial [video] on Youtube. https://www.youtube.com/watch?v=iv8rSLsi1xo&pp=ygUfaG93IHRvIHVzZSBnaXRodWIgZm9yIGJlZ2lubmVycw%3D%3D
- Window functions tutorial [video] on Youtube. https://www.youtube.com/watch?v=rIcB4zMYMas&pp=ygUUd2luZG93IGZ1bmN0aW9ucyBzcWw%3D
- Window function in SQL on https://www.geeksforgeeks.org/sql/window-functions-in-sql/