SQL for Cars Retail Data Exploration

2024

Anis Dela Desela
Jakarta, Indonesia

Overview

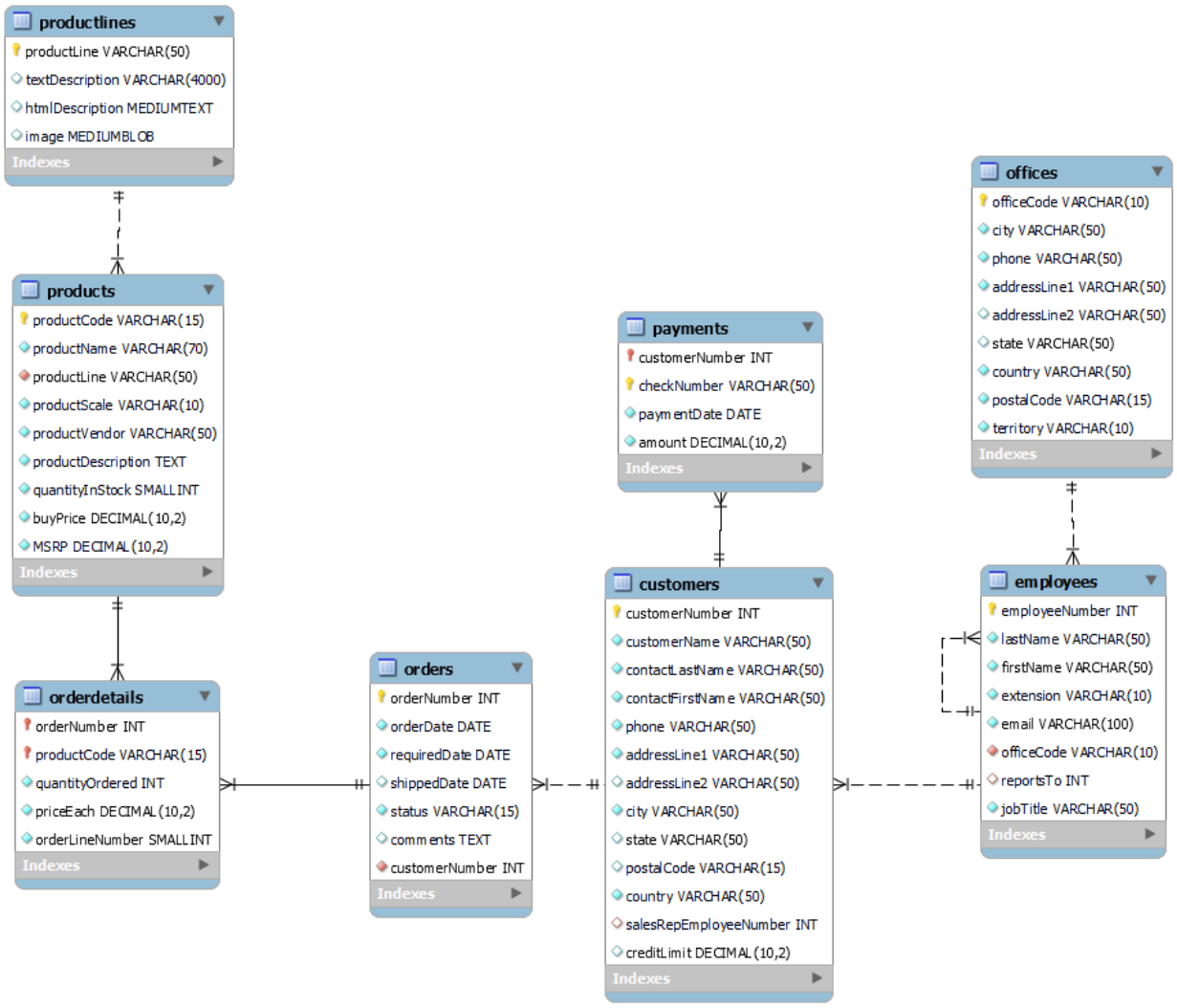
The Cars retail is a miniature vehicle toy store. The store has a database that stores information about its business processes. The store owner wants to analyze the existing database to develop the retail Cars business.

Methodology

I. ERD Dataset

The dataset utilized in this project is for exploration based on use cases for cars retail data, including various parameters.

The ERD of the dataset is:



II. Tools

In conducting this analysis, I utilized SQL. The SQL script for this purpose is also accessible at : [Cars Retail Data - Anis' Github]

Analysis

I. The store owner is interested in knowing the Nth most expensive product ordered by each customer.

CUSTOMERNAME	PRODUCTNAME	PRICEEACH	RANK_PRODUCT
Atelier graphique	1948 Porsche Type 356 Roadster	117.26	2
Signal Gift Stores	1917 Grand Touring Sedan	153.00	2
Australian Collectors, Co.	1952 Alpine Renault 1300	188.58	2
La Rochelle Gifts	2003 Harley-Davidson Eagle Drag Bike	154.93	2
Baane Mini Imports	2001 Ferrari Enzo	193.25	2

Pict 1. The-Nth Most Expensive Product Bought by Each Customer

QUERY:

```
USE cars_retail;

-- Create Procedure for the Expensive Product
DELIMITER //
CREATE PROCEDURE EXPENSIVE_PRODUCT(
IN RANK_PRODUCT_PARAM INT
)

-- Gather the customer data needed
BEGIN
WITH CUSTOMER_DATA AS(
      SELECT C.CUSTOMERNUMBER, C.CUSTOMERNAME,
                  O.ORDERNUMBER
      FROM CUSTOMERS AS C
      JOIN ORDERS AS O ON C.CUSTOMERNUMBER = O.CUSTOMERNUMBER)

-- Gather the product details that include the product price
, PRODUCT_DATA AS(
      SELECT OD.ORDERNUMBER,
                  OD.PRODUCTCODE,
                  OD.PRICEEACH,
                  P.PRODUCTNAME
      FROM ORDERDETAILS AS OD
      JOIN PRODUCTS AS P
      ON P.PRODUCTCODE = OD.PRODUCTCODE)

-- Combine the product and customer data
, RAW_DATA AS(
      SELECT C.CUSTOMERNUMBER, C.CUSTOMERNAME, C.ORDERNUMBER,
                  P.PRODUCTCODE, P.PRODUCTNAME, P.PRICEEACH,
```

```
                    ROW_NUMBER() OVER(PARTITION BY C.CUSTOMERNUMBER,
C.CUSTOMERNAME ORDER BY P.PRICEEACH DESC) AS RANK_PRODUCT
       FROM CUSTOMER_DATA AS C
       JOIN PRODUCT_DATA AS P
       ON C.ORDERNUMBER = P.ORDERNUMBER)

SELECT CUSTOMERNAME, PRODUCTNAME, PRICEEACH, RANK_PRODUCT
FROM RAW_DATA
    WHERE RANK_PRODUCT=RANK_PRODUCT_PARAM;
END //
DELIMITER ;

-- Show the procedure status
SHOW PROCEDURE STATUS;

-- To acces the procedure, use 'CALL'
CALL EXPENSIVE_PRODUCT('2');
```

II. The store owner wants to see the customers who placed the first and last orders in each country.

FIRST_CUST	LAST_CUST	COUNTRY	MAXDATE	MINDATE
Australian Collectors, Co.	Souveniers And Things Co.	Australia	2005-05-29	2003-04-29
Salzburg Collectables	Salzburg Collectables	Austria	2005-05-17	2003-04-28
Royale Belge	Petit Auto	Belgium	2005-05-30	2003-04-11
Québec Home Shopping Network	Québec Home Shopping Network	Canada	2005-05-01	2003-11-05
Danish Wholesale Imports	Danish Wholesale Imports	Denmark	2005-04-15	2003-02-11
Suominen Souveniers	Toys of Finland, Co.	Finland	2005-02-09	2003-08-01
La Corne D'abondance, Co.	La Rochelle Gifts	France	2005-05-31	2003-04-01

Pict 2. Customers with the First and Latest Order

QUERY:

```
-- Get the first and last date of order for each country
WITH FIRST_LAST_CUST AS(
SELECT C.CUSTOMERNUMBER, C.CUSTOMERNAME,
       C.COUNTRY,
         O.ORDERDATE,
         MAX(O.ORDERDATE) OVER(PARTITION BY COUNTRY) AS MAXDATE,
         MIN(O.ORDERDATE) OVER(PARTITION BY COUNTRY) AS MINDATE
FROM CUSTOMERS AS C
JOIN ORDERS AS O
ON C.CUSTOMERNUMBER = O.CUSTOMERNUMBER
ORDER BY COUNTRY)

-- Gather the first customer for each country
, FIRST_ORDER_CUST AS(
SELECT * FROM FIRST_LAST_CUST
```

```
WHERE ORDERDATE = MINDATE)

-- Gather the last customer for each country
, LAST_ORDER_CUST AS(
SELECT * FROM FIRST_LAST_CUST
WHERE ORDERDATE = MAXDATE)

-- Combine the first and latest customer for each country
SELECT F.CUSTOMERNAME AS FIRST_CUST,
      L.CUSTOMERNAME AS LAST_CUST,
        F.COUNTRY,
        L.MAXDATE,
        F.MINDATE
FROM LAST_ORDER_CUST AS L
JOIN FIRST_ORDER_CUST AS F
ON F.COUNTRY = L.COUNTRY;
```

III. The store owner wants to see the monthly and yearly sales and transaction trends.

MONTH_YEAR_ORDER	TOTAL_ORDER	TOTAL_SALES
January 2003	5	1357
February 2003	3	1449
March 2003	6	1755
April 2003	7	1993
May 2003	6	2017
June 2003	7	1685
July 2003	7	2145
August 2003	5	1974
September 2003	8	2510

Pict 3.a Monthly Trend

YEAR_ORDER	TOTAL_ORDER	TOTAL_SALES
2003	111	36439
2004	151	49487
2005	64	19590

Pict 3.b Yearly Trend

QUERY:

```
-- TOTAL ORDER RAW
CREATE VIEW RAW_ORDER_SALES AS
WITH RAW_ORDER AS(
SELECT ORDERDATE,
      ORDERNUMBER
FROM ORDERS
ORDER BY ORDERDATE)

-- TOTAL PENJUALAN RAW
, RAW_SALES AS(
SELECT OD.ORDERNUMBER,
      OD.QUANTITYORDERED,
```

```
        O.ORDERDATE
FROM ORDERDETAILS AS OD
JOIN ORDERS AS O
ON OD.ORDERNUMBER = O.ORDERNUMBER
ORDER BY ORDERDATE)

-- TOTAL TRANSAKSI DAN ORDER
SELECT O.ORDERNUMBER,
      O.ORDERDATE,
        S.QUANTITYORDERED
FROM RAW_SALES AS S
JOIN RAW_ORDER AS O
ON S.ORDERNUMBER = O.ORDERNUMBER
ORDER BY O.ORDERDATE;
SELECT * FROM RAW_ORDER_SALES;

-- GET MONTHLY ORDER & SALES
SELECT DATE_FORMAT(ORDERDATE, '%M %Y') AS MONTH_YEAR_ORDER,
      COUNT(DISTINCT ORDERNUMBER) AS TOTAL_ORDER,
        SUM(QUANTITYORDERED) AS TOTAL_SALES
FROM RAW_ORDER_SALES
GROUP BY DATE_FORMAT(ORDERDATE, '%M %Y')
ORDER BY MIN(ORDERDATE);

-- GET YEARLY ORDER & SALES
SELECT DATE_FORMAT(ORDERDATE, '%Y') AS YEAR_ORDER,
      COUNT(DISTINCT ORDERNUMBER) AS TOTAL_ORDER,
        SUM(QUANTITYORDERED) AS TOTAL_SALES
FROM RAW_ORDER_SALES
GROUP BY DATE_FORMAT(ORDERDATE, '%Y')
ORDER BY MIN(ORDERDATE);
```

IV. From the transactions that have been made in the store, the store owner is interested in knowing how many total customer payments are above or below the average total payments.

CUSTOMERNUMBER	CUSTOMERNAME	NUMBER_PAYMENT	AVG_PAYMENT	PAYMENT_STATUS
103	Atelier graphique	3	2.7857	Above Average
112	Signal Gift Stores	3	2.7857	Above Average
114	Australian Collectors, Co.	4	2.7857	Above Average
119	La Rochelle Gifts	3	2.7857	Above Average
121	Baane Mini Imports	4	2.7857	Above Average

Pict 4. Customer's Payment Status

QUERY:

```
-- Count total payments each customer
```

```
WITH COUNT_PAYMENT AS(
SELECT P.CUSTOMERNUMBER,
       C.CUSTOMERNAME,
       COUNT(DISTINCT P.CHECKNUMBER) AS NUMBER_PAYMENT
FROM PAYMENTS AS P
JOIN CUSTOMERS AS C
ON P.CUSTOMERNUMBER = C.CUSTOMERNUMBER
GROUP BY CUSTOMERNUMBER)

-- Get the average number of all payments
, AVERAGE_PAYMENTS AS(
SELECT CUSTOMERNUMBER,
       CUSTOMERNAME,
       NUMBER_PAYMENT,
       AVG(NUMBER_PAYMENT) OVER() AS AVG_PAYMENT,
         CASE
              WHEN AVG(NUMBER_PAYMENT)>NUMBER_PAYMENT THEN 'Below Average'
              ELSE 'Above Average'
       END AS PAYMENT_STATUS
FROM COUNT_PAYMENT
GROUP BY CUSTOMERNUMBER)

SELECT *
FROM AVERAGE_PAYMENTS;
```

V. The store owner plans to create a customer loyalty program by providing special facilities to customers who fall into the Loyal Customer category. Before implementing it, he requests to categorize customers based on their order frequency.

A. If a customer has ordered once, they are categorized as a One-time customer.
B. If a customer has ordered twice, they are categorized as a Repeated customer.
C. If a customer has ordered three times, they are categorized as a Frequent customer.
D. If a customer has ordered at least four times, they are categorized as a Loyal customer.

CUSTOMERNUMBER	CUSTOMERNAME	TOTAL_ORDER	CUSTOMER_TYPE
103	Atelier graphique	3	Frequent Customer
112	Signal Gift Stores	3	Frequent Customer
114	Australian Collectors, Co.	5	Loyal Customer
119	La Rochelle Gifts	4	Loyal Customer

Pict 5. Customer's Loyalty Type

QUERY:

```
SELECT DISTINCT(O.CUSTOMERNUMBER),
       C.CUSTOMERNAME,
         COUNT(O.ORDERDATE) AS TOTAL_ORDER,
         CASE
              WHEN COUNT(O.ORDERDATE) = 1 THEN 'One-Time Customer'
              WHEN COUNT(O.ORDERDATE) = 2 THEN 'Repeated Customer'
              WHEN COUNT(O.ORDERDATE) = 3 THEN 'Frequent Customer'
              WHEN COUNT(O.ORDERDATE) >= 4 THEN 'Loyal Customer'
              ELSE 'Not Customer'
       END AS CUSTOMER_TYPE
FROM ORDERS AS O
JOIN CUSTOMERS AS C
ON O.CUSTOMERNUMBER = C.CUSTOMERNUMBER
GROUP BY CUSTOMERNUMBER
ORDER BY CUSTOMERNUMBER;
```

VI. The store owner is interested in understanding product purchase trends in each country. He requests to find out the most ordered product category in each country.

COUNTRY	PRODUCTCODE	PRODUCTNAME	PRODUCTLINE	TOTAL_ORDER	PRODUCT_RANK
Australia	S18_2949	1913 Ford Model T Speedster	Vintage Cars	6	1
Australia	S12_1666	1958 Setra Bus	Trucks and Buses	5	2
Australia	S18_1097	1940 Ford Pickup Truck	Trucks and Buses	5	3
Australia	S10_1949	1952 Alpine Renault 1300	Classic Cars	4	4
Australia	S10_2016	1996 Moto Guzzi 1100i	Motorcycles	4	5
Austria	S12_3380	1968 Dodge Charger	Classic Cars	3	1

Pict 6. Product Rank for Each Country

QUERY:

```
-- Gather Country, Customer Number, and the Order Number
WITH CUST_COUNTRY AS(
SELECT O.CUSTOMERNUMBER, O.ORDERNUMBER, C.COUNTRY
FROM CUSTOMERS AS C
JOIN ORDERS AS O ON C.CUSTOMERNUMBER = O.CUSTOMERNUMBER)

-- Gather the Product Category
, PRODUCT_CATEGORY AS(
SELECT OD.ORDERNUMBER, OD.PRODUCTCODE, P.PRODUCTNAME, P.PRODUCTLINE
FROM ORDERDETAILS AS OD
JOIN PRODUCTS AS P ON OD.PRODUCTCODE = P.PRODUCTCODE)

-- Combine the product category with the customer country CTE table
, CATEGORY_COUNTRY AS(
SELECT C.ORDERNUMBER, C.CUSTOMERNUMBER, C.COUNTRY,
       P.PRODUCTCODE, P.PRODUCTNAME, P.PRODUCTLINE
```

```
FROM CUST_COUNTRY AS C
JOIN PRODUCT_CATEGORY AS P
ON C.ORDERNUMBER = P.ORDERNUMBER)

-- Create rank based on most ordered product
, RANK_PRODUCT AS(
SELECT COUNTRY,
      PRODUCTCODE, PRODUCTNAME, PRODUCTLINE,
        COUNT(DISTINCT(ORDERNUMBER)) AS TOTAL_ORDER,
        ROW_NUMBER() OVER(PARTITION BY COUNTRY ORDER BY COUNT(DISTINCT
ORDERNUMBER) DESC) as PRODUCT_RANK
FROM CATEGORY_COUNTRY
GROUP BY COUNTRY, PRODUCTCODE
ORDER BY COUNTRY, PRODUCT_RANK)

SELECT * FROM RANK_PRODUCT
WHERE PRODUCT_RANK <= 5;
```

VII. The store owner wants to know the average time it takes for customers to place a repeat order.

CUSTOMERNUMBER	CUSTOMERNAME	AVG_REPEAT_ORDER
239	Collectable Mini Designs Co.	7.00
211	King Kong Collectables, Co.	16.00
249	Amica Models & Co.	23.00
141	Euro+ Shopping Channel	34.04

Pict 7. Average Time of Customers to Repeat Order

QUERY:

```
-- Calculate diffdate with the previous order for each customer
WITH DIFF_DATE AS(
SELECT O.CUSTOMERNUMBER, O.ORDERDATE,
      C.CUSTOMERNAME,
      DATEDIFF(O.ORDERDATE, LAG(O.ORDERDATE) OVER(PARTITION BY
O.CUSTOMERNUMBER ORDER BY O.ORDERDATE)) AS DIFFERENCE_DATE
FROM ORDERS AS O
JOIN CUSTOMERS AS C
ON O.CUSTOMERNUMBER = C.CUSTOMERNUMBER
ORDER BY CUSTOMERNUMBER, ORDERDATE)

-- Calculate the average diffdate
SELECT DISTINCT(CUSTOMERNUMBER),
      CUSTOMERNAME,
        ROUND(AVG(DIFFERENCE_DATE) OVER(PARTITION BY CUSTOMERNUMBER),2)
AS AVG_REPEAT_ORDER
FROM DIFF_DATE
```

```
WHERE DIFFERENCE_DATE IS NOT NULL
ORDER BY AVG_REPEAT_ORDER;
```

VIII. The store owner wants to see the dates and transaction amounts of the payments made by customers when they placed their first orders.

CUSTOMERNUMBER	FIRST_PAYMENT	AMOUNT
103	2003-06-05	14571.44
112	2003-06-06	32641.98
114	2003-05-20	45864.03
119	2004-08-08	47924.19
121	2003-02-16	50218.95
124	2003-04-11	11044.30

Pict 8.Each Customer First Payment

QUERY:

```
-- Calculate the first payment
WITH FIRST_PAYMENT_DATE AS(
SELECT DISTINCT(CUSTOMERNUMBER),
      MIN(PAYMENTDATE) OVER(PARTITION BY CUSTOMERNUMBER) AS
FIRST_PAYMENT
FROM PAYMENTS)

SELECT DISTINCT F.CUSTOMERNUMBER,
      F.FIRST_PAYMENT,
      P.AMOUNT
FROM FIRST_PAYMENT_DATE AS F
JOIN PAYMENTS AS P
ON F.CUSTOMERNUMBER = P.CUSTOMERNUMBER
AND F.FIRST_PAYMENT = P.PAYMENTDATE;
```