Incubyte Sales Analysis using mySQL

Dataset Analysis

- The dataset contains 500,000 rows and 19 columns related to sales transactions. Key observations:
- Some columns have missing values (e.g., CustomerID, TransactionDate, PaymentMethod, StoreType, Region, ProductName).
- The TransactionDate column is stored as object (string) and may need to be converted to a datetime format.
- Returned and IsPromotional columns are categorical (Yes/No values).
- It includes key attributes such as TransactionAmount, Quantity, DiscountPercent, ShippingCost, LoyaltyPoints, and DeliveryTimeDays.

Data Cleaning Process

-- 1. Convert TransactionDate to proper DATETIME format

```
UPDATE test_assessment.assessment_dataset

SET TransactionDate = STR_TO_DATE(TransactionDate, '%m/%d/%Y %H:%i')


ALTER TABLE test_assessment.assessment_dataset

MODIFY COLUMN TransactionDate DATETIME
```

-- 2. Identify missing values in key columns

```
SELECT TransactionDate, COUNT(*) AS missing_count

FROM test_assessment.assessment_dataset

WHERE TransactionDate IS NULL

GROUP BY TransactionDate
```

-- 3. Handling missing value in CustomerID

--- CustomerID

```
UPDATE test_assessment.assessment_dataset

SET CustomerID = -1

WHERE CustomerID IS NULL OR CustomerID = 0
```

--- TransactionDate

First check with existing CustomerID

```
UPDATE test_assessment.assessment_dataset s

LEFT JOIN (
    SELECT CustomerID, MAX(TransactionDate) AS LastTransaction
    FROM test_assessment.assessment_dataset
    WHERE TransactionDate IS NOT NULL
    GROUP BY CustomerID
) t ON s.CustomerID = t.CustomerID
SET s.TransactionDate = t.LastTransaction
WHERE s.TransactionDate IS NULL
```

--- Update with '2000-01-01 00:00:00'

```
UPDATE test_assessment.assessment_dataset
SET TransactionDate = '2000-01-01 00:00:00'
WHERE TransactionDate IS NULL
```

-- PaymentMethod with 'Unknown' with Coalesce Function

```
UPDATE test_assessment.assessment_dataset
SET PaymentMethod = COALESCE(PaymentMethod, 'Unknown')
```

-- ProductName with 'Unknown'

```
UPDATE test_assessment.assessment_dataset
SET ProductName = COALESCE(ProductName, 'Unknown')
```

-- StoreType with 'Unknown'

```
UPDATE test_assessment.assessment_dataset
SET StoreType = COALESCE(StoreType, 'Unknown')
```

-- 4. Standardize text formats (lowercase, trim spaces)

```
UPDATE test_assessment.assessment_dataset
SET City = TRIM(LOWER(City)), PaymentMethod = TRIM(LOWER(PaymentMethod))
```

-- 5 Identify duplicate rows and Delete

```
WITH duplicates AS (
    SELECT *, ROW_NUMBER() OVER (PARTITION BY TransactionID ORDER BY TransactionDate) AS row_num
    FROM test_assessment.assessment_dataset
)
```

```
DELETE FROM test_assessment.assessment_dataset

WHERE TransactionID IN (SELECT TransactionID FROM duplicates WHERE row_num > 1)
```

Sales Analysis

-- 1. Total Revenue & Orders

```
SELECT COUNT(DISTINCT TransactionID) AS total_orders,

            ROUND(SUM(TransactionAmount),2) AS total_revenue

FROM test_assessment.assessment_dataset

WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown'
```

Output:
Total Sales Revenue: ₹10,156,929,717.25

Total Orders: 4,50,000

-- 2. Average Order Value (AOV)

```
SELECT ROUND(SUM(TransactionAmount) / COUNT(DISTINCT TransactionID),2) AS avg_order_value

FROM test_assessment.assessment_dataset

WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown'
```

Output:

Average Order Value (AOV): ₹22,570.95

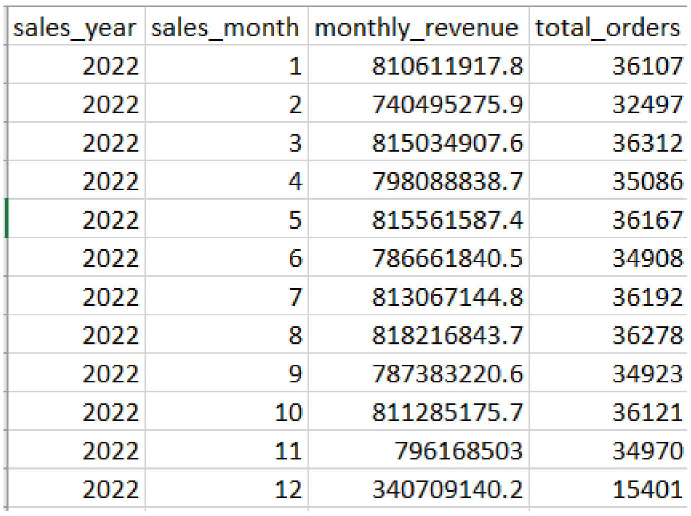
-- 3. Monthly Sales Trends with EXTRACT FUNCTION

```
SELECT EXTRACT(year FROM TransactionDate)  AS sales_year,

            EXTRACT(month FROM TransactionDate) AS sales_month,

            ROUND(SUM(TransactionAmount),2) AS monthly_revenue,

            COUNT(DISTINCT TransactionID) AS total_orders

FROM test_assessment.assessment_dataset

WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown'

GROUP BY sales_year,sales_month

HAVING sales_year <> 2000

ORDER BY sales_year,sales_month
```

-- Monthly Sales Trends with MONTH and YEAR FUNCTION

```
SELECT YEAR(TransactionDate)  AS sales_year,

            MONTH(TransactionDate) AS sales_month,

            ROUND(SUM(TransactionAmount),2) AS monthly_revenue,

            COUNT(DISTINCT TransactionID) AS total_orders

FROM test_assessment.assessment_dataset

WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown'

GROUP BY sales_year,sales_month

HAVING sales_year <> 2000

ORDER BY sales_year,sales_month
```

Output:

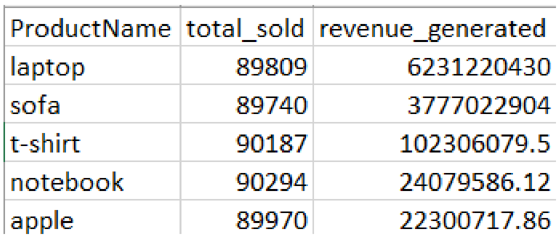


sales_year	sales_month	monthly_revenue	total_orders
2022	1	810611917.8	36107
2022	2	740495275.9	32497
2022	3	815034907.6	36312
2022	4	798088838.7	35086
2022	5	815561587.4	36167
2022	6	786661840.5	34908
2022	7	813067144.8	36192
2022	8	818216843.7	36278
2022	9	787383220.6	34923
2022	10	811285175.7	36121
2022	11	796168503	34970
2022	12	340709140.2	15401

-- 4. Best 5 Selling Products

```
SELECT ProductName,

          COUNT(*) AS total_sold,

          ROUND(SUM(TransactionAmount),2) AS revenue_generated

FROM test_assessment.assessment_dataset

GROUP BY ProductName

HAVING ProductName <> 'unknown'

ORDER BY total_sold DESC

LIMIT 5;
```

Output:

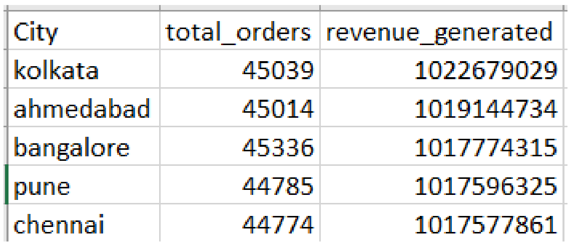


ProductName	total_sold	revenue_generated
laptop	89809	6231220430
sofa	89740	3777022904
t-shirt	90187	102306079.5
notebook	90294	24079586.12
apple	89970	22300717.86

-- 5. Top 5 Cities by Sales Revenue

```
SELECT City,
            COUNT(DISTINCT TransactionID) AS total_orders,
            ROUND(SUM(TransactionAmount),2) AS revenue_generated
FROM test_assessment.assessment_dataset
WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown'
GROUP BY City
ORDER BY revenue_generated DESC
LIMIT 5
```

Output:

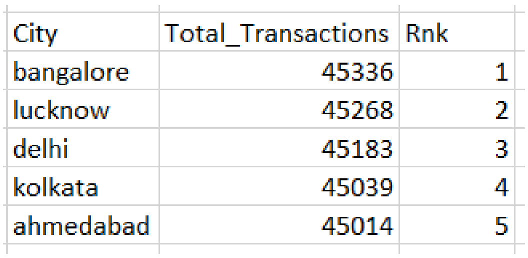


City	total_orders	revenue_generated
kolkata	45039	1022679029
ahmedabad	45014	1019144734
bangalore	45336	1017774315
pune	44785	1017596325
chennai	44774	1017577861

-- With Rank() Function Top 5 cities by Total Transactions

```
SELECT City,
      COUNT(*) AS Total_Transactions,
      RANK() OVER (ORDER BY COUNT(*) DESC) AS Rnk
FROM test_assessment.assessment_dataset
WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown'
GROUP BY City
LIMIT 5;
```

Output:

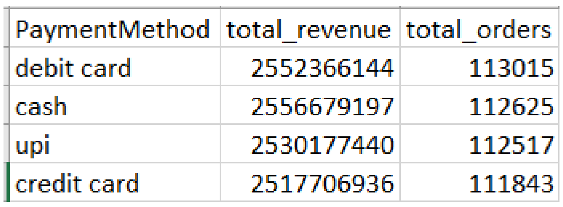


City	Total_Transactions	Rnk
bangalore	45336	1
lucknow	45268	2
delhi	45183	3
kolkata	45039	4
ahmedabad	45014	5

-- 6. Most Preferred Payment Methods

```
SELECT PaymentMethod,
        ROUND(SUM(TransactionAmount),2) AS total_revenue,
        COUNT(DISTINCT TransactionID) AS total_orders
FROM test_assessment.assessment_dataset
GROUP BY PaymentMethod
HAVING PaymentMethod <> 'unknown'
ORDER BY total_orders DESC
```

Output:

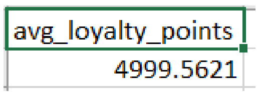


PaymentMethod	total_revenue	total_orders
debit card	2552366144	113015
cash	2556679197	112625
upi	2530177440	112517
credit card	2517706936	111843

-- 7. Average loyalty points across all customers.

```
SELECT AVG(LoyaltyPoints) AS avg_loyalty_points
FROM test_assessment.assessment_dataset
WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown'
```

Output:

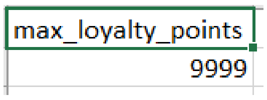


avg_loyalty_points
4999.5621

-- 8. Maximum loyalty points earned by a customer: 9,999

```
SELECT Max(LoyaltyPoints) AS max_loyalty_points
FROM test_assessment.assessment_dataset
WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown'
```

Output:

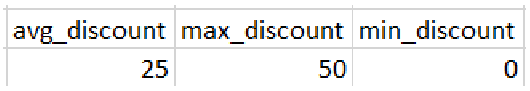


max_loyalty_points
9999

-- 9. Discount & Pricing Insights

```
SELECT ROUND(AVG(DiscountPercent),2) AS avg_discount,
	ROUND(MAX(DiscountPercent),2) AS max_discount,
	ROUND(MIN(DiscountPercent),2) AS min_discount
FROM test_assessment.assessment_dataset
WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown'
```

Output:

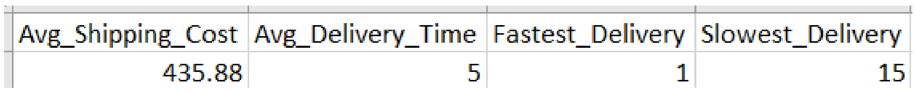


avg_discount	max_discount	min_discount
25	50	0

-- 10 Shipping & Delivery Performance

```
SELECT ROUND(AVG(ShippingCost),2) AS Avg_Shipping_Cost,
	ROUND(AVG(DeliveryTimeDays),0) AS Avg_Delivery_Time,
	MIN(DeliveryTimeDays) AS Fastest_Delivery,
	MAX(DeliveryTimeDays) AS Slowest_Delivery
FROM test_assessment.assessment_dataset
WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown'
```

Output:



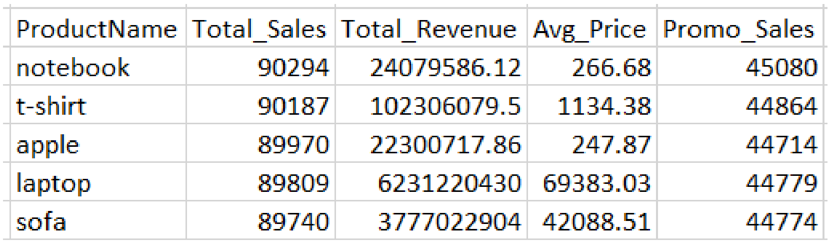
Avg_Shipping_Cost	Avg_Delivery_Time	Fastest_Delivery	Slowest_Delivery
435.88	5	1	15

-- 11. Product Sales Breakdown

```
SELECT ProductName,
	COUNT(*) AS Total_Sales,
	ROUND(SUM(TransactionAmount),2) AS Total_Revenue,
	ROUND(AVG(TransactionAmount), 2) AS Avg_Price,
	SUM(CASE WHEN IsPromotional = 'Yes' THEN 1 ELSE 0 END) AS Promo_Sales
FROM test_assessment.assessment_dataset
GROUP BY ProductName
HAVING ProductName <> 'unknown'
```

```
ORDER BY Total_Sales DESC

LIMIT 10
```

Output:

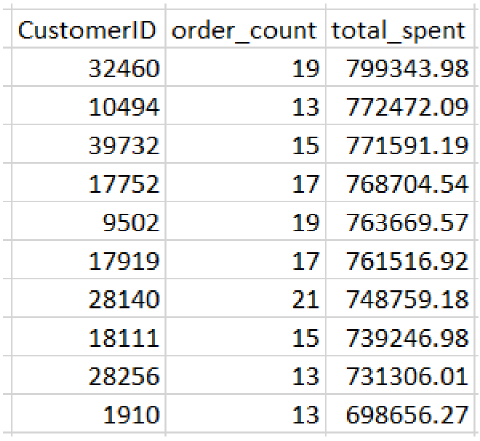


ProductName	Total_Sales	Total_Revenue	Avg_Price	Promo_Sales
notebook	90294	24079586.12	266.68	45080
t-shirt	90187	102306079.5	1134.38	44864
apple	89970	22300717.86	247.87	44714
laptop	89809	6231220430	69383.03	44779
sofa	89740	3777022904	42088.51	44774

-- 12. Customer Purchase Frequency

```
SELECT CustomerID,

            COUNT(DISTINCT TransactionID) AS order_count,

            ROUND(SUM(TransactionAmount),2) AS total_spent

FROM test_assessment.assessment_dataset

WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown'

GROUP BY CustomerID

HAVING CustomerID <> -1

ORDER BY total_spent DESC

LIMIT 10
```

Output:



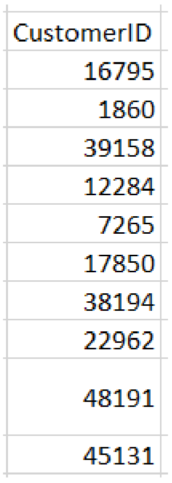
CustomerID	order_count	total_spent
32460	19	799343.98
10494	13	772472.09
39732	15	771591.19
17752	17	768704.54
9502	19	763669.57
17919	17	761516.92
28140	21	748759.18
18111	15	739246.98
28256	13	731306.01
1910	13	698656.27

-- 13. Finding Repeat Customers

```
WITH CustomerOrders AS (

   SELECT CustomerID, COUNT(TransactionID) AS Order_Count

   FROM test_assessment.assessment_dataset

   WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown'
```

```
    GROUP BY CustomerID
    HAVING CustomerID <> -1
)
SELECT CustomerID
FROM CustomerOrders
WHERE Order_Count > 1
```

Output: TOTAL REPEAT CUSTOMERS : 48,882

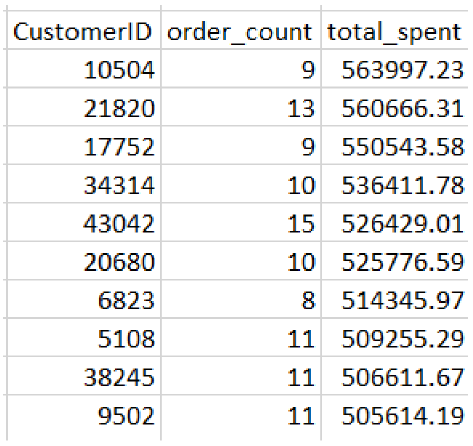


CustomerID
16795
1860
39158
12284
7265
17850
38194
22962
48191
45131

-- 14. Customer Purchase Frequency With No Return

```
SELECT CustomerID,
      COUNT(DISTINCT TransactionID) AS order_count,
      ROUND(SUM(TransactionAmount),2) AS total_spent
FROM test_assessment.assessment_dataset
WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown' and Returned =
"No"
GROUP BY CustomerID
HAVING CustomerID <> -1
ORDER BY total_spent DESC
LIMIT 10
```

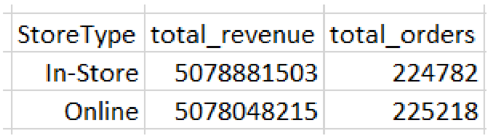
Output:



CustomerID	order_count	total_spent
10504	9	563997.23
21820	13	560666.31
17752	9	550543.58
34314	10	536411.78
43042	15	526429.01
20680	10	525776.59
6823	8	514345.97
5108	11	509255.29
38245	11	506611.67
9502	11	505614.19

```sql
-- 15. StoreType Sales Breakdown

SELECT

    StoreType,

    ROUND(SUM(TransactionAmount),2) AS total_revenue,

    COUNT(DISTINCT TransactionID) AS total_orders

FROM test_assessment.assessment_dataset

GROUP BY StoreType

HAVING StoreType <> 'unknown'

ORDER BY total_revenue DESC
```

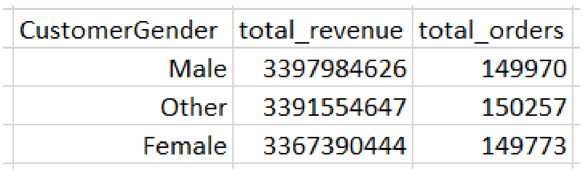
Output:



StoreType	total_revenue	total_orders
In-Store	5078881503	224782
Online	5078048215	225218

```sql
-- 16. Gender Sales Breakdown

SELECT

    CustomerGender,

    ROUND(SUM(TransactionAmount),2) AS total_revenue,

    COUNT(DISTINCT TransactionID) AS total_orders

FROM test_assessment.assessment_dataset

GROUP BY CustomerGender

HAVING CustomerGender IS NOT NULL

ORDER BY total_revenue DESC
```

Output:

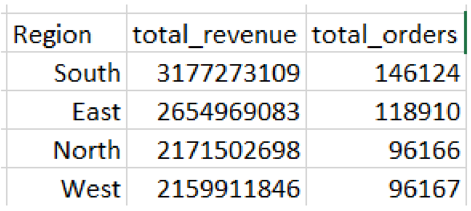


CustomerGender	total_revenue	total_orders
Male	3397984626	149970
Other	3391554647	150257
Female	3367390444	149773

-- 17. Region Sales Breakdown

```
SELECT

   Region,

   ROUND(SUM(TransactionAmount),2) AS total_revenue,

   COUNT(DISTINCT TransactionID) AS total_orders

FROM test_assessment.assessment_dataset

GROUP BY Region

HAVING Region IS NOT NULL

ORDER BY total_revenue DESC
```

Output:

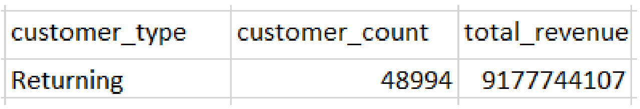


Region	total_revenue	total_orders
South	3177273109	146124
East	2654969083	118910
North	2171502698	96166
West	2159911846	96167

-- 18. Customer Retention Analysis (New vs. Repeat Customers)

```
SELECT

   CASE

          WHEN CustomerID IN (SELECT DISTINCT CustomerID FROM test_assessment.assessment_dataset WHERE
TransactionDate < DATE_SUB(CURDATE(), INTERVAL 1 YEAR)) THEN 'Returning'

          ELSE 'New'

   END AS customer_type,

   COUNT(DISTINCT CustomerID) AS customer_count,

   ROUND(SUM(TransactionAmount),2) AS total_revenue

FROM test_assessment.assessment_dataset

WHERE CustomerID <> -1

GROUP BY customer_type
```

Output:

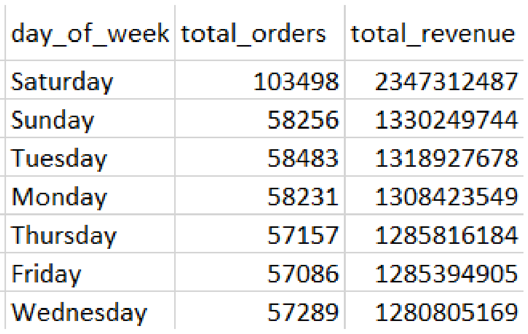


customer_type	customer_count	total_revenue
Returning	48994	9177744107

-- 19. Day of the Week Sales Analysis (Peak Shopping Days)

```
SELECT
   DAYNAME(TransactionDate) AS day_of_week,
   COUNT(DISTINCT TransactionID) AS total_orders,
   ROUND(SUM(TransactionAmount),2) AS total_revenue
FROM test_assessment.assessment_dataset
WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown'
GROUP BY day_of_week
ORDER BY total_revenue DESC;
```

Output:

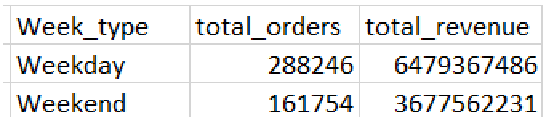


day_of_week	total_orders	total_revenue
Saturday	103498	2347312487
Sunday	58256	1330249744
Tuesday	58483	1318927678
Monday	58231	1308423549
Thursday	57157	1285816184
Friday	57086	1285394905
Wednesday	57289	1280805169

-- 20. WeekEnd vs WeekDay Sales Analysis

```
SELECT
   CASE
          WHEN DAYNAME(TransactionDate) IN ('Saturday','Sunday') THEN 'Weekend'
          WHEN DAYNAME(TransactionDate) NOT IN ('Saturday','Sunday') THEN 'Weekday'
   End as Week_type,
   COUNT(DISTINCT TransactionID) AS total_orders,
   ROUND(SUM(TransactionAmount),2) AS total_revenue
FROM test_assessment.assessment_dataset
WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown'
GROUP BY Week_type
ORDER BY total_revenue DESC
```

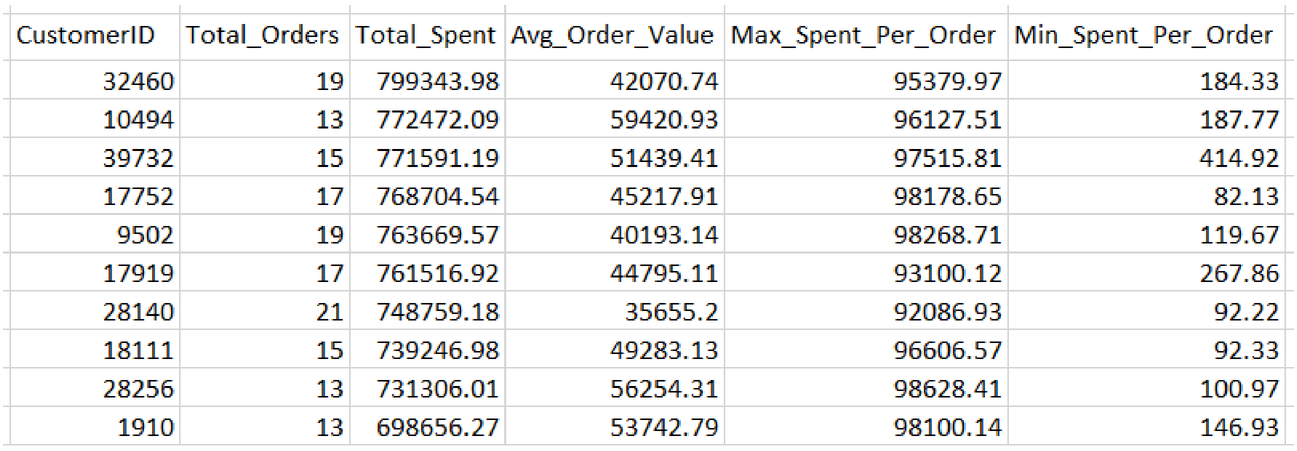
Output:



Week_type	total_orders	total_revenue
Weekday	288246	6479367486
Weekend	161754	3677562231

```
-- 21. Customer Behavior Analysis

SELECT

    cs.CustomerID,

    cs.Total_Orders,

    cs.Total_Spent,

    cs.Avg_Order_Value,

    (SELECT MAX(sd.TransactionAmount)

     FROM test_assessment.assessment_dataset sd

     WHERE sd.CustomerID = cs.CustomerID) AS Max_Spent_Per_Order,

    (SELECT MIN(sd.TransactionAmount)

     FROM test_assessment.assessment_dataset sd

     WHERE sd.CustomerID = cs.CustomerID) AS Min_Spent_Per_Order

FROM (

    SELECT

        CustomerID,

        COUNT(TransactionID) AS Total_Orders,

        ROUND(SUM(TransactionAmount),2) AS Total_Spent,

        ROUND(AVG(TransactionAmount),2) AS Avg_Order_Value

    FROM test_assessment.assessment_dataset

    WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown'

    GROUP BY CustomerID

    HAVING CustomerID <> -1

) AS cs

ORDER BY cs.Total_Spent DESC

LIMIT 10
```

Output:



CustomerID	Total_Orders	Total_Spent	Avg_Order_Value	Max_Spent_Per_Order	Min_Spent_Per_Order
32460	19	799343.98	42070.74	95379.97	184.33
10494	13	772472.09	59420.93	96127.51	187.77
39732	15	771591.19	51439.41	97515.81	414.92
17752	17	768704.54	45217.91	98178.65	82.13
9502	19	763669.57	40193.14	98268.71	119.67
17919	17	761516.92	44795.11	93100.12	267.86
28140	21	748759.18	35655.2	92086.93	92.22
18111	15	739246.98	49283.13	96606.57	92.33
28256	13	731306.01	56254.31	98628.41	100.97
1910	13	698656.27	53742.79	98100.14	146.93

-- 22. Returned Rate Product Wise

```
With CTE As

(SELECT

   ProductName,

   COUNT(CASE WHEN Returned = 'Yes' THEN 1 END) AS total_returns,

   COUNT(*) AS total_orders

FROM test_assessment.assessment_dataset

GROUP BY ProductName

HAVING ProductName <> 'unknown')

SELECT ProductName,

   total_returns,

   total_orders,

          ROUND((total_returns / total_orders) * 100, 2) AS return_rate_percentage

FROM CTE

ORDER BY return_rate_percentage DESC;
```

Output:

ProductName	total_returns	total_orders	return_rate_percentage
Apple	45033	89970	50.05
Laptop	44904	89809	50
notebook	45061	90294	49.9
Sofa	44696	89740	49.81
t-shirt	44783	90187	49.66

-- 23. Customer Segmentation Based on Loyalty Points

```
SELECT
  CASE
    WHEN LoyaltyPoints < 100 THEN 'Low Loyalty'
    WHEN LoyaltyPoints BETWEEN 100 AND 500 THEN 'Medium Loyalty'
    ELSE 'High Loyalty'
  END AS loyalty_tier,
  COUNT(DISTINCT customerId) AS customer_count,
  ROUND(SUM(TransactionAmount),2) AS total_sales
FROM test_assessment.assessment_dataset
WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown'
GROUP BY loyalty_tier;
```

Output:

loyalty_tier	customer_count	total_sales
High Loyalty	48986	9647993180
Low Loyalty	3874	102392918.1
Medium Loyalty	13843	406543619.3

-- 24. Customer Segmentation (RFM Analysis)

```
WITH customer_rfm AS (
  SELECT CustomerID,
      COUNT(DISTINCT TransactionID) AS frequency,
      ROUND(SUM(TransactionAmount),2) AS monetary,
      MAX(TransactionDate) AS last_purchase,
      DATEDIFF(CURRENT_DATE, MAX(TransactionDate)) AS recency
  FROM test_assessment.assessment_dataset
  WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown'
  GROUP BY CustomerID
  HAVING CustomerID <> -1
)
SELECT CustomerID,
    recency,
    frequency,
    monetary,
    NTILE(4) OVER (ORDER BY recency DESC) AS recency_quartile,
    NTILE(4) OVER (ORDER BY frequency DESC) AS frequency_quartile,

    NTILE(4) OVER (ORDER BY monetary DESC) AS monetary_quartile

FROM customer_rfm
```

Output: Total Customers: 48990

CustomerID	recency	frequency	monetary	recency_quartile	frequency_quartile	monetary_quartile
32460	804	19	799344	4	1	1
10494	794	13	772472.1	4	1	1
39732	838	15	771591.2	2	1	1
17752	796	17	768704.5	4	1	1
9502	792	19	763669.6	4	1	1
17919	804	17	761516.9	4	1	1
28140	818	21	748759.2	3	1	1
18111	808	15	739247	3	1	1
28256	827	13	731306	2	1	1
1910	796	13	698656.3	4	1	1

-- 25. 7 days moving Avg Revenue Growth Analysis using Window Functions

```
WITH Moving_Avg AS

(SELECT DATE(TransactionDate) as TransactionDate,

    ROUND(SUM(TransactionAmount),2) AS daily_revenue

FROM test_assessment.assessment_dataset

WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown'

GROUP BY TransactionDate

HAVING TransactionDate <> '2000-01-01'

ORDER BY TransactionDate)

SELECT *,ROUND(SUM(daily_revenue) OVER (ORDER BY TransactionDate ROWS BETWEEN 6 PRECEDING AND
CURRENT ROW),2) AS 7_day_moving_avg

FROM Moving_Avg
```

Output:

TransactionDate	daily_revenue	7_day_moving_avg
01-01-2022	1910.91	1910.91
01-01-2022	58590.27	60501.18
01-01-2022	280.21	60781.39
01-01-2022	385.89	61167.28
01-01-2022	450.4	61617.68
01-01-2022	58286.23	119903.91
01-01-2022	158.1	120062.01
01-01-2022	40999.26	159150.36

-- 26. Market Basket Analysis (Frequently Bought Together)

```
WITH product_pairs AS (
   SELECT LEAST(a.ProductName, b.ProductName) AS product_1,
         GREATEST(a.ProductName, b.ProductName) AS product_2,
         COUNT(*) AS frequency
   FROM test_assessment.assessment_dataset a
   JOIN test_assessment.assessment_dataset b ON a.TransactionID = b.TransactionID AND a.ProductName <>
b.ProductName
   GROUP BY product_1, product_2
)
SELECT product_1, product_2, frequency
FROM product_pairs
ORDER BY frequency DESC
LIMIT 10
```

Output: Currently There are No Products Bought Together by Any Customer

-- 27. Promotion Effectiveness (Pre vs. Post Discount Sales)

```
WITH sales_comparison AS (
   SELECT
      CASE WHEN IsPromotional = 'Yes' THEN 'Post-Discount' ELSE 'Pre-Discount' END AS discount_status,
      ROUND(SUM(TransactionAmount),2) AS total_revenue,
      COUNT(DISTINCT TransactionID) AS order_count,
      ROUND(AVG(TransactionAmount),2) AS avg_order_value
   FROM test_assessment.assessment_dataset
   WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown'
   GROUP BY discount_status
)
SELECT * FROM sales_comparison
```

Output:

discount_status	total_revenue	order_count	avg_order_value
Post-Discount	5075964989	224211	22639.23
Pre-Discount	5080964728	225789	22503.15

-- 28. Customer Retention Rate

```
WITH customer_orders AS (
   SELECT CustomerID, MIN(TransactionDate) AS first_order, MAX(TransactionDate) AS last_order
   FROM test_assessment.assessment_dataset
   WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown'
   GROUP BY CustomerID
   HAVING CustomerID <> -1
)
SELECT COUNT(CustomerID) AS total_customers,
     COUNT(CASE WHEN DATEDIFF(last_order, first_order) > 90 THEN 1 END) AS retained_customers,
     (COUNT(CASE WHEN DATEDIFF(last_order, first_order) > 90 THEN 1 END) / COUNT(CustomerID)) * 100 AS
retention_rate
FROM customer_orders
```

Output:

total_customers	retained_customers	retention_rate
48990	48189	98.365

-- 29. Churned Customers (Inactive Users)

```
SELECT CustomerID, MAX(TransactionDate) AS last_purchase_date,
     DATEDIFF(CURRENT_DATE, MAX(TransactionDate)) AS days_since_last_purchase
FROM test_assessment.assessment_dataset
WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown'
GROUP BY CustomerID
HAVING days_since_last_purchase > 180 and CustomerID <> -1
ORDER BY days_since_last_purchase DESC
```

Output:

CustomerID	last_purchase_date	days_since_last_purchase
46895	03-01-2022 14:26	1137
38663	05-01-2022 00:38	1135
38726	06-01-2022 21:11	1134
2040	07-01-2022 05:56	1133
14563	09-01-2022 17:01	1131
35168	10-01-2022 14:27	1130
35171	15-01-2022 02:59	1125

-- 30. Average Time Between Purchases (Customer Lifetime Value Insight)

```
WITH purchase_intervals AS (
   SELECT CustomerID,
         TransactionDate,
         LAG(TransactionDate) OVER (PARTITION BY CustomerID ORDER BY TransactionDate) AS previous_purchase
   FROM test_assessment.assessment_dataset
   WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown'
)
SELECT CustomerID,
      AVG(DATEDIFF(TransactionDate, previous_purchase)) AS avg_days_between_purchases
FROM purchase_intervals
WHERE previous_purchase IS NOT NULL
GROUP BY CustomerID
HAVING CustomerID <> -1
ORDER BY avg_days_between_purchases ASC
```

Output:

CustomerID	avg_days_between_purchases
29496	0
2185	0
3540	1
7418	1
16570	1
29447	1.5
30005	2
38726	2
5101	3
14630	3
47987	3.5

-- 31. Year-over-Year (YoY) Sales Growth (%)

```
WITH YearlySales AS (
   SELECT
      YEAR(TransactionDate) AS sales_year,
      ROUND(SUM(TransactionAmount), 2) AS yearly_revenue
   FROM test_assessment.assessment_dataset
   WHERE PaymentMethod <> 'unknown' AND StoreType <> 'Unknown' AND ProductName <> 'unknown'
   GROUP BY sales_year
)

SELECT

   sales_year,

   yearly_revenue,

   LAG(yearly_revenue) OVER (ORDER BY sales_year) AS prev_year_revenue,

   ROUND(((yearly_revenue - LAG(yearly_revenue) OVER (ORDER BY sales_year)) / LAG(yearly_revenue) OVER
(ORDER BY sales_year)) * 100, 2) AS yoy_growth_percentage

FROM YearlySales

WHERE sales_year <> 2000
```

Output:

sales_year	yearly_revenue	prev_year_revenue	yoy_growth_percentage
2022	9133284396		

-- 32. Month-over-Month (MoM) Sales Growth (%)

```
WITH MonthlySalesGrowth AS

(SELECT YEAR(TransactionDate) AS year,

     MONTH(TransactionDate) AS month,

     ROUND(SUM(TransactionAmount),2) AS total_revenue

FROM test_assessment.assessment_dataset

WHERE PaymentMethod <> 'unknown' and StoreType <> 'Unknown' and ProductName <> 'unknown'

GROUP BY year, month

ORDER BY year, month)

SELECT year,month,total_revenue,

            LAG(total_revenue) OVER (PARTITION BY year ORDER BY month) AS prev_month_revenue,

     ROUND(((total_revenue - LAG(total_revenue) OVER (PARTITION BY year ORDER BY month)) / LAG(total_revenue)
OVER (PARTITION BY year ORDER BY month)) * 100, 2) AS MoM_Growth_Percentage

FROM MonthlySalesGrowth

WHERE year <> 2000;
```

Output:

year	month	total_revenue	prev_month_revenue	MoM_Growth_Percentage
2022	1	810611917.8		
2022	2	740495275.9	810611917.8	-8.65
2022	3	815034907.6	740495275.9	10.07
2022	4	798088838.7	815034907.6	-2.08
2022	5	815561587.4	798088838.7	2.19
2022	6	786661840.5	815561587.4	-3.54

2022	7	813067144.8	786661840.5	3.36
2022	8	818216843.7	813067144.8	0.63
2022	9	787383220.6	818216843.7	-3.77
2022	10	811285175.7	787383220.6	3.04
2022	11	796168503	811285175.7	-1.86
2022	12	340709140.2	796168503	-57.21