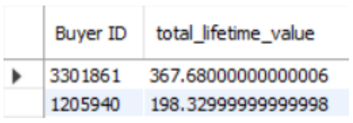
Revenue360: CLTV, Churn & Product Profitability Analysis

Customer Lifetime Value (CLTV) & Revenue Analysis

1. What is the total revenue contributed by each customer?

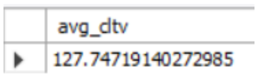
```
SELECT `Buyer ID`, SUM(`Final Revenue`) AS total_lifetime_value
FROM orders_table
GROUP BY `Buyer ID`;
```



Buyer ID	total_lifetime_value
3301861	367.68000000000006
1205940	198.32999999999998

2. What is the average CLTV across all customers?

```
SELECT AVG(total_spent) AS avg_cltv
FROM (
  SELECT `Buyer ID`, SUM(`Final Revenue`) AS total_spent
  FROM orders_table
  GROUP BY `Buyer ID`
) AS customer_values;
```

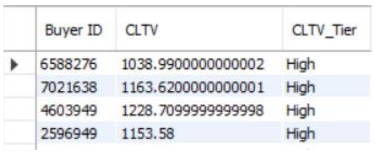


avg_cltv
127.74719140272985

3. What is the distribution of customers across value tiers? (Low, Medium, High)

```
SELECT `Buyer ID`,
      SUM(`Final_Revenue`) AS CLTV,
      CASE
        WHEN SUM(`Final_Revenue`) > 1000 THEN 'High'
        WHEN SUM(`Final_Revenue`) BETWEEN 500 AND 1000 THEN 'Medium'
        ELSE 'Low'
      END AS CLTV_Tier
```

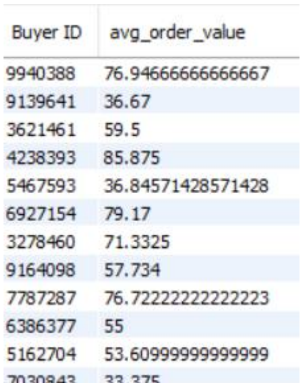
```
FROM orders_table
GROUP BY `Buyer ID`;
```



Buyer ID	CLTV	CLTV_Tier
6588276	1038.9900000000002	High
7021638	1163.6200000000001	High
4603949	1228.7099999999998	High
2596949	1153.58	High

4. What is the average order value (AOV) per customer?

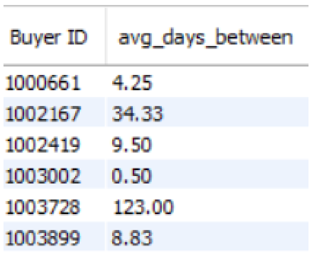
```
SELECT `Buyer ID`,

    SUM(`Final Revenue`) / COUNT(*) AS avg_order_value

FROM orders_table

GROUP BY `Buyer ID`;
```



Buyer ID	avg_order_value
9940388	76.94666666666667
9139641	36.67
3621461	59.5
4238393	85.875
5467593	36.84571428571428
6927154	79.17
3278460	71.3325
9164098	57.734
7787287	76.72222222222223
6386377	55
5162704	53.60999999999999
7030843	33.375

5. What is the average time between purchases for each customer?

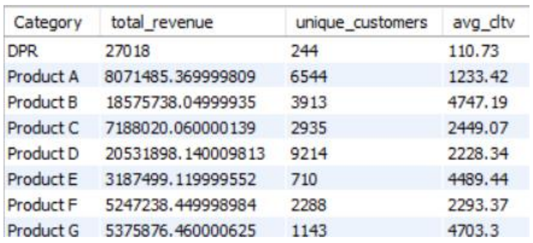
```
SELECT `Buyer ID`,

    ROUND(AVG(DATEDIFF(next_date, curr_date)), 2) AS avg_days_between

FROM (

  SELECT `Buyer ID`,

      STR_TO_DATE(`Date`, '%d/%m/%Y') AS curr_date,

      LEAD(STR_TO_DATE(`Date`, '%d/%m/%Y')) OVER (

        PARTITION BY `Buyer ID`

        ORDER BY STR_TO_DATE(`Date`, '%d/%m/%Y')

      ) AS next_date

  FROM orders_table

) sub

WHERE next_date IS NOT NULL

GROUP BY `Buyer ID`;
```



Buyer ID	avg_days_between
1000661	4.25
1002167	34.33
1002419	9.50
1003002	0.50
1003728	123.00
1003899	8.83

6. Which customer segment gives the highest lifetime value?

```
SELECT p.Category,
       SUM(o.`Final Revenue`) AS total_revenue,
       COUNT(DISTINCT o.`Buyer ID`) AS unique_customers,
       ROUND(SUM(o.`Final Revenue`) / COUNT(DISTINCT o.`Buyer ID`), 2) AS avg_cltv
FROM orders_table o
JOIN products_table p ON o.`Item ID` = p.`Item ID`
GROUP BY p.Category;
```

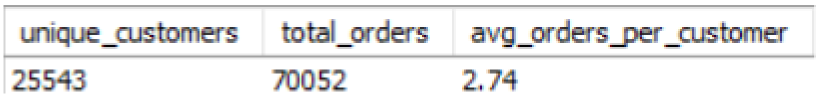


Category	total_revenue	unique_customers	avg_cltv
DPR	27018	244	110.73
Product A	8071485.369999809	6544	1233.42
Product B	18575738.04999935	3913	4747.19
Product C	7188020.060000139	2935	2449.07
Product D	20531898.140009813	9214	2228.34
Product E	3187499.119999552	710	4489.44
Product F	5247238.449998984	2288	2293.37
Product G	5375876.460000625	1143	4703.3

Customer Behaviour & Churn

1. What is the repeat purchase rate?

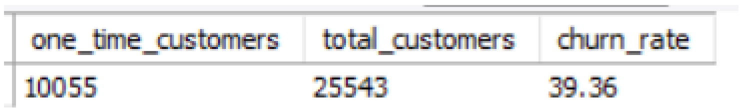
```
SELECT
  COUNT(DISTINCT `Buyer ID`) AS unique_customers,
  COUNT(*) AS total_orders,
  ROUND(COUNT(*) * 1.0 / COUNT(DISTINCT `Buyer ID`), 2) AS avg_orders_per_customer
FROM orders_table;
```



unique_customers	total_orders	avg_orders_per_customer
25543	70052	2.74

2. What is churn rate (customers who never returned after first order)?

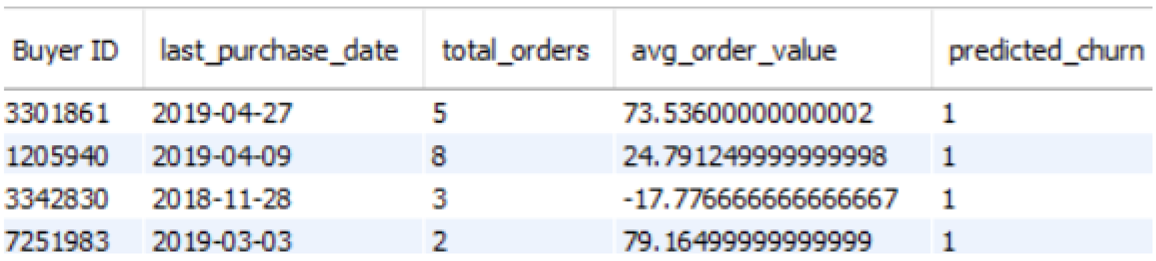
```
SELECT
  COUNT(*) AS one_time_customers,
  (SELECT COUNT(DISTINCT `Buyer ID`) FROM orders_table) AS total_customers,
  ROUND(100.0 * COUNT(*) / (SELECT COUNT(DISTINCT `Buyer ID`) FROM orders_table), 2) AS churn_rate
FROM (
  SELECT `Buyer ID`
  FROM orders_table
  GROUP BY `Buyer ID`
  HAVING COUNT(*) = 1
) AS one_time;
```



one_time_customers	total_customers	churn_rate
10055	25543	39.36

3. Rule-Based Churn Prediction (Simulated Classification)

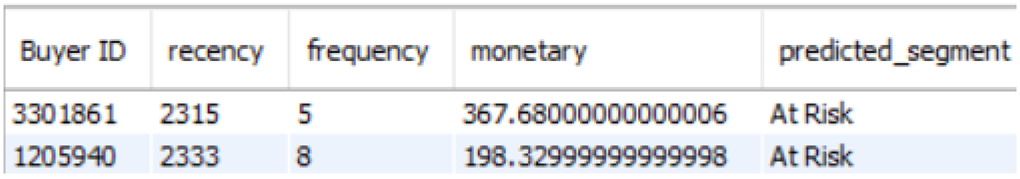
```
SELECT
 `Buyer ID`,
 MAX(STR_TO_DATE(`Date`, '%d/%m/%Y')) AS last_purchase_date,
 COUNT(*) AS total_orders,
 AVG(`Final Revenue`) AS avg_order_value,
 CASE
  WHEN COUNT(*) = 1 THEN 1
  WHEN DATEDIFF(CURDATE(), MAX(STR_TO_DATE(`Date`, '%d/%m/%Y'))) > 60 THEN 1
  WHEN AVG(`Final Revenue`) < 200 THEN 1
  ELSE 0
 END AS predicted_churn
FROM orders_table
GROUP BY `Buyer ID`;
```



Buyer ID	last_purchase_date	total_orders	avg_order_value	predicted_churn
3301861	2019-04-27	5	73.53600000000002	1
1205940	2019-04-09	8	24.791249999999998	1
3342830	2018-11-28	3	-17.776666666666667	1
7251983	2019-03-03	2	79.16499999999999	1

4. CLTV Prediction via RFM Segmentation (Simulated Regression)

```
WITH rfm AS (
  SELECT
    `Buyer ID`,
    DATEDIFF(CURDATE(), MAX(STR_TO_DATE(`Date`, '%d/%m/%Y'))) AS recency,
    COUNT(*) AS frequency,
    SUM(`Final Revenue`) AS monetary
  FROM orders_table
  GROUP BY `Buyer ID`
)
SELECT *,
  CASE
    WHEN recency <= 30 AND frequency >= 5 AND monetary >= 1000 THEN 'Top Customer'
    WHEN recency <= 60 AND frequency >= 3 THEN 'Potential Loyalist'
    WHEN frequency = 1 THEN 'One-time Buyer'
    ELSE 'At Risk'
  END AS predicted_segment
FROM rfm;
```

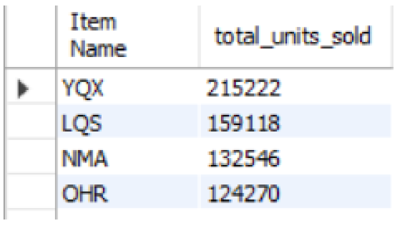


Buyer ID	recency	frequency	monetary	predicted_segment
3301861	2315	5	367.68000000000006	At Risk
1205940	2333	8	198.32999999999998	At Risk

Product & Profitability Analysis

1. Top 5 products by units sold

```
SELECT p.`Item Name`, SUM(o.`Final Quantity`) AS total_units_sold
FROM orders_table o
JOIN products_table p ON o.`Item ID` = p.`Item ID`
GROUP BY p.`Item Name`
ORDER BY total_units_sold DESC
LIMIT 5;
```



Item Name	total_units_sold
YQX	215222
LQS	159118
NMA	132546
OHR	124270

2. Which version sells the most?

SELECT p.`Version`, SUM(o.`Purchased Item Count`) AS total_sold

FROM orders_table o

JOIN products_table p ON o.`Item ID` = p.`Item ID`

GROUP BY p.`Version`

ORDER BY total_sold DESC

LIMIT 1;

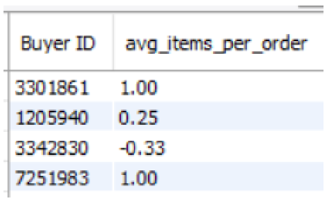
3. What's the average number of products per order for each customer?

SELECT `Buyer ID`,

ROUND(AVG(`Final Quantity`), 2) AS avg_items_per_order

FROM orders_table

GROUP BY `Buyer ID`;



Buyer ID	avg_items_per_order
3301861	1.00
1205940	0.25
3342830	-0.33
7251983	1.00

4. What is the impact of refunds on lifetime value?

SELECT `Buyer ID`,

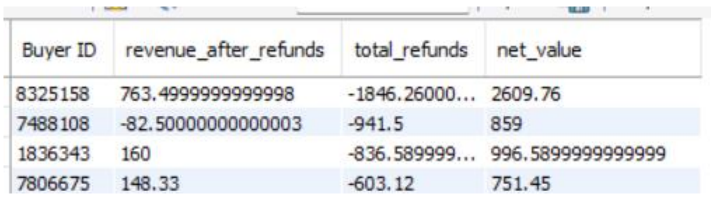
SUM(`Final Revenue`) AS revenue_after_refunds,

SUM(`Refunds`) AS total_refunds,

SUM(`Final Revenue`) - SUM(Refunds) AS net_value

FROM orders_table

GROUP BY `Buyer ID`;



Buyer ID	revenue_after_refunds	total_refunds	net_value
8325158	763.4999999999998	-1846.26000...	2609.76
7488108	-82.50000000000003	-941.5	859
1836343	160	-836.589999...	996.5899999999999
7806675	148.33	-603.12	751.45

Refunds & Risk Analysis

1. Total number of refunded items

SELECT SUM(`Refunded Item Count`) AS total_refunded_items FROM orders_table;

-- Refund ratio by category

SELECT p.`Category`,

SUM(o.`Refunded Item Count`) AS refunded,

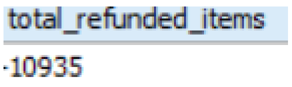
SUM(o.`Purchased Item Count`) AS purchased,

*ROUND(100.0 * SUM(o.`Refunded Item Count`) / NULLIF(SUM(o.`Purchased Item Count`), 0), 2) AS refund_ratio*

FROM orders_table o

JOIN products_table p ON o.`Item ID` = p.`Item ID`

GROUP BY p.`Category`;



total_refunded_items
-10935

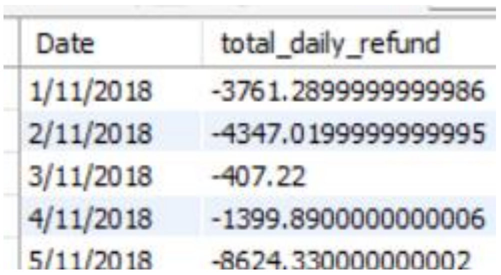
2. Daily refund trend

SELECT `Date`, SUM(`Refunds`) AS total_daily_refund

FROM orders_table

GROUP BY `Date`

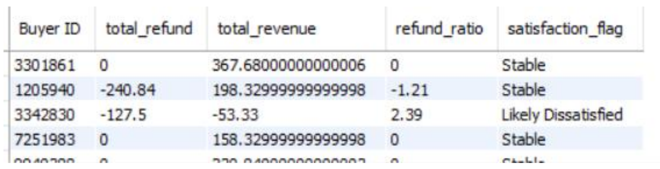
ORDER BY STR_TO_DATE(`Date`, '%d/%m/%Y');



Date	total_daily_refund
1/11/2018	-3761.2899999999986
2/11/2018	-4347.0199999999995
3/11/2018	-407.22
4/11/2018	-1399.8900000000006
5/11/2018	-8624.330000000002

3. Refund Impact Modeling (Customer Sentiment Proxy)

```sql
SELECT `Buyer ID`,
    SUM(`Refunds`) AS total_refund,
    SUM(`Final Revenue`) AS total_revenue,
    ROUND(SUM(`Refunds`) / NULLIF(SUM(`Final Revenue`), 0), 2) AS refund_ratio,
    CASE
      WHEN ROUND(SUM(`Refunds`) / NULLIF(SUM(`Final Revenue`), 0), 2) > 0.3 THEN 'Likely Dissatisfied'
      ELSE 'Stable'
    END AS satisfaction_flag
FROM orders_table
GROUP BY `Buyer ID`;
```

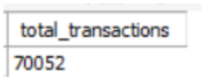


Buyer ID	total_refund	total_revenue	refund_ratio	satisfaction_flag
3301861	0	367.68000000000006	0	Stable
1205940	-240.84	198.32999999999998	-1.21	Stable
3342830	-127.5	-53.33	2.39	Likely Dissatisfied
7251983	0	158.32999999999998	0	Stable

Revenue & Transaction Metrics

1. Total number of transactions

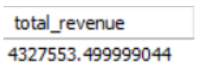
```sql
SELECT COUNT(*) AS total_transactions FROM orders_table;
```



total_transactions
70052

2. Total Revenue Generated

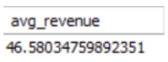
```sql
SELECT SUM(`Total Revenue`) AS total_revenue FROM orders_table;
```



total_revenue
4327553.499999044

3. Average revenue per transaction

```sql
SELECT AVG(`Final Revenue`) AS avg_revenue FROM orders_table;
```

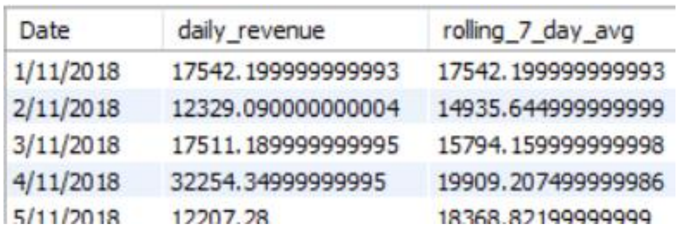


avg_revenue
46.58034759892351

Trends & Forecasting

1. Trend-Based Forecasting via Moving Averages (Simulated Time Series)

```
SELECT
  `Date`,
  SUM(`Final Revenue`) AS daily_revenue,
  AVG(SUM(`Final Revenue`)) OVER (
    ORDER BY STR_TO_DATE(`Date`, '%d/%m/%Y')
    ROWS BETWEEN 6 PRECEDING AND CURRENT ROW
  ) AS rolling_7_day_avg
FROM orders_table
GROUP BY `Date`
ORDER BY STR_TO_DATE(`Date`, '%d/%m/%Y');
```



Date	daily_revenue	rolling_7_day_avg
1/11/2018	17542.199999999993	17542.199999999993
2/11/2018	12329.090000000004	14935.644999999999
3/11/2018	17511.189999999995	15794.159999999998
4/11/2018	32254.34999999995	19909.207499999986
5/11/2018	12207.28	18368.82199999999