Executive Summary -- Customer Segmentation & Retention Strategy Using SQL

Business Context

Company ABC is an e-commerce retailer specializing in catering to customers across multiple countries and product lines (Bikes, Accessories, and Clothing).

The leadership team wanted to understand ***which customer segments drive the most value***, how their engagement behaviours differ, and where opportunities exist to expand business and improve retention.

Business Goal

Strengthen customer retention and drive sustainable revenue growth by analyzing high-value (VIP) customer behaviour and uncovering opportunities to expand engagement across segments and product lines.

Analytical Approach

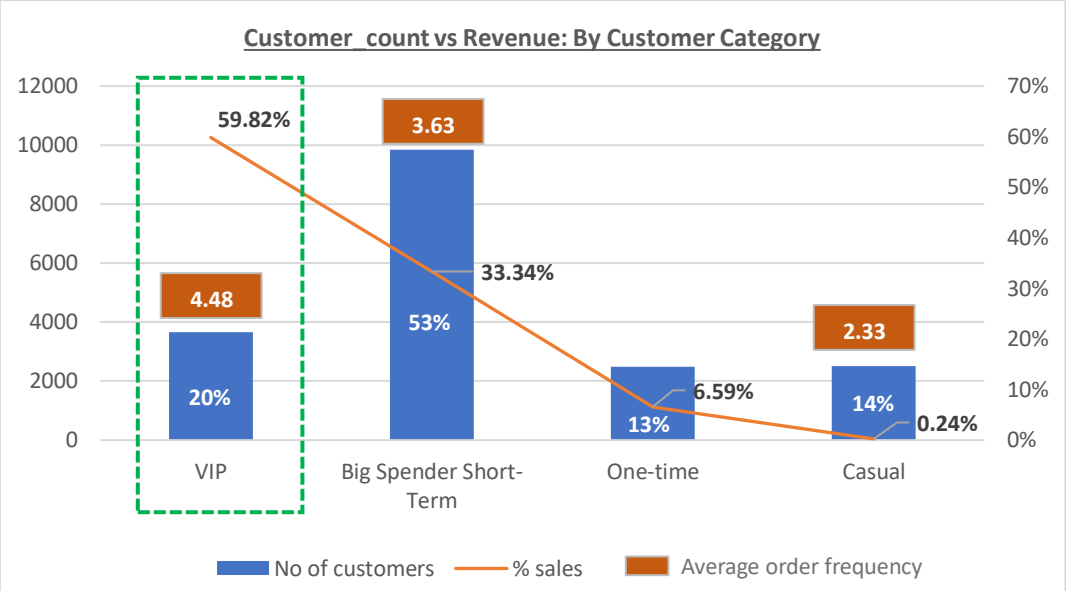
Data from **customers, products, and orders** tables was cleaned and analysed using **SQL (CTEs, window functions, segmentation logic)** to:

- Segment customers into four groups based on lifespan and engagement.
- Identify key behavioural (order frequency, contribution to revenue) and product-level differences across segments.
- Evaluate of retention patterns.

Strategic Takeaways

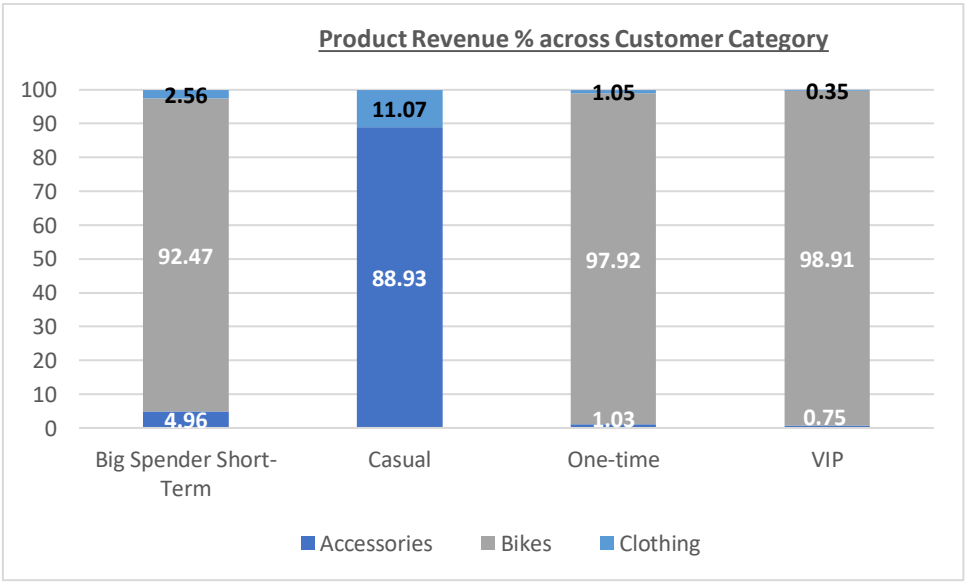
1. **Value is concentrated but fragile:** A small group of customers contribute the majority of revenue, yet their engagement depth remains shallow, posing a risk for long-term revenue stability.
2. **Category diversification is key:** Overreliance on a single high-ticket category (Bikes) limits repeat purchases and cross-sell potential.
3. **Retention ROI is highest in early lifecycle:** Targeting Big Spenders within their first 60–90 days post-purchase can yield faster gains than broad-based acquisition efforts.
4. **Data-driven loyalty design:** Combining behavioural metrics (frequency, recency) with transaction data can create smarter segmentation for future marketing interventions.

Insight 1: VIPs drive disproportionate value but their engagement is shallow.



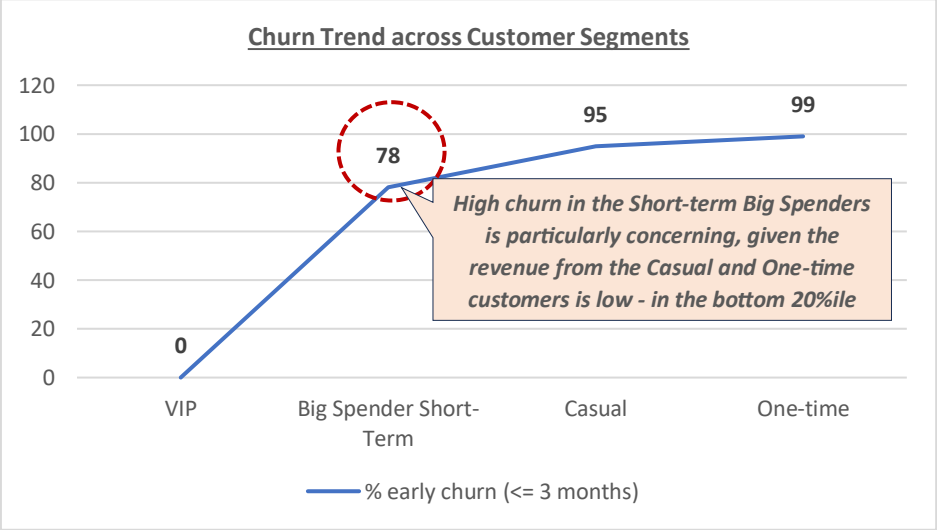
- **Observation:** VIPs form only **20% of customers**, yet contribute **60% of total sales** and have the **highest average order size** (₹4.8K+).
- **However**, their **average order frequency (≈4.5)** is only marginally higher than other segments.
- **Interpretation:** High-value customers are transacting big, but not often.
- **Actionable takeaway:** Retention programs need to focus on increasing purchase frequency, not ticket size.

Insight 2: Product concentration risk – Bikes dominate VIP spend.



- **Observation:** Among all the bike purchase, **61% revenue is from VIPs**. The other categories of Accessories and Clothing are underpenetrated by the VIPs i.e. <20%. But Bikes also make up **98% of the total VIP revenue**.
- **Interpretation:** The brand is over-reliant on one product category. This exposes the business to cyclical demand shocks and reduces repeat potential (since Bikes have a long lifecycle).
- **Actionable takeaway:**
    - Cross-sell complementary categories (Accessories, Clothing) post-bike purchase.
    - Introduce loyalty bundles or early upgrade programs for existing bike customers.

Insight 3: Early churn is significant across segments – which is concerning especially among "Big Spender Short-Term" customers.



- **Observation:** Nearly **80% of Big Spenders churn within 3 months**.
- **Interpretation:** These customers are capable of high-value purchases but don't find recurring value. Could be due to a product-fit, app experience, etc.
- **Actionable takeaway:**
    - Build retention triggers right after high-value purchases (or within 30-60 days of purchase) like personalized communication, upgrade options, referral bonuses, discounts for complementary products.
    - Track early engagement signals to predict and prevent churn.

----- SQL Queries ---

1. ***Assigning price ranges for products***

```
WITH product_segments AS (
    SELECT
        p.product_key,
        p.category,
        p.product_name,
        p.cost,
        o.sales_amount,
        CASE WHEN p.cost < 100 THEN 'Below 100 $'
            WHEN p.cost BETWEEN 100 AND 500 THEN '100-500 $'
            WHEN p.cost BETWEEN 500 AND 1000 THEN '500-1000 $'
            ELSE 'Above 1000 $'
        END AS cost_range
    FROM products p
        JOIN orders o
        ON p.product_key = o.product_key)

SELECT
    category,
    cost_range,
    COUNT(product_key) AS total_products,
    ROUND(SUM(sales_amount),0) AS total_sales_per_product
FROM product_segments
GROUP BY cost_range, category
ORDER BY category, total_sales_per_product DESC;
```

2. ***Analysing purchase trends - repeat and single purchases***

```
– 1st CTE: defining the lifespan
WITH cust_lifespan AS(
SELECT
        c.country,
        c.customer_key,
        COUNT(o.order_number) as total_orders,
        MAX(o.order_date) as last_order,
        MIN(o.order_date) as first_order,
        CONCAT (EXTRACT (YEAR FROM (AGE( MAX(order_date), MIN(order_date)))), ' years ',
        EXTRACT (MONTH FROM (AGE( MAX(order_date), MIN(order_date)))), ' months') as lifespan
FROM customers c
        LEFT JOIN orders o
        ON c.customer_key = o.customer_key
GROUP BY c.country, c.customer_key
ORDER BY COUNT(order_number) DESC),

– 2nd CTE: grouping the lifespan intervals
lifespan_category AS(
SELECT
        country,
        customer_key,
        lifespan,
CASE WHEN last_order-first_order <= 12 THEN '0-12 months'
        WHEN (last_order-first_order > 12) AND (last_order-first_order <= 24) THEN '1-2 years'
        WHEN (last_order-first_order > 24) AND (last_order-first_order <= 36) THEN '2-3 years'
        ELSE 'Up to 4 years'
```

```
END category
FROM cust_lifespan)

– Main query
SELECT
country,
category,
COUNT(customer_key) as no_of_customers
FROM lifespan_category
GROUP BY category, country
ORDER BY category ASC, COUNT(customer_key) DESC;
```

3. Customer Value Segmentation - # of orders, lifespan

— CTE 1:

```
WITH percentile_rank AS(
SELECT
    c.country,
    c.customer_key,
    CONCAT(c.first_name, ' ', c.last_name) as customer_name,
    COUNT(order_number) as total_orders,
    SUM(sales_amount) as total_sales,
    --DENSE_RANK() OVER (PARTITION BY c.country ORDER BY
(COUNT(order_number)) DESC) as customer_sales_rank,
    NTILE (100) OVER (PARTITION BY c.country ORDER BY
(COUNT(order_number)) ASC) as orders_percentile,
    NTILE (100) OVER (PARTITION BY c.country ORDER BY
(SUM(sales_amount)) ASC) as sales_percentile,
    MIN(o.order_date) as first_order,
    MAX(o.order_date) as last_order,
```

CONCAT(EXTRACT (YEAR FROM (AGE (MAX(o.order_date), MIN(o.order_date)))), ' years ', EXTRACT (MONTH FROM (AGE(MAX(o.order_date), MIN(o.order_date)))),' months') as lifespan,

(MAX(o.order_date) - MIN(o.order_date))/ 30 as lifespan_months

FROM customers c

LEFT JOIN orders o

ON c.customer_key = o.customer_key

GROUP BY c.country, c.customer_key, CONCAT(c.first_name, ' ', c.last_name)

ORDER BY c.country ASC, total_orders DESC),

— CTE 2:

cust_category AS(

SELECT

pr.customer_key,

CASE WHEN total_orders = 1 THEN 'One-time'

WHEN sales_percentile > 20 AND EXTRACT (YEAR FROM (AGE (last_order, first_order))) >= 1 THEN 'VIP'

WHEN sales_percentile > 20 AND EXTRACT (YEAR FROM (AGE (last_order, first_order))) < 1 THEN 'Big Spender Short-Term'

WHEN sales_percentile <= 20 AND EXTRACT (YEAR FROM (AGE (last_order, first_order))) >= 1 THEN 'Loyal Low-Spend'

ELSE 'Casual'

END category

FROM percentile_rank pr)

— Main Query 1:

SELECT

pr.country,

pr.customer_key,

pr.total_orders,

pr.total_sales,

pr. orders_percentile,

pr. sales_percentile,

pr.lifespan,

cc.category

FROM percentile_rank pr

JOIN cust_category cc

ON pr.customer_key = cc.customer_key

ORDER BY pr.country, pr.total_sales DESC ;

— Main Query 2: customer_category across countries

SELECT

- *cc.category,*
- *pr.country,*
- *--pr.customer_key*
- *COUNT (pr.customer_key),*
- *ROW_NUMBER() OVER (PARTITION BY cc.category ORDER B*

COUNT (pr.customer_key) DESC)

- *ROW_NUMBER() OVER (PARTITION BY cc.category ORDER BY COUNT (pr.customer_key) DESC),*
- *ROUND(100.0 * COUNT(pr.customer_key) / SUM(COUNT(pr.customer_key)) OVER (PARTITION BY pr.country),2) AS pct_of_country*

FROM percentile_rank pr

JOIN cust_category cc

ON pr.customer_key = cc.customer_key

--WHERE category = 'Big Spender Short-Term'

GROUP BY pr.country, cc.category

ORDER BY cc.category ASC;

— Main Query 3: customer_category avg_lifespan

SELECT

cc.category,

```
   ROUND(AVG (pr.last_order - pr.first_order)/30,2) AS
category_avg_lifespan_months

FROM percentile_rank pr

JOIN cust_category cc

ON pr.customer_key = cc.customer_key

GROUP BY cc.category


— Main Query 4: category-wise core statistics

SELECT

cc.category,

        COUNT(pr.customer_key) AS total_customers_category,

        ROUND(AVG(pr.total_orders),2) AS avg_orders_category,

        ROUND(SUM(pr.total_sales),2) AS total_sales_category,

        AVG(pr.total_sales) AS avg_sales_category,

        PERCENTILE_CONT (0.5) WITHIN GROUP (ORDER BY pr.total_sales)
OVER () AS median_sales_category, — this doesn't need OVER() as GROUP BY
already defines the scope + percent_cont is aggregate function

        ROUND(100.0*COUNT(pr.customer_key)/ SUM(COUNT(pr.customer_key))
OVER (),2) AS pct_customers, — these need OVER () as the denominator needs to
run across ALL groups

        ROUND(100.0*SUM(pr.total_sales)/ SUM(SUM(pr.total_sales)) OVER (),2)
AS pct_sales

FROM percentile_rank pr

JOIN cust_category cc

ON pr.customer_key = cc.customer_key

GROUP BY cc.category

ORDER BY total_sales_category DESC;


— Main Query 5: category-wise product lines

SELECT

        cc.category,

        p.category,
```

```
    --p.subcategory,
    COUNT(*) AS total_products,
    SUM(sales_amount) AS total_sales,
    ROUND(100*SUM(sales_amount)/ SUM(SUM(sales_amount)) OVER
(PARTITION BY cc.category),2) as pct_sales_cust,
    ROUND(100*SUM(sales_amount)/ SUM(SUM(sales_amount)) OVER
(PARTITION BY p.category),2) as pct_sales_product
FROM cust_category cc
LEFT JOIN orders o ON cc.customer_key = o.customer_key
LEFT JOIN products p ON o.product_key = p.product_key
GROUP BY cc.category, p.category
--p.subcategory
ORDER BY cc.category, SUM(sales_amount) DESC;
```

4. Customer Segment Wise - Early churn (<=3 months)

— CASE based conditional aggregation:

```
SELECT
```

- cc.category,
- COUNT(DISTINCT pr.customer_key) AS total_cust_category,
- COUNT(DISTINCT CASE WHEN pr.lifespan_months <= 3 THEN pr.customer_key END) AS early_churn_customers,
- ROUND(100*COUNT (DISTINCT CASE WHEN pr.lifespan_months <= 3 THEN pr.customer_key END)/ COUNT(DISTINCT pr.customer_key)) AS pct_early_churn

```
FROM percentile_rank pr
JOIN cust_category cc
ON pr.customer_key = cc.customer_key
GROUP BY category
```

— Using CTEs:

```
-- CTE 1: Early churners per category
WITH early_churn AS (
    SELECT
        cc.category,
        COUNT(DISTINCT pr.customer_key) AS early_churn_customers
    FROM percentile_rank pr
    JOIN cust_category cc
        ON pr.customer_key = cc.customer_key
    WHERE pr.lifespan_months <= 3
    GROUP BY cc.category),

-- CTE 2: Total customers per category
total_cust AS (
    SELECT
        cc.category,
        COUNT(DISTINCT pr.customer_key) AS total_customers
    FROM percentile_rank pr
    JOIN cust_category cc
        ON pr.customer_key = cc.customer_key
    GROUP BY cc.category)

-- Final output
SELECT
    - t.category,
    - t.total_customers,
    - COALESCE (e.early_churn_customers, 0) AS early_churn_customers,
    - ROUND(100.0 * COALESCE(e.early_churn_customers, 0) /
      t.total_customers, 2) AS pct_early_churn
FROM total_cust t
LEFT JOIN early_churn e
```

```
    ON t.category = e.category
ORDER BY pct_early_churn DESC;
```