**Optimizing Online Sports Revenue**

The core objective of the project involves analysing product data for an online sports retail enterprise to optimize revenue.

Background of the Project

Sports clothing is a booming enterprise. Many athleisure brands such as Lululemon, Adidas and Nike are competing for a pie of the billion dollar apparel market. Sports clothing and athleisure attire is a large industry, worth approximately $193 billion in 2021 with a strong growth forecast over the next decade.

Emerging players are leveraging on technology in creating competitive advantages and delivering value to consumers.

The sports company under review, Malulu Inc. adopted a digital business model whereby they set up a digital store. Customers can browse, place products in carts and purchase at the click of a button. They can also track their delivery.

Malulu Inc. would like to improve their revenue margins. The sports company’s product manager assigns you (Data Analyst) the responsibility of analysing the company’s product data and recommend feasible strategies of improving the company’s revenue margins.

The product data provided entails numeric, string, and timestamp data on pricing and revenue, ratings, reviews, descriptions, and website traffic.

Analytical Tools

SQL and Python are the primary analytical tools that will be used by the data analyst in analysing Malulu Inc.’s product data.

Project Tasks

1. **Counting Missing Values**

The database provided to us***, sports***, contains five tables, with product\_id being the primary key for all of them:

Steps for this phase:

* Count the total number of products, along with the number of non-missing values in description, listing\_price, and last\_visited.
* Count all columns from the info table, aliasing as total\_rows.
* Count the number of non-missing values for description as count\_description, listing\_price as count\_listing\_price, and last\_visited as count\_last\_visited.
* Join the info table with finance and traffic, matching on product\_id.
* The line postgresql:///sports is used to connect to the database; don't remove it.

#SQL Code Snippet

%%sql

postgresql:///sports

-- *Count all columns as total\_rows*

*-- Count the number of non-missing entries for description, listing\_price, and last\_visited*

*-- Join info, finance, and traffic*

SELECT COUNT(\*) AS total\_rows,

COUNT(i.description) AS count\_description,

COUNT(f.listing\_price) AS count\_listing\_price,

COUNT(t.last\_visited) AS count\_last\_visited

FROM info AS i

INNER JOIN finance AS f

ON i.product\_id = f.product\_id

INNER JOIN traffic AS t

ON t.product\_id = f.product\_id;

Based on output of the query above, there are 3,179 products

1. **Nike vs Adidas Pricing**

How do the price points of Nike and Adidas products differ?

#SQL Code Snippet

%%sql

*-- Select the brand, listing\_price as an integer, and a count of all products in finance*

*-- Join brands to finance on product\_id*

*-- Filter for products with a listing\_price more than zero*

*-- Aggregate results by brand and listing\_price, and sort the results by listing\_price in descending order*

SELECT b.brand, f.listing\_price::integer, COUNT(f.\*)

FROM finance AS f

INNER JOIN brands AS b

ON f.product\_id = b.product\_id

WHERE listing\_price > 0

GROUP BY b.brand, f.listing\_price

ORDER BY listing\_price DESC;

The output is 77 rows detailing price ranges of each brands in descending order

1. **Labelling and Price Ranges**

It turns out there are 77 unique prices for the products in our database, which makes the output of our last query quite difficult to analyse.

Let's build on our previous query by assigning labels to different price ranges, grouping by brand and label. We will also include the total revenue for each price range and brand.

#SQL Code Snippet

%%sql

-- *Select the brand, a count of all products in the finance table, and total revenue*

*-- Create four labels for products based on their price range, aliasing as price\_category*

*-- Join brands to finance on product\_id and filter out products missing a value for brand*

*-- Group results by brand and price\_category, sort by total\_revenue*

SELECT b.brand, COUNT(f.\*), SUM(f.revenue) as total\_revenue,

CASE WHEN f.listing\_price < 42 THEN 'Budget'

WHEN f.listing\_price >= 42 AND f.listing\_price < 74 THEN 'Average'

WHEN f.listing\_price >= 74 AND f.listing\_price < 129 THEN 'Expensive'

ELSE 'Elite' END AS price\_category

FROM finance AS f

INNER JOIN brands AS b

ON f.product\_id = b.product\_id

WHERE b.brand IS NOT NULL

GROUP BY b.brand, price\_category

ORDER BY total\_revenue DESC;

The output is 8 rows detailing all products of all brands grouped by price category and sorted by revenue in descending order.

1. **Average Discount by Brand**

Insight: Adidas items generate more total revenue regardless of price category! Specifically, "Elite" Adidas products priced $129 or more typically generate the highest revenue, so the company can potentially increase revenue by shifting their stock to have a larger proportion of these products

Analysing revenue margins via discount perspective will enable us determine how discounts offered by brands influence revenue margins earned by the company.

#SQL Code Snippet

%%sql

-- *Select brand and average\_discount as a percentage*

*-- Join brands to finance on product\_id*

*-- Aggregate by brand*

*-- Filter for products without missing values for brand*

SELECT b.brand, AVG (f.discount) \* 100 AS average\_discount

FROM brands AS b

INNER JOIN finance AS f

ON b.product\_id = f.product\_id

GROUP BY b.brand

HAVING b.brand IS NOT NULL

ORDER BY average\_discount;

Adidas offers discounts at 33% thus contributing to its high revenue margins. Nike does not offer discounts on its products.

Recommendation: the company could try to reduce the amount of discount offered on Adidas products, and monitor sales volume to see if it remains stable. Alternatively, it could try offering a small discount on Nike products. This would reduce average revenue for these products, but may increase revenue overall if there is an increase in the volume of Nike products sold.

1. **Correlation between Revenue and Reviews**

From a marketing point of view, do reviews influence sales? This can be determined by examining the correlation between revenue and reviews

#SQL Code Snippet

%%sql

*-- Calculate the correlation between reviews and revenue as review\_revenue\_corr*

*-- Join the reviews and finance tables on product\_id*

SELECT corr(r.reviews, f.revenue) AS review\_revenue\_corr

FROM reviews AS r

INNER JOIN finance AS f

ON r.product\_id = f.product\_id;

Output is 0.65, depicting a strong positive correlation between revenue and reviews. Increasing positive ratings subsequently increases sales generated by the e-commerce company.

1. **Ratings and reviews by product description length**

Here we analyse impact of length of product description on product reviews and ratings, hence consequent impact on revenue.

#SQL Code Snippet

%%sql

*-- Calculate description\_length*

*-- Convert rating to a numeric data type and calculate average\_rating*

*-- Join info to reviews on product\_id and group the results by description\_length*

*-- Filter for products without missing values for description, and sort results by description\_length*

SELECT TRUNC(LENGTH(i.description), -2) AS description\_length,

ROUND(AVG(r.rating::numeric), 2) AS average\_rating

FROM info AS i

INNER JOIN reviews AS r

ON i.product\_id = r.product\_id

WHERE i.description IS NOT NULL

GROUP BY description\_length

ORDER BY description\_length;

Based on the output of the above code, there is no relationship between the length of the product’s description and its rating.

1. **Reviews by Brand and Month**

Let us analyse reviews by month to see if there are any trends or gaps to exploit.

#SQL Code Snippet

%%sql

*-- Select brand, month from last\_visited, and a count of all products in reviews aliased as num\_reviews*

*-- Join traffic with reviews and brands on product\_id*

*-- Group by brand and month, filtering out missing values for brand and month*

*-- Order the results by brand and month*

SELECT b.brand, DATE\_PART('month', t.last\_visited) AS month, COUNT(r.\*) AS num\_reviews

FROM brands AS b

INNER JOIN traffic AS t

ON b.product\_id = t.product\_id

INNER JOIN reviews AS r

ON t.product\_id = r.product\_id

GROUP BY b.brand, month

HAVING b.brand IS NOT NULL

AND DATE\_PART('month', t.last\_visited) IS NOT NULL

ORDER BY b.brand, month;

Based on the output of the code, product reviews are the highest in the first quarter of the year. There is scope to run experiments aiming to increase the volume of reviews in the other nine months.

1. **Footwear Product Performance**

Let’s switch our attention to the type of products being sold. As there are no labels for product type, we will create a Common Table Expression (CTE) that filters description for keywords, then use the results to find out how much of the company's stock consists of footwear products and the median revenue generated by these items.

#SQL Code Snippet

%%sql

*-- Create the footwear CTE, containing description and revenue*

*-- Filter footwear for products with a description containing %shoe%, %trainer, or %foot%*

*-- Also filter for products that are not missing values for description*

*-- Calculate the number of products and median revenue for footwear products*

WITH footwear AS

(

SELECT i.description, f.revenue

FROM info AS i

INNER JOIN finance AS f

ON i.product\_id = f.product\_id

WHERE i.description ILIKE '%shoe%'

OR i.description ILIKE '%trainer%'

OR i.description ILIKE '%foot%'

AND i.description IS NOT NULL

)

SELECT COUNT(\*) AS num\_footwear\_products,

percentile\_disc(0.5) WITHIN GROUP (ORDER BY revenue) AS median\_footwear\_revenue

FROM footwear;

Based on the output, there are 2,700 footwear products with a median average of $3186.

1. **Clothing Product Performance**

To assess whether footwear's median revenue is good or bad compared to other products, let's examine how this differs to clothing products.

#SQL Code Snippet

%%sql

*-- Copy the footwear CTE from the previous task*

*-- Calculate the number of products in info and median revenue from finance*

*-- Inner join info with finance on product\_id*

*-- Filter the selection for products with a description not in footwear*

WITH footwear AS

(

SELECT i.description, f.revenue

FROM info AS i

INNER JOIN finance AS f

ON i.product\_id = f.product\_id

WHERE i.description ILIKE '%shoe%'

OR i.description ILIKE '%trainer%'

OR i.description ILIKE '%foot%'

AND i.description IS NOT NULL

)

SELECT COUNT(i.\*) AS num\_clothing\_products,

percentile\_disc(0.5) WITHIN GROUP (ORDER BY f.revenue) AS median\_clothing\_revenue

FROM info AS i

INNER JOIN finance AS f on i.product\_id = f.product\_id

WHERE i.description NOT IN (SELECT description FROM footwear);