Project Diary

Dataset

This project was created using the *E-commerce Business Transaction* dataset from the Kaggle source below.

Source: https://www.kaggle.com/datasets/gabrielramos87/an-online-shop-business

Project scenario

We are working for a young business that sells a variety of products. Overall, the company is performing well but they want to look at improving their profitability by looking into adjusting the product lines they sell to see if they can either streamline the types of products sold to save on manufacturing costs or find a product category to focus on.

Goals

For this project, we aim to create a Power BI report that allows the business to make decisions based on the following questions:

What products / categories are most profitable? Does this differ on country?

Of these profitable products, are they purchased by many customers or are they purchased in large quantities by a small selection of customers?

What products have had most recent improvement?

We will utilise SQL in *MySQL* to prepare and explore our initial data before importing into Power BI.

Importing data

We took our data and first brought it into MySQL to prepare and cleanse the data, as well as conduct some exploratory analysis. Due to the size of the data (over 500,000 rows), importing the data directly was not possible with the available hardware.

To complete the import of data, I created a table with appropriate columns matching our dataset and used *'LOAD DATA LOCAL INLINE*' to import data directly into the table. The first line was ignored as this was headers.

```
LOAD DATA LOCAL INFILE 'C:/Users/jakeh/Desktop/Data Analysis/Projects/E-commerce Project/SalesDatabase.csv' INTO TABLE Sales
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES;
```

Data preparation and cleansing

We first checked for null values. This returned 0 rows, indicating no null values.

```
SELECT * FROM Sales
WHERE TransactionNo IS NULL
    OR OrderDate IS NULL
    OR ProductNo IS NULL
    OR ProductName IS NULL
    OR Price IS NULL
    OR Quantity IS NULL
    OR CustomerNo IS NULL
    OR Country IS NULL;
```

We also checked the count of orders grouped by *OrderDate*. This was to ensure that the date imported correctly, as different date formats can cause issues.

```
SELECT OrderDate, COUNT(*) FROM Sales
GROUP BY OrderDate
ORDER BY COUNT(*) DESC;
```

We initially wanted to assign categories and themes to each row based on *ProductName*. Categories would refer to the *type* of product, and theme to *events* or *topics.* This was to allow us to conduct category/theme-based analysis to answer business question 1. However, theme-related investigations were dropped as not enough themes were present, with most products being general products not relating to any specific topics.

The categories selected were selected to ensure a reasonable number of results for each category. Category tags were applied using the *CASE* and *LIKE* keywords. We selected numerous words relating to each category to reach as many products as possible.

```
UPDATE Sales
SET Category =
    CASE
        WHEN LOWER(ProductName) LIKE '%bag%' THEN 'Bags'
        WHEN LOWER(ProductName) LIKE '%wallet%' OR
                LOWER(ProductName) LIKE '%purse%' THEN 'Wallets and Purses'
        WHEN LOWER(ProductName) LIKE '%tin%' OR
                LOWER(ProductName) LIKE '%box%' THEN 'Tins and Boxes'
```

Exploratory analysis

We then proceeded to complete exploratory analysis and find answers to our business questions.

Our first question was to investigate product and category profitability, taking country into account. We first ranked products based on the total earnings of that product across all sales. This showed us the top-earning products.

```
SELECT RANK() OVER (ORDER BY SUM(SaleIncome) DESC) AS EarningsRanking,
    ProductName,
    ROUND(SUM(SaleIncome),2) AS TotalEarnings,
    Category,
    Theme
FROM Sales
GROUP BY ProductName, Category, Theme
ORDER BY SUM(SaleIncome) DESC;
```

After this, we investigated how many of the top 50 products belonged to each category, as well as the category earnings for these products. This highlighted that *Kitchen* products were particularly plentiful in the top-ranking products with the highest-total earnings over these 50 products.

```
WITH SalesRanking AS (
    SELECT ProductName,
    SUM(SaleIncome) AS TotalEarnings,
    Category,
    Theme
    FROM Sales
    GROUP BY ProductName, Category, Theme
    ORDER BY SUM(SaleIncome) DESC
    LIMIT 50
)
SELECT Category,
    COUNT(Category) AS CategoryProducts,
    ROUND(SUM(TotalEarnings),2) AS CategoryEarnings,
    ROUND(SUM(TotalEarnings) / COUNT(Category),2) AS EarningsPerUniqueProduct
FROM SalesRanking
GROUP BY Category
ORDER BY SUM(TotalEarnings) DESC;
```

To incorporate country-level sales data, we first looked at the total earnings per country. This showed that 82% of sales were made in the UK, making up the vast majority of earnings for the company.

We then looked at the number of times a product was ranked in the top 5 products across all countries in our database to see if there was correlation between this and total earnings. These results were somewhat expected, with high earning products tending to be ranked in many countries. However, this was not the rule, with some exceptions being present.

The second question was to see *how* products were purchased and returned. To do this, we first looked at the number of sales and returns were made for the top-ranking products. This showed us that most products had a return rate % of between 0.5% and 3%. There were some abnormal product returns through, with some having returns well over 5% indicating a higher-level of risk.

```
WITH PositiveAndNegative AS (
    SELECT ProductName,
        Category,
        CustomerNo,
        CASE WHEN Quantity > 0 THEN Quantity ELSE 0 END AS qty_purchased,
        CASE WHEN Quantity < 0 THEN -Quantity ELSE 0 END AS qty_returned,
        CASE WHEN Quantity > 0 THEN SaleIncome ELSE 0 END AS income,
        CASE WHEN Quantity < 0 THEN -SaleIncome ELSE 0 END AS refunded
    FROM Sales
)
SELECT
    ProductName,
    Category,
    SUM(qty_purchased) AS QuantityPurchased,
    SUM(qty_returned) AS QuantityReturned,
    SUM(qty_purchased) - SUM(qty_returned) AS FinalQuantitySold,
    ROUND((SUM(qty_returned) / SUM(qty_purchased) * 100),2) AS PercentageReturned,
    ROUND(AVG(qty_purchased),2) AS AverageQuantityPurchased,
    MAX(qty_purchased) AS LargestQuantityPurchased,
    ROUND(SUM(income), 2) AS Income,
    ROUND(SUM(refunded), 2) AS ReturnedPayments,
    ROUND(SUM(income) - SUM(refunded), 2) AS TotalProfit,
    COUNT(DISTINCT CASE WHEN qty_purchased > 0 THEN CustomerNo END) AS UniquePurchasers,
    COUNT(DISTINCT CASE WHEN qty_returned  > 0 THEN CustomerNo END) AS UniqueRefunders
FROM PositiveAndNegative
GROUP BY ProductName, Category
ORDER BY SUM(income) - SUM(refunded) DESc
LIMIT 10;
```

We then repeated this process on a categorical level. This highlighted some useful statistics; the return rate % for both *Kitchen* and *Stationary* products was high (over 8% and 17% respectively), average quantity of products purchased per order was fairly consistent over most categories (between 8 and 15 items for the majority), and

unique customers scaled linearly with quantity of products sold.

```
WITH ProductEarnings AS (
    SELECT
        ProductName,
        Category,
        Theme,
        Country,
        ROUND(SUM(SaleIncome),2) AS TotalEarnings
    FROM Sales
    GROUP BY ProductName, Category, Theme, Country
),
CountryRanks AS (
    SELECT
        ProductName,
        Category,
        Theme,
        Country,
        TotalEarnings,
        RANK() OVER (PARTITION BY Country ORDER BY TotalEarnings DESC) AS EarningsRanking
    FROM ProductEarnings
)
SELECT
    ProductName,
    COUNT(*) AS TimesRanked,
    ROUND(SUM(TotalEarnings),2) AS ProductEarnings
FROM CountryRanks
WHERE EarningsRanking > 6
GROUP BY ProductName
ORDER BY TimesRanked DESC;
```

```sql
WITH Categories AS (
    SELECT Category,
        CustomerNo,
        CASE WHEN Quantity > 0 THEN Quantity ELSE 0 END AS qty_purchased,
        CASE WHEN Quantity < 0 THEN -Quantity ELSE 0 END AS qty_returned,
        CASE WHEN Quantity > 0 THEN SaleIncome ELSE 0 END AS income,
        CASE WHEN Quantity < 0 THEN -SaleIncome ELSE 0 END AS refunded
    FROM Sales
)
SELECT
    Category,
    SUM(qty_purchased) AS QuantityPurchased,
    SUM(qty_returned) AS QuantityReturned,
    SUM(qty_purchased) - SUM(qty_returned) AS FinalQuantitySold,
    ROUND((SUM(qty_returned) / SUM(qty_purchased) * 100),2) AS PercentageReturned,
    ROUND(AVG(qty_purchased),2) AS AverageQuantityPurchased,
    MAX(qty_purchased) AS LargestQuantityPurchased,
    ROUND(SUM(income), 2) AS Income,
    ROUND(SUM(refunded), 2) AS ReturnedPayments,
    ROUND(SUM(income) - SUM(refunded), 2) AS TotalProfit,
    COUNT(DISTINCT CASE WHEN qty_purchased > 0 THEN CustomerNo END) AS UniquePurchasers,
    COUNT(DISTINCT CASE WHEN qty_returned  > 0 THEN CustomerNo END) AS UniqueRefunders
FROM Categories
GROUP BY Category
ORDER BY SUM(income) - SUM(refunded) DESC;
```

The last question to investigate was to see which products have shown the most change in sales over the last quarter. We calculated this as a % increase/decrease by product. This data had a very large range and had potential to be useful but would be more easily analysed visually.

```sql
WITH RECURSIVE Bounds AS (
    SELECT DATE_FORMAT(MIN(OrderDate),'%Y-%m-01') AS StartDate,
        DATE_FORMAT(MAX(OrderDate),'%Y-%m-01') AS EndDate
    FROM Sales
),
Calendar AS (
    SELECT CAST(StartDate AS DATE) AS CalendarStart
    FROM Bounds
    UNION ALL
    SELECT DATE_ADD(CalendarStart, INTERVAL 1 MONTH)
    FROM Calendar, Bounds
    WHERE CalendarStart < EndDate
),
-- 2. Add Data
Setup AS (
    SELECT DATE_FORMAT(c.CalendarStart, '%Y-%m') AS Date,
        SUM(SaleIncome) AS TotalSales,
        ProductName,
        c.CalendarStart
    FROM Calendar c
    LEFT JOIN Sales s ON s.OrderDate >= c.CalendarStart
    AND s.OrderDate < DATE_ADD(c.CalendarStart, INTERVAL 1 MONTH)
    GROUP BY c.CalendarStart,ProductName
),
-- 3. Measure sales 3 months ago
Lags AS (
    SELECT Date,
        TotalSales,
        ProductName,
        LAG(TotalSales,3) OVER (PARTITION BY ProductName ORDER BY Date) AS 3MonthsAgo,
        CalendarStart
    FROM Setup
),

    -- Make a percentage increase/decrease between this month and 3 months ago
    Deltas AS (
        SELECT Date,
            TotalSales,
            ProductName,
            ((TotalSales - 3MonthsAgo) / 3MonthsAgo) * 100 AS 3MonthMomentum,
            CalendarStart
        FROM Lags
    ),
    -- To measure latest month
    MaxMonth AS (
        SELECT DATE_FORMAT(MAX(OrderDate), '%Y-%m-01') AS LatestMonthStart
        FROM Sales
    )
    -- Final Query
    SELECT Date,
        ProductName,
        TotalSales,
        CONCAT(ROUND(3MonthMomentum,2),'%') AS 3MonthMomentum
    FROM Deltas d
    JOIN MaxMonth m ON d.CalendarStart = m.LatestMonthStart
    ORDER BY 3MonthMomentum DESC;
```

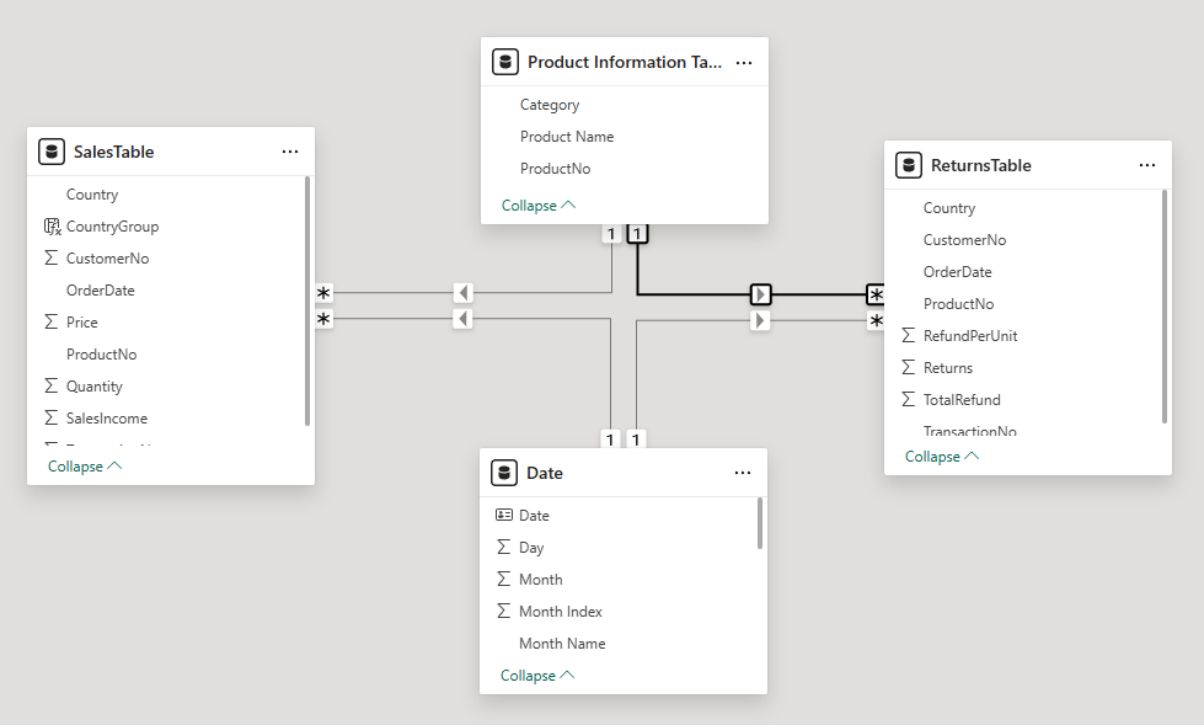
The last step to undertake in MySQL was to export our transformed data. We decided to export three tables: A product table, sales table and returns table. As the table size was again large, we used *INTO OUTFILE* to export the data.

Power BI Data Preparation

We imported our 3 MySQL queries into 3 separate tables in our PowerBI project. We adjusted data types and formatting using Power Query to ensure our data was presented correctly.

At this stage we created a *date table* using *AUTOCALENDAR()*. In this table we included *quarters* to allow for easy *QoQ Growth* measure creation.

Following this, we created relationships between our tables in *Model View* to create our semantic model. We treated the *Product Information Table* as our dimension table, with both the *SalesTable* and *ReturnsTable* as fact tables. The *date table* also connected to both fact tables to allow for data intelligence functions to be completed. All relationships were *many to one, active* relationships.



We also created another table called *Measures Table* to store and organise measures. A table named *Growth Rate Adjuster* was also created for use on a specific slicer in the report.

Report creation

In order to create an efficient report that can be utilised by different teams within the company effectively, we decided to create a number of pages in our report for

specific purposes which can be distributed as deemed appropriate. The pages to be designed were:

- A high-level overview of our dataset. This will give the consumer a clear picture of how the company is currently performing, as well as insight into ongoing performance.
- A product-focused page. This will allow consumers to delve deeper into the data, allowing them to see which individual products are performing best. They will also be able to look at products belonging only to specific categories.
- A country-focused page. This will give consumers the ability to analyse geographical sales data to help them make informed decisions.
- A *what-if* scenario generator page. This page will give consumers the ability to adjust growth rates and product lines to see how future change will affect the business.
- All pages have a nav-bar on the left-hand side to allow them to navigate through the report.
- No page interacts with other pages in terms of selection and filters for simplicity.

Overview Page

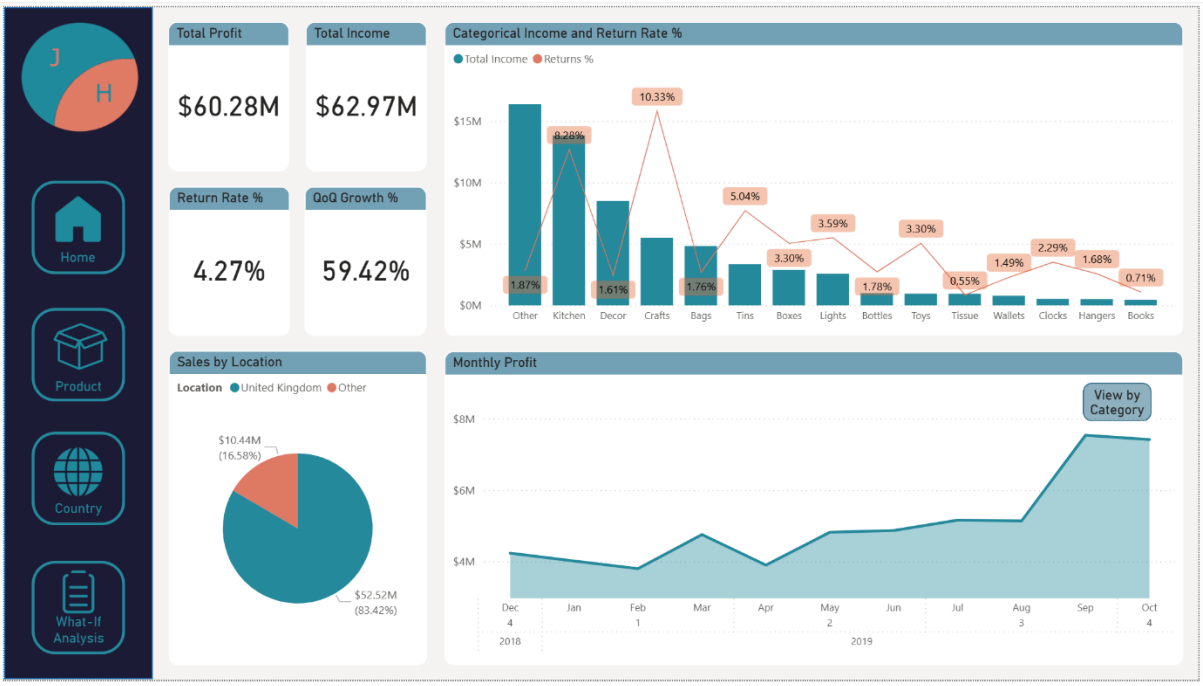
The overview page consisted of 4 visuals: *KPI* cards, line and column chart, pie chart, and a swappable column to line chart.

The cards provide key statistics on the business' income, return rate, and quarterly growth. These are not affected by filters and selection data.

The line and column chart provide data on the income and return rates for each category. The columns can be selected to view categorical data in the pie chart and line chart.

The pie chart showcases sales in two groups: *United Kingdom* and *Other*. Due to sales in the UK making up over 83% of total sales, it made sense to showcase this data in the overview like this for quick and easy understanding. All charts are affected by selecting the segments of this chart.

The swappable column and line charts can be viewed interchangeably using a small button in the top right of the chart. The column chart shows the QoQ Growth % for each individual category. The line chart shows profit over time for the business. You can highlight data in the other charts by selecting either a time point on the line chart or categorical data in the pie chart by selecting a column.



Product Page

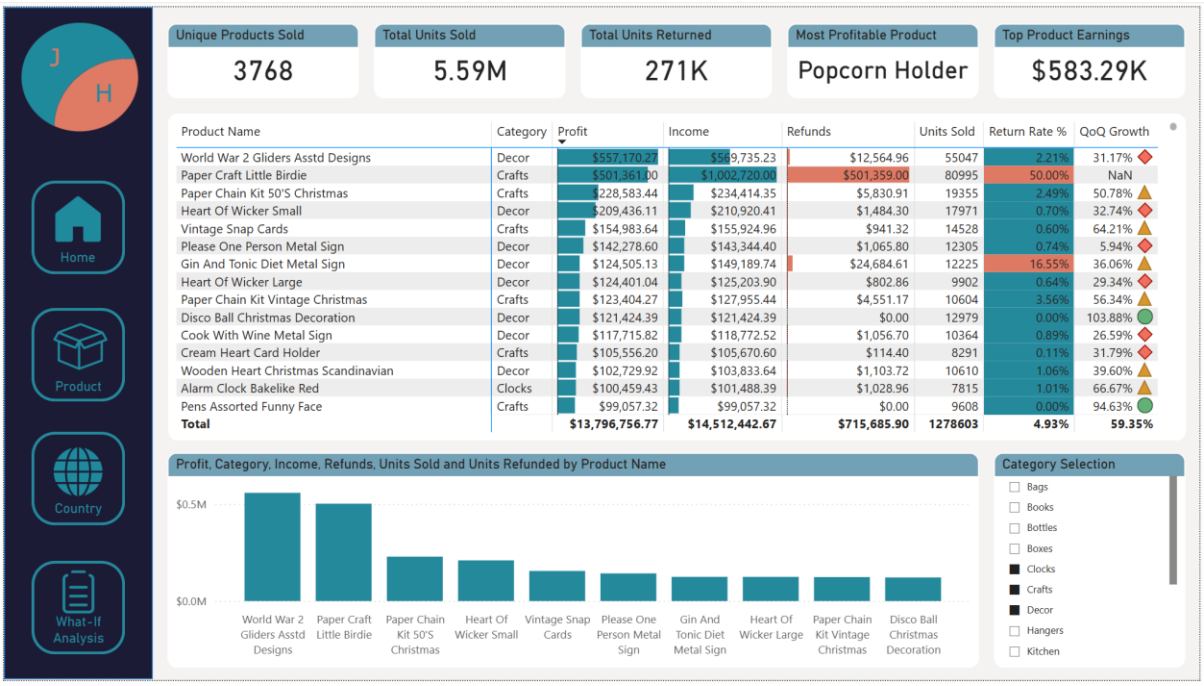
The product page consists of a selection of *KPI* cards, a matrix, column chart and category selection slicer.

The cards provide information on the products sold. These are used to highlight hero products as well as give an overview of business-wide product-related sales data. These are not affected by filters and selection data.

The matrix takes up the majority of the page and shows product name, category, profit, income, refund loss, units sold, return rate % and QoQ Growth. This allows consumers to get deeper insight into individual products. The chart is colour-coded to allow for quick and easy understanding of how that product is performing.

The column chart highlights the top 10 products. It stores information on these products which can be shown in the tooltip that appears when hovering over that specific product.

The category selection slicer allows you to filter the matrix and column chart by one or more categories. This can be useful for more in-depth categorical analysis.



Country Page

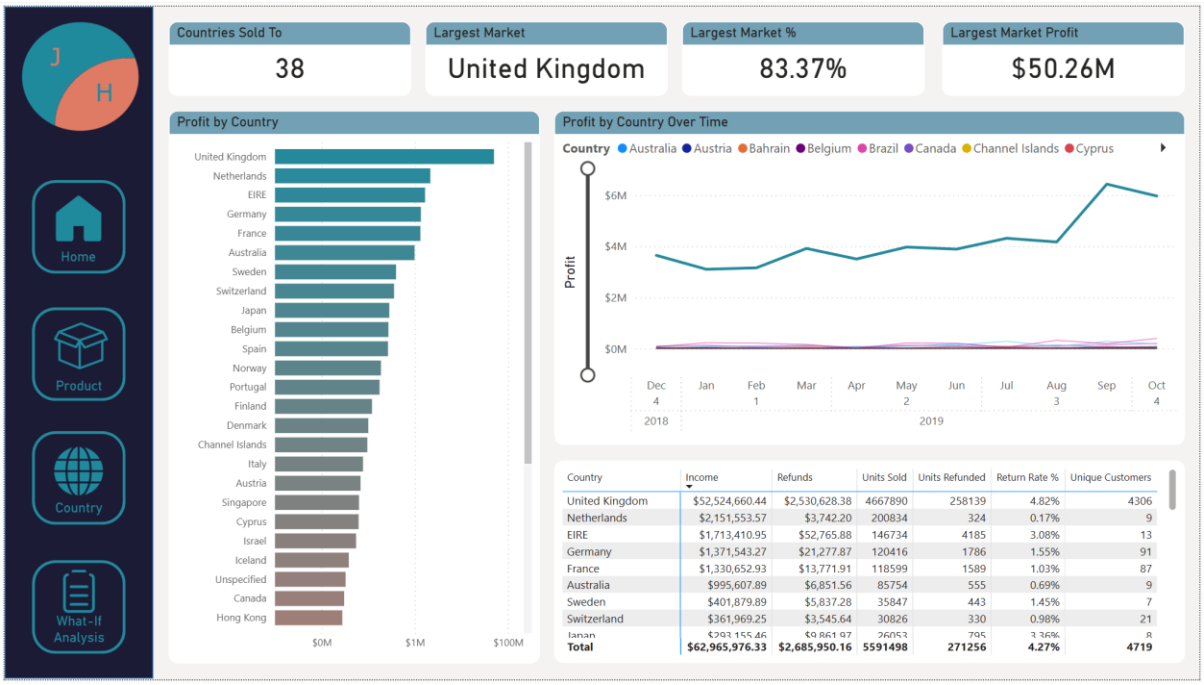
The country page consists of a selection of *KPI* cards, bar chart, line chart and a matrix.

The cards highlight geographical related data. These mostly relate to the top market for the business.

The bar chart highlights the level of profit for each country in the sales data. It is colour-coded to highlight how well the country is performing. It has a logarithmic x axis representing profit to allow for easier comparison of countries with lower incomes. This is due to the extreme difference in profit levels between the top and bottom countries which renders the chart almost useless for comparison with a standard axis scale. Selecting bars here allows you to filter the other visuals on the page.

The line chart showcases individual countries' profit over time. Each country has its own line. The scale on this chart is not logarithmic but features a bar to allow the consumer to scale the area of chart they are earning. This allows them to compare countries effectively but still highlights the considerable difference between the top earning country and others.

The matrix allows the consumer to see statistics on units sold and refunded, profit, and the number of unique customers in that region.



Country	Income	Refunds	Units Sold	Units Refunded	Return Rate %	Unique Customers
United Kingdom	$52,524,660.44	$2,530,628.38	4667890	258139	4.82%	4306
Netherlands	$2,151,553.57	$3,742.20	200834	324	0.17%	9
EIRE	$1,713,410.95	$52,765.88	146734	4185	3.08%	13
Germany	$1,371,543.27	$21,277.87	120416	1786	1.55%	91
France	$1,330,652.93	$13,771.91	118599	1589	1.03%	87
Australia	$995,607.89	$6,851.56	85754	555	0.69%	9
Sweden	$401,879.89	$5,837.28	35847	443	1.45%	7
Switzerland	$361,969.25	$3,545.64	30826	330	0.98%	21
Total	$62,965,976.33	$2,685,950.16	5591498	271256	4.27%	4719

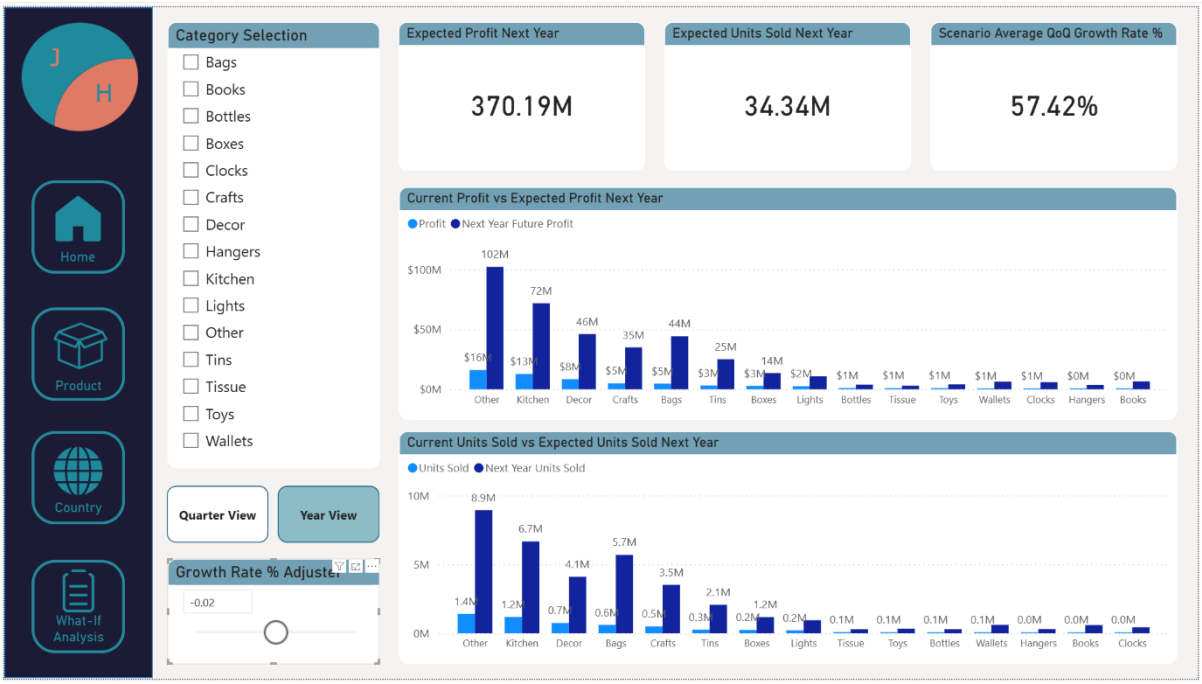
Scenario Page

The scenario page consists of a selection of slicers and buttons, several *KPI* cards and two column charts.

The slicers and buttons are the tools that allow the consumer to analyse the company's performance based on their selection. There is a category selection slicer, buttons to swap between quarterly and yearly analysis, and a *growth rate % adjuster* slicer. The final slicer applies a % change to the number of units sold and refunded to simulate a hypothetical scenario where the demand for products increases or decreases.

The cards on this page *do change* contrary to the other pages. This is to provide as much data as possible relating to this scenario. These cards highlight profit, units sold, and the average *QOQ Growth Rate %* for all products in the selection.

The two column charts compare profit and units sold from the current financial year in comparison to the forecasted levels.



Conclusion

Overall, I would consider this project successful at achieving the goals set out at the beginning of this project. We have taken our initial dataset, cleansed and prepared it, and created a report allowing the business to make data-driven decisions on the questions in our initial scenario.