Retail Sales Analytics Using Excel, SQL, Powe BI and Python

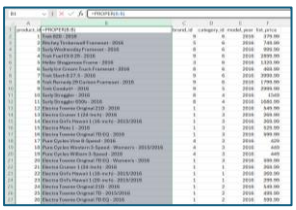
CAPSTONE PROJECT

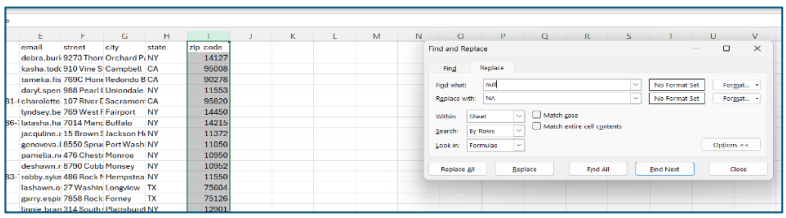
Problem Statement:
In a competitive retail market, businesses need to track sales performance, understand customer behaviour, and identify growth opportunities. This project aims to build an end-to-end analytics solution that enables insights generation from raw transactional data using a multi-tool approach.

Retail company: Gear up

Phase 1: Excel – Data Cleaning & Preparation

1. Standardize Column Headers
   Ensure all columns across sheets (like customers, orders, products) are consistently named.
   - Used Proper function – standardised the Column headers.
2. Remove Duplicates
   Identify and remove any duplicate records.
   - Data Tab – Remove duplicates
3. Handle Missing Values
   Use filtering and formulas to find blanks/nulls.
   Fill missing ZIP codes or phone numbers with placeholders or “NA”.
   - For null values used ctrl + H > replaced null to “NA”
4. Data Type Conversion
   Convert order dates, shipped dates, and required dates into Excel Date format.
   Ensure numeric fields (quantity, price) are set to number format.
   - Home tab > Number Formatting

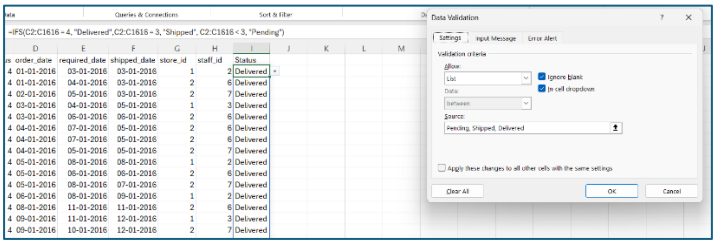




5. Data Validation

   Use dropdowns for fields like order_status (Pending, Shipped, Delivered) to ensure consistency.

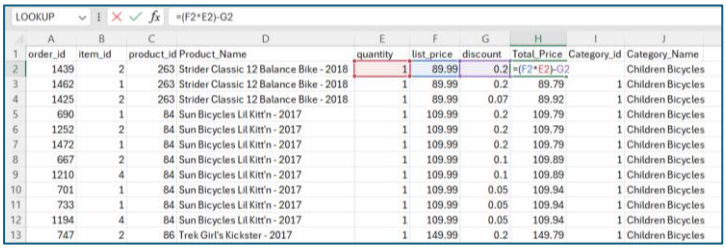
- New column “Status” > Data Tab > Data Validation > Lists



6. Create New Derived Columns

   Add a column in order_items to calculate total_price.

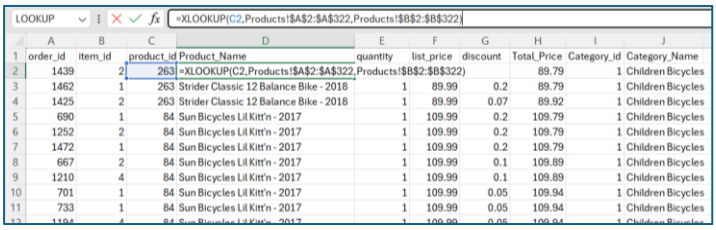
- Formula – (List_price * quantity) – Discount



7. Merge Lookup Data

   Use VLOOKUP or XLOOKUP to merge product names into order_items using product_id.

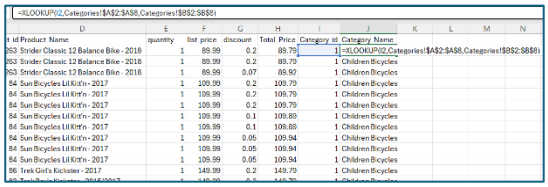
- Xlookup function - =Xlookup(product_id, product_id range, product_name range)

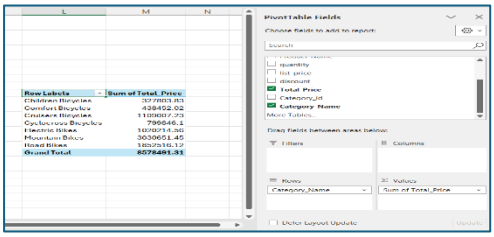


8. Create Basic Pivot Table

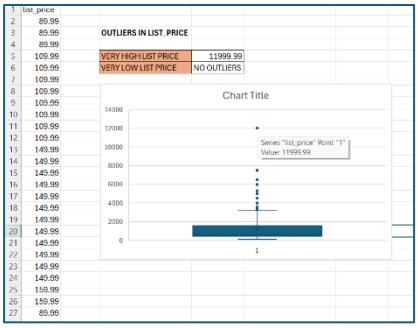
   Build a pivot table summarizing total sales by product category.

- Using Xlookup – inserted Category_id and Category_name in workbook.
- Insert > Pivot Table > Sales in values & Category_name in rows





9. Sort and Filter for Outliers
   Find outliers in pricing
   - Select column >Insert> Recommended charts > Boxplot chart

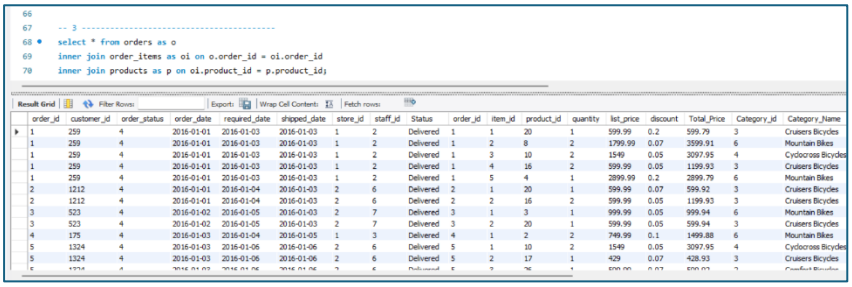


10. Prepare Final CSVs
    Save cleaned sheets as CSVs named: customers.csv, orders.csv, etc., for SQL import.

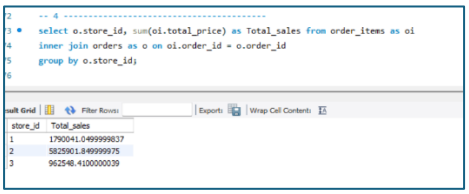
Phase 2: SQL – Database Management and Querying

1. Create Tables Based on ERD
   Use CREATE TABLE statements to replicate the exact structure of the ER diagram (with constraints).

2. Import CSVs into SQL
   Load cleaned Excel files using LOAD DATA or MySQL Workbench import feature.
   - Database created RETAIL_SALES > Tables > Import Wizard > Clean dataset loaded > Alter table > changed data types and added primary key constraint
3. Inner Join for Order Details
   Join orders, order_items, and products to display detailed line items.



4. Total Sales by Store
   Write a query to group sales (total_price) by each store_id.



5. Top 5 Selling Products
   Use ORDER BY and LIMIT to get the top 5 most sold products by quantity.

```sql
-- 5 ----------------------------------------
select p.product_name,sum(oi.quantity) as sum_qty from order_items as oi
inner join products as p on oi.product_id = p.product_id
group by p.product_name
order by sum_qty desc
limit 5;
```

product_name	sum_qty
Electra Cruiser 1 (24-Inch) - 2016	296
Electra Townie Original 7D EQ - 2016	290
Electra Townie Original 21D - 2016	289
Electra Girl's Hawaii 1 (16-inch) - 2015/2016	269
Surly Ice Cream Truck Frameset - 2016	167

6. Customer Purchase Summary

   For each customer, return total orders placed, total items purchased, and total revenue.

```sql
-- 6 ----------------------------------------
select o.customer_id, count(oi.order_id) as total_orders, count(oi.item_id) as total_items, sum(oi.total_price) as total_revenue from order_items as oi
inner join orders as o on oi.order_id = o.order_id
group by o.customer_id;
```

customer_id	total_orders	total_items	total_revenue
1	11	11	30644.78
2	10	10	21652.87
3	13	13	26248.659999999993
4	9	9	24197.809999999998
5	8	8	19442.01
6	11	11	35856.829999999994
7	4	4	7707.64
8	3	3	2603.58
9	9	9	26678.97
10	11	11	37800.42
11	5	5	4079.49
12	8	8	27157.21
13	8	8	15830.32

7. Segment Customers by Total Spend

   Write a query to classify customers into spending brackets (e.g., low, medium, high).

```sql
-- 7 ----------------------------------------
select o.customer_id, c.first_name, sum(oi.total_price) as total_spending,
case
    when sum(oi.total_price) >= 15000 then "High"
    when sum(oi.total_price) between 8000 and 15000 then "Medium"
    else "Low"
end as Spending_brackets
from order_items as oi
inner join orders as o on oi.order_id = o.order_id
inner join customers as c on o.customer_id = c.customer_id
group by o.customer_id, c.first_name;
```

customer_id	first_name	total_spending	Spending_brackets
1	Debra	30644.78	High
2	Kasha	21652.87	High
3	Tameka	26248.659999999993	High
4	Daryl	24197.809999999998	High
5	Charolette	19442.01	High
6	Lyndsey	35856.829999999994	High
7	Latasha	7707.64	Low
8	Jacquline	2603.58	Low
9	Genoveva	26678.97	High
10	Pamela	37800.42	High
11	Deshawn	4079.49	Low
12	Robby	27157.21	High
13	Lashawn	15830.32	High

8. Staff Performance Analysis

   Analyze total revenue generated by each staff member based on their handled orders.

```sql
-- 8 ----------------------------------------
select o.staff_id, s.first_name, count(oi.order_id) as handled_orders, sum(oi.total_price) as total_revenue from order_items as oi
inner join orders as o on oi.order_id = o.order_id
inner join staffs as s on o.staff_id = s.staff_id
group by o.staff_id, s.first_name;
```

staff_id	first_name	handled_orders	total_revenue
2	Mireya	462	837375.7800000024
3	Genna	544	952665.2700000049
6	Marcelene	1615	2938714.119999979
7	Venita	1580	2887187.7299999804
8	Kali	269	516667.66000000003
9	Layla	252	445880.7500000001

9. Stock Alert Query

   Write a query to list products where stock quantity < 10 in any store.

```sql
-- 9 ----------------------------------------
select s.store_id, s.store_name, sum(sk.quantity) as total_quantity, p.product_name from stores as s
inner join stocks as sk on s.store_id = sk.store_id
inner join products as p on p.product_id = sk.product_id
group by s.store_id, s.store_name, p.product_name
having sum(sk.quantity) > 10;
```

store_id	store_name	total_quantity	product_name
1	Santa Cruz Bikes	27	Trek 820 - 2016
1	Santa Cruz Bikes	23	Trek Fuel EX 8 29 - 2016
1	Santa Cruz Bikes	22	Heller Shagamaw Frame - 2016
1	Santa Cruz Bikes	11	Trek Conduit+ - 2016
1	Santa Cruz Bikes	15	Surly Straggler - 2016
1	Santa Cruz Bikes	26	Electra Townie Original 21D - 2016
1	Santa Cruz Bikes	37	Electra Cruiser 1 (24-Inch) - 2016
1	Santa Cruz Bikes	27	Electra Girl's Hawaii 1 (16-inch) - 2015/2016
1	Santa Cruz Bikes	20	Electra Townie Original 7D EQ - 2016
1	Santa Cruz Bikes	16	Pure Cycles Western 3-Speed - Women's - 2015...
1	Santa Cruz Bikes	26	Electra Townie Original 7D EQ - Women's - 2016
1	Santa Cruz Bikes	21	Surly Big Dummy Frameset - 2017
1	Santa Cruz Bikes	20	Surly Karate Monkey 27.5+ Frameset - 2017

10. Create Final Segmentation Table

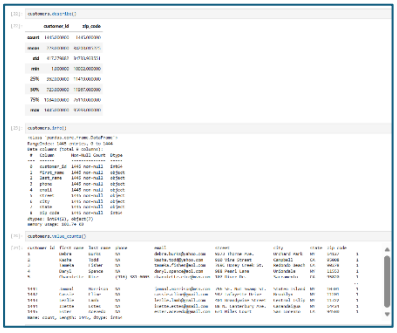
Create a table customer_segments that will be populated from Python ML results later.

Phase 3: Python

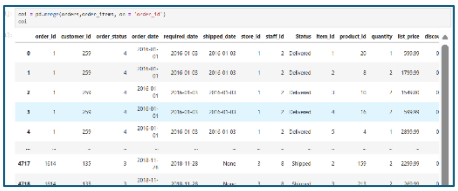
1. Load Data from SQL
   Use pandas.read_sql() to pull the orders, order_items, and customers tables into a DataFrame.

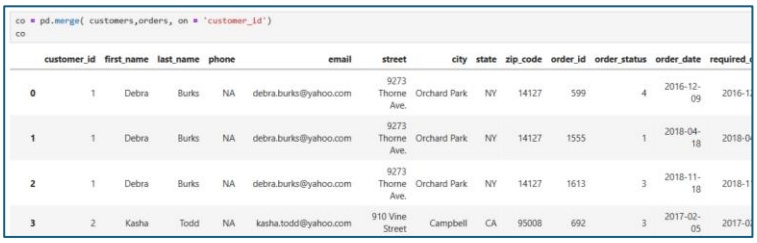
```
from sqlalchemy import create_engine

engine = create_engine("mysql+pymysql://root:12345@localhost:3306/retail_sales")

customers = pd.read_sql("SELECT * FROM customers;", engine)

order_items = pd.read_sql("SELECT * FROM order_items;", engine)

orders = pd.read_sql("SELECT * FROM orders;", engine)
```

2. Basic EDA (Exploratory Data Analysis)
   Use df.describe(), df.info(), and df.value_counts() to summarize the dataset.



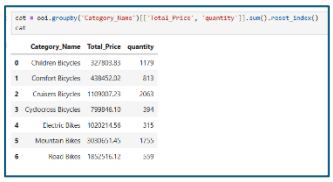
Merged the tables:



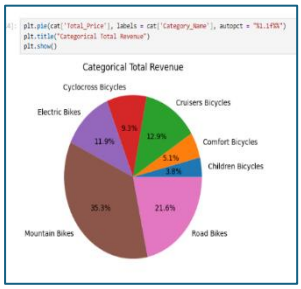


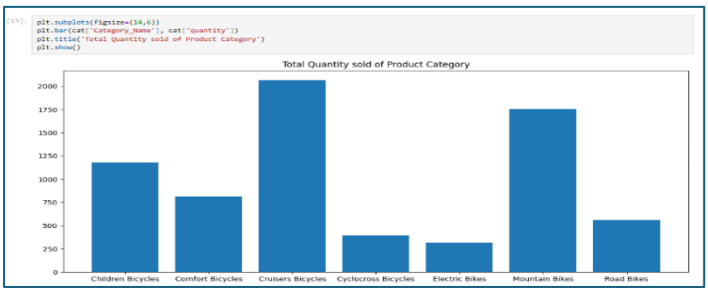
customer_id	first_name	last_name	phone	email	street	city	state	zip_code	order_id	order_status	order_date	required_
0	1	Debra	Burks	NA	debra.burks@yahoo.com	9273 Thorne Ave.	Orchard Park	NY	14127	599	4	2016-12-09	2016-1
1	1	Debra	Burks	NA	debra.burks@yahoo.com	9273 Thorne Ave.	Orchard Park	NY	14127	1555	1	2018-04-18	2018-0
2	1	Debra	Burks	NA	debra.burks@yahoo.com	9273 Thorne Ave.	Orchard Park	NY	14127	1613	3	2018-11-18	2018-1
3	2	Kasha	Todd	NA	kasha.todd@yahoo.com	910 Vine Street	Campbell	CA	95008	692	3	2017-02-05	2017-0

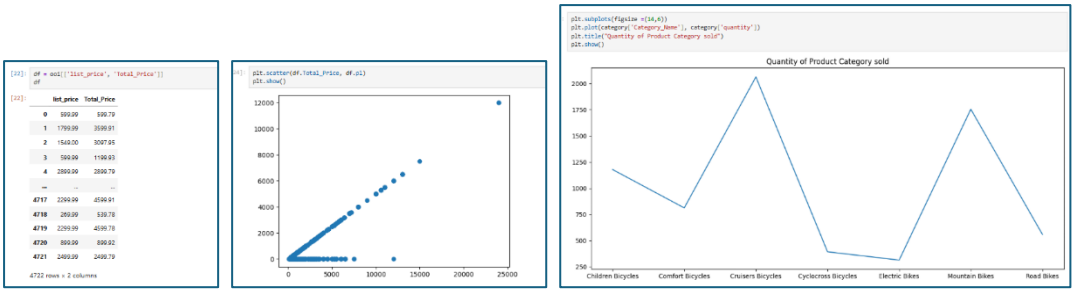
Plot basic charts using matplotlib or seaborn.



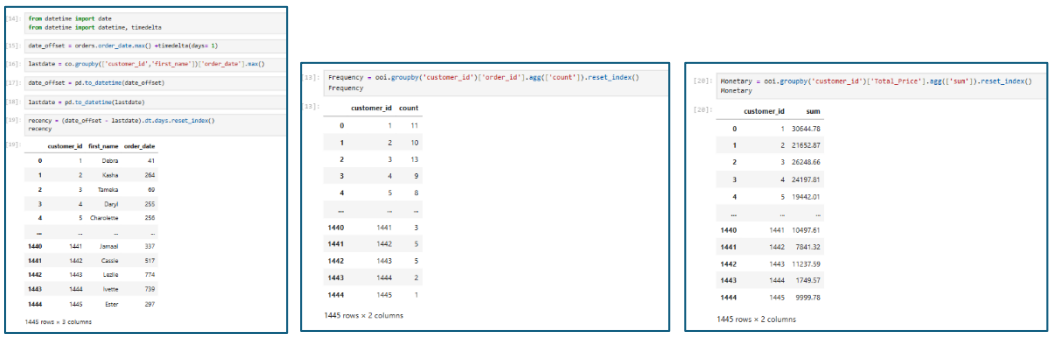
	Category_Name	Total_Price	quantity
0	Children Bicycles	327803.83	1179
1	Comfort Bicycles	438452.02	813
2	Cruisers Bicycles	1109007.23	2063
3	Cyclocross Bicycles	799916.12	394
4	Electric Bikes	1020214.58	315
5	Mountain Bikes	3083651.45	1755
6	Road Bikes	1852516.12	559



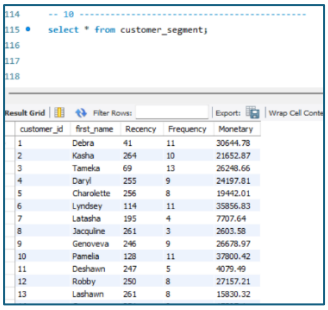




3. Calculate RFM Features for Customers
   Compute Recency, Frequency, and Monetary values for each customer.
   - Recency: Days since last order
   - Frequency: Number of orders
   - Monetary: Total value of all purchases



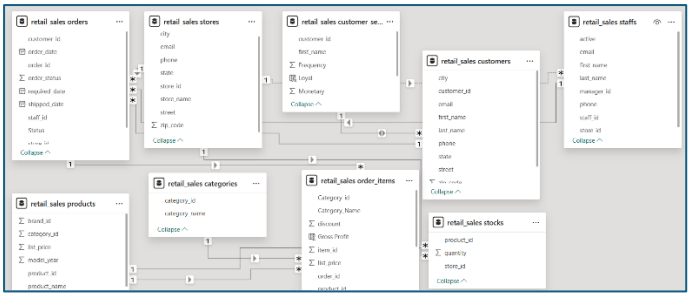
4. Export Segmentation Results to SQL
   Save the rfm_data with the segment label to SQL as customer_segments table.



customer_id	first_name	Recency	Frequency	Monetary
1	Debra	41	11	30644.78
2	Kasha	264	10	21652.87
3	Tameka	69	13	26248.66
4	Daryl	255	9	24197.81
5	Charolette	256	8	19442.01
6	Lyndsey	114	11	35856.83
7	Latasha	195	4	7707.64
8	Jacquline	261	3	2603.58
9	Genoveva	246	9	26678.97
10	Pamelia	128	11	37800.42
11	Deshawn	247	5	4079.49
12	Robby	250	8	27157.21
13	Lashawn	261	8	15830.32

Phase 4: Power BI – Visualization & Dashboarding Mandatory Tasks

1. Connect Power BI to SQL
   Import key tables like orders, products, order_items, customers, stores, and staffs from the SQL database.
   - Get data> Database> MySQL Database> Server name & database name> Navigator (selection of tables)> transform data
2. Create Relationships Between Tables
   Use Model View to define foreign key relationships based on the ER diagram (e.g., link orders.customer_id to customers.customer_id).



3. Sales Overview Report

   Create visuals for:

   Total sales over time (line or area chart)

   Monthly sales trend

   Total orders placed

4. Top Products by Sales

   Bar chart or table listing top-selling products by revenue or quantity.

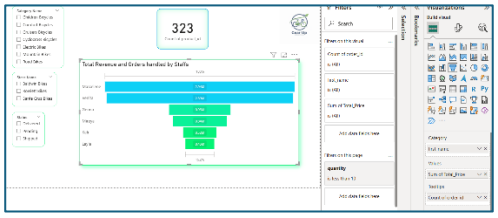
5. Customer Purchase Analysis

   Add a stacked bar chart to show purchase patterns by city or state using the customers and orders table.

6. Sales by Store Map

   Use the map visual with store.state or store.zip_code to show sales distribution geographically.

7. Low Stock Alert Dashboard
   Use conditional formatting and cards to display products with stock levels below a threshold (e.g., 10 units).
8. Interactive Filters and Slicers
   Add slicers for:
   - Order Status
   - Product Category
   - Store
9. Staff Performance Report
   Table or chart showing total sales/revenue per staff member based on handled orders.



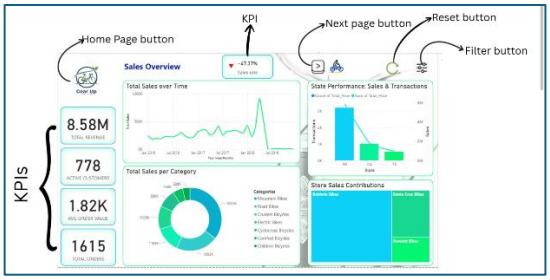
10. Consolidated Dashboard Page
    Final report page with KPIs:
    - Total Revenue
    - Active Customers
    - Avg Order Value
    - Total Orders

Buttons:

Sales Overview:

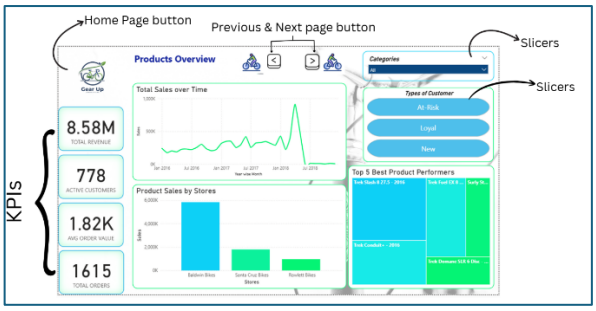
- Total sales Over time
- Total sales per category
- State performance: Sales & Transactions

- Store Sales Contribution



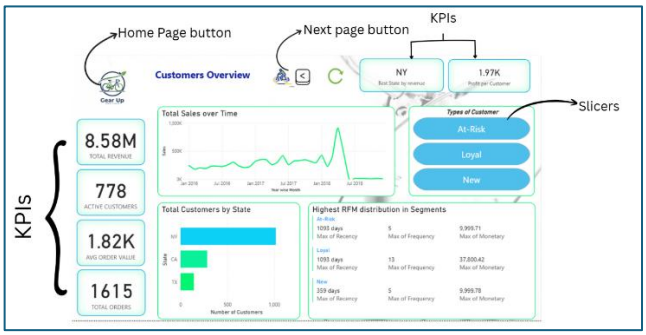
Products Overview:

- Total Sales over Time
- Product sales by stores
- Top 5 best product performers



Customers overview:

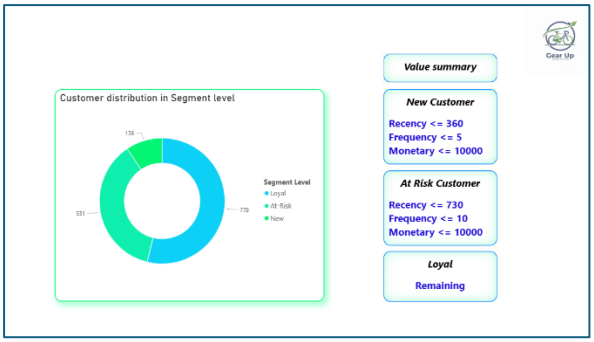
- Total sales over time
- Total customers by state
- Highest RFM distribution in segments



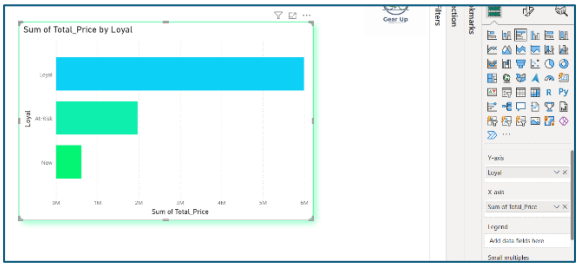
11. Import customer_segments Table
    Load the segmentation result (exported from Python) from SQL into Power BI.

➢ Get data> Database> MySQL Database> Server name & database name> Navigator (selection of tables)> transform data

12. Visualize Customer Segments
    Use Pie Chart / Donut / TreeMap to show the distribution of segments (e.g., Loyal, New, At-Risk).



13. Segment-Level Revenue Breakdown
    Add a bar chart that shows total revenue per customer segment.



14. Use Segments as Report Filters
    Enable filtering across dashboards by customer segment using slicers.

Overall insights:

- Sales was high during mid (April, June, September) of the calendar year, which is the summer season.

Q1	Jan, Feb, Mar	Low Sales
Q2	April, May, June	High Sales
Q3	July, Aug, Sept	High sales
Q4	Oct, Nov, Dec	Low Sales

- Start of the year and end of the year has low sales because of fall and winter seasons.
- Most sold product category is Mountain bikes: which clear mountain riders rides during summer.
- But least sales are of children category bicycles: in past three years they have bought during summer season.
- In the last one year (2018) more than 100 new customers have come which gave average profit of 1.27k.
- Since 2016 to 2018, loyal customers are higher (778) but at-risk customers are little closer to the loyal customers' count.