

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.

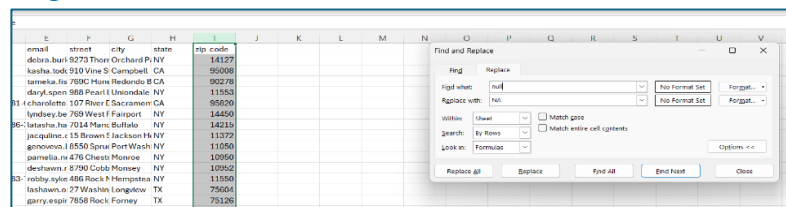
Retail company: Gear up

Phase 1: Excel – Data Cleaning & Preparation

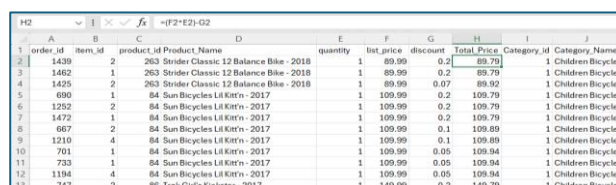
1. Standardize Column Headers
Ensure all columns across sheets (like customers, orders, products) are consistently named.
 - Used Proper function – standardised the Column headers.
2. Remove Duplicates
Identify and remove any duplicate records.
 - Data Tab – Remove duplicates
3. Handle Missing Values
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



order_id	item_id	product_id	product_name	quantity	list_price	discount	total_price	category_id	category_name
1	1430	2	263 Strider Classic 12 Balance Bike - 2018	1	89.99	0.2	89.79	1	Children Bicycles
3	1462	1	263 Strider Classic 12 Balance Bike - 2018	1	89.99	0.07	89.92	1	Children Bicycles
4	1425	2	263 Strider Classic 12 Balance Bike - 2018	1	89.99	0.07	89.92	1	Children Bicycles
5	690	1	84 Sun Bicycles Lil'Kite'n - 2017	1	109.99	0.2	109.79	1	Children Bicycles
6	1252	2	84 Sun Bicycles Lil'Kite'n - 2017	1	109.99	0.2	109.79	1	Children Bicycles
7	1472	1	84 Sun Bicycles Lil'Kite'n - 2017	1	109.99	0.2	109.79	1	Children Bicycles
8	667	2	84 Sun Bicycles Lil'Kite'n - 2017	1	109.99	0.1	109.89	1	Children Bicycles
9	1210	4	84 Sun Bicycles Lil'Kite'n - 2017	1	109.99	0.1	109.89	1	Children Bicycles
10	701	1	84 Sun Bicycles Lil'Kite'n - 2017	1	109.99	0.05	109.94	1	Children Bicycles
11	733	1	84 Sun Bicycles Lil'Kite'n - 2017	1	109.99	0.05	109.94	1	Children Bicycles
12	1194	4	84 Sun Bicycles Lil'Kite'n - 2017	1	109.99	0.05	109.94	1	Children Bicycles
13	747	2	86 Trek Girl's Kickster - 2017	1	149.99	0.2	149.79	1	Children Bicycles



email	street	city	state	zip_code	phone_number
debra.ban	5273 Thon Orchard Pl	NY		14127	
linda.hok	510 Vine St	CA		95008	
lameka.jia	769C Hone Redondo B	CA		90278	
daryl.spen	868 Pearl I	NY		11553	
l.chasleto	127 Shaw C	CA		95026	
lyndsey.be	769 West F	NY		14450	
latasha.jae	7014 Marc	NY		14215	
jac.gulme	113 Brown I	CA		11372	
gonnevo	16550 Spru	NY		11050	
panella.n	478 Chest	NY		10950	
dehanna	7790 Cold	NY		10952	
robby.syl	456 Rock P	NY		11550	
lashawn.o	27 Washn	TX		75604	
garry.wash	7658 Rock	TX		75126	
joan.juan	344 South	NY		12081	



order_id	item_id	product_id	product_name	quantity	list_price	discount	total_price	category_id	category_name
1	1430	2	263 Strider Classic 12 Balance Bike - 2018	1	89.99	0.2	89.79	1	Children Bicycles
3	1462	1	263 Strider Classic 12 Balance Bike - 2018	1	89.99	0.07	89.92	1	Children Bicycles
4	1425	2	263 Strider Classic 12 Balance Bike - 2018	1	89.99	0.07	89.92	1	Children Bicycles
5	690	1	84 Sun Bicycles Lil'Kite'n - 2017	1	109.99	0.2	109.79	1	Children Bicycles
6	1252	2	84 Sun Bicycles Lil'Kite'n - 2017	1	109.99	0.2	109.79	1	Children Bicycles
7	1472	1	84 Sun Bicycles Lil'Kite'n - 2017	1	109.99	0.2	109.79	1	Children Bicycles
8	667	2	84 Sun Bicycles Lil'Kite'n - 2017	1	109.99	0.1	109.89	1	Children Bicycles
9	1210	4	84 Sun Bicycles Lil'Kite'n - 2017	1	109.99	0.1	109.89	1	Children Bicycles
10	701	1	84 Sun Bicycles Lil'Kite'n - 2017	1	109.99	0.05	109.94	1	Children Bicycles
11	733	1	84 Sun Bicycles Lil'Kite'n - 2017	1	109.99	0.05	109.94	1	Children Bicycles
12	1194	4	84 Sun Bicycles Lil'Kite'n - 2017	1	109.99	0.05	109.94	1	Children Bicycles
13	747	2	86 Trek Girl's Kickster - 2017	1	149.99	0.2	149.79	1	Children Bicycles

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

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

The screenshot shows the 'Data Validation' dialog box in Microsoft Excel. The 'Settings' tab is active, displaying a list of validation rules for the range D1:M18. The first rule is selected, showing 'Data' as 'Text', 'Allow' as 'any blank', and 'Show custom message' checked. The 'Error Message' tab is also visible, showing a custom message: 'Feeding, Shipping, Delivered'. The 'OK' button is highlighted.

6. Create New Derived Columns

Add a column in order items to calculate total price.

➤ Formula – (List_price * quantity) – Discount

	A	B	C	D	E	F	G	H	I	J
	Order ID	Item ID	Product ID	Product Name	Quantity	List Price	Discount	Total Price	Category ID	Category Name
1	1439	2	263	Strider Classic 12 Balance Bike - 2018	=F2/E2-G2	89.99	0.21			Children Bicycles
2	1462	1	263	Strider Classic 12 Balance Bike - 2018		89.99	0.2			Children Bicycles
3	1462	2	263	Strider Classic 12 Balance Bike - 2018		89.99	0.07	89.92		Children Bicycles
4	690	1	84	Sun Bicycles Lil Kirtin - 2017		109.99	0.2	109.79		Children Bicycles
5	1252	2	84	Sun Bicycles Lil Kirtin - 2017		109.99	0.2	109.79		Children Bicycles
6	1472	1	84	Sun Bicycles Lil Kirtin - 2017		109.99	0.2	109.79		Children Bicycles
7	667	2	84	Sun Bicycles Lil Kirtin - 2017		109.99	0.1	109.89		Children Bicycles
8	1210	4	84	Sun Bicycles Lil Kirtin - 2017		109.99	0.1	109.89		Children Bicycles
9	701	1	84	Sun Bicycles Lil Kirtin - 2017		109.99	0.05	109.94		Children Bicycles
10	733	1	84	Sun Bicycles Lil Kirtin - 2017		109.99	0.05	109.94		Children Bicycles
11	1194	4	84	Sun Bicycles Lil Kirtin - 2017		109.99	0.05	109.94		Children Bicycles
12	747	2	86	Trek Gyr® Kickster - 2017		149.99	0.2	149.79		Children Bicycles

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)

LOOKUP		=XLOOKUP(C2,Products!\$A\$2:\$A\$32,Products!\$B\$2:\$B\$32)									
	A	B	D		E	F	G	H	I	J	
	order_id	item_id	product_id	Product Name	quantity	list price	discount	Total Price	Category_id	Category Name	
1	1439	2	263	=XLOOKUP(C2,Products!\$A\$2:\$A\$32,Products!\$B\$2:\$B\$32)				89.79	1	Children Bicycles	
2	1462	1	263	Strider Classic 12 Balance Bike - 2018	1	89.99	0.2	89.79	1	Children Bicycles	
4	1425	2	263	Strider Classic 12 Balance Bike - 2018	1	89.99	0.07	89.92	1	Children Bicycles	
5	690	1	84	Sun Bicycles Lil Kitten - 2017	1	109.99	0.2	109.79	1	Children Bicycles	
6	1252	2	84	Sun Bicycles Lil Kitten - 2017	1	109.99	0.2	109.79	1	Children Bicycles	
7	1472	1	84	Sun Bicycles Lil Kitten - 2017	1	109.99	0.2	109.79	1	Children Bicycles	
8	667	2	84	Sun Bicycles Lil Kitten - 2017	1	109.99	0.1	109.89	1	Children Bicycles	
9	1210	4	84	Sun Bicycles Lil Kitten - 2017	1	109.99	0.1	109.89	1	Children Bicycles	
10	701	1	84	Sun Bicycles Lil Kitten - 2017	1	109.99	0.05	109.94	1	Children Bicycles	
11	733	1	84	Sun Bicycles Lil Kitten - 2017	1	109.99	0.05	109.94	1	Children Bicycles	

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

C	D	E	F	G	H	I	J	K	L	M
product_id	Product Name	quantity	last price	discount	Total Price	Category id	Category Name			
263	Srider Classics 12 Balance Bike - 2018	1	89.99	0.2	89.79	-XLOOLQUPJ...Products\$B\$2-\$A\$327, Products\$D\$2-\$A\$327				
263	Srider Classics 12 Balance Bike - 2018	1	89.99	0.2	89.79	-XLOOLQUPJ...Products\$B\$2-\$A\$327, Products\$D\$2-\$A\$327				
262	Srider Classics 12 Balance Bike - 2018	1	89.99	0.07	89.92	1	Children Bicycles			
262	Srider Classics 12 Balance Bike - 2018	1	109.99	0.2	109.79	1	Children Bicycles			
84	Sun Bicycles L&M Inn - 2017	1	109.99	0.1	109.89	1	Children Bicycles			
84	Sun Bicycles L&M Inn - 2017	1	109.99	0.1	109.89	1	Children Bicycles			
84	Sun Bicycles L&M Inn - 2017	1	109.99	0.2	109.79	1	Children Bicycles			
84	Sun Bicycles L&M Inn - 2017	1	109.99	0.1	109.89	1	Children Bicycles			
84	Sun Bicycles L&M Inn - 2017	1	109.99	0.1	109.89	1	Children Bicycles			
84	Sun Bicycles L&M Inn - 2017	1	109.99	0.1	109.89	1	Children Bicycles			
84	Sun Bicycles L&M Inn - 2017	1	109.99	0.05	109.94	1	Children Bicycles			
84	Sun Bicycles L&M Inn - 2017	1	109.99	0.05	109.94	1	Children Bicycles			

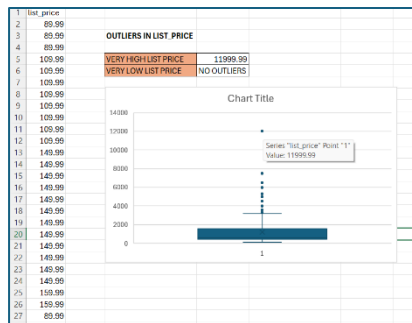
[XLOOKUP]_Categories (\$A\$2:\$A\$8;Categories (\$B\$2:\$B\$8))									
		G	H	I	J	K	L	M	N
Id of Product Name	quantity	price	discount	Total Price	Category	Category Name			
60 San Ruyes Classic II Balance Bio: 2018	1	89.99	0.2	89.79	1	[XLOOKUP]_Categories (\$A\$2:\$A\$8;Categories (\$B\$2:\$B\$8))			
60 San Ruyes Classic II Balance Bio: 2018	1	89.99	0.2	89.79	1	Children Bicycles			
60 San Ruyes Classic II Balance Bio: 2018	1	89.99	0.07	89.90	1	Children Bicycles			
60 San Ruyes Classic II Balance Bio: 2018	1	109.99	0.2	109.79	1	Children Bicycles			
60 San Ruyes Ld Korm: 2017	1	109.99	0.2	109.79	1	Children Bicycles			
60 San Ruyes Ld Korm: 2017	1	109.99	0.2	109.79	1	Children Bicycles			
60 San Ruyes Ld Korm: 2017	1	109.99	0.1	109.89	1	Children Bicycles			
60 San Ruyes Ld Korm: 2017	1	109.99	0.1	109.89	1	Children Bicycles			
60 San Ruyes Ld Korm: 2017	1	109.99	0.05	109.94	1	Children Bicycles			
60 San Ruyes Ld Korm: 2017	1	109.99	0.05	109.94	1	Children Bicycles			
60 San Ruyes Ld Korm: 2017	1	109.99	0.05	109.94	1	Children Bicycles			
60 San Ruyes Ld Korm: 2017	1	109.99	0.05	109.94	1	Children Bicycles			

[illegible]

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.

```

66 -- 3
67
68 * select * from orders as o
69 inner join order_items as oi on o.order_id = oi.order_id
70 inner join products as p on oi.product_id = p.product_id

```

order_id	customer_id	order_status	order_date	required_date	shipped_date	store_id	staff_id	Status	order_id	item_id	product_id	quantity	list_price	discount	Total Price	Category_id	Category Name
1	259	4	2016-01-01	2016-01-03	2016-01-03	1	2	Delivered	1	1	20	1	999.99	0.2	999.79	3	Cruisers Bicycles
1	259	4	2016-01-01	2016-01-03	2016-01-03	1	2	Delivered	1	2	8	2	1799.99	0.07	3599.93	6	Mountain Bikes
1	259	4	2016-01-01	2016-01-03	2016-01-03	1	2	Delivered	1	3	10	2	1549	0.05	3097.95	4	Cyclocross Bicycles
1	259	4	2016-01-01	2016-01-03	2016-01-03	1	2	Delivered	1	4	16	2	599.99	0.05	1199.93	3	Cruisers Bicycles
1	259	4	2016-01-01	2016-01-03	2016-01-03	1	2	Delivered	1	5	4	1	2899.99	0.2	2899.79	6	Mountain Bikes
2	1212	4	2016-01-01	2016-01-04	2016-01-03	2	6	Delivered	2	1	20	1	999.99	0.07	999.92	3	Cruisers Bicycles
2	1212	4	2016-01-01	2016-01-04	2016-01-03	2	6	Delivered	2	2	16	2	599.99	0.05	1199.93	3	Cruisers Bicycles
3	523	4	2016-01-02	2016-01-05	2016-01-03	2	7	Delivered	3	1	3	1	999.99	0.05	999.94	6	Mountain Bikes
3	523	4	2016-01-02	2016-01-05	2016-01-03	2	7	Delivered	3	2	20	1	599.99	0.05	599.94	3	Cruisers Bicycles
4	175	4	2016-01-03	2016-01-04	2016-01-05	1	3	Delivered	4	1	2	2	749.99	0.1	1499.88	6	Mountain Bikes
5	1324	4	2016-01-03	2016-01-06	2016-01-06	2	6	Delivered	5	1	10	2	1549	0.05	3097.95	4	Cyclocross Bicycles
5	1324	4	2016-01-03	2016-01-06	2016-01-06	2	6	Delivered	5	2	17	1	429	0.07	428.93	3	Cruisers Bicycles
6	1754	4	2016-01-03	2016-01-06	2016-01-06	2	6	Delivered	6	2	16	1	599.99	0.07	599.92	3	Cruisers Bicycles

4. Total Sales by Store

Write a query to group sales (total_price) by each store_id.

```

2 -- 4
3
4 * select o.store_id, sum(oi.total_price) as Total_sales from order_items as oi
5 inner join orders as o on oi.order_id = o.order_id
6 group by o.store_id
7

```

store_id	Total_sales
1	1790041.0499999837
2	5825901.849999975
3	962548.4100000039

5. Top 5 Selling Products

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

product_name	sum_qty
Electra Cruiser 1 (24-Inch) - 2016	296
Electra Townie Original 20 (Eq - 2016)	290
Electra Townie Original 2.0 (- 2016)	289
Electra Girls Hawk 1 (16-inch) - 2015/2016	269
Sunly Ice Cream Truck Frameset - 2016	167

6. Customer Purchase Summary

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

```

84 -- 6 .....
85 = select o.customer_id, count(o.order_id) as total_orders, count(i.item_id) as total_items, sum(i.total_price) as total_revenue from order_items as oi
86 inner join orders as o on oi.order_id = o.order_id
87 group by o.customer_id
88

```

Result Grid Filter Rows: Exports: Wrap Cell Contents: ☐ Fetch rows:

	customer_id	total_orders	total_items	total_revenue
1	11	11		30644.78
2	10	10		21652.87
3	13	13		26346.559999999993
4	9	9		24197.809999999998
5	8	8		19442.01
6	11	11		30556.829999999994
7	9	4		7767.64
8	3	3		2603.58
9	9	9		26678.97
10	11	11		37900.42
11	5	5		4979.49
12	8	8		27157.21
13	8	8		15830.32
14	9	9		23903.209999999996

7. Segment Customers by Total Spend

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

```

80 -- 7 -----
81 case
82 select o.customer_id, c.first_name, sum(o.total_price) as total_spending,
83        1
84        when sum(o.total_price) >= 15000 then "high"
85        when sum(o.total_price) between 8000 and 15000 then "medium"
86        else "Low"
87 end as Spending_brackets
88 from order_items as o
89 inner join orders as o on o.order_id = o.order_id
90 inner join customers as c on o.customer_id = c.customer_id
91 group by o.customer_id, c.first_name;
92
93
94
95
96
97
98
99
100

```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

Fetch rows:

	customer_id	first_name	total_spending	Spending_brackets
1	Delora	3064.78	High	
2	Kaela	21652.87	High	
3	Tamela	26248.899999999993	High	
4	Daryl	24137.899999999996	High	
5	Charlette	19442.01	High	
6	Lynette	30955.899999999994	High	
7	Lakana	7702.64	Low	
8	Jacqueline	2603.58	Low	
9	Gervina	26378.97	High	
10	Pemeka	37800.42	High	
11	Deaphani	4077.49	Low	
12	Rudby	27137.21	High	
13	Lapham	15830.32	High	

8. Staff Performance Analysis

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

```
-- 8
2 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
3 inner join orders as o on oi.order_id = o.order_id
4 inner join staffs as s on o.staff_id = s.staff_id
5 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	2938714.1199999979
7	Verita	1580	2887187.7299999804
8	Kali	269	516567.6600000003
9	Laila	715	446980.7200000001

9. Stock Alert Query

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

```

107 -->
108 * select s.store_id, s.store_name, sum(sk.quantity) as total_quantity, p.product_name from stores as s
109 inner join stocks as sk on s.store_id = sk.store_id
110 inner join products as p on p.product_id = sk.product_id
111 group by s.store_id, s.store_name, p.product_name
112 having sum(sk.quantity) > 10;

```

Result Grid Filter Rows: Export: Wrap Cell Contents:

store_id	store_name	total_quantity	product_name
1	Santa Cruz Bikes	27	Trek 620 - 2016
1	Santa Cruz Bikes	23	The Fuel EX 9.9 - 2016
1	Santa Cruz Bikes	22	heller Shogakukan frame - 2016
1	Santa Cruz Bikes	11	Trek Conquest - 2016
1	Santa Cruz Bikes	15	Santa Shogakukan - 2016
1	Santa Cruz Bikes	26	Electra Touring Original 2D - 2016
1	Santa Cruz Bikes	37	Electra Cruiser 1 (24-26d) - 2016
1	Santa Cruz Bikes	37	Electra Grit Henge 1 (16-mdb) - 2015/2016
1	Santa Cruz Bikes	20	Electra Touring Original 2D EC - 2016
1	Santa Cruz Bikes	16	Pure Cycles Wierem 3 Speed - Wierem's - 2015
1	Santa Cruz Bikes	20	Electra Touring Original 2D EC - Wierem's - 2016
1	Santa Cruz Bikes	21	Santa Cruz Phantom - 2017
1	Santa Cruz Bikes	20	Santa Kante Mariner 27.5 S - Framment - 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.

```
[1]: customers.describe()

customer_id  city_code
count  1400000.000000
mean  1000000.000000
std  1000000.000000
min  0.000000
max  1400000.000000

[2]: customers.info()

Out[2]: pandas.core.frame.DataFrame
RangeIndex: 1400000 entries, 0 to 1399999
Data columns (total 11 columns):
0  customer_id    int64  1400000
1  city_code      object  1400000
2  first_name     object  1400000
3  last_name      object  1400000
4  phone         object  1400000
5  email         object  1400000
6  street         object  1400000
7  city          object  1400000
8  state         object  1400000
9  zip_code      object  1400000
10 order_id      int64  1400000
dtypes: object(8), int64(3)
memory usage: 10.0 MB

[3]: customers.info(verbose=True)

Out[3]: customer_id    int64  1400000
city_code      object  1400000
first_name     object  1400000
last_name      object  1400000
phone         object  1400000
email         object  1400000
street         object  1400000
city          object  1400000
state         object  1400000
zip_code      object  1400000
order_id      int64  1400000
dtypes: object(8), int64(3)
memory usage: 10.0 MB
```

Merged the tables:

```
out = pd.merge(customers,orders, on = 'customer_id')
out

Out[1]: customer_id  order_status  order_date  required_date  shipped_date  store_id  staff_id  Status  item_id  product_id  quantity  list_price  discount
0  1  220  4  2016-01-01  2016-01-01  2016-01-01  1  2  Delivered  1  20  1  280.00  0
1  1  220  4  2016-01-01  2016-01-01  2016-01-01  1  2  Delivered  2  8  2  1790.00  0
2  1  220  4  2016-01-01  2016-01-01  2016-01-01  1  2  Delivered  3  10  7  1400.00  0
3  1  220  4  2016-01-01  2016-01-01  2016-01-01  1  2  Delivered  4  16  7  1400.00  0
4  1  220  4  2016-01-01  2016-01-01  2016-01-01  1  2  Delivered  5  4  1  280.00  0
...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
4717 1034 103 3  2018-11-18  2018-11-18  2018-11-18  3  8  Shipped  2  139  2  220.00  0
4718 1034 103 3  2018-11-18  2018-11-18  2018-11-18  3  8  Shipped  2  139  2  220.00  0
```

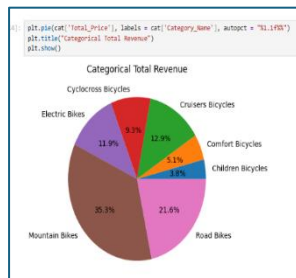
```
co = pd.merge(customers,orders, on = 'customer_id')
co

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

Plot basic charts using matplotlib or seaborn.

```
cat = pd.groupby('Category_Name')[['Total_Price', 'quantity']].sum().reset_index()
cat

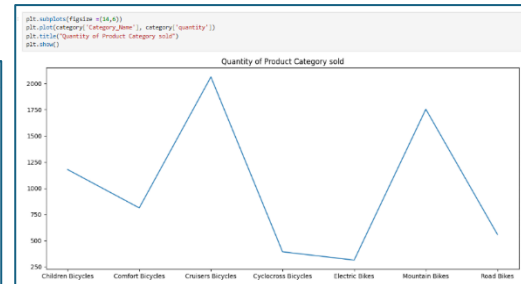
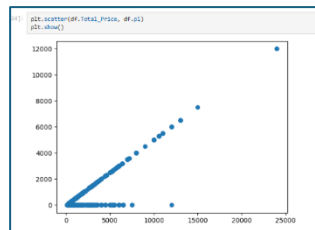
Out[1]: Category_Name  Total_Price  quantity
0  Children Bicycles  327810.583  1179
1  Comfort Bicycles  438620.02  813
2  Cruisers Bicycles  1100007.23  2063
3  Cyclocross Bicycles  799016.10  394
4  Electric Bikes  1020214.56  315
5  Mountain Bikes  3030610.45  1755
6  Road Bikes  1805516.12  559
```



```
df = df[['last_price', 'total_price']]
df
```

	last_price	Total_Price
0	500.00	500.79
1	1799.00	3099.01
2	1549.00	3007.05
3	500.00	1100.03
4	3800.00	2800.79
...
4717	2200.00	4590.01
4718	299.00	530.78
4719	2200.00	4590.78
4720	500.00	500.02
4721	3499.00	2499.79

4722 rows x 2 columns



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

```
from datetime import date
from datetime import datetime
data_offset = orders.order_date.max() - datetime.utcnow()
lastdate = pd.to_datetime(lastdate)
data_offset = pd.to_datetime(data_offset)
lastdate = pd.to_datetime(lastdate)
recency = (data_offset - lastdate).dt.days.reset_index()
recency
```

customer_id	first_name	order_date
0	Debra	41
1	Kasha	264
2	Tamela	69
3	Daryl	255
4	Charolette	256
...
1440	Lyndey	114
1441	Lataha	195
1442	Jacqueline	261
1443	Genevieve	246
1444	Pamela	128
1445	Deborah	247
1446	Robby	250
1447	Lashawn	251

1445 rows x 2 columns

```
Frequency = orders.groupby('customer_id')['order_id'].agg('count').reset_index()
Frequency
```

customer_id	count
0	11
1	10
2	13
3	9
4	8
...	...
1440	3
1441	5
1442	5
1443	2
1444	1

1445 rows x 2 columns

```
Monetary = orders.groupby('customer_id')['total_price'].agg('sum').reset_index()
Monetary
```

customer_id	sum
0	30644.78
1	21652.87
2	26240.66
3	24197.81
4	19442.01
...	...
1440	10497.61
1441	7841.32
1442	11237.39
1443	1748.57
1444	9999.78

1445 rows x 2 columns

4. Export Segmentation Results to SQL

Save the rfm_data with the segment label to SQL as customer_segments table.

```
-- 10 -----
select * from customer_segment;
```

customer_id	first_name	Recency	Frequency	Monetary
1	Debra	41	11	30644.78
2	Kasha	264	10	21652.87
3	Tamela	69	13	26240.66
4	Daryl	255	9	24197.81
5	Charolette	256	8	19442.01
6	Lyndey	114	11	10497.61
7	Lataha	195	4	7707.64
8	Jacqueline	261	5	2605.58
9	Genevieve	246	9	26678.97
10	Pamela	128	11	37800.42
11	Deborah	247	5	4079.49
12	Robby	250	8	27157.21
13	Lashawn	251	8	13830.32

Phase 4: Power BI – Visualization & Dashboarding Mandatory Tasks

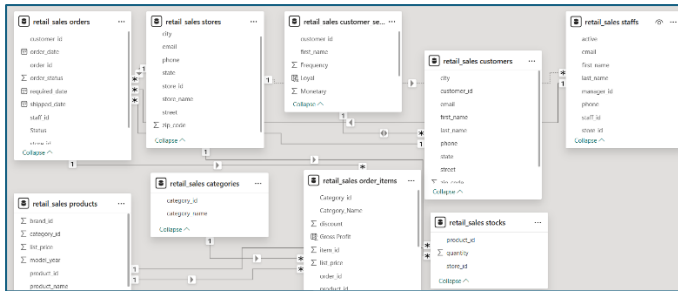
1. Connect Power BI to SQL

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).



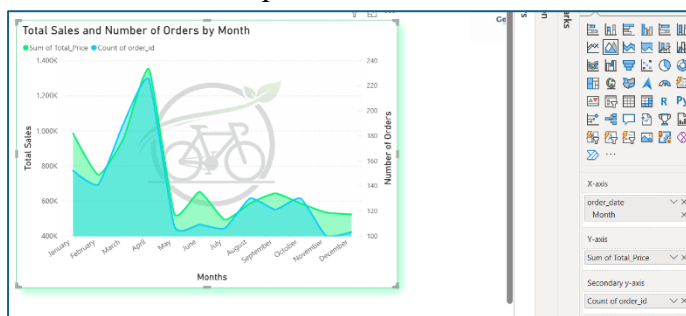
3. Sales Overview Report

Create visuals for:

Total sales over time (line or area chart)

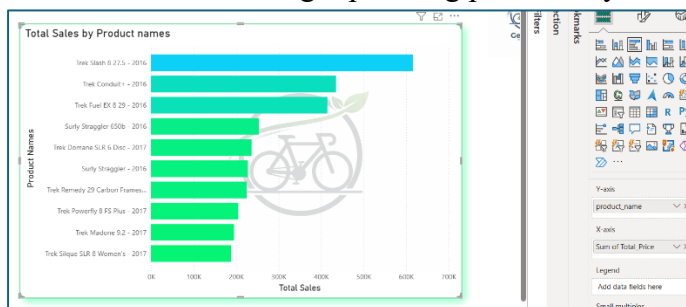
Monthly sales trend

Total orders placed



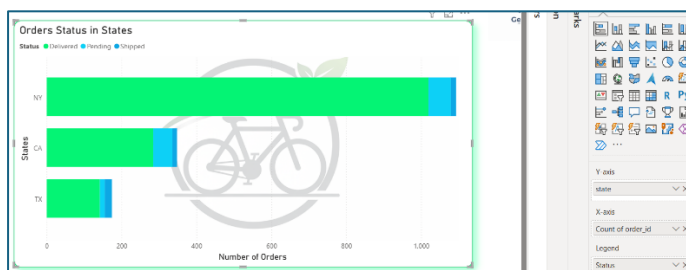
4. Top Products by Sales

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:

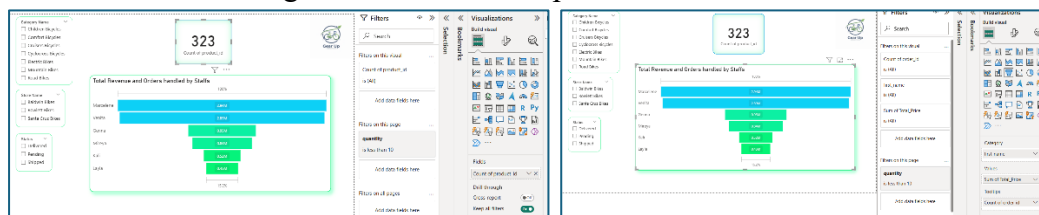
Order Status

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



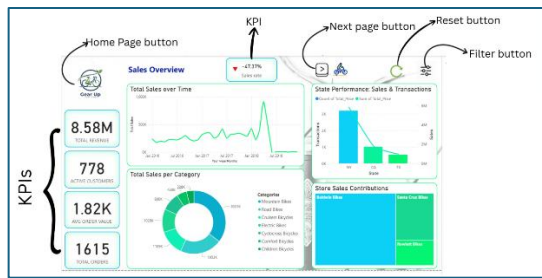
Buttons:



Sales Overview:

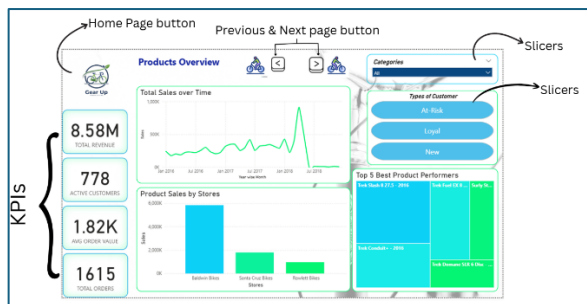
- Total sales Over time
- Total sales per category
- State performance: Sales & Transactions

- Store Sales Contribution



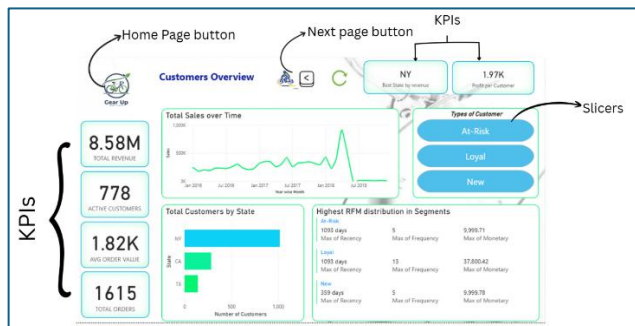
Products Overview:

- 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



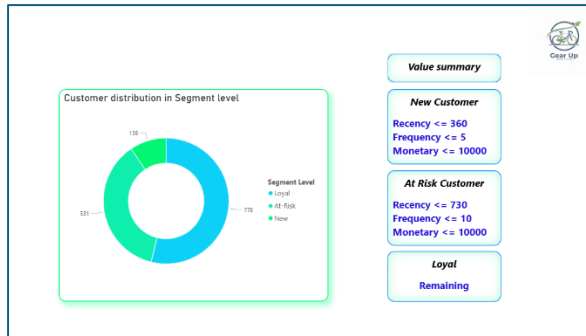
11. Import customer_segments Table

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

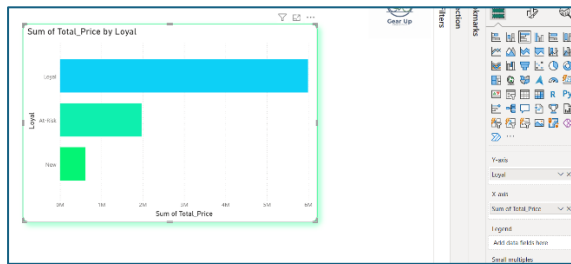
12. Visualize Customer Segments

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.