Bicycle Stores Sales Data

PRESENTED BY HIMA BINDU

Today's Topics

OUR DISCUSSION FLOW

Introduction to Data set
Data cleaning using MYSQL
Querying data using MYSQL
Analysing the data through POWERBI
Gaining insights through the analysed dataset.

OVERVIEW

The bicycle stores database is designed to manage key business information for a bike retail operation, including customers, products, orders, staff, and inventory. It captures customer details and allows tracking of orders with linked order items, each associated with specific products. Products are categorized by brand and type, offering insights into inventory and brand popularity. Staff and stores information is integrated, showing where and by whom orders are processed. The database also maintains stock records per store, ensuring inventory levels can be monitored across locations. This comprehensive setup supports efficient management of sales, inventory, and customer interactions in a bike retail environment.

Data Cleaning

REMOVING DUPLICATES

Checking each table for duplicate entries, especially in primary keys like customer_id, order_id, product_id

HANDLING MISSING VALUES

Identify missing values by querying columns for NULL values:

STANDARDIZING FORMATS

Ensure data consistency in columns like dates, phone numbers, and names.

Data Analysis Using MYSQL

- AVERAGE ORDER VALUE BY STORES
- BEST SELLING PRODUCTS
- CUSTOMER PURCHASE PATTERNS
- MONTHLY SALES TRENDS
- PRODUCT CATEGORY AND SALES CONTRIBUTION
- SALES PERFORMANCE
- SALES BY STAFF AND STORE
- STAFF RANKING BY STORE
- TOP SPENT CUSTOMER

Average order value by store

```
CREATE TABLE bike.AvgOrderValue_ByStore
WITH store_sales AS (
SELECT
o.store_id,
o.order_id,
SUM(oi.quantity * oi.list_price) AS order_value
FROM bike.orders o
JOIN bike.order_items oi ON o.order_id = oi.order_id
GROUP BY o.store_id, o.order_id
)
SELECT
store_id,
AVG(order_value) AS avg_order_value
FROM store_sales
GROUP BY store_id;
```

RESULT TABLE

Result Grid | Filter Rows:

store_id	avg_order_value
1	5144.09744252871 5
2	5330.505224153743
3	5532.188275862061

Visualisation

Avg order value of Store

store_id	Avg order value
3	5,532.19
2	5,330.51
1	5,144.10

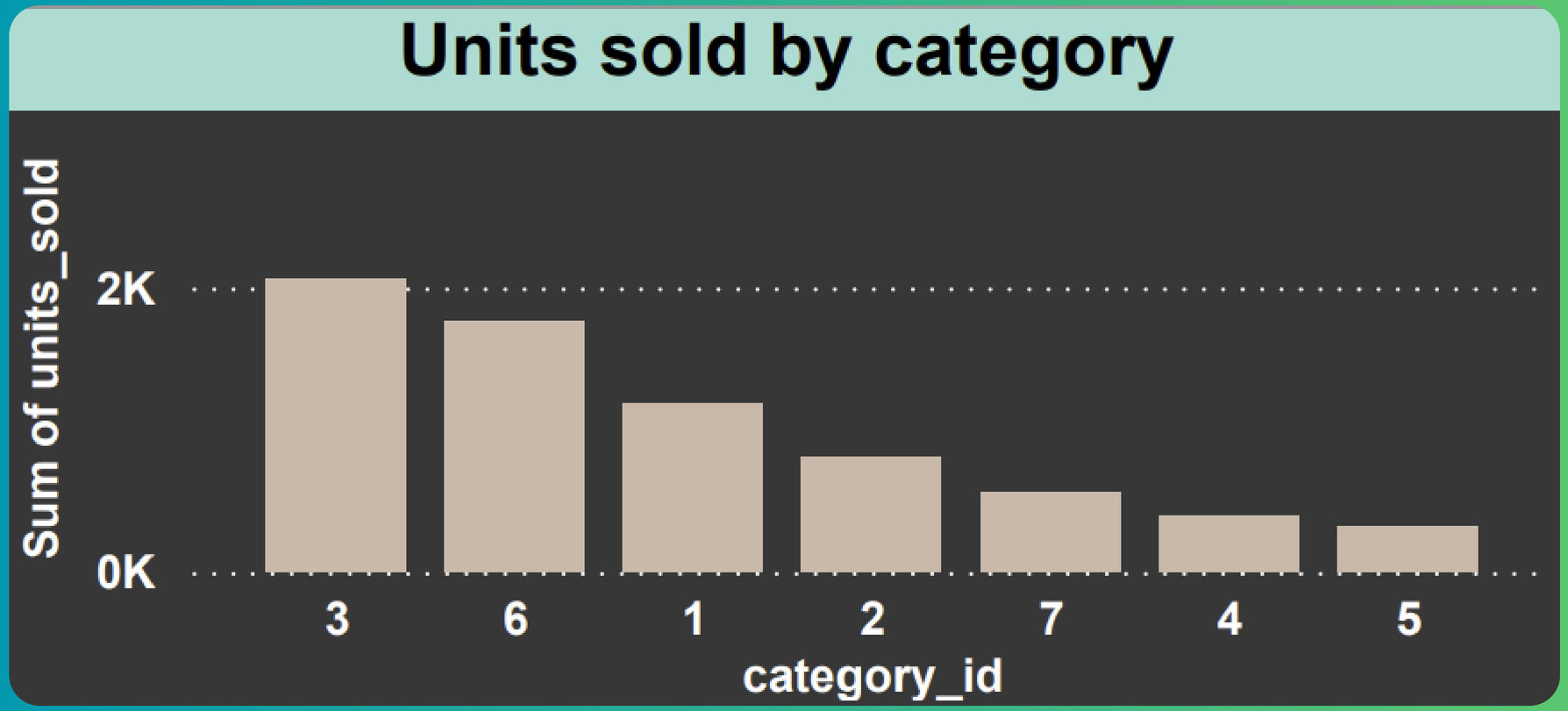
Best Selling Products

```
CREATE TABLE bike.BestSelling_products
SELECT
    p.category_id,
    p.product_id,p.product_name,
    SUM(oi.quantity) AS units_sold
FROM bike.products p
JOIN bike.order_items oi ON p.product_id =
oi.product_id
GROUP BY p.category_id,
p.product_id,p.product_name
ORDER BY units_sold DESC;
```

RESULT TABLE

category_id	product_id	product_name	units_sold
6	6	Surly Ice Cream Truck Frameset - 2016	167
3	13	Electra Cruiser 1 (24-Inch) - 2016	157
3	16	Electra Townie Original 7D EQ - 2016	156
6	7	Trek Slash 8 27.5 - 2016	154
1	1	Electra Girl's Hawaii 1 (20-inch) - 2015/2016	154
3	12	Electra Townie Original 21D - 2016	153
4	11	Surly Straggler 650b - 2016	151
2	25	Electra Townie Original 7D - 2015/2016	148
4	10	Surly Straggler - 2016	147
5	9	Trek Conduit+ - 2016	145
1	22	Electra Girl's Hawaii 1 (16-inch) - 2015/2016	145
6	4	Trek Fuel EX 8 29 - 2016	143
1	21	Electra Cruiser 1 (24-Inch) - 2016	139

Visualisation



Customer purchase patterns

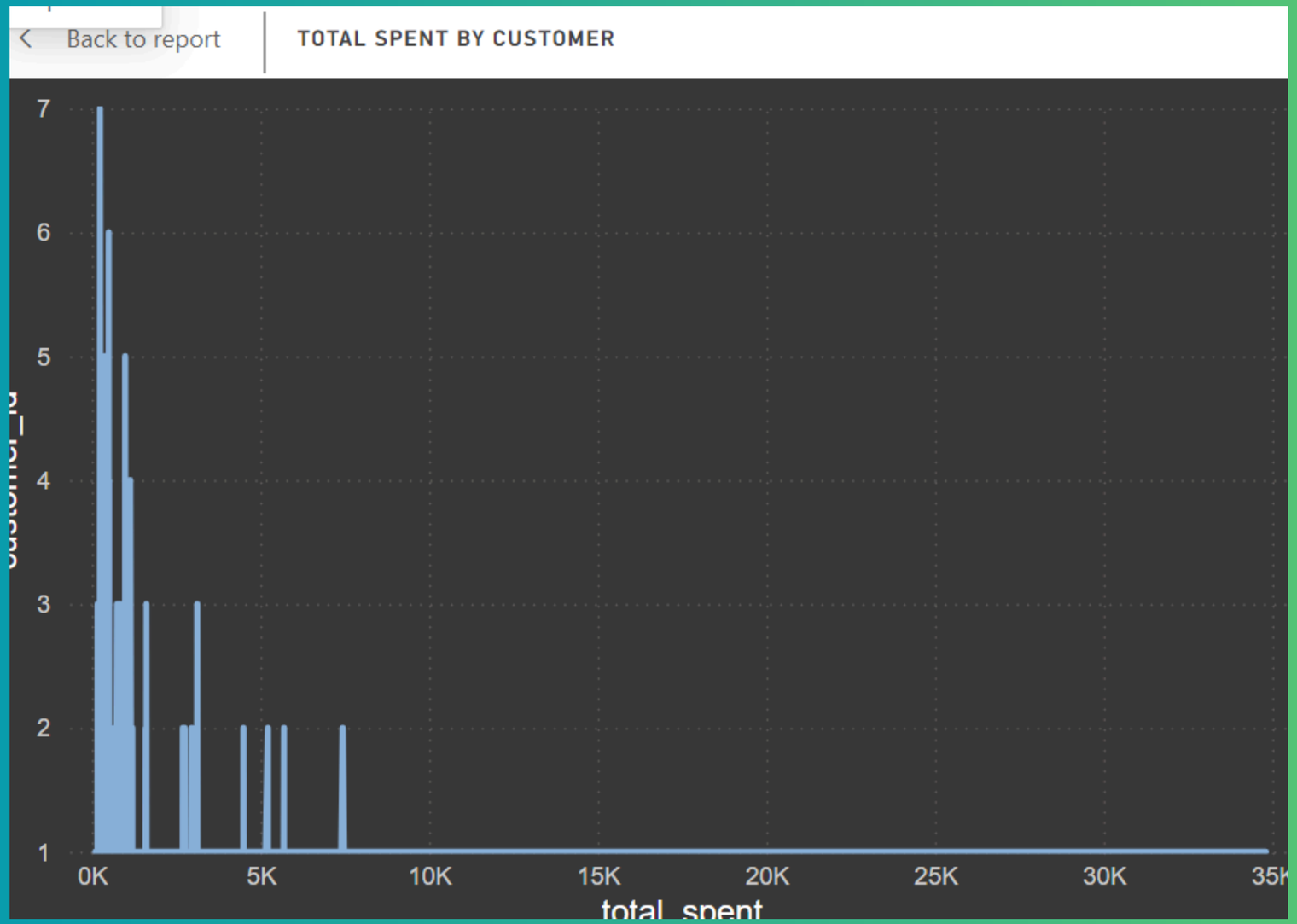
```
CREATE TABLE bike.customer_purchase_patterns
SELECT
    c.customer_id,
    COUNT(o.order_id) AS total_orders,
    SUM(oi.quantity * oi.list_price *(1-discount)) AS
total_spent
FROM bike.customers c
JOIN bike.orders o ON c.customer_id =
o.customer_id
JOIN bike.order_items oi ON o.order_id = oi.order_id
GROUP BY c.customer_id;
```

RESUL TABLE

Result Grid | Filter Rows:

customer_id	total_orders	total_spent
259	5	10231.0464
1212	2	1697.9717
523	2	1519.981
175	4	13023.926
1324	3	3900.0606999999995
94	10	34807.93919999999
324	3	2165.0816999999997
1204	2	1372.4719
60	6	9919.3294
442	1	242.991
1326	3	3155.9565
91	6	18856.877
873	4	3278.054
258	1	437.09069999999997

Visualisation



Monthly sales Trend

```
CREATE TABLE BIKE.MONTHLYSALESTREND
WITH MONTHLY_SALES AS (
SELECT
DATE_FORMAT(O.ORDER_DATE,'%Y-%M') AS MONTH,
SUM(OI.QUANTITY * OI.LIST_PRICE *(1-DISCOUNT)) AS
MONTHLY_TOTAL_SALES
FROM BIKE.ORDERS O
JOIN BIKE.ORDER_ITEMS OI ON O.ORDER_ID = OI.ORDER_ID
GROUP BY MONTH
)
SELECT
MONTH,
MONTHLY_TOTAL_SALES
FROM MONTHLY_SALES
ORDER BY MONTH;
```

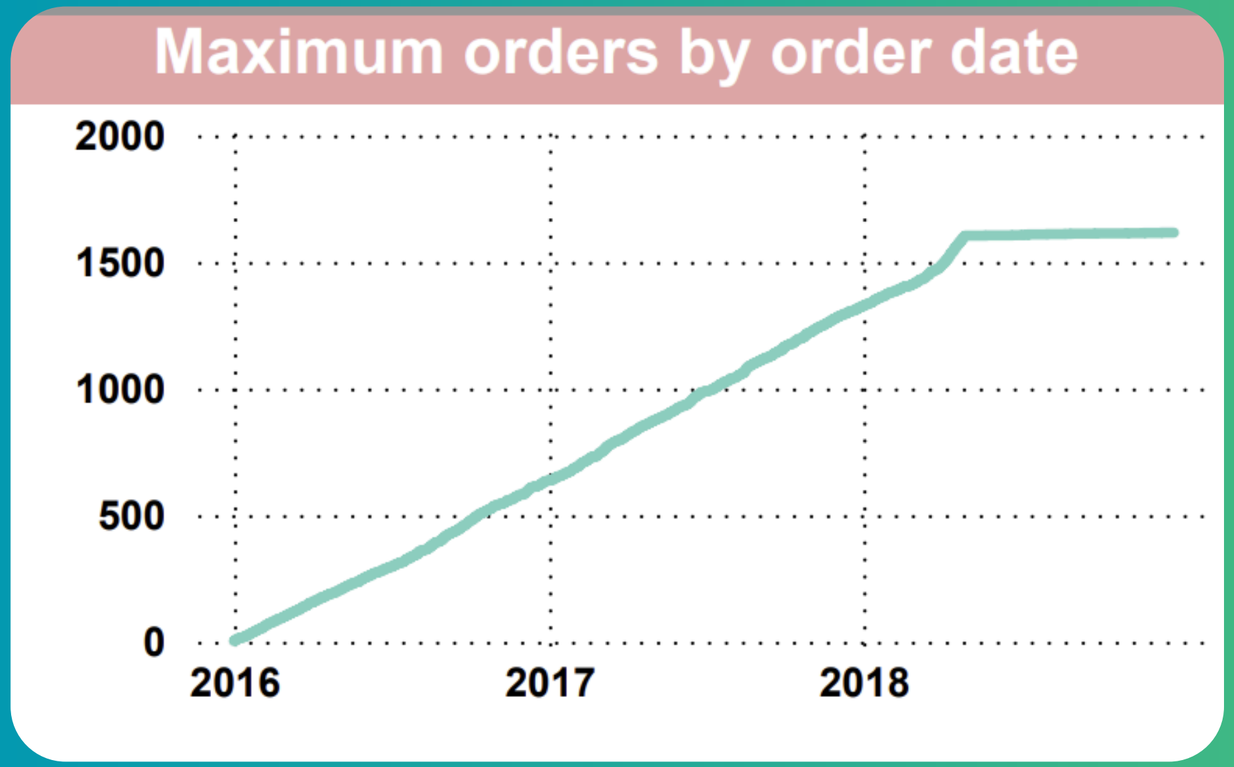
RESULT TABLE

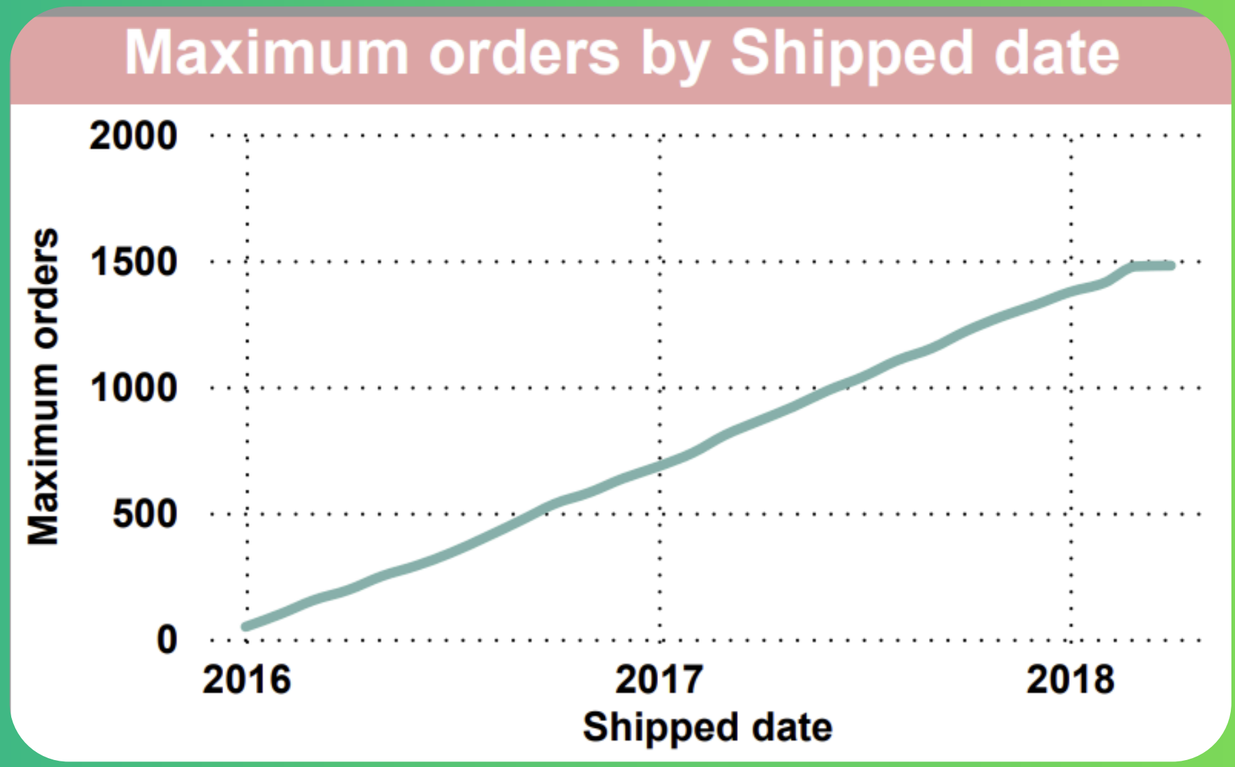
Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

order_id	customer_id	order_status	order_date	required_date	shipped_date	store_id	staff_id
1	259	4	2016-01-01	2016-01-03	2016-01-03	1	2
2	1212	4	2016-01-01	2016-01-04	2016-01-03	2	6
3	523	4	2016-01-02	2016-01-05	2016-01-03	2	7
4	175	4	2016-01-03	2016-01-04	2016-01-05	1	3
5	1324	4	2016-01-03	2016-01-06	2016-01-06	2	6
6	94	4	2016-01-04	2016-01-07	2016-01-05	2	6
7	324	4	2016-01-04	2016-01-07	2016-01-05	2	6
8	1204	4	2016-01-04	2016-01-05	2016-01-05	2	7
9	60	4	2016-01-05	2016-01-08	2016-01-08	1	2
10	442	4	2016-01-05	2016-01-06	2016-01-06	2	6
11	1326	4	2016-01-05	2016-01-08	2016-01-07	2	7
12	91	4	2016-01-06	2016-01-08	2016-01-09	1	2
13	873	4	2016-01-08	2016-01-11	2016-01-11	2	6
14	258	4	2016-01-09	2016-01-11	2016-01-12	1	3

alestrend 1

Visualisation





product category and sales contribution

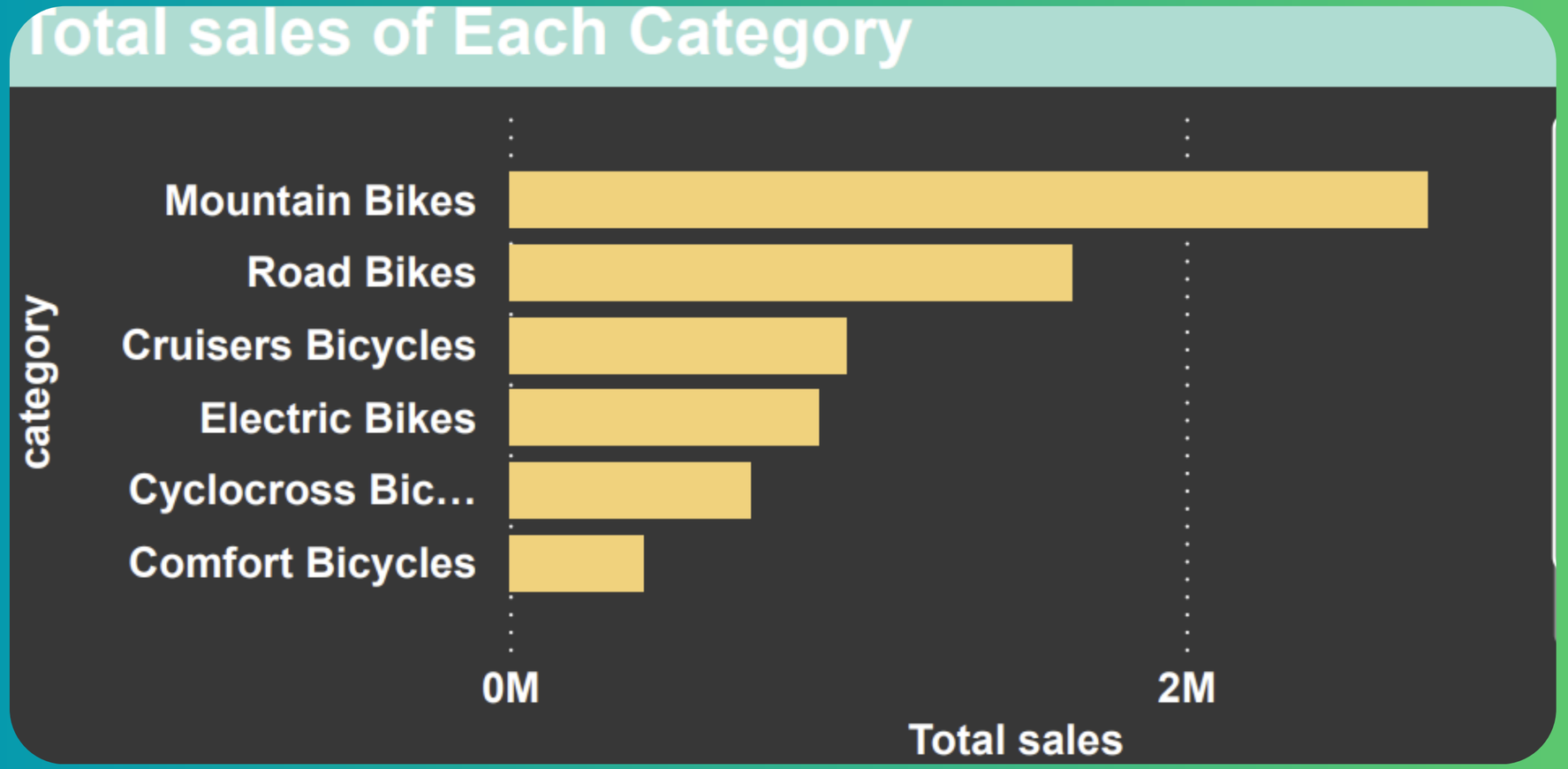
```
CREATE TABLE bike.Product_CategorySales_Contribution
WITH category_sales AS (
SELECT
c.category_id, c.category_name,
SUM(oi.quantity * oi.list_price * (1-discount)) AS category_total_sales
FROM bike.categories c
JOIN bike.products p ON c.category_id = p.category_id
JOIN bike.order_items oi ON p.product_id = oi.product_id
GROUP BY c.category_id, category_name
),
total_sales AS (
SELECT SUM(category_total_sales) AS overall_sales
FROM category_sales
)
SELECT
cs.category_id,cs.category_name,
cs.category_total_sales,
(cs.category_total_sales / ts.overall_sales) * 100 AS sales_contribution_pct
FROM category_sales cs, total_sales ts;
```

RESULT TABLE

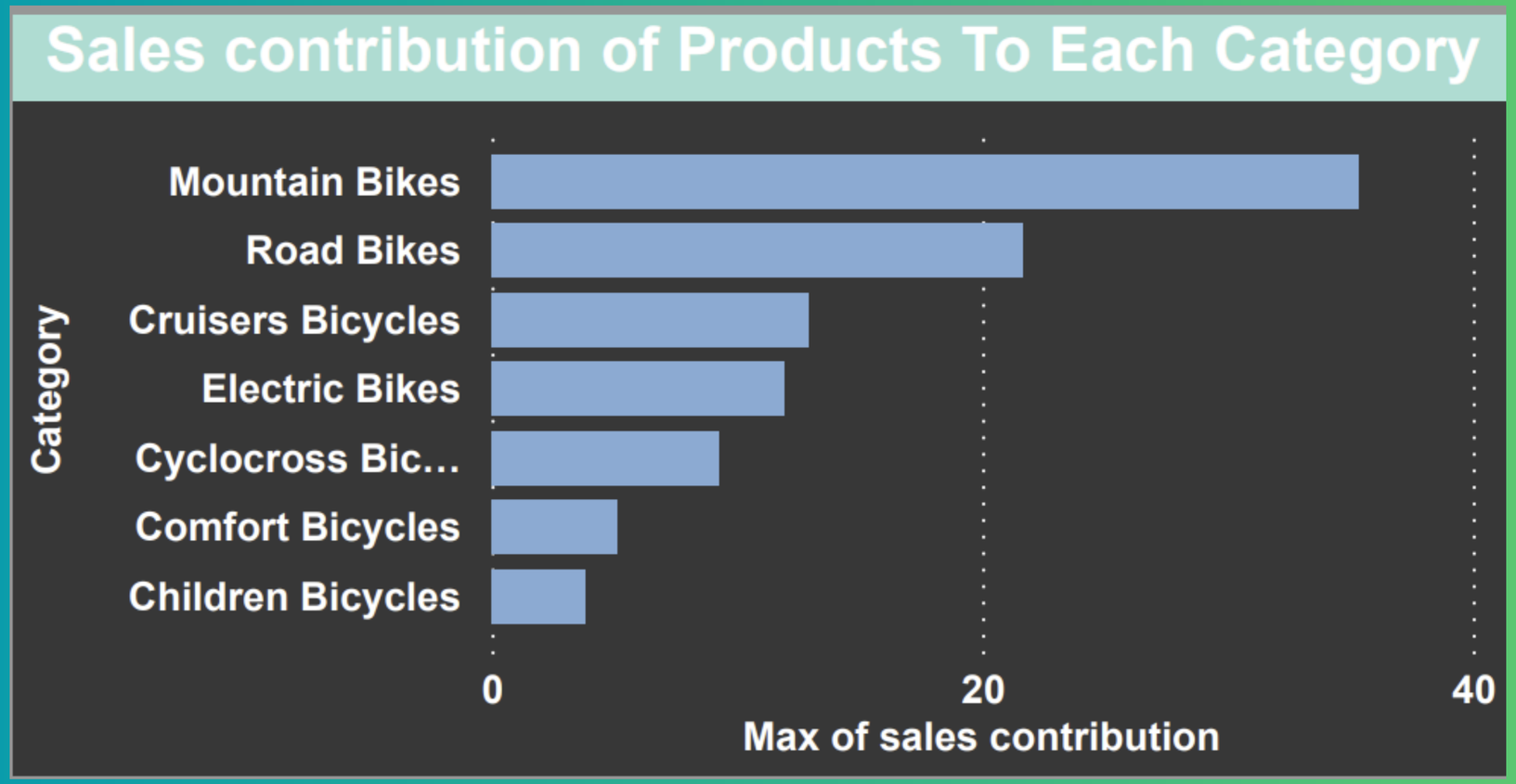
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category_id	category_name	category_total_sales	sales_contribution_pct
3	Cruisers Bicycles	995032.6237000015	12.940792563698377
6	Mountain Bikes	2715079.533699997	35.31068248687666
4	Cyclocross Bicycles	711011.8359000003	9.24698957477354
2	Comfort Bicycles	394020.09810000093	5.124387114545006
5	Electric Bikes	916684.7800000007	11.921847888935176
1	Children Bicycles	292189.1982000012	3.800036012085139 5
7	Road Bikes	1665098.487999998	21.65526397638228

Visualisation



Visualisation 2



Sales Performance

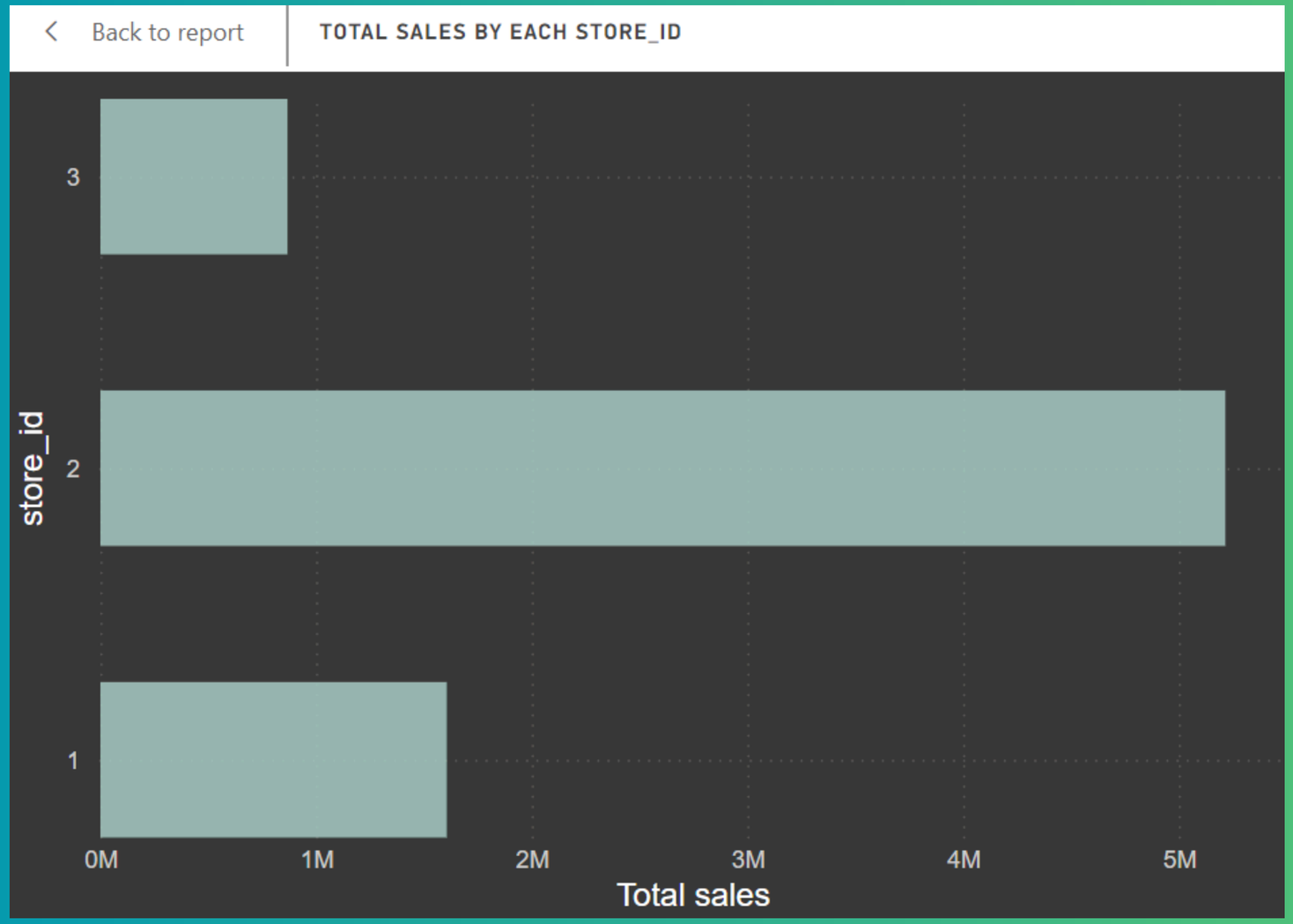
```
CREATE TABLE BIKE.SALES_PERFROMANCE AS
SELECT
    O.STORE_ID,
SUM(OI.QUANTITY * OI.LIST_PRICE * (1 - OI.DISCOUNT)) AS
    TOTAL_SALES
FROM BIKE.ORDERS O
JOIN BIKE.ORDER_ITEMS OI ON O.ORDER_ID = OI.ORDER_ID
GROUP BY O.STORE_ID;
```

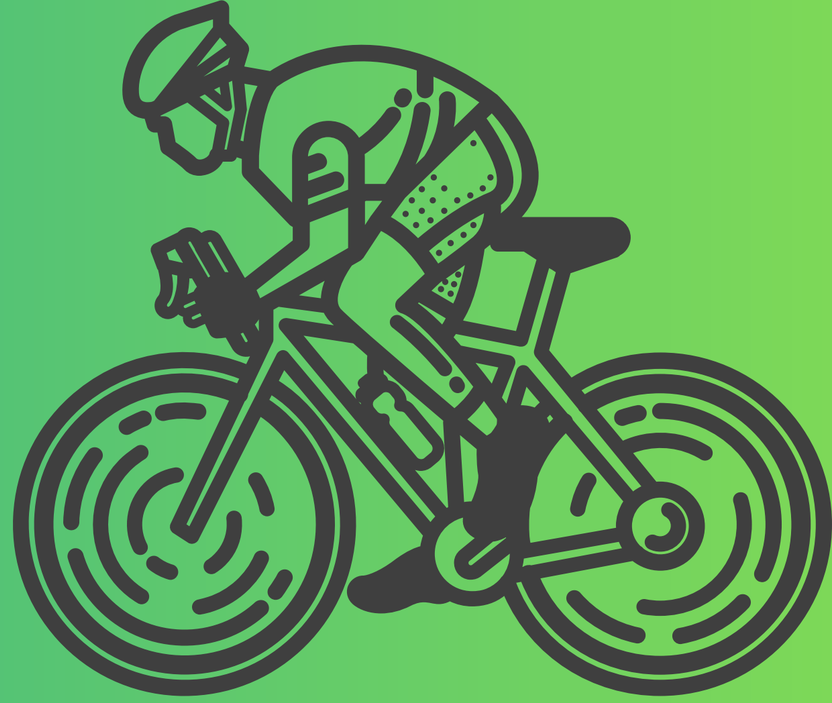
RESULT TABLE

Result Grid | Filter Rows:

store_id	total_sales
1	1605823.0364999974
2	5215751.277499983
3	867542.2436000014

Visualisation





Sales by Staff and Stores

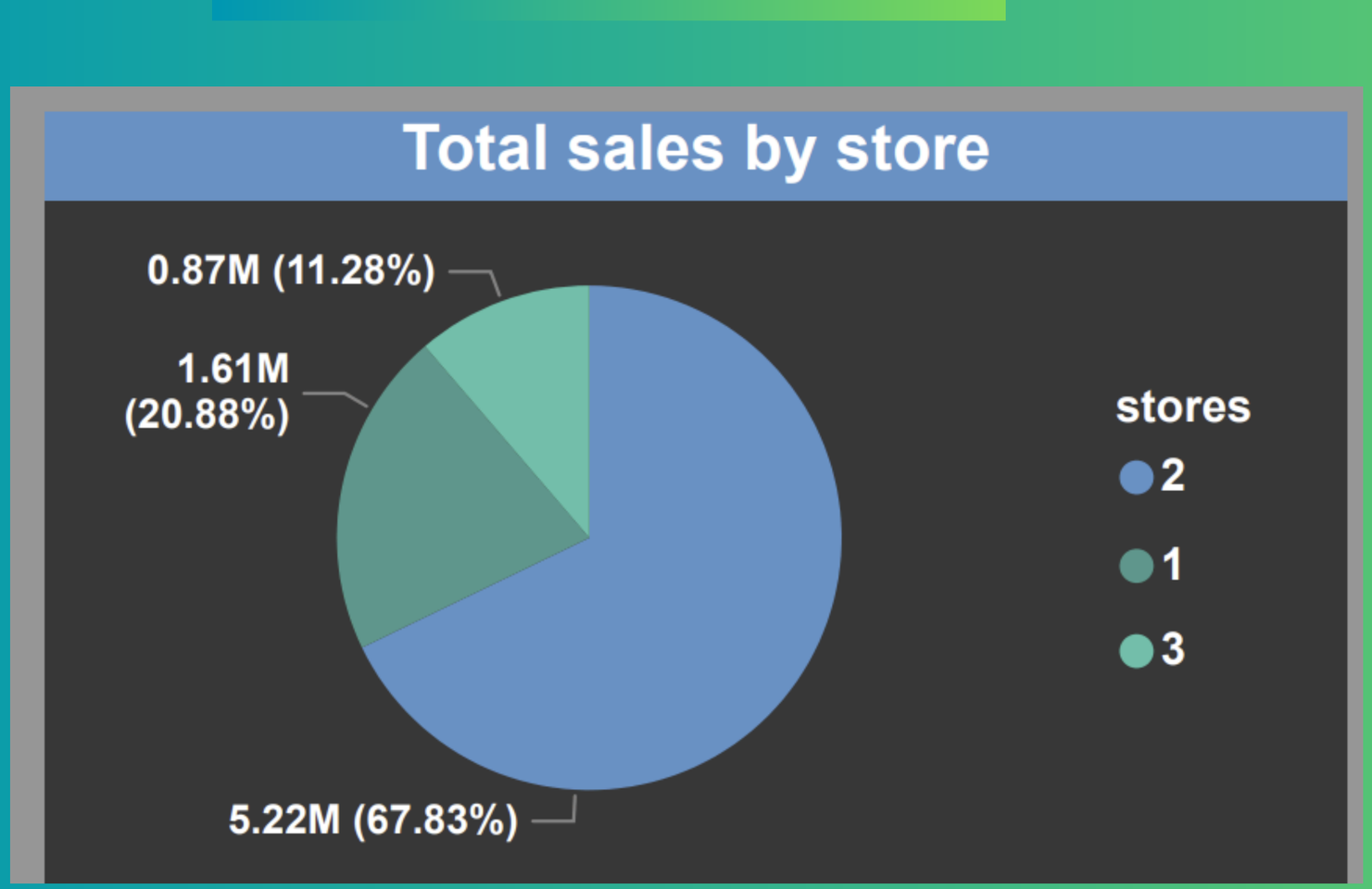
```
CREATE TABLE BIKE.SALESBYSTAFF_AND_STORES
SELECT
ST.STORE_ID,
SF.STAFF_ID,STAFF_NAME,
SUM(OI.QUANTITY * OI.LIST_PRICE *(1-DISCOUNT)) AS
TOTAL_SALES
FROM BIKE.STAFFS SF
JOIN BIKE.ORDERS O ON SF.STAFF_ID = O.STAFF_ID
JOIN BIKE.ORDER_ITEMS OI ON O.ORDER_ID = OI.ORDER_ID
JOIN BIKE.STORES ST ON SF.STORE_ID = ST.STORE_ID
GROUP BY ST.STORE_ID, SF.STAFF_ID,STAFF_NAME;
```

RESULT TABLE

Result Grid | Filter Rows: | Export:

store_id	staff_id	staff_name	total_sales
1	2	Mireya Copeland	752535.6776000011
2	6	Marcelene Boyer	2624120.652999998
2	7	Venita Daniel	2591630.6244999943
1	3	Genna Serrano	853287.3589000016
3	8	Kali Vargas	463918.3046000002
3	9	Layla Terrell	403623.9390000005

Visualisation



Staff ranking by stores

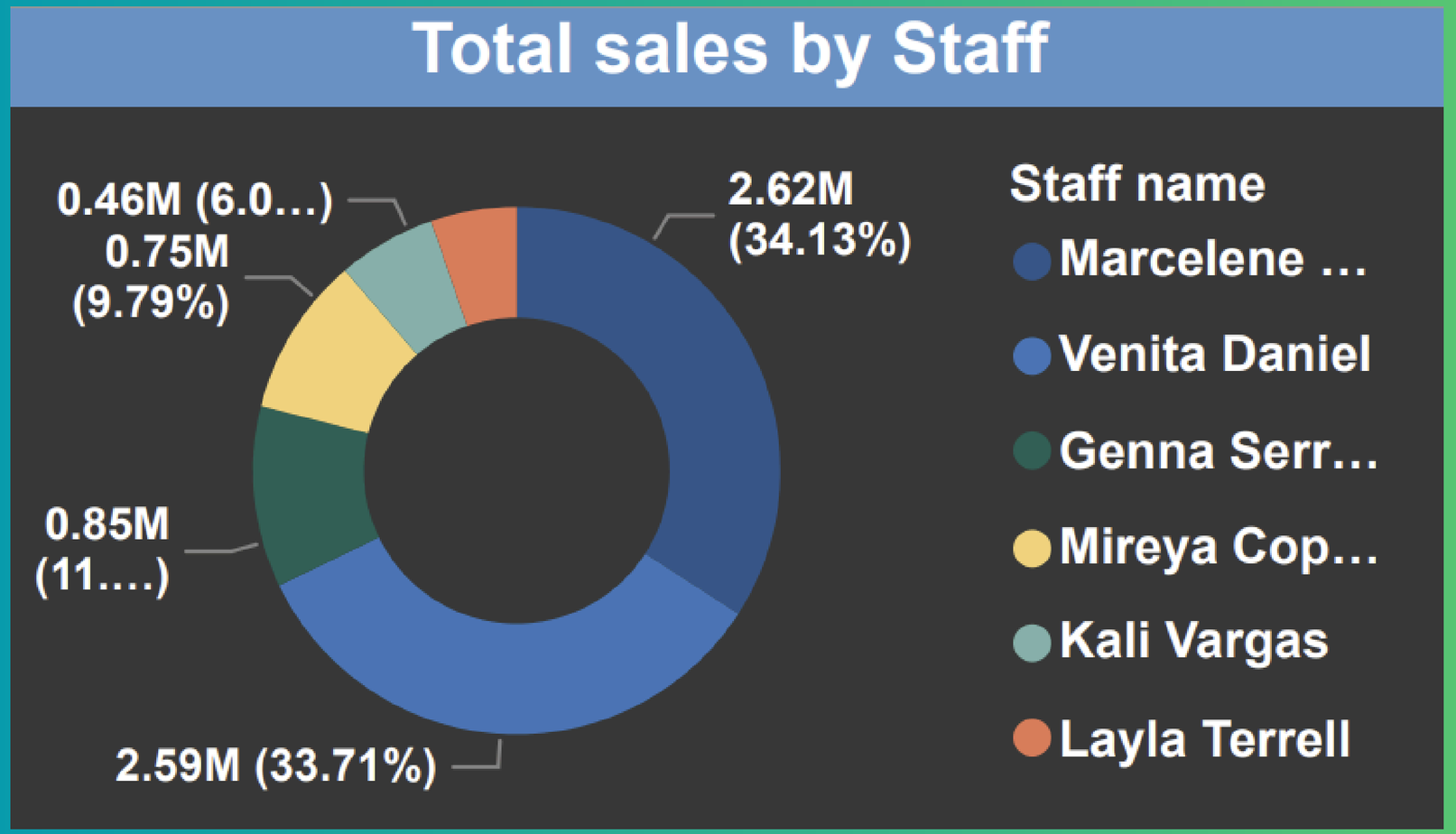
```
CREATE TABLE bike.Staff_Ranking_ByStore
SELECT store_id, staff_id,  total_sales,
rank_value
FROM ( SELECT  store_id,  staff_id,  total_sales,
ROW_NUMBER() OVER (PARTITION BY store_id ORDER BY total_sales DESC) AS
rank_value
FROM (
SELECT
  s.store_id,
  sf.staff_id,
SUM(oi.quantity * oi.list_price * (1 - oi.discount)) AS total_sales
FROM bike.staffs sf
JOIN bike.orders o ON sf.staff_id = o.staff_id
JOIN bike.order_items oi ON o.order_id = oi.order_id
JOIN bike.stores s ON sf.store_id = s.store_id
GROUP BY s.store_id, sf.staff_id
) staff_sales
) ranked_staff
WHERE rank_value <= 5;
```

RESULT TABLE

Result Grid | Filter Rows: | Export:

store_id	staff_id	total_sales	rank_value
1	3	853287.3589000016	1
1	2	752535.6776000011	2
2	6	2624120.652999998	1
2	7	2591630.6244999943	2
3	8	463918.3046000002	1
3	9	403623.9390000005	2

Visualisation



TOP SPENT CUSTOMER

```
CREATE TABLE BIKE.TOPSPENTCUSTOMER
SELECT CUSTOMER_ID, TOTAL_SPENT,CUSTOMER_NAME
FROM (
    SELECT
        C.CUSTOMER_ID,C.CUSTOMER_NAME,
        SUM(OI.QUANTITY * OI.LIST_PRICE *(1-DISCOUNT)) AS
TOTAL_SPENT
    FROM BIKE.CUSTOMERS C
    JOIN BIKE.ORDERS O ON C.CUSTOMER_ID =
O.CUSTOMER_ID
    JOIN BIKE.ORDER_ITEMS OI ON O.ORDER_ID =
OI.ORDER_ID
    GROUP BY C.CUSTOMER_ID,C.CUSTOMER_NAME
    ) AS CUSTOMER_SPENDING
    ORDER BY TOTAL_SPENT DESC
    LIMIT 1;
```

RESULT TABLE

Result Grid | Filter Rows: | Export

customer_id	total_spent	customer_name
94	34807.93919999999	Sharyn Hopkins

Visualisation

Top spent Customer

Customer	Total spent
Sharyn Hopkins	34,807.94

Valuable Insights

📊 Store Performance

Highest Average Order Value: Store 3 takes the lead with $5,532.19, followed by Store 2 ($5,330.51) and Store 1 ($5,144.10).
Total Sales: Store 3 also dominates with $5.22M, contributing 67.83% of total sales, while Store 1 and Store 2 add $1.61M (20.88%) and $0.87M (11.28%), respectively.

📅 Order Trends

Maximum Orders by Order Date: Peaks are observed in mid-2017 and late-2018, indicating seasonal or promotional effects driving sales.
Shipped Orders Trends: Shipping patterns align with order peaks, suggesting efficient order fulfillment processes during busy periods.

Category Insights

Units Sold by Category: Mountain Bikes lead by a significant margin, with Road Bikes and Cruisers following.
Total Sales by Category: Mountain Bikes again dominate revenue generation, reinforcing their importance in inventory and marketing strategies.

👥 Customer Contributions

Top Spender: Sharyn Hopkins is the most valuable customer, spending $34,807.94.
Customer Spending Trends: Consistently high spending across top customer IDs highlights loyalty opportunities.

Staff Performance

Total Sales by Staff: Marcelene leads with $2.62M, closely followed by Venita ($2.59M), showcasing strong individual contributions to team success.

🚀 Product Performance

Sales Contribution Across Categories: Mountain and Road Bikes show the highest product-category contributions, suggesting these are key focus areas for scaling.

Thankyou