# SQL PROJECT ON JENKINS BIKE STORE SALES

# ABOUT OUR BIKE SALES PROJECT

## 🚴 Bike Store Sales Analysis Using SQL 🔍

### 📌 Project Overview

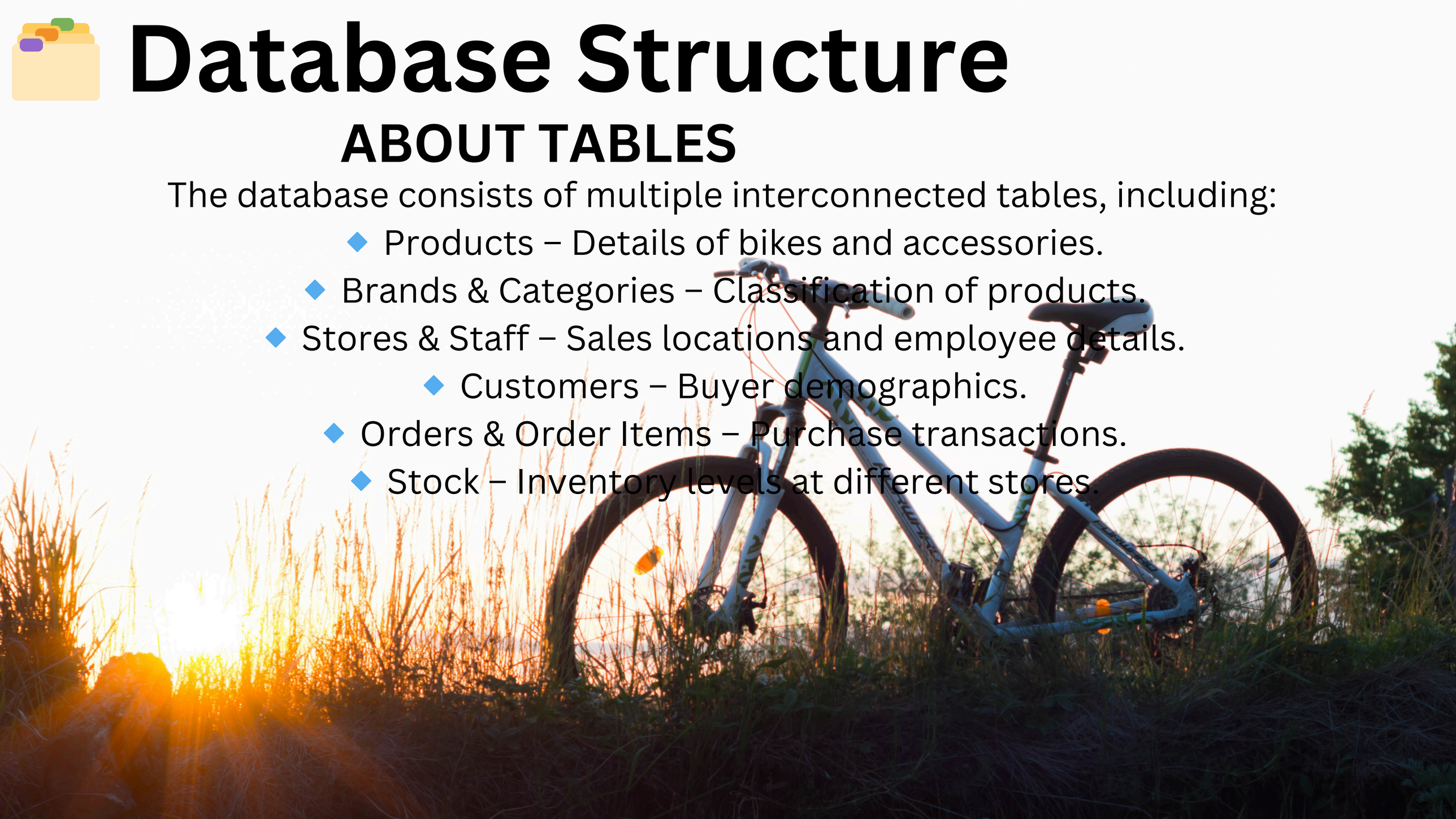
This project focuses on analyzing sales performance and customer insights for Jenkins, bike stores, using SQL. By querying a structured database, we extract valuable insights into sales trends, product performance, and customer behavior.

# Database Structure

## ABOUT TABLES

The database consists of multiple interconnected tables, including:

- Products – Details of bikes and accessories.
- Brands & Categories – Classification of products.
- Stores & Staff – Sales locations and employee details.
- Customers – Buyer demographics.
- Orders & Order Items – Purchase transactions.
- Stock – Inventory levels at different stores.



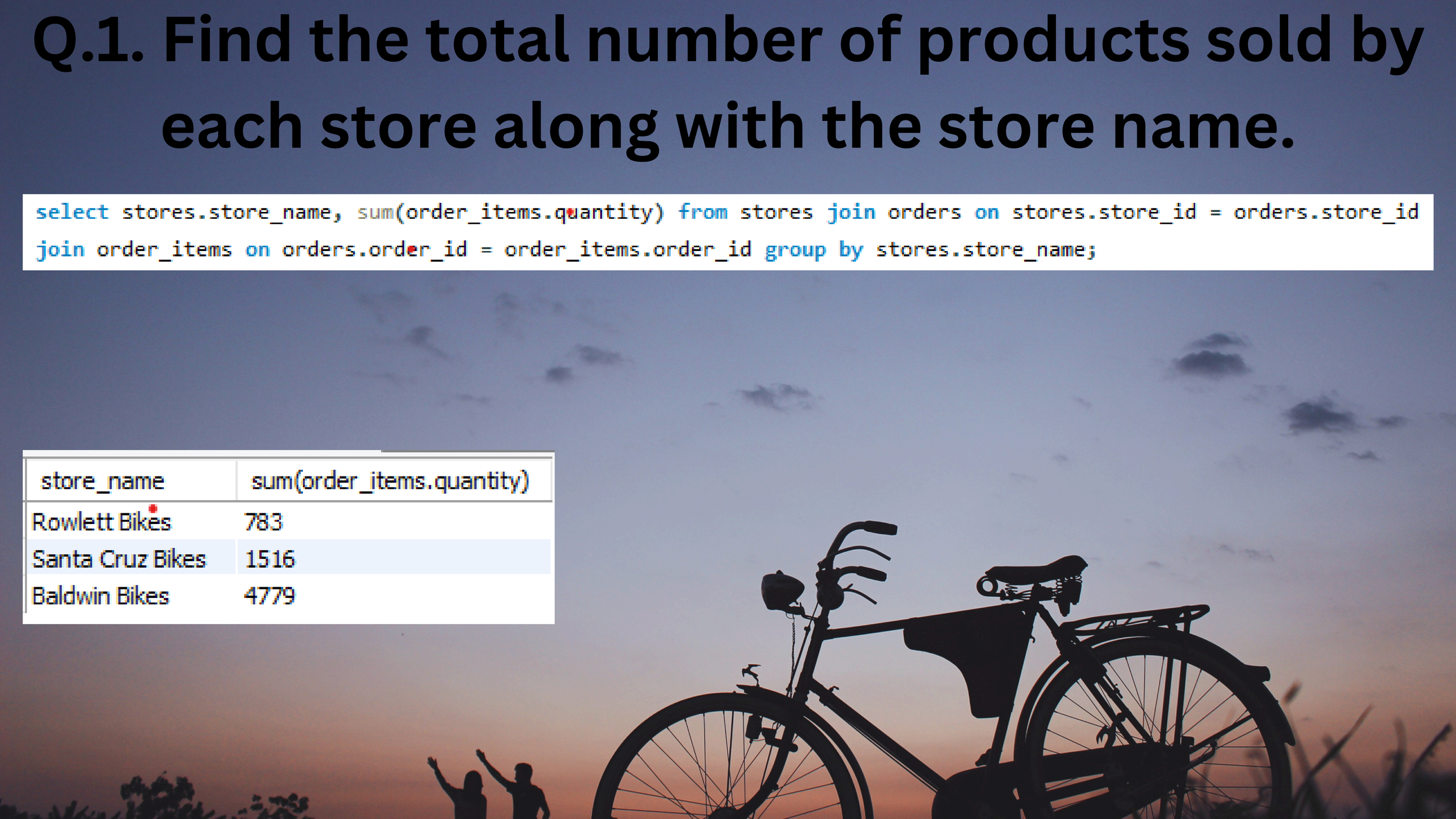
# Schema Relationships & Connections

- One-to-Many:
  - A brand can have multiple products
  - A category can have multiple products
  - A customer can place multiple orders
  - A store has multiple staff members
  - An order contains multiple items
- Many-to-Many:
  - Products & Stores (via stock table)
  - Orders & Products (via orders_items table)

# Q.1. Find the total number of products sold by each store along with the store name.

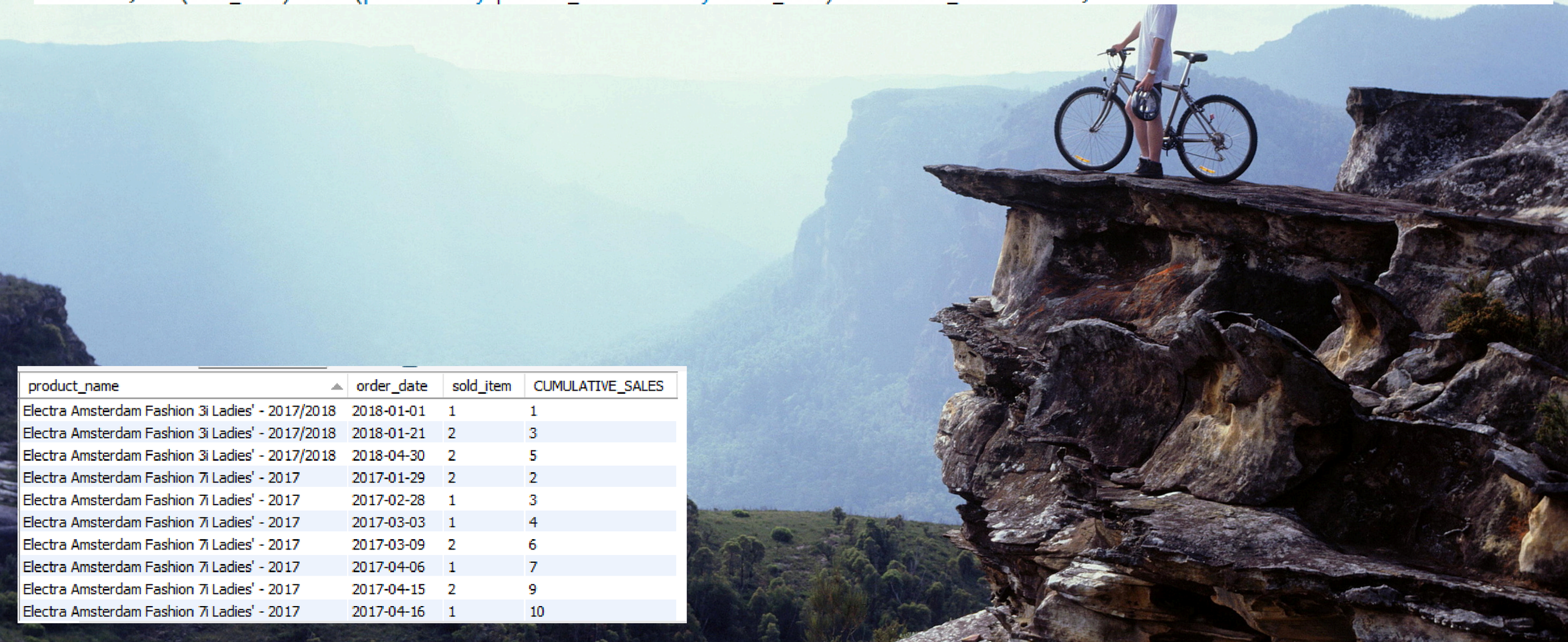
```
select stores.store_name, sum(order_items.quantity) from stores join orders on stores.store_id = orders.store_id
join order_items on orders.order_id = order_items.order_id group by stores.store_name;
```

| store_name | sum(order_items.quantity) |
|---|---|
| Rowlett Bikes | 783 |
| Santa Cruz Bikes | 1516 |
| Baldwin Bikes | 4779 |



# Q.2. Calculate the cumulative sum of quantities sold for each product over time.

```
with a as (select products.product_name, orders.order_date,sum(order_items.quantity)sold_item from products join order_items on products.product_id
= order_items.product_id join orders on orders.order_id = order_items.order_id group by products.product_name,orders.order_date)
select *, sum(sold_item) over (partition by product_name order by order_date) CUMULATIVE_SALES from a;
```



| product_name | order_date | sold_item | CUMULATIVE_SALES |
|---|---|---|---|
| Electra Amsterdam Fashion 3i Ladies' - 2017/2018 | 2018-01-01 | 1 | 1 |
| Electra Amsterdam Fashion 3i Ladies' - 2017/2018 | 2018-01-21 | 2 | 3 |
| Electra Amsterdam Fashion 3i Ladies' - 2017/2018 | 2018-04-30 | 2 | 5 |
| Electra Amsterdam Fashion 7i Ladies' - 2017 | 2017-01-29 | 2 | 2 |
| Electra Amsterdam Fashion 7i Ladies' - 2017 | 2017-02-28 | 1 | 3 |
| Electra Amsterdam Fashion 7i Ladies' - 2017 | 2017-03-03 | 1 | 4 |
| Electra Amsterdam Fashion 7i Ladies' - 2017 | 2017-03-09 | 2 | 6 |
| Electra Amsterdam Fashion 7i Ladies' - 2017 | 2017-04-06 | 1 | 7 |
| Electra Amsterdam Fashion 7i Ladies' - 2017 | 2017-04-15 | 2 | 9 |
| Electra Amsterdam Fashion 7i Ladies' - 2017 | 2017-04-16 | 1 | 10 |

# Q.3. Find the product with the highest total sales (quantity * price) for each category.

```
with a as (select products.product_name,categories.category_name,sum(order_items.quantity * order_items.list_price) total_sales from products join
order_items on products.product_id = order_items.product_id join categories on products.category_id = categories.category_id group by
products.product_name, categories.category_name)
select * from (select *, rank() over( partition by category_name order by total_sales desc)rnk from a)b where rnk = 1;
```

| product_name | category_name | total_sales | rnk |
|---|---|---|---|
| Electra Girl's Hawaii 1 (20-inch) - 2015/2016 | Children Bicycles | 4619846.00 | 1 |
| Electra Townie Original 7D EQ - 2016 | Comfort Bicycles | 8039866.00 | 1 |
| Electra Townie Original 7D EQ - 2016 | Cruisers Bicycles | 9359844.00 | 1 |
| Surly Straggler 650b - 2016 | Cyclocross Bicycles | 25382949.00 | 1 |
| Trek Conduit+ - 2016 | Electric Bikes | 43499855.00 | 1 |
| Trek Slash 8 275 - 2016 | Mountain Bikes | 61599846.00 | 1 |
| Trek Domane SLR 6 Disc - 2017 | Road Bikes | 23649957.00 | 1 |

# Q.4 Find the customer who spent the most money on orders.

```sql
select customers.first_name,customers.customer_id,sum(order_items.quantity*order_items.list_price)sales from customers
join orders on customers.customer_id = orders.customer_id join order_items on orders.order_id = order_items.order_id
group by customers.first_name,customers.customer_id order by sales limit  1;
```

| first_name | customer_id | sales |
|---|---|---|
| Stephanie | 850 | 10999.00 |

# Q.5.Find the highest-priced product for each category name.

```sql
select categories.category_name,sum(products.list_price)highest_price from products
join categories on products.category_id = categories.category_id
group by categories.category_name order by highest_price desc;
```

| category_name | highest_price |
|---|---|
| Road Bikes | 18423890.00 |
| Mountain Bikes | 9047540.00 |
| Electric Bikes | 7875976.00 |
| Cruisers Bicycles | 5452452.00 |
| Cyclocross Bicycles | 2082740.00 |
| Comfort Bicycles | 2046370.00 |
| Children Bicycles | 1697941.00 |

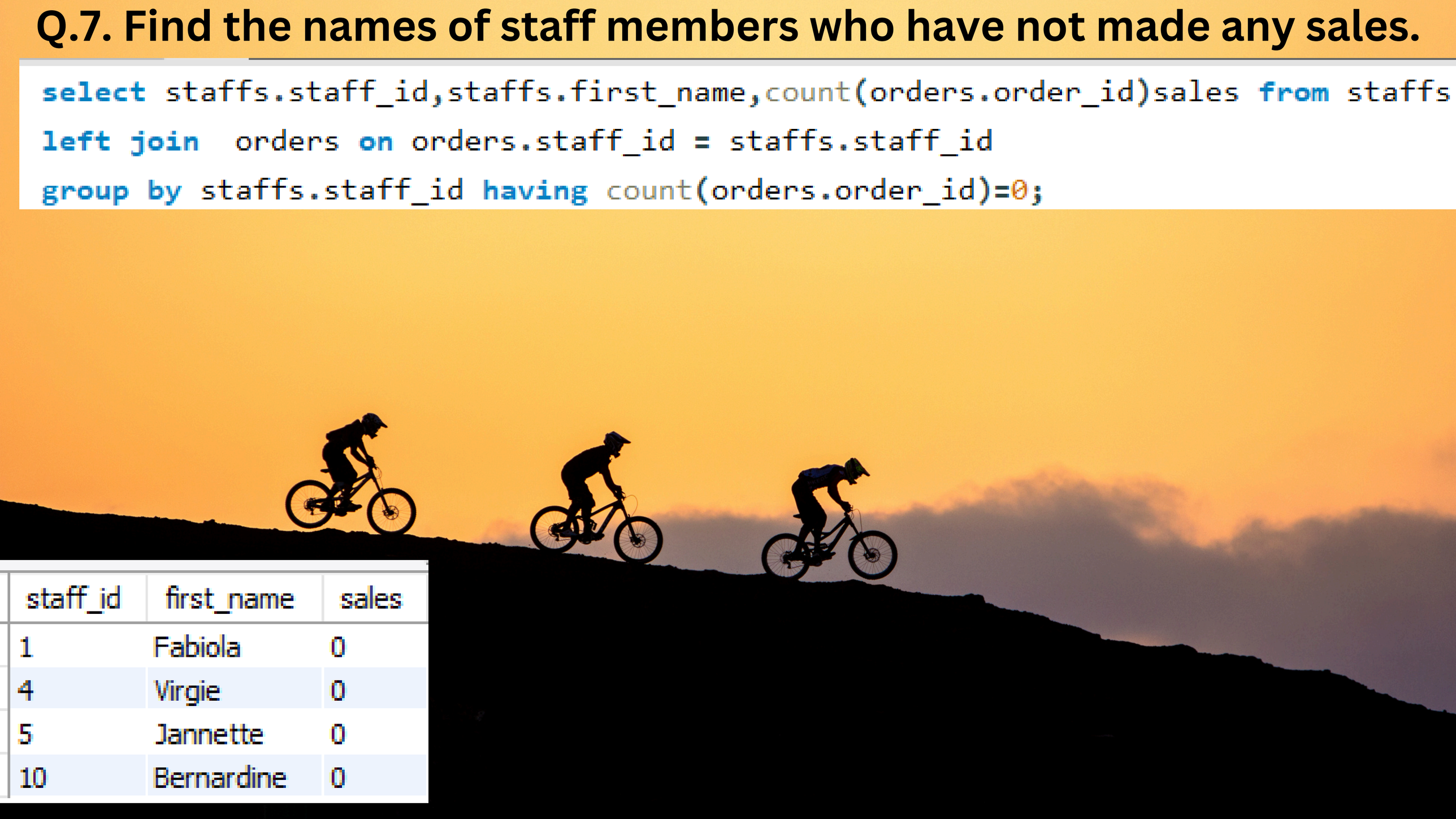
# Q.6 Find the total number of orders placed by each customer per store.

```
select stores.store_name,concat(customers.first_name,' ',customers.last_name) full_name,customers.customer_id,
count(orders.order_id)orders from customers join orders on customers.customer_id = orders.customer_id
join stores on stores.store_id = orders.store_id group by stores.store_name,
concat(customers.first_name,' ',customers.last_name), customers.customer_id
order by full name;
```

| store_name | full_name | customer_id | orders |
|---|---|---|---|
| Baldwin Bikes | Aaron Knapp | 1174 | 1 |
| Baldwin Bikes | Abbey Pugh | 338 | 1 |
| Baldwin Bikes | Abby Gamble | 75 | 2 |
| Rowlett Bikes | Abram Copeland | 1224 | 1 |
| Santa Cruz Bikes | Adam Henderson | 673 | 1 |
| Baldwin Bikes | Adam Thornton | 1085 | 1 |
| Baldwin Bikes | Addie Hahn | 195 | 1 |
| Santa Cruz Bikes | Adelaida Hancock | 1261 | 1 |
| Baldwin Bikes | Adelle Larsen | 22 | 2 |
| Baldwin Bikes | Adena Blake | 1023 | 1 |

# Q.7. Find the names of staff members who have not made any sales.

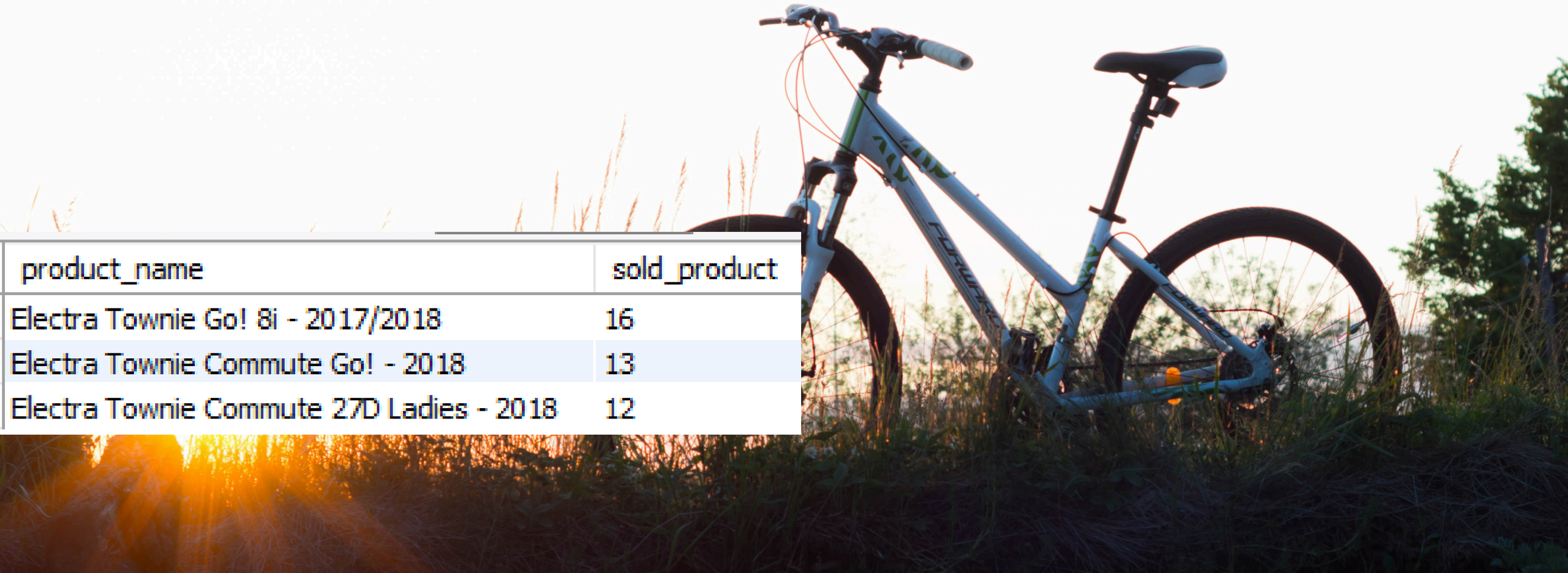
```
select staffs.staff_id,staffs.first_name,count(orders.order_id)sales from staffs
left join  orders on orders.staff_id = staffs.staff_id
group by staffs.staff_id having count(orders.order_id)=0;
```



| staff_id | first_name | sales |
|---|---|---|
| 1 | Fabiola | 0 |
| 4 | Virgie | 0 |
| 5 | Jannette | 0 |
| 10 | Bernardine | 0 |

# Q.8. Find the top 3 most sold products in terms of quantity.

```
select products.product_name,sum(order_items.quantity) sold_product from products
join order_items on products.product_id = order_items.order_id
group by  products.product_name order by sold_product desc limit 3;
```



| product_name | sold_product |
|---|---|
| Electra Townie Go! 8i - 2017/2018 | 16 |
| Electra Townie Commute Go! - 2018 | 13 |
| Electra Townie Commute 27D Ladies - 2018 | 12 |

# Q.9. Find the median value of the price list.

```sql
WITH a AS (SELECT list_price, ROW_NUMBER() OVER (ORDER BY list_price) AS rn,COUNT(*) OVER () AS n
    FROM order_items)
SELECT CASE WHEN n % 2 = 0
        THEN (SELECT AVG(list_price) FROM a WHERE rn IN (n/2, (n/2) + 1))
        ELSE (SELECT list_price FROM a WHERE rn = (n/2) + 1)
    END AS median FROM a LIMIT 1;
```

| median |
| --- |
| 59999.000000 |

# Q.10. List all products that have never been ordered.(use Exists)

```sql
select products.product_id,products.product_name,order_items.order_id from products left join
order_items on products.product_id=order_items.product_id
where not exists (select 1 from order_items WHERE order_items.product_id = products.product_id );
```

| product_id | product_name | order_id |
|---|---|---|
| 1 | Trek 820 - 2016 | NULL |
| 121 | Surly Krampus Frameset - 2018 | NULL |
| 125 | Trek Kids' Dual Sport - 2018 | NULL |
| 154 | Trek Domane SLR 6 Disc Women's - 2018 | NULL |
| 195 | Electra Townie Go! 8i Ladies' - 2018 | NULL |
| 267 | Trek Precaliber 12 Girl's - 2018 | NULL |
| 284 | Electra Savannah 1 (20-inch) - Girl's - 2018 | NULL |
| 291 | Electra Sweet Ride 1 (20-inch) - Girl's - 2018 | NULL |
| 316 | Trek Checkpoint ALR 4 Women's - 2019 | NULL |
| 317 | Trek Checkpoint ALR 5 - 2019 | NULL |

# Q.11. List the names of staff members who have made more sales than the average number of sales by all staff members.

```
with a as (select staffs.staff_id, coalesce(sum(order_items.list_price*order_items.quantity),0)sales
from staffs left join orders on staffs.staff_id = orders.staff_id
left join order_items on orders.order_id =order_items.order_id group by staffs.staff_id)
select * from a where sales > (select avg(sales) from a);
```


| staff_id | sales |
|---|---|
| 3 | 95272226.00 |
| 6 | 293888873.00 |
| 7 | 288735348.00 |

# Q.12. Identify the customers who have ordered all types of products (i.e., from every category).

```sql
select customers.customer_id, count(order_items.product_id) All_type_products from customers join orders
on customers.customer_id = orders.customer_id join order_items on orders.order_id = order_items.order_id
join products on order_items.product_id = products.product_id group by customers.customer_id
having count(distinct products.category_id) = (select count( category_id)from categories);
```

| customer_id | All_type_products |
|---|---|
| 9 | 9 |