Bike Stores Relational Database

DATA MINING PROJECT

SAKARTH, RISHAV, ANIRUDH

SCOPE

Business Operations Management- day-to-day operations within a retail business setting

Customer Details Management - maintains comprehensive records of customers.

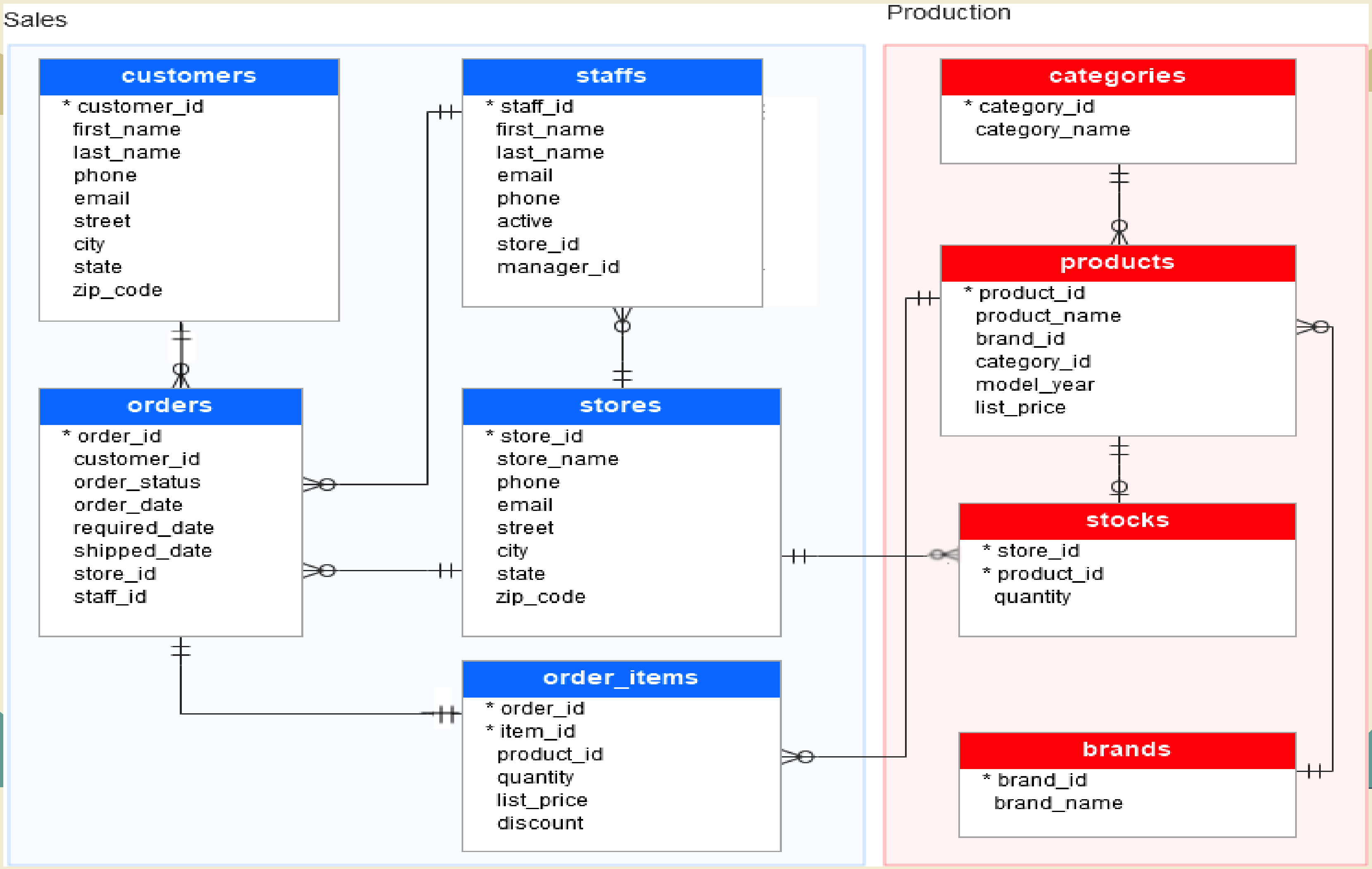
Staff Information Tracking - The database tracks staff details, roles, and performance metrics.

Product Catalogue Maintenance - provides a structured system for managing the product catalogue.

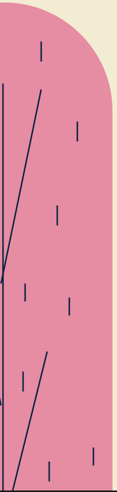
Sales Analysis and Reporting - enables detailed sales analysis and reporting

REPRESENTATION

Our database has 9 tables, 3 triggers, 7 views, 2 indexes and 1 transaction


Sales
Production
customers
* customer_id
first_name
last_name
phone
email
street
city
state
zip_code
staffs
* staff_id
first_name
last_name
email
phone
active
store_id
manager_id
orders
* order_id
customer_id
order_status
order_date
required_date
shipped_date
store_id
staff_id
stores
* store_id
store_name
phone
email
street
city
state
zip_code
order_items
* order_id
* item_id
product_id
quantity
list_price
discount
categories
* category_id
category_name
products
* product_id
product_name
brand_id
category_id
model_year
list_price
stocks
* store_id
* product_id
quantity
brands
* brand_id
brand_name

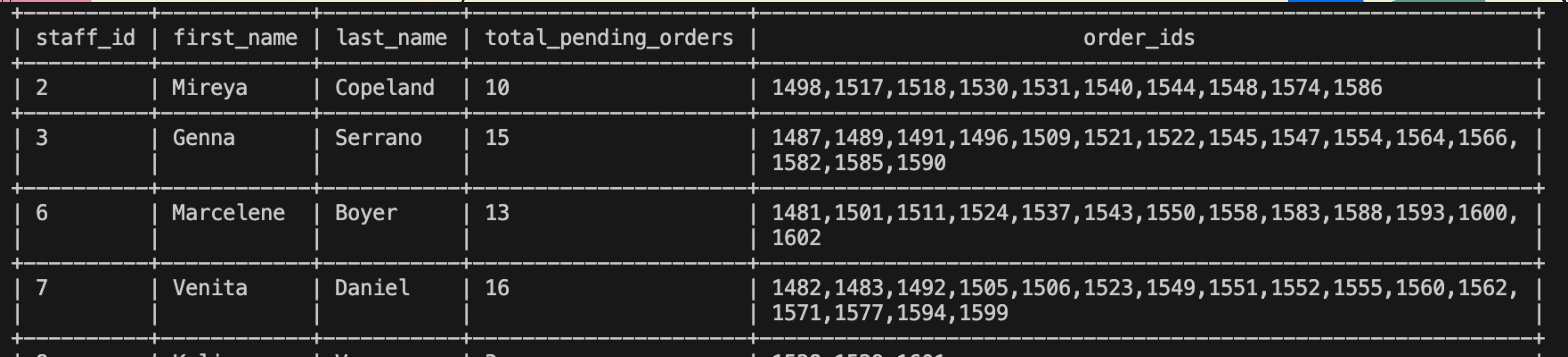
SOME QUERIES



A normal day at work for bike store's owner

I want to check the pending work of each staff and reminding them to complete it.

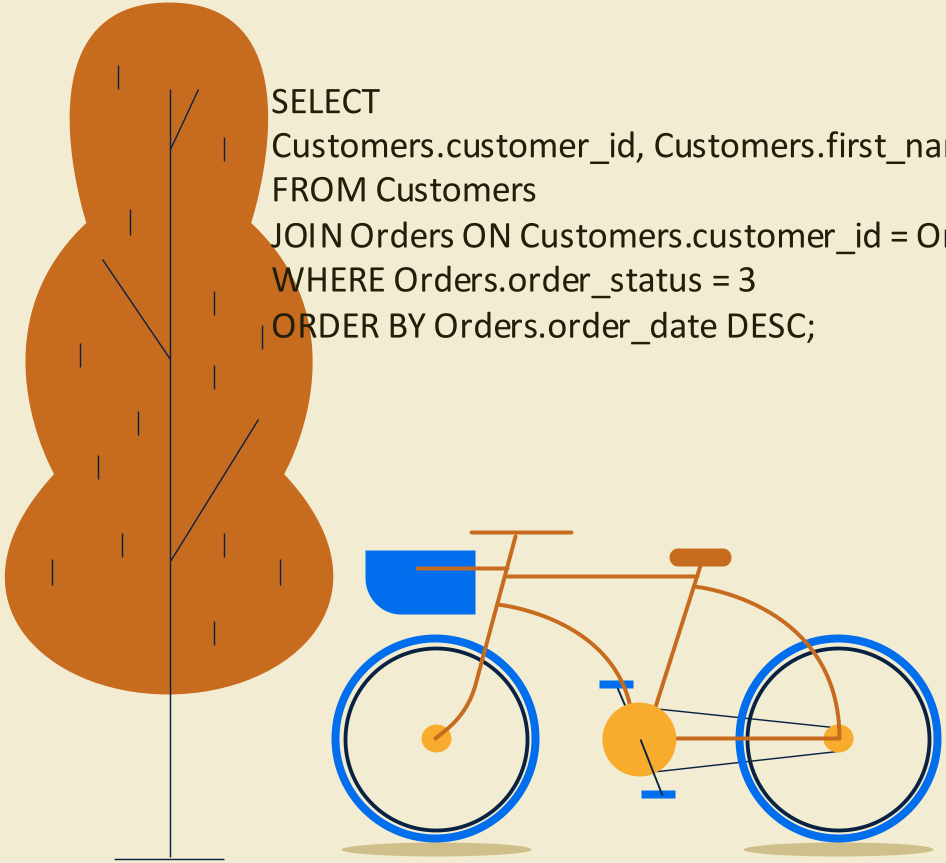
```
SELECT
Staffs.staff_id, Staffs.first_name, Staffs.last_name,
COUNT(Orders.order_id) AS total_pending_orders,
GROUP_CONCAT(Orders.order_id) AS order_ids
FROM Staffs
JOIN Orders ON Staffs.staff_id = Orders.staff_id
WHERE Orders.order_status = 1
GROUP BY Staffs.staff_id;
```

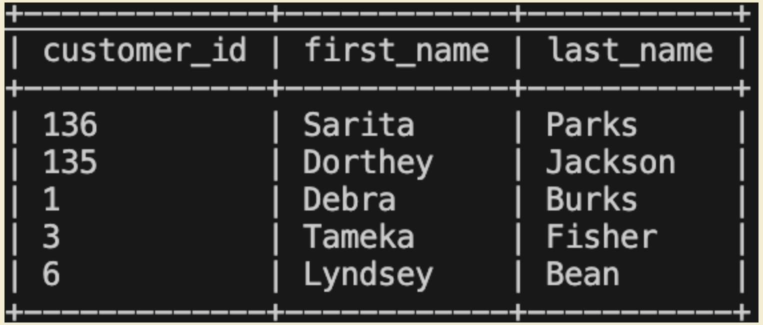


```
+----------+------------+-----------+----------------------+----------------------------------------------------------------------------+
| staff_id | first_name | last_name | total_pending_orders |                                 order_ids                                  |
+----------+------------+-----------+----------------------+----------------------------------------------------------------------------+
| 2        | Mireya     | Copeland  | 10                   | 1498,1517,1518,1530,1531,1540,1544,1548,1574,1586                          |
+----------+------------+-----------+----------------------+----------------------------------------------------------------------------+
| 3        | Genna      | Serrano   | 15                   | 1487,1489,1491,1496,1509,1521,1522,1545,1547,1554,1564,1566,               |
|          |            |           |                      | 1582,1585,1590                                                             |
+----------+------------+-----------+----------------------+----------------------------------------------------------------------------+
| 6        | Marcelene  | Boyer     | 13                   | 1481,1501,1511,1524,1537,1543,1550,1558,1583,1588,1593,1600,               |
|          |            |           |                      | 1602                                                                       |
+----------+------------+-----------+----------------------+----------------------------------------------------------------------------+
| 7        | Venita     | Daniel    | 16                   | 1482,1483,1492,1505,1506,1523,1549,1551,1552,1555,1560,1562,               |
|          |            |           |                      | 1571,1577,1594,1599                                                        |
+----------+------------+-----------+----------------------+----------------------------------------------------------------------------+
```

I want to send sorry mails to customers whose orders are rejected

```
SELECT
Customers.customer_id, Customers.first_name, Customers.last_name
FROM Customers
JOIN Orders ON Customers.customer_id = Orders.customer_id
WHERE Orders.order_status = 3
ORDER BY Orders.order_date DESC;
```

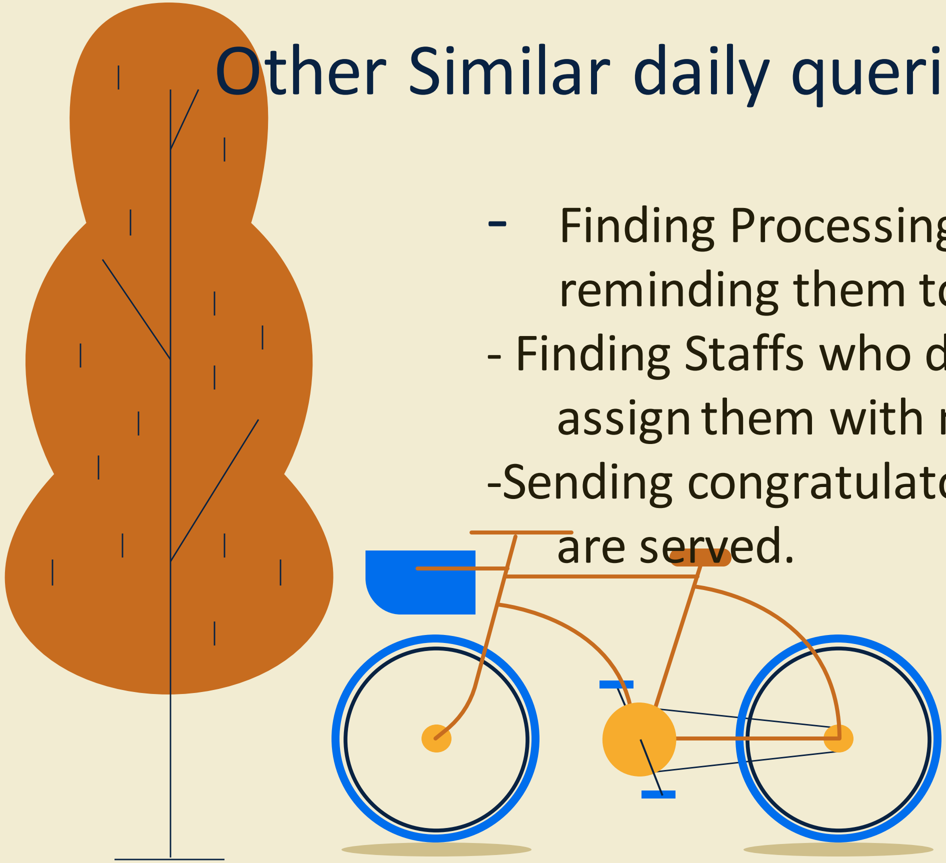




```
+-------------+------------+-----------+
| customer_id | first_name | last_name |
+-------------+------------+-----------+
| 136         | Sarita     | Parks     |
| 135         | Dorthey    | Jackson   |
| 1           | Debra      | Burks     |
| 3           | Tameka     | Fisher    |
| 6           | Lyndsey    | Bean      |
+-------------+------------+-----------+
```

Other Similar daily queries with this database

- Finding Processing work of each staff and reminding them to complete it.
- Finding Staffs who does not have assigned work and assign them with new work
- Sending congratulatory mail to customers whose orders are served.

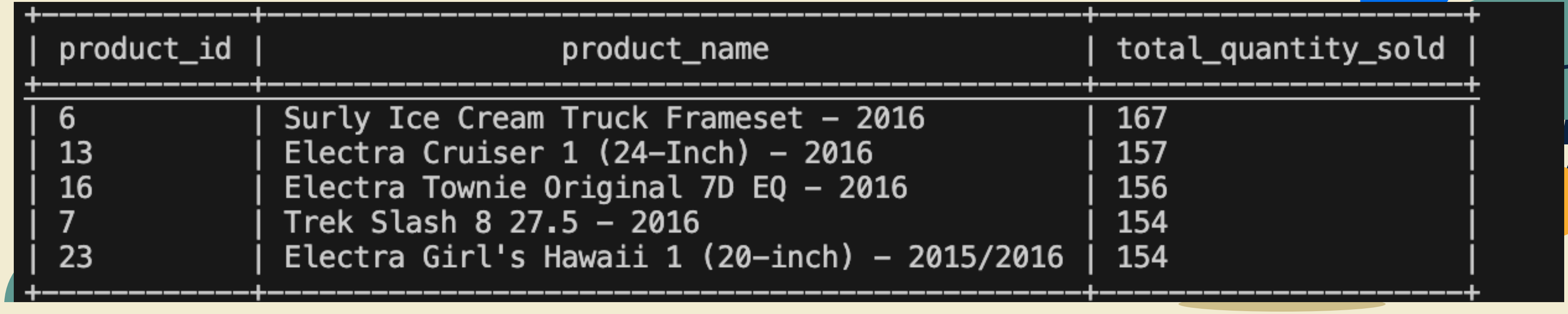


VIEWS

Frequent data analysis for the seller done through views

I want to keep track of top performing products by sales

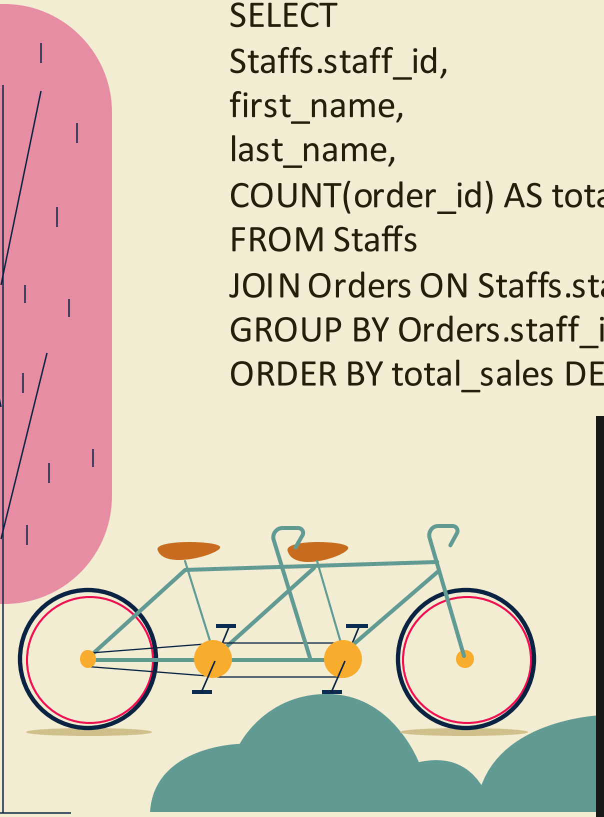
```
CREATE VIEW ProductSalesView AS
SELECT Products.product_id, product_name, SUM(quantity)  AS total_quantity_sold
FROM Order_Items
JOIN Products ON Order_Items.product_id = Products.product_id
GROUP BY Order_Items.product_id, product_name
ORDER BY total_quantity_sold DESC;
```

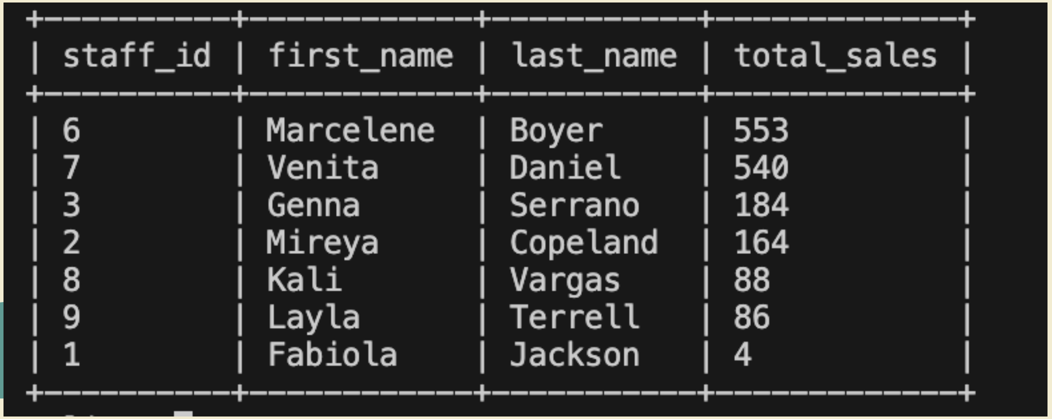


```
+------------+----------------------------------------------+---------------------+
| product_id |                 product_name                 | total_quantity_sold |
+------------+----------------------------------------------+---------------------+
| 6          | Surly Ice Cream Truck Frameset - 2016        | 167                 |
| 13         | Electra Cruiser 1 (24-Inch) - 2016           | 157                 |
| 16         | Electra Townie Original 7D EQ - 2016         | 156                 |
| 7          | Trek Slash 8 27.5 - 2016                     | 154                 |
| 23         | Electra Girl's Hawaii 1 (20-inch) - 2015/2016 | 154                 |
+------------+----------------------------------------------+---------------------+
```

I as a seller want to see the top performing staff by sales

```
CREATE VIEW StaffSalesView AS
SELECT
Staffs.staff_id,
first_name,
last_name,
COUNT(order_id) AS total_sales
FROM Staffs
JOIN Orders ON Staffs.staff_id = Orders.staff_id
GROUP BY Orders.staff_id
ORDER BY total_sales DESC;
```





```
+----------+------------+-----------+-------------+
| staff_id | first_name | last_name | total_sales |
+----------+------------+-----------+-------------+
| 6        | Marcelene  | Boyer     | 553         |
| 7        | Venita     | Daniel    | 540         |
| 3        | Genna      | Serrano   | 184         |
| 2        | Mireya     | Copeland  | 164         |
| 8        | Kali       | Vargas    | 88          |
| 9        | Layla      | Terrell   | 86          |
| 1        | Fabiola    | Jackson   | 4           |
+----------+------------+-----------+-------------+
```

Other views for data analysis

-- Average available stock of products in three stores

-- Top performing store by sales

-- Top performing category by sales

-- Top performing brand by sales

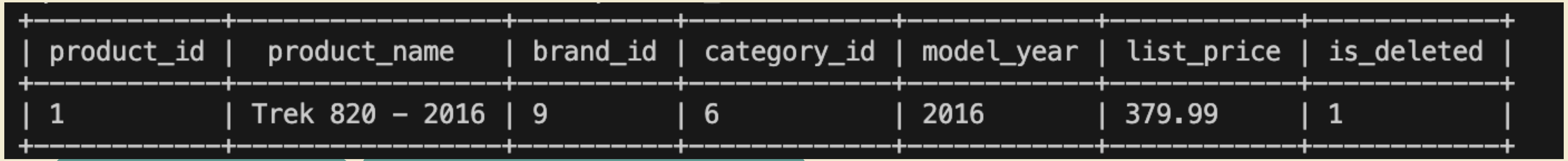
-- Top performing customers by spending

--View for knowing the active products

TRIGGERS

Seller receives frequent complaints about product so he decides to remove the product temporarily (soft deletion)

```
CREATE TRIGGER soft_delete_product
INSTEAD OF DELETE ON Active_Products
FOR EACH ROW
BEGIN
    UPDATE Products
    SET is_deleted = 1
    WHERE product_id = OLD.product_id;
END;
```



```
+------------+------------------+----------+-------------+------------+------------+------------+
| product_id |   product_name   | brand_id | category_id | model_year | list_price | is_deleted |
+------------+------------------+----------+-------------+------------+------------+------------+
| 1          | Trek 820 - 2016  | 9        | 6           | 2016       | 379.99     | 1          |
+------------+------------------+----------+-------------+------------+------------+------------+
```

Similarly, we have got trigger for soft insertion

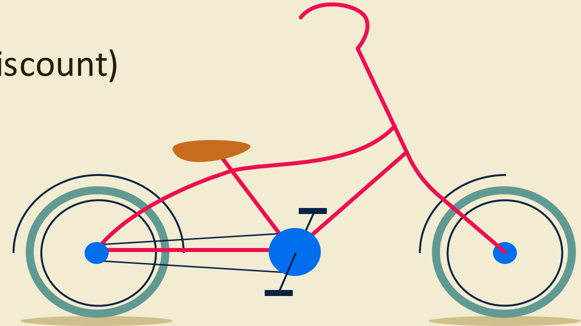
Trigger to Automatically update stock after sale

```
CREATE TRIGGER reduce_stock_after_order
AFTER INSERT ON Order_Items
FOR EACH ROW
BEGIN
    UPDATE Stocks
    SET quantity = quantity - NEW.quantity
    WHERE product_id = NEW.product_id AND store_id = (SELECT store_id
FROM Orders WHERE order_id = NEW.order_id);
END;
```

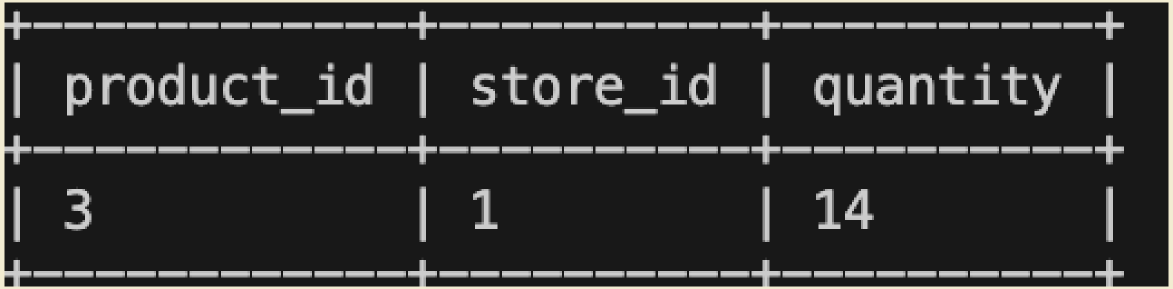
New Inersertion in order and order_item

```
INSERT INTO Orders (order_id, customer_id, order_status, order_date, required_date, shipped_date,
store_id, staff_id)
VALUES (1623, 1, 2, '2024-03-13', '2024-03-20', null, 1, 1);

INSERT INTO Order_Items (order_id, item_id, product_id, quantity, list_price, discount)
VALUES (1623, 1, 3, 6, 100.00, 0.00);
```

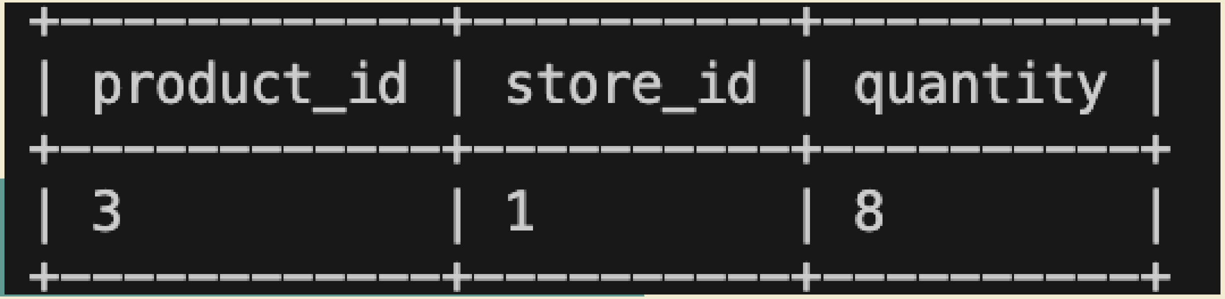


Stock before insertion



```
+------------+----------+----------+
| product_id | store_id | quantity |
+------------+----------+----------+
| 3          | 1        | 14       |
+------------+----------+----------+
```

Automatically updating stock table after insertion of order



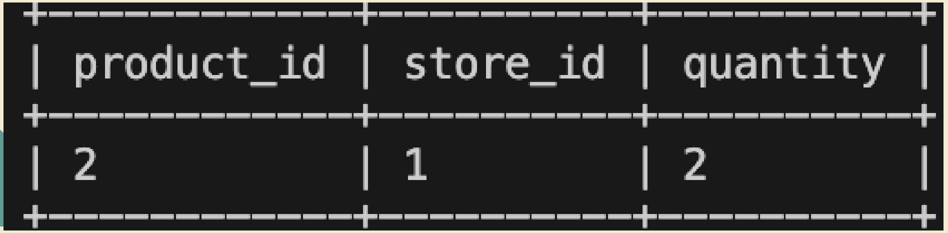
```
+------------+----------+----------+
| product_id | store_id | quantity |
+------------+----------+----------+
| 3          | 1        | 8        |
+------------+----------+----------+
```

TRANSACTIONS

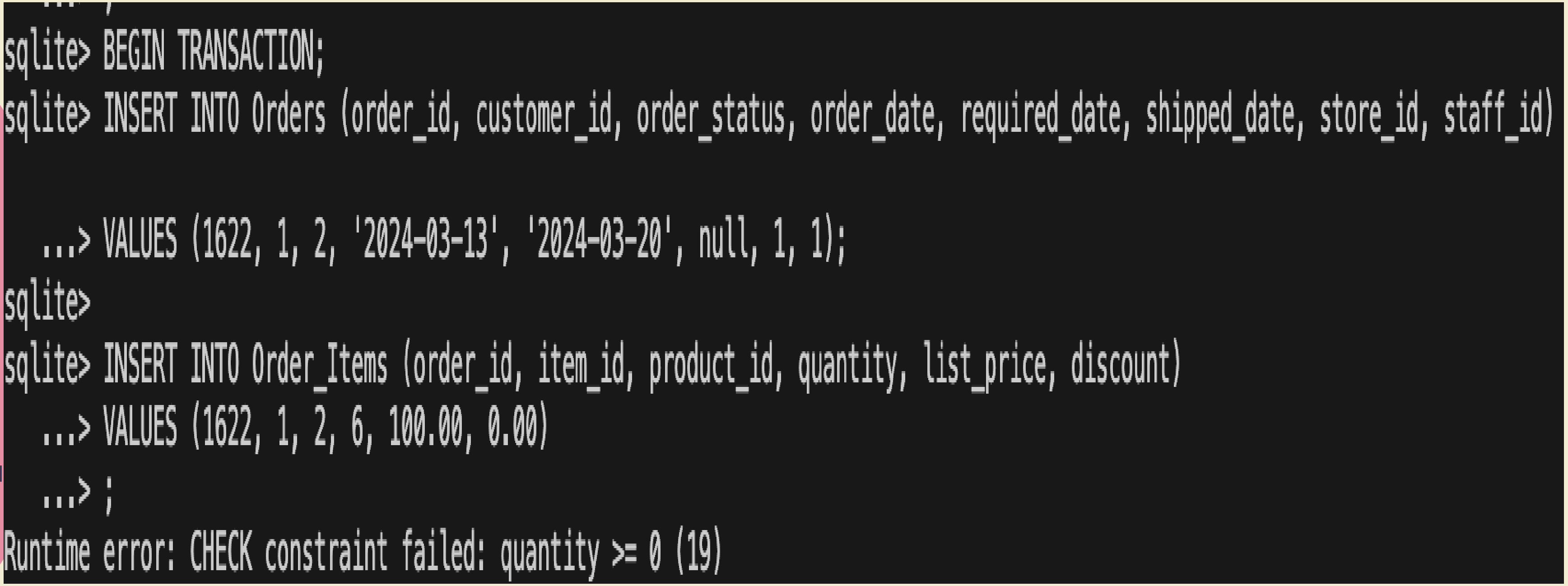
Option for roll-back in case of insufficient stock after order placement

```
BEGIN TRANSACTION;
INSERT INTO Orders (order_id, customer_id, order_status, order_date,
required_date, shipped_date, store_id, staff_id)
VALUES (1622, 1, 2, '2024-03-13', '2024-03-20', null,  1, 1);

INSERT INTO Order_Items (order_id, item_id, product_id, quantity,
list_price, discount)
VALUES (1622, 1, 2, 6, 100.00, 0.00);
```



```
+------------+----------+----------+
| product_id | store_id | quantity |
+------------+----------+----------+
| 2          | 1        | 2        |
+------------+----------+----------+
```

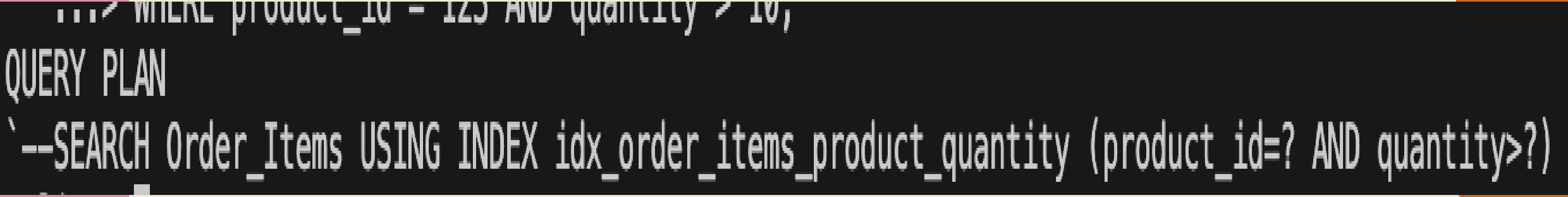


```
sqlite> BEGIN TRANSACTION;
sqlite> INSERT INTO Orders (order_id, customer_id, order_status, order_date, required_date, shipped_date, store_id, staff_id)


   ...> VALUES (1622, 1, 2, '2024-03-13', '2024-03-20', null, 1, 1);
sqlite>
sqlite> INSERT INTO Order_Items (order_id, item_id, product_id, quantity, list_price, discount)
   ...> VALUES (1622, 1, 2, 6, 100.00, 0.00)
   ...> ;
Runtime error: CHECK constraint failed: quantity >= 0 (19)
```

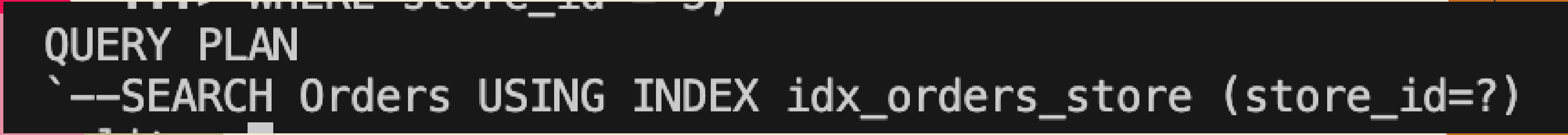
INDEXES

```
CREATE INDEX idx_order_items_products_brands
ON Order_Items(product_id, quantity);
```



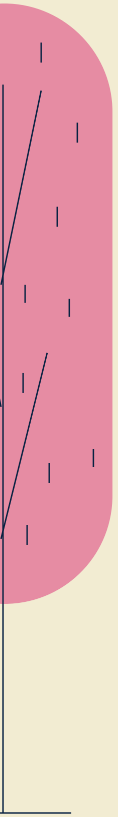
```
   ...> WHERE product_id = 123 AND quantity > 10;
QUERY PLAN
`--SEARCH Order_Items USING INDEX idx_order_items_product_quantity (product_id=? AND quantity>?)
```

CREATE INDEX idx_orders_store ON Orders (store_id)



```
   ...> WHERE store_id = 5);
QUERY PLAN
`--SEARCH Orders USING INDEX idx_orders_store (store_id=?)
```

LIMITATIONS



Our database is currently designed to cater to one owner and all his stores.

As the volume of data grows (customers, orders, products), the current database design may experience performance issues, as we have only created index for two tables as per current size of the dataset.

THANKS!