

An abstract digital background featuring a grid of binary code (0s and 1s) in white and red. Overlaid on this are various mathematical formulas and symbols in white and red, including $1+x+y+2a+21$, $\lim_{h \rightarrow 0} \frac{1}{h}$, $x=0 \times n$, $(1+x+y+2a+21) \times (g+x)$, $x+y+2a+21$, $2+3y+2a$, $2a+21$, and $2a+21$. There are also geometric shapes like circles and lines, some in red and some in white, scattered across the image. The overall color scheme is dark blue and black with red and white highlights.



A retail company dealing with various products across multiple stores



Key focus areas: customer behavior, staff performance, inventory management, and store operations



Project Objective: Utilize SQL to derive insights from the given dataset

OVERVIEW OF JENSON USA

MySQL80 x

View Query Database Server Tools Scripting Help

brands categories customers order_items orders products staffs stocks stores

Limit to 1000 rows

```
1 -- Find the total number of products sold by each
2 -- store along with the store name.
3 • select * from jenkins;
4 • select stores.store_name,
5   sum(order_items.quantity) total_products_sold
6   from stores join orders
7   on stores.store_id = orders.store_id
8   join order_items
9   on order_items.order_id = orders.order_id
10  group by stores.store_name;
```

Result Grid

store_name	total_products_sold
Santa Cruz Bikes	1516
Baldwin Bikes	4779
Rowlett Bikes	783

QUERY 1: TOTAL NUMBER OF PRODUCTS SOLD BY STORE

- **Insight:** Identifies top-performing stores based on sales volume

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: brands categories customers order_items orders products staffs stocks stores SQL File 3* SQL File 12* SQL File 13* x

SCHEMAS

Filter objects

Tables

- brands
- categories
- customers
- order_items
- orders
- products
- staffs
- stocks
- stores
- Views

Administration Schemas

Information

Schema: jenkins

```
4 with a as (select products.product_name,  
5 orders.order_date,  
6 sum(order_items.quantity) total_orders  
7 from products join order_items  
8 on products.product_id = order_items.product_id  
9 join orders  
10 on orders.order_id = order_items.order_id  
11 group by products.product_name,  
12 orders.order_date)  
13  
14 select *, sum(total_orders)  
15 over(partition by product_name order by order_date)  
16 from a;
```

Result Grid

Filter Rows: Export: Wrap Cell Content: Fetch rows:

product_name	order_date	total_orders	sum(total_orders) over(partition by product_name order by order_date)
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

Object Info Session Result 1 x Read Only

► **Insight:** Helps in inventory demand forecasting

QUERY 2: CUMULATIVE QUANTITY SOLD PER PRODUCT OVER TIME

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

brands categories customers order_items orders products staffs stocks stores SQL File 3* SQL File 12* SQL File 13* SQL File 15* x SQL File 16

Limit to 1000 rows

```
3 with a as (select categories.category_name,
4 products.product_name,
5     sum(order_items.quantity * order_items.list_price) as total_sales
6     from order_items join products
7     on order_items.product_id = products.product_id
8     join categories
9     on categories.category_id = products.category_id
10    group by categories.category_name,
11    products.product_name)
12 select *, rank() over(partition by category_name order by total_sales desc)
13 from a;
14
15
```

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

category_name	product_name	total_sales	rank() over(partition by category_name order by total_sales desc)
Children Bicycles	Electra Girl's Hawaii 1 (20-inch) - 2015/2016	4619846.00	1
Children Bicycles	Electra Girl's Hawaii 1 (16-inch) - 2015/2016	3914855.00	2
Children Bicycles	Electra Cruiser 1 (24-Inch) - 2016	3752861.00	3
Children Bicycles	Electra Townie 3i EQ (20-inch) - Boys' - 2017	1910961.00	4
Children Bicycles	Electra Girl's Hawaii 1 16" - 2017	1409953.00	5
Children Bicycles	Trek Precaliber 24 (21-Speed) - Girls - 2017	1364961.00	6
Children Bicycles	Electra Townie 7D (20-inch) - Boys' - 2017	1359960.00	7
Children Bicycles	Electra Savannah 3i (20-inch) - Girls - 2017	1154967.00	8
Children Bicycles	Electra Moto 3i (20-inch) - Boy's - 2017	1084969.00	9
Children Bicycles	Electra Moto 3i (20-inch) - Girl's - 2017	1084969.00	10

Result 2 x

Read Only

► **Insight:** Determines best-selling products in each category

QUERY 3: PRODUCT WITH HIGHEST SALES PER CATEGORY

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

brands categories customers order_items orders products staffs stocks stores SQL File 3* SQL File 12* SQL File 13* SQL File 15* SQL File 16* x

Limit to 1000 rows

```
1 -- Find the customer who spent the most money on orders.--
2
3 select orders.customer_id,
4 sum(order_items.quantity*order_items.list_price) as Sales
5 from orders join order_items
6 on orders.order_id = order_items.order_id
7 group by orders.customer_id
8 order by sales desc
9 limit 1;
```

Result Grid

customer_id	Sales
10	3780184.00

Result 1 x Read Only

► **Insight:** Identifies the most valuable customer.

QUERY 4: CUSTOMER WITH MAXIMUM SPENDING

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

brands categories customers order_items orders products staffs stocks st

Limit to 1000 rows

```
1 -- Find the highest-priced product for each category name.--
2 • select categories.category_name,
3   products.product_name, products.list_price
4   from products JOIN categories on products.category_id = categories.category_id
5   where products.list_price = (
6     select max(list_price)
7     from products
8     where category_id = products.category_id);
9
```

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

	category_name	product_name	list_price
▶	Road Bikes	Trek Domane SLR 9 Disc - 2018	1199999.00

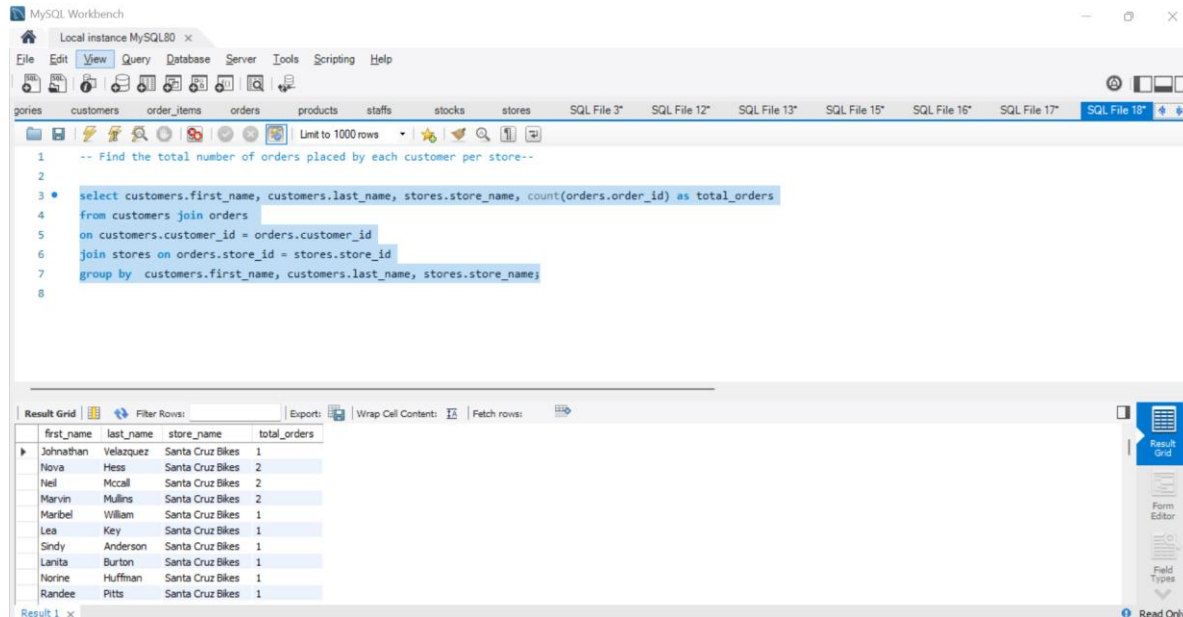
Result 1

► **Insight:** Useful for pricing strategies.

QUERY 5: HIGHEST PRICED PRODUCT IN EACH CATEGORY

QUERY 6: ORDERS PER CUSTOMER PER STORE

- **Insight:** Understands customer shopping patterns.



The screenshot displays the MySQL Workbench interface. The top toolbar includes icons for file operations, query execution, and database management. The main editor window contains the following SQL query:

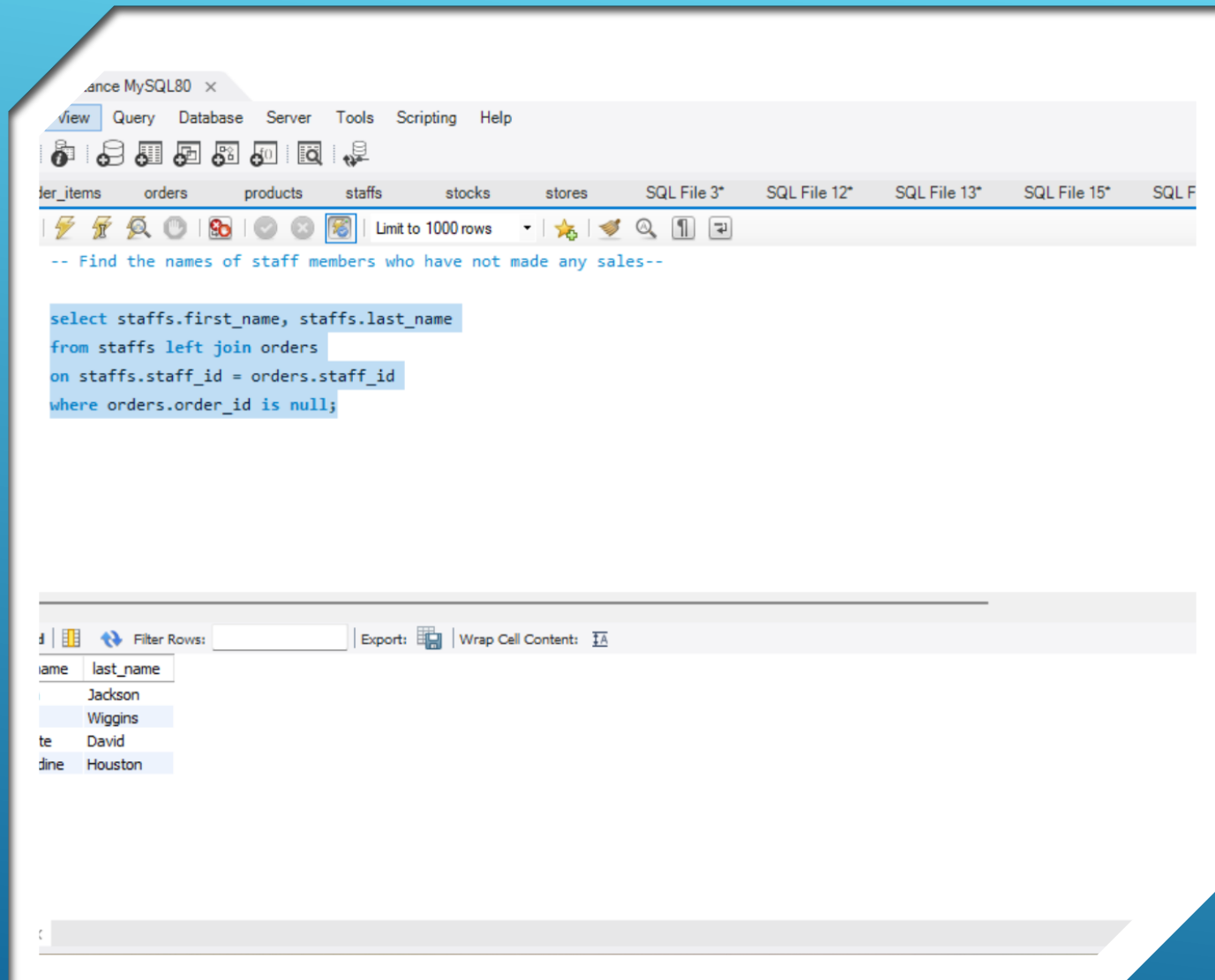
```
-- Find the total number of orders placed by each customer per store--  
select customers.first_name, customers.last_name, stores.store_name, count(orders.order_id) as total_orders  
from customers join orders  
on customers.customer_id = orders.customer_id  
join stores on orders.store_id = stores.store_id  
group by customers.first_name, customers.last_name, stores.store_name;
```

Below the query editor, the 'Result Grid' tab is active, showing the results of the query. The results are displayed in a table with the following columns: first_name, last_name, store_name, and total_orders. The data is as follows:

first_name	last_name	store_name	total_orders
Johnathan	Velazquez	Santa Cruz Bikes	1
Nova	Hess	Santa Cruz Bikes	2
Neil	Mccall	Santa Cruz Bikes	2
Marvin	Mullins	Santa Cruz Bikes	2
Maribel	William	Santa Cruz Bikes	1
Lea	Key	Santa Cruz Bikes	1
Sindy	Anderson	Santa Cruz Bikes	1
Lanita	Burton	Santa Cruz Bikes	1
Norine	Huffman	Santa Cruz Bikes	1
Randee	Pitts	Santa Cruz Bikes	1

QUERY 7: STAFF WITH NO SALES

- **Insight:** Helps in staff performance evaluations.

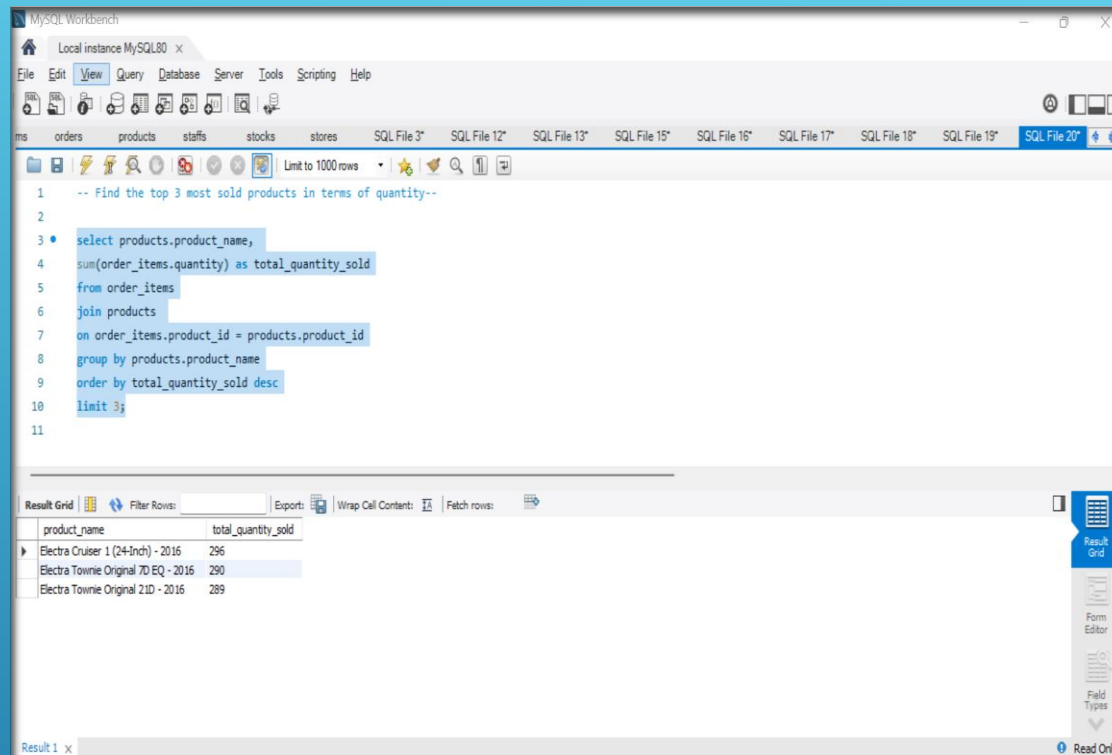


The screenshot shows the MySQL Workbench interface. The top menu bar includes View, Query, Database, Server, Tools, Scripting, and Help. Below the menu is a toolbar with various icons. The main window displays a SQL query in the script editor:

```
-- Find the names of staff members who have not made any sales--  
  
select staffs.first_name, staffs.last_name  
from staffs left join orders  
on staffs.staff_id = orders.staff_id  
where orders.order_id is null;
```

Below the query editor, the results pane shows a table with two columns: first_name and last_name. The table contains four rows of data:

first_name	last_name
Jackson	
Wiggins	
David	
Houston	



The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
-- Find the top 3 most sold products in terms of quantity--  
  
select products.product_name,  
sum(order_items.quantity) as total_quantity_sold  
from order_items  
join products  
on order_items.product_id = products.product_id  
group by products.product_name  
order by total_quantity_sold desc  
limit 3;
```

The results are displayed in the Result Grid below the query editor:

product_name	total_quantity_sold
Electra Cruiser 1 (24-Inch) - 2016	296
Electra Townie Original 7D EQ - 2016	290
Electra Townie Original 21D - 2016	289

► **Insight:** Determines the most in-demand products.

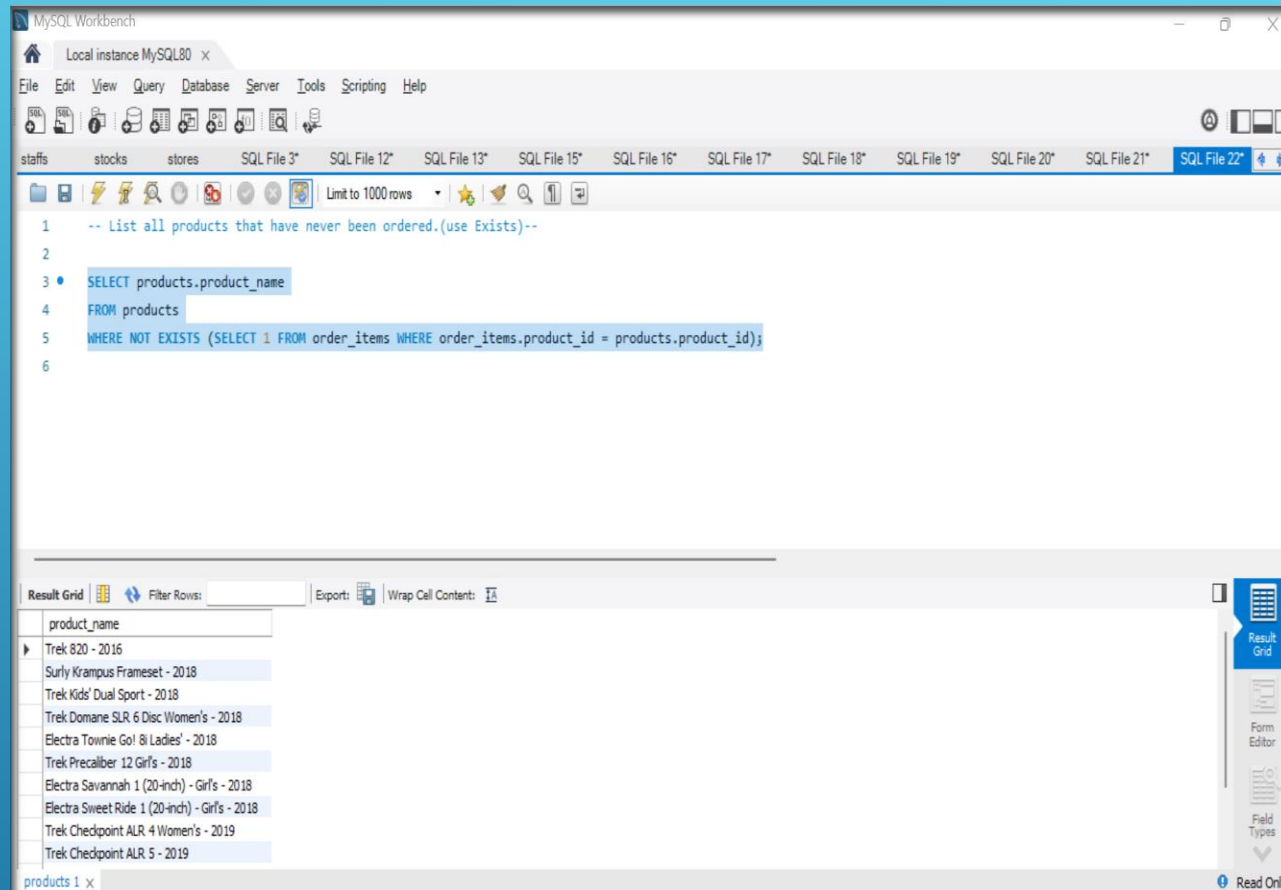
QUERY 8: TOP 3 MOST SOLD PRODUCTS BY QUANTITY

```
1 -- Find the median value of the price list--
2
3 SELECT AVG(list_price) AS median_price
4 FROM (
5     SELECT list_price,
6            ROW_NUMBER() OVER (ORDER BY list_price) AS row_num,
7            COUNT(*) OVER () AS total_rows
8     FROM products
9 ) AS ranked_prices
10 WHERE row_num IN (FLOOR((total_rows + 1) / 2), CEIL((total_rows + 1) / 2));
11
12
```

median_price
74999.000000

- **Insight:** Helps in pricing strategy decisions.

QUERY 9: MEDIAN PRICE OF PRODUCTS



► **Insight:** Helps in stock optimization.

QUERY 10: PRODUCTS NEVER ORDERED

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

stores SQL File 3* SQL File 12* SQL File 13* SQL File 15* SQL File 16* SQL File 17* SQL File 18* SQL File 19* SQL File 20* SQL File 21* SQL File 22* SQL File 23* x

Limit to 1000 rows

```
1 -- List the names of staff members who have made more sales than the average number of sales by all staff members--
2
3 WITH sales_per_staff AS (
4     SELECT orders.staff_id, COUNT(orders.order_id) AS total_sales
5     FROM orders
6     GROUP BY orders.staff_id
7 ),
8 average_sales AS (
9     SELECT AVG(total_sales) AS avg_sales FROM sales_per_staff
10 )
11 SELECT staffs.first_name, staffs.last_name
12 FROM staffs
13 JOIN sales_per_staff ON staffs.staff_id = sales_per_staff.staff_id
14 WHERE sales_per_staff.total_sales > (SELECT avg_sales FROM average_sales);
15
```

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

first_name	last_name
Marcelene	Boyer
Venita	Daniel

Result 1 x Read Only

- **Insight:** Identifies high-performing employees.

QUERY 11: STAFF MEMBERS WITH ABOVE-AVERAGE SALES

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

SQL File 12* SQL File 13* SQL File 15* SQL File 16* SQL File 17* SQL File 18* SQL File 19* SQL File 20* SQL File 21* SQL File 22* SQL File 23* SQL File 24* x SQL File 25

Limit to 1000 rows

```
1 -- Identify the customers who have ordered all types of products (i.e., from every category)--
2
3 • SELECT customers.first_name, customers.last_name
4 FROM customers
5 JOIN orders ON customers.customer_id = orders.customer_id
6 JOIN order_items ON orders.order_id = order_items.order_id
7 JOIN products ON order_items.product_id = products.product_id
8 JOIN categories ON products.category_id = categories.category_id
9 GROUP BY customers.customer_id, customers.first_name, customers.last_name
10 HAVING COUNT(DISTINCT categories.category_id) = (SELECT COUNT(*) FROM categories);
11
```

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

first_name	last_name
Genoveva	Baldwin

Result 1 x | Read Only

► **Insight:**
Identifies
highly
engaged
customers.

QUERY 12: CUSTOMERS ORDERING FROM ALL CATEGORIES

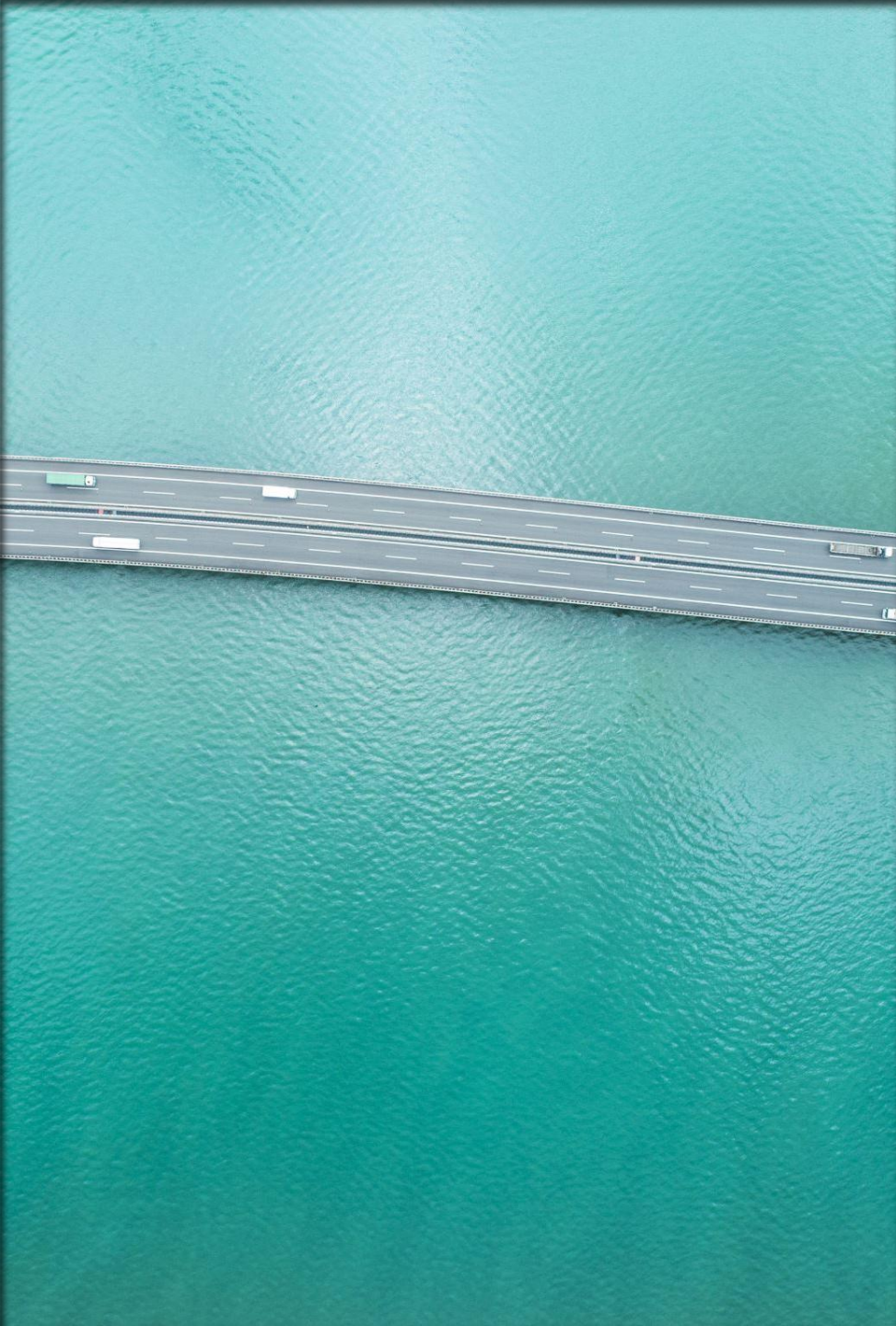
Key Findings:

- Best-performing stores and products identified.
- Top customers and high-revenue items determined.
- Underperforming staff and low-demand products analyzed.

Business Recommendations:

- Improve inventory and demand forecasting.
- Implement targeted marketing strategies for high-value customers.
- Optimize staff performance through sales-based incentives.
- Reevaluate pricing and stocking strategies for slow-moving products.

CONCLUSION & RECOMMENDATIONS



THANK YOU!

