

# Jenson USA

-by Shivani Nagar

**MYSQL | DATA ANALYSIS**





INSPIRING  
PEOPLE TO **RIDE,**  
**EXPERIENCE,**  
AND **EXPLORE.**


**MISSION :** Jenson USA is an e-commerce cycling retail business. Our purpose is to inspire people to Ride, Experience, and Explore.

We bring our culture to life through our values.

We are focused on a vision to “grow” beyond our limitations.

**MYSQL | DATA ANALYSIS**





# Table of content

01

Problem  
Statement

03

SQL query

05

Output

07

Conclusion

02

Contact

[Link to uncleaned dataset](#)



-- Find the total number of products sold by each store along with the store name.

```
-- 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;
```

Result Grid			Filter Rows:
	store_name	sum(order_items.quantity)	
▶	Santa Cruz Bikes	1516	
	Baldwin Bikes	4779	
	Rowlett Bikes	783	

-- Calculate the cumulative sum of quantities sold for each product over time.

```
-- Calculate the cumulative sum of quantities sold for each product over time.
```

```
SELECT product_id,order_date,quantity,  
sum(quantity) over(partition by product_id order by order_date) cumsum  
FROM
```

```
(SELECT products.product_id, orders.order_date, sum(order_items.quantity) quantity  
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_id, orders.order_date) temp;
```

Result Grid				
Filter Rows:				
	product_id	order_date	quantity	cumsum
▶	2	2016-01-03	2	2
	2	2016-01-14	2	4
	2	2016-01-18	1	5
	2	2016-02-05	1	6
	2	2016-02-09	1	7
	2	2016-02-26	1	8
	2	2016-02-28	2	10
	2	2016-03-08	1	11
	2	2016-03-14	2	13
	2	2016-03-20	4	17


-- Find the product with the highest total sales (quantity \* price) for each category.

```
with a as(SELECT category_name, product_id, product_name, Highest_sales,  
rank() over(partition by category_name order by Highest_sales desc) rnk from  
(SELECT products.product_id, products.product_name, categories.category_name,  
sum(order_items.quantity * (order_items.list_price - order_items.discount)) Highest_sales  
FROM categories JOIN products  
ON categories.category_id = products.category_id  
JOIN order_items  
ON order_items.product_id = products.product_id  
GROUP BY products.product_id, products.product_name, categories.category_name) temp)  
  
SELECT * from a  
WHERE rnk = 1;
```

-- Find the customer who spent the most money on orders.

```
with a as(SELECT concat(customers.first_name," ",customers.last_name) Cust_Name,  
orders.order_id,  
sum(order_items.quantity * (order_items.list_price - order_items.discount)) sales  
FROM orders join customers  
ON orders.customer_id = customers.customer_id  
JOIN order_items  
ON order_items.order_id = orders.order_id  
GROUP BY concat(customers.first_name," ",customers.last_name), orders.order_id)  
  
SELECT * FROM  
(SELECT *,rank() over(order by sales desc) rnk FROM a) temp  
where rnk = 1;
```

Result Grid


Filter Rows:

Export


	Cust_Name	order_id	sales	rnk
▶	Pamelia Newman	1541	3232863.00	1



-- Find the highest-priced product for each category name.

```
with a as
  (SELECT categories.category_id, categories.category_name, products.product_name,
  products.list_price,
  rank() over(partition by categories.category_id order by products.list_price desc) rnk
  FROM categories JOIN products
  ON categories.category_id = products.category_id)

SELECT * from a
WHERE rnk = 1;
```








-- Find the total number of orders placed by each customer per store.

```
SELECT concat(customers.first_name," ",customers.last_name) Cust_Name,  
stores.store_name,stores.store_id,count(orders.order_id) Total_Orders  
FROM customers JOIN orders  
ON customers.customer_id = orders.customer_id  
JOIN stores  
ON stores.store_id = orders.store_id  
GROUP BY concat(customers.first_name," ",customers.last_name),  
stores.store_name,stores.store_id;
```



-- Find the names of staff members who have not made any sales.

```
SELECT staffs.staff_id,  
concat(staffs.first_name," ",staffs.last_name) as Fullname  
FROM staffs  
WHERE NOT EXISTS  
(SELECT * FROM orders WHERE orders.staff_id = staffs.staff_id);
```

Result Grid			Filter Rows:
	staff_id	Fullname	
▶	1	Fabiola Jackson	
	4	Virgie Wiggins	
	5	Jannette David	
	10	Bernardine Houston	

-- Find the top 3 most sold products in terms of quantity.

```
SELECT product_name,product_id FROM
(SELECT products.product_id,products.product_name, sum(order_items.quantity),
rank() over(order by sum(order_items.quantity) desc) rnk
FROM order_items JOIN products
ON products.product_id = order_items.product_id
GROUP BY products.product_id,products.product_name) temp
WHERE rnk <=3;
```

Result Grid			Filter Rows:	E
	product_name	product_id		
	Surly Ice Cream Truck Frameset - 2016	6		
	Electra Cruiser 1 (24-Inch) - 2016	13		
	Electra Townie Original 7D EQ - 2016	16		

-- Find the median value of the price list.

```
with temp as(SELECT list_price,row_number() over(order by list_price) rn,  
count(*) over() cn FROM order_items)  
  
SELECT CASE  
    WHEN cn % 2 = 0 THEN (SELECT avg(list_price) FROM temp  
    WHERE rn in (cn/2,cn/2 + 1))  
    ELSE (SELECT list_price FROM temp  
    WHERE rn = (cn+1)/2)  
END AS MEDIAN FROM temp limit 1;
```

Result Grid		Filter
	MEDIAN	
▶	59999.000000	



-- List all products that have never been ordered.  
(use Exists)

```
SELECT products.product_id,products.product_name  
FROM products WHERE NOT EXISTS  
(SELECT order_items.product_id FROM order_items  
WHERE order_items.product_id = products.product_id);
```

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

```
SELECT staffs.staff_id,  
coalesce(sum(order_items.quantity * (order_items.list_price - order_items.discount)),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  
HAVING sum(order_items.quantity * (order_items.list_price - order_items.discount)) >  
(SELECT avg(sales) FROM  
(SELECT staffs.staff_id,  
coalesce(sum(order_items.quantity * (order_items.list_price - order_items.discount)),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) temp);
```

Result Grid			Filter Rows
	staff_id	sales	
▶	3	95269072.00	
	6	293879916.00	
	7	288726544.00	

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

```
SELECT customers.customer_id
FROM customers JOIN orders
ON customers.customer_id = orders.customer_id
JOIN order_items
ON orders.order_id = order_items.order_id
JOIN products p
ON p.product_id = order_items.product_id
GROUP BY customers.customer_id
HAVING count(distinct p.category_id) = (SELECT count(categories.category_id) FROM categories);
```


Result Grid	
	customer_id
▶	9



## Conclusion: Key Insights from Jenson USA SQL Analysis

- **Store Sales Performance:** Identifies top and underperforming stores.
- **Product Demand Trends:** Tracks cumulative sales to forecast inventory needs.
- **Top-Selling Products by Category:** Highlights revenue-driving products.
- **Top-Spending Customers:** Pinpoints high-value customers for targeted engagement.
- **Pricing Strategy:** Assesses highest-priced products and median price points.
- **Customer Loyalty:** Analyzes order frequency per store.
- **Staff Performance:** Identifies top performers and those needing improvement.
- **Product Popularity:** Lists top-selling and never-ordered products.
- **Engaged Shoppers:** Recognizes customers purchasing across all categories.

These insights aid in optimizing operations, enhancing customer relations, and driving sales growth.







**Liked ?  
Connect for more..**

**Email**

**shivaninagar2nd@gmail.com**

**Linkedin :**

**[https://www.linkedin.com/in/  
shivani-nagar-928028275/](https://www.linkedin.com/in/shivani-nagar-928028275/)**



**MYSQL | DATA ANALYSIS**



**Thank You**