# Jenson USA Data analysis using MySQL


Present by: Rahul Dahal

04 September 2024


# Problem Statement:

The goal of this analysis is to derive actionable insights from Jensen's dataset, focusing on customer behavior, staff performance, inventory management, and store operations. By analyzing key metrics such as product sales, customer spending, staff contributions, and order patterns, the objective is to identify opportunities for improving operational efficiency and enhancing overall business performance. The findings from this analysis will support data-driven decision-making to optimize various aspects of the business.

JensonUSA

# Objective:

To extract actionable insights from Jensen's dataset to improve customer behavior analysis, staff performance, inventory management, and store operations.

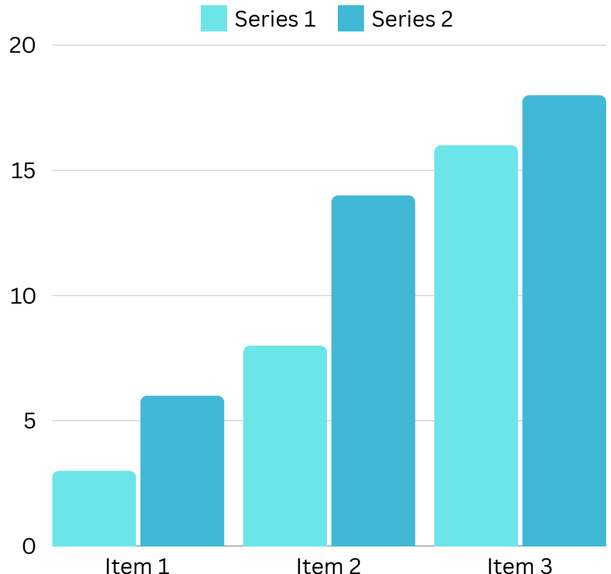
## Jenson USA

As a data analyst at Jensen's, craft SQL queries to derive insights on customer behavior, staff performance, inventory management, and store operations.

**Dataset Link:** https://drive.google.com/drive/folders/1feFkCInYME7Be3kjmz-TD2PV1uVkXNAN

**Questions**

1. Find the total number of products sold by each store along with the store name.
2. Calculate the cumulative sum of quantities sold for each product over time.
3. Find the product with the highest total sales (quantity * price) for each category.
4. Find the customer who spent the most money on orders.
5. Find the highest-priced product for each category name.
6. Find the total number of orders placed by each customer per store.
7. Find the names of staff members who have not made any sales.
8. Find the top 3 most sold products in terms of quantity.
9. Find the median value of the price list.
10. List all products that have never been ordered.(use Exists)
11. List the names of staff members who have made more sales than the average number of sales by all staff members.
12. Identify the customers who have ordered all types of products (i.e., from every category).



/04

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

Here to get the data simply join function was utilized as the store table and order_id table had no common column to retrieve data from order_id table fopr quanity ordered hence extra join was used to obtain data from orders table linked it with store table then joined with order_id table. in conclusion two join was utilized.

```sql
#Find the total number of products sold by each store along with the store name.

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

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

| store_name | total_quantity |
|---|---|
| Santa Cruz Bikes | 1516 |
| Baldwin Bikes | 4779 |
| Rowlett Bikes | 783 |

JensonUSA

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

- As the question was asking for the cumilative sum of qunatity ordered over time and product so, we had to use the product_id, order_date and quanity column from two different tables. however the desired resutlt was cumilative sum hence, the data was first gouped by on normal join by grouping it with non aggregated columns . after that to get the cumilative sum based on change in date windows function was needed to be utilized hence partition and order by was used over the selected row which was order_date and product_id.

order_items | SQL File 5 | products

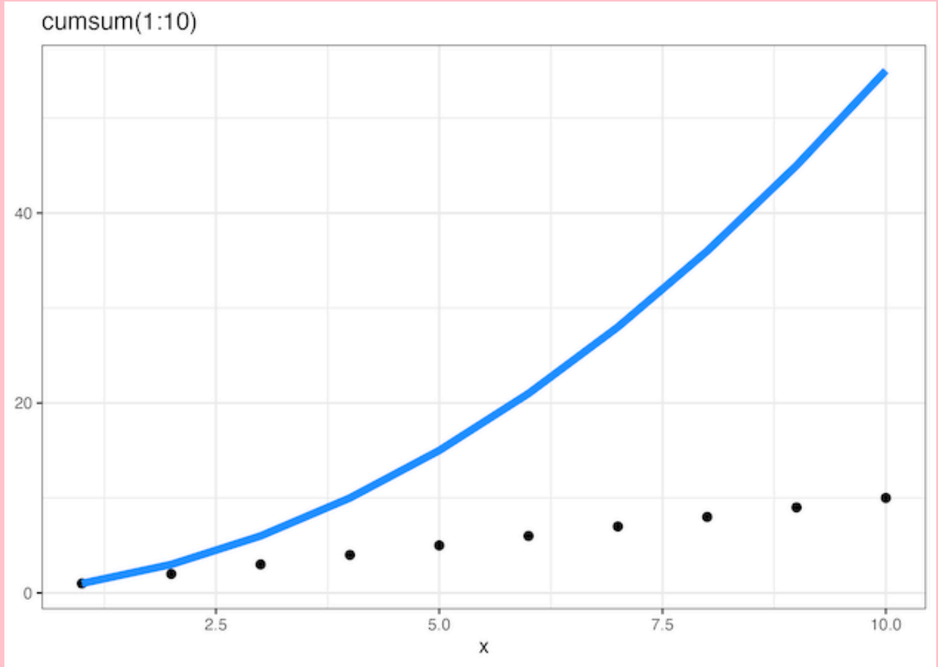
Don't Limit

```sql
#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) as cumilative_sum
from
(select order_items.product_id, orders.order_date, sum(order_items.quantity) quantity
from orders join order_items
on orders.order_id = order_items.order_id
group by orders.order_date, order_items.product_id) as a;
```

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

| product_id | order_date | quantity | cumilative_sum |
|---|---|---|---|
| 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 |




JensonUSA

/06

JensonUSA

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

Here the question was quite tricky as had to include multiple subqueries to get the data out of it. the solution could be different accroding to the person however according to me i first formed the subquery including all the products, id, category and total sales of the product then joined all different tables to retrive the datas. on second included one more subquery to rank the data in top to buttom orders olus partition and order by based on category and total sales respectively and finally main query where we obtained the data where rank is equal to 1 which was the top sales product from each category.

```
3
4   select * from
5   (select product_name,product_id,category_name, total_sales,
6   rank()over(partition by category_name order by total_sales desc)as RK from
7   (select products.product_name, products.product_id,categories.category_name,
8    sum(order_items.quantity*order_items.list_price)
9   as total_sales
10  from products join order_items
11  on products.product_id = order_items.product_id
12  join categories on categories.category_id = products.category_id
13  group by products.product_name, products.product_id,categories.category_name) as a
14  ) as b
15  where Rk = 1;
```

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

| product_name | product_id | category_name | total_sales | RK |
|---|---|---|---|---|
| Electra Girl's Hawaii 1 (20-inch) - 2015/2016 | 23 | Children Bicycles | 4619846.00 | 1 |
| Electra Townie Original 7D EQ - 2016 | 26 | Comfort Bicycles | 8039866.00 | 1 |
| Electra Townie Original 7D EQ - 2016 | 16 | Cruisers Bicycles | 9359844.00 | 1 |
| Surly Straggler 650b - 2016 | 11 | Cyclocross Bicycles | 25382949.00 | 1 |
| Trek Conduit+ - 2016 | 9 | Electric Bikes | 43499855.00 | 1 |

/07

JensonUSA

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

Here again multiple sub queries were utilized to retrive the data additioanlly with some windows functions.

as first the question was asking for the top ranked customer who spent more money. so had to find a new column as sum of order times price and quanity, once the total sales was found new query was utilized to rank the sales column based on sales from top to buttom and finally main query where the condition was to get data of top_spent (Customer)

```
2
3   select * from
4   (select *, rank() over (order by sales desc) as Top_spent
5   from
6   (Select customers.customer_id,
7   concat(customers.first_name,"",customers.last_name),
8   sum(order_items.quantity*(order_items.list_price - order_items.discount))sales
9   from customers join orders
10  on customers.customer_id = orders.customer_id
11  join order_items
12  on orders.order_id = order_items.order_id
13  group by customers.customer_id, concat(customers.first_name,customers.last_name))as a)b
14  where Top_spent = 1;
```

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

| customer_id | concat(customers.first_name,"",customers.last_na | sales | Top_spent |
|---|---|---|---|
| 10 | PameliaNewman | 3780140.00 | 1 |

/07

JensonUSA

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

Here in this section simply both of the tables were joined based on common column and list prick section was first ranked from highest to lowest partition by category name, then it was made as a subquery once its done a main query was formed with a condition where the data was retrived by asking the top values which was rank = one.

order_items | SQL File 6* | SQL File 7* | SQL File 8* | SQL File 9* | SQL File 7*

Don't Limit

```sql
# Find the highest-price product for each category name.

select * from
(select categories.category_name, products.product_name, products.list_price,
rank()over(partition by categories.category_name order by products.list_price desc) as Rnk
from categories join products
on categories.category_id = products.category_id) as a
where Rnk = 1;
```

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

| category_name | product_name | list_price | Rnk |
|---|---|---|---|
| Children Bicycles | Electra Straight 8 3i (20-inch) - Boy's - 2017 | 48999.00 | 1 |
| Children Bicycles | Electra Townie 3i EQ (20-inch) - Boys' - 2017 | 48999.00 | 1 |
| Children Bicycles | Trek Superfly 24 - 2017/2018 | 48999.00 | 1 |
| Comfort Bicycles | Electra Townie Go! 8i - 2017/2018 | 259999.00 | 1 |
| Cruisers Bicycles | Electra Townie Commute Go! - 2018 | 299999.00 | 1 |

JensonUSA

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

As here we were looking for the total orders placed by the customer based on customer name and specific store hence being three different columns joins were utilized and as the desried outcome was meant to be on the category name and price hence windows function was used where partition and order by both were utilized to get the desired outcome/

```
3 •  Select customers.customer_id,
4    concat(customers.first_name,customers.last_name) as customer_name,
5    stores.store_name,
6    count(orders.order_id)
7    over(partition by customers.customer_id order by stores.store_name)
8    as total_orders_per_customer
9    from stores join orders
10   on stores.store_id = orders.store_id
11   join customers
12   on customers.customer_id = orders.customer_id;
```

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

| customer_id | customer_name | store_name | total_orders_per_customer |
|---|---|---|---|
| 1 | DebraBurks | Baldwin Bikes | 3 |
| 1 | DebraBurks | Baldwin Bikes | 3 |
| 1 | DebraBurks | Baldwin Bikes | 3 |
| 2 | KashaTodd | Santa Cruz Bikes | 3 |
| 2 | KashaTodd | Santa Cruz Bikes | 3 |

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

To get the staff members who have not made any sales, here “exists” function was utilized. as one of the table was missing the staffs id numbers which was meant to be the employess who had no mad any sles. hence where not exists was used by providing condition of both table’s column being equals to, which gave the values of id which were not exist on orders column.

```
3 •  select staffs.staff_id,
4    concat(staffs.first_name," ", staffs.last_name)
5    from staffs
6    where not exists
7    (select orders.staff_id from orders
8    where staffs.staff_id = orders.staff_id);
9
```

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

| staff_id | concat(staffs.first_name," ", staffs.last_name) |
|---|---|
| 1 | Fabiola Jackson |
| 4 | Virgie Wiggins |
| 5 | Jannette David |
| 10 | Bernardine Houston |

JensonUSA

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

Here to get the top three product based on quantity sales, the data was first joined with the product table and then with the order_details where the quantity was mentioned with additinally summing the quanity to get the total quantity sales. the data is again ranked based on top to buttom using windows function and finally called a condition for the products  which were coming on the top 3 positions and utilized sub query with mainn query.

```
2
3 •  select * from
4    (Select products.product_id, products.product_name,
5    sum(order_items.quantity)quantity,
6    rank()over(order by sum(order_items.quantity) desc) Rnk
7     from products join order_items
8     on products.product_id = order_items.product_id
9     group by products.product_id,products.product_name) as a
10    where Rnk<=3;
```

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

| product_id | product_name | quantity | Rnk |
|---|---|---|---|
| 6 | Surly Ice Cream Truck Frameset - 2016 | 167 | 1 |
| 13 | Electra Cruiser 1 (24-Inch) - 2016 | 157 | 2 |
| 16 | Electra Townie Original 7D EQ - 2016 | 156 | 3 |


# 9. Find the median value of the price list

JensonUSA

As in the My SQL there is not a direct function to get the median value, hence a customize query was needed to get the data. Here first the required column was retrived from the table then created a additional column as row number and finally counted all the rows to get the lenth of column.
once the length was retrived then with the help of common table expression and case statement the desired value was obtained. the condition was splited based on even and odd numbers.

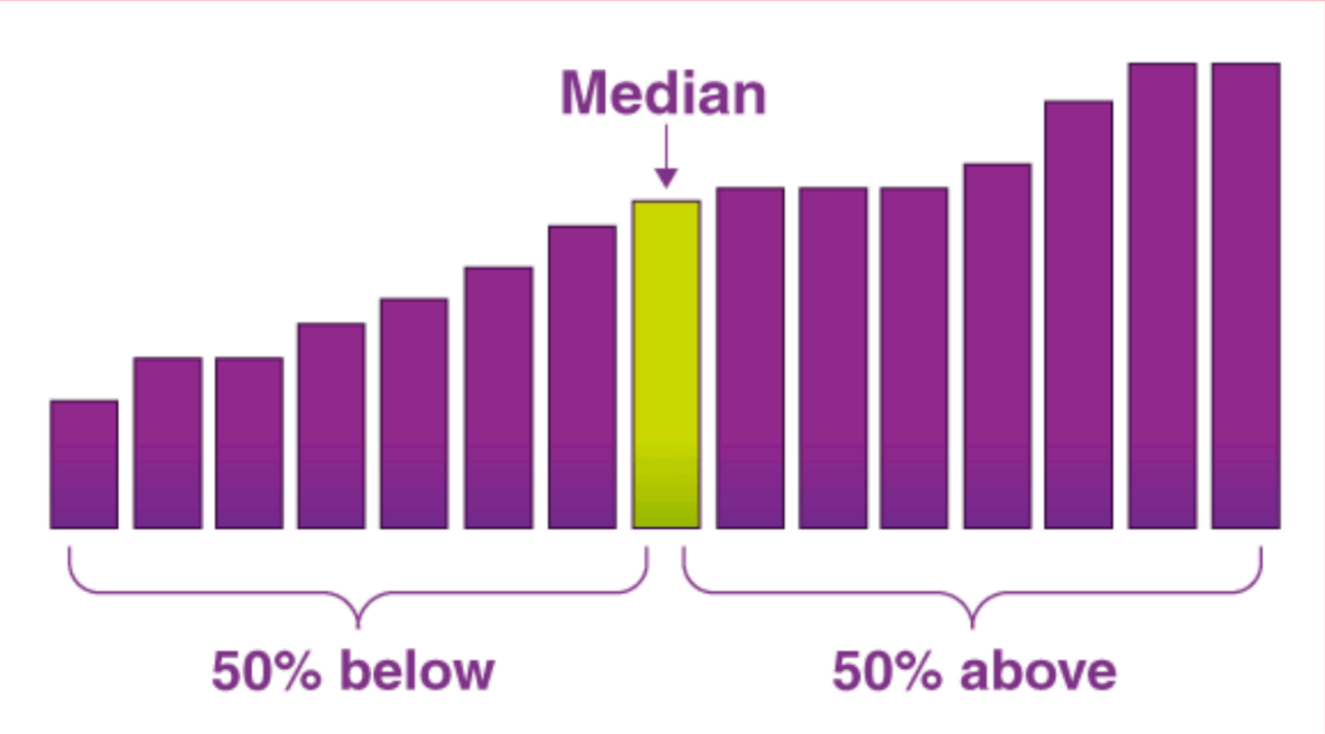
```sql
# Find the median value of the price list

with a as (select list_price,
row_number()over(order by list_price)as Rn,
count(list_price)over()as length
from order_items)

select case
when length % 2 = 0 then (select avg(list_price) from a
where rn in (length/2, (length/2)+1) )
else (select (list_price) from a
where rn = (length+1)/2)
end as median from a limit 1;
```

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

| median |
|---|
| 59999.000000 |



JensonUSA

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

Here again similar to previous query exist function is utilized to retrive the data.

```
3 •  select products.product_id, products.product_name
4    from products
5    where not exists
6    (select order_items.product_id from order_items
7    where products.product_id = order_items.product_id);
8
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

| product_id | product_name |
|---|---|
| 1 | Trek 820 - 2016 |
| 121 | Surly Krampus Frameset - 2018 |
| 125 | Trek Kids' Dual Sport - 2018 |
| 154 | Trek Domane SLR 6 Disc Women's - 2018 |
| 195 | Electra Townie Go! 8i Ladies' - 2018 |

JensonUSA

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

Use of multiple joins as of the difference on common columns and utilization of having clause as of the aggerated column- based condition

```sql
Select staffs.staff_id,
concat(staffs.first_name, staffs.last_name) as full_name,
coalesce(sum(order_items.quantity*(order_items.list_price - order_items.discount)),0) as total_sales
from staffs left join orders
on staffs.staff_id = orders.staff_id
left join order_items
on order_items.order_id = orders.order_id
group by staffs.staff_id,concat(staffs.first_name, staffs.last_name)
    having sum(order_items.quantity*(order_items.list_price - order_items.discount)) >

(select avg(total_sales)
from
(Select staffs.staff_id,
concat(staffs.first_name, staffs.last_name) as full_name,
coalesce(sum(order_items.quantity*(order_items.list_price - order_items.discount)),0) as total_sales
from staffs left join orders
on staffs.staff_id = orders.staff_id
left join order_items
on order_items.order_id = orders.order_id
group by staffs.staff_id,concat(staffs.first_name, staffs.last_name))as a);
```

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

| staff_id | full_name | total_sales |
|---|---|---|
| 3 | GennaSerrano | 95269072.00 |
| 6 | MarceleneBoyer | 293879916.00 |
| 7 | VenitaDaniel | 288726544.00 |

JensonUSA

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

As there was a condition for three different days so simply count function was used by making date distinct using having clause.

```
4
5   select customers.customer_id
6   from customers
7   join orders
8   on customers.customer_id = orders.customer_id
9   join order_items
10  on order_items.order_id = orders.order_id
11  join products
12  on products.product_id = order_items.product_id
13  group by customers.customer_id
14  having count(distinct products.category_id)=(select count(category_id)from categories);
15
```

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

| customer_id |
|---|
| 9 |

# Conclusion

**The analysis of Jensen's dataset has provided valuable insights into customer behavior, staff performance, inventory management, and store operations. By identifying top-performing products, understanding customer spending patterns, evaluating staff contributions, and optimizing inventory, the findings offer clear recommendations for enhancing operational efficiency and driving business growth. These data-driven strategies will help Jensen's make informed decisions to improve customer satisfaction, streamline processes, and increase profitability.**

# Thank You

rahuldahal2053@gmail.com

+61-413654026

Rahul Dahal

Daw park, Adelaide, 5041

August 2024

Wscube tech