JENSON USA

Using Advanced SQL

PRESENTED BY : DISHA SINDHI

About the Jenson USA

Jenson USA is a prominent online and brick-and-mortar retailer specializing in bicycles, cycling components, accessories, and gear. Established in 1994, the company has built a strong reputation in the cycling community for offering a wide range of products catering to road, mountain, and gravel cyclists, as well as casual riders.

Product Range: Jenson USA offers complete bikes, bike frames, wheels, tires, apparel, tools, and maintenance products from renowned brands such as Shimano, SRAM, Fox, Specialized, and many more.

Custom Bike Builder: One of their standout features is the ability to custom-build bikes on their website, allowing riders to personalize components to suit their preferences and needs.Price Match Guarantee: The company is known for competitive pricing, offering a price match policy to ensure customers get the best value.

INTRODUCTION: WINDOW FUNCTION

The WINDOW function in SQL is a powerful tool used for performing calculations across a set of table rows that are related to the current row. Unlike aggregate functions, which return a single result for a group of rows, window functions allow you to calculate values across rows without collapsing them into a single result.

Key Features of SQL Window Functions:

Preserve Row Details: Unlike GROUP BY, which aggregates rows into one, window functions operate on individual rows while retaining the full result set.

Partitioning: You can define a subset of rows (a "window") over which the function operates using the PARTITION BY clause.

Ordering: Rows can be ordered within each partition using the ORDER BY clause to define the sequence for calculations.

Sliding Window: Use ROWS or RANGE clauses to specify a sliding subset of rows within the partition.

QUERIES TO DERIVE INSIGHTS

01

Customer behavior

Eco-conscious customers might prefer products with sustainable materials or companies with green initiatives.

02

Staff performance

Evaluating and enhancing staff performance ensures efficiency, excellent customer service, and business growth.

03

Inventory management

Efficient inventory processes ensure customers find what they need while avoiding overstock or stockouts.

04

Store operations

e-commerce business, are the backbone of ensuring efficient workflows, excellent customer service, and profitability. Store operations cover all activities required to run a store successfully, from customer service to inventory management.

Questions

Q1.Find the total number of products sold by each store along with the store name.

Q2.Calculate the cumulative sum of quantities sold for each product over time.

Q3.Find the product with the highest total sales (quantity * price) for each category.

Q4.Find the customer who spent the most money on orders.

Q5.Find the highest-priced product for each category name.

Q6.Find the total number of orders placed by each customer per store.

Q7.Find the names of staff members who have not made any sales.

Q8.Find the top 3 most sold products in terms of quantity.

Q9.Find the median value of the price list.

Q10.List all products that have never been ordered.(use Exists)

Q11.List the names of staff members who have made more sales than the average number of sales by all staff members.

Q12.Identify the customers who have ordered all types of products (i.e., from every category).

Customer behavior

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

```sql
SELECT
    customers.customer_id
FROM
    customers
        JOIN
    orders ON customers.customer_id = orders.customer_id
        JOIN
    order_items ON order_items.order_id = orders.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
9

Find the customer who spent the most money on orders.

```sql
with a as(select customers.customer_id,
concat(customers.first_name, " ", customers.last_name) full_name,
sum(order_items.quantity*(order_items.list_price-order_items.discount)) 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.customer_id,concat(customers.first_name, customers.last_name))

Select * from
(select *, rank() over(order by sales desc) rnk
from a) b
where rnk = 1;
```

customer_id	full_name	sales	rnk
10	Pamelia Newman	3780140.00	1

Staff performance

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

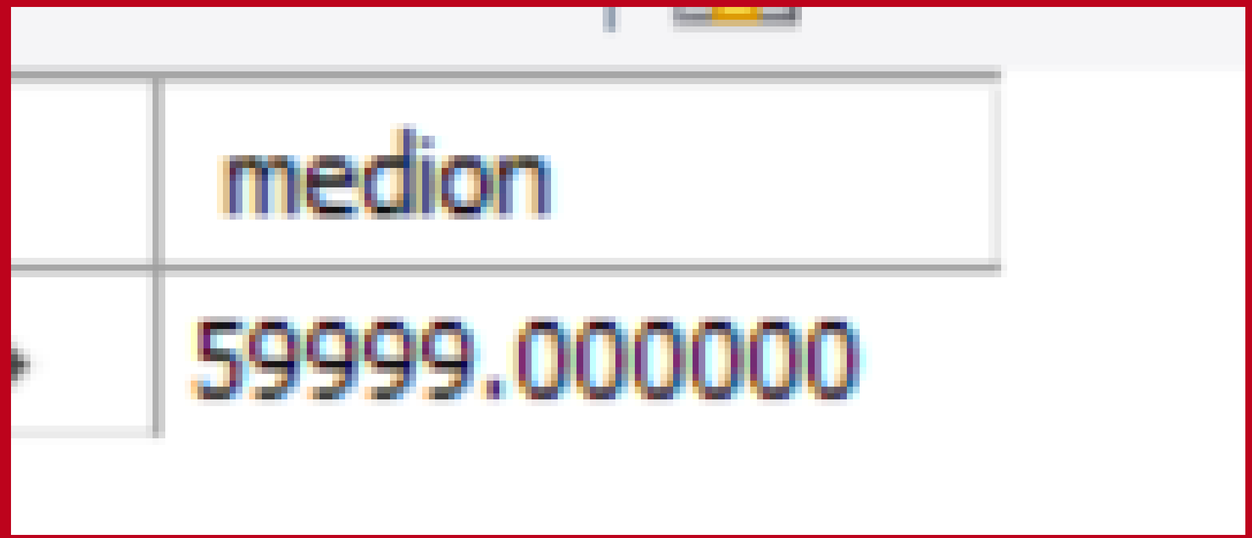
```
select * from (Select products.product_name,order_items.product_id,
sum(order_items.quantity),
rank() over(order by sum(order_items.quantity) desc)ranks
from order_items
join products
on order_items.product_id = products.product_id
group by products.product_name,order_items.product_id) sold_q
where ranks <= 3;
```

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

Find the median value of the price list.

```
with a as(Select list_price ,
row_number() over (order by list_price) lp,
count(*) over() co
from order_items)

Select case
when co%2=0 then(select avg(list_price) from a
where lp in((co/2),(co/2)+1))
else (select List_price from a
Where lp = (co+1)/2)
end as medion from a limit 1;
```



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

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

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
267	Trek Precaliber 12 Girl's - 2018
284	Electra Savannah 1 (20-inch) - Girl's - 2018

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

```
with a as(Select categories.category_id,categories.category_name,
products.product_name,products.product_id, sum(products.list_price * order_items.quantity) totoal_amount
from categories
join products
on categories.category_id = products.category_id
join order_items
on products.product_id = order_items.product_id
group by categories.category_id,categories.category_name,
products.product_name,products.product_id),

b as(select category_id,category_name,product_name,product_id,totoal_amount,
dense_rank() over(partition by category_id order by totoal_amount desc) ranks
from a)

select category_id,category_name,product_name,product_id,totoal_amount from b where ranks = 1;
```

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

Find the highest-priced product for each category name.

```
Select * from
(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) a

where rnk = 1;
```

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

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

```
Select Products.Product_name,order_items.product_id,order_items.quantity,orders.order_date,
sum(order_items.quantity) over(partition by order_items.product_id order by orders.order_date)
from Products
join order_items
on Products.product_id = order_items.product_id
join orders
on order_items.order_id = orders.order_id;
```

Product_name	product_id	quantity	order_date	sum(order_items.quantity) over(partition by order_items.product_id order by orders.order_date)
Ritchey Timberwolf Frameset - 2016	2	2	2016-01-03	2
Ritchey Timberwolf Frameset - 2016	2	2	2016-01-14	4
Ritchey Timberwolf Frameset - 2016	2	1	2016-01-18	5
Ritchey Timberwolf Frameset - 2016	2	1	2016-02-05	6
Ritchey Timberwolf Frameset - 2016	2	1	2016-02-09	7

Inventory management

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

```sql
SELECT
    staffs.staff_id,
    CONCAT(staffs.first_name, ' ', staffs.last_name) fullname,
    orders.order_id
FROM
    staffs
        LEFT JOIN
    orders ON staffs.staff_id = orders.staff_id
WHERE
    orders.order_id IS NULL
```

staff_id	fullname	order_id
1	Fabiola Jackson	NULL
4	Virgie Wiggins	NULL
5	Jannette David	NULL
10	Bernardine Houston	NULL

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  orders right join staffs
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  orders right join staffs
on staffs.staff_id = orders.staff_id
left join order_items
on orders.order_id = order_items.order_id
group by staffs.staff_id) as a);
```

staff_id	sales
3	95269072.00
6	293879916.00
7	288726544.00

Store operations

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

```
SELECT
    stores.store_name, SUM(order_items.quantity) total_products
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	total_products
Santa Cruz Bikes	1516
Baldwin Bikes	4779
Rowlett Bikes	783

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

```sql
SELECT
    customer_id, store_id, COUNT(order_id) total_orders
FROM
    orders
GROUP BY customer_id , store_id
```

customer_id	store_id	total_orders
259	1	1
1212	2	1
523	2	1
175	1	2
1324	2	1

Thank you

Connect with me

GMAIL

dishasindhi7@gmail.com

LINKEDIN

www.linkedin.com/in/disha-sindhi-b0092732a