Derived Insights Using Advance MySQL

JENSON USA
WE KEEP YOU PEDALING

MySQL

Presented By
Thomas Sangala

INTRODUCTION

Focus is to pivotal in transforming raw data into actionable insights that drive business decisions. Through the use of SQL, you explore various aspects of the company's operations, including customer behavior, staff performance, inventory management, and store efficiency. By crafting precise queries, you uncover trends and patterns that help optimize sales strategies, enhance staff productivity, and ensure effective inventory management. Your work supports the organization in making data-driven decisions, ultimately contributing to its growth and success.

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

FIND THE TOTAL NUMBER OF PRODUCTS SOLD BY EACH STORE ALONG WITH THE STORE NAME.

```sql
SELECT
    stores.store_name,
    SUM(order_items.quantity) AS total_product_sold
FROM
    orders
        JOIN
    order_items ON order_items.order_id = orders.order_id
        JOIN
    stores ON stores.store_id = orders.store_id
GROUP BY stores.store_name;
```

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

```sql
select
    product_id, order_date, quantity, sum(quantity)
over(partition by product_id order by order_date) as cumulative_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 order_items.product_id, orders.order_date) a

```

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

```sql
with a as
  (select categories.category_id, categories.category_name, products.product_id, products.product_name,
sum(order_items.quantity *(order_items.list_price -order_items.discount)) sales
from
  order_items
join
  products on products.product_id = order_items.product_id
join
  categories on categories.category_id = products.category_id
group by categories.category_id, categories.category_name, products.product_id, products.product_name)
select * from
(select *, rank() over (partition by category_id order by sales desc ) as rnk from a) b
where rnk =1;
```

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 order_items.order_id = orders.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;
```

Find the highest-priced product for each category name.

```sql
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)
from products
join categories
on products.category_id = categories.category_id) a
where rnk = 1 ;
```

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

```sql
SELECT
    store_id, customer_id, COUNT(Order_id) AS Total_orders
FROM
    orders
GROUP BY store_id , customer_id
ORDER BY store_id , customer_id;

```

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

```sql
SELECT
    staffs.staff_id,
    CONCAT(staffs.first_name, ' ', staffs.last_name) full_name
FROM
    staffs
WHERE
    NOT EXISTS( SELECT
            staff_id
        FROM
            orders
        WHERE
            orders.staff_id = staffs.staff_id);
```

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

.

```sql
select product_name from
(select products.product_id, products.product_name,
sum(order_items.quantity) quantity,
rank() over (order by sum(order_items.quantity) desc) rnk
from products
join order_items
on products.product_id = order_items.product_id
group by products.product_id, products.product_name) a
where rnk <= 3;
```

Find the median value of the price list.

```sql
with m as (select list_price, row_number()
over (order by list_price)  rn,
count(list_price) over() cn
from order_items)
select case when cn % 2 = 0 then (select avg(list_price)
 from m
where rn in (cn/2, (cn/2)+1))
else (select list_price from m where rn = (cn+1)/2)
end as median
from m limit 1;

```

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

```sql
SELECT
    products.product_id, products.product_name
FROM
    products
WHERE
    NOT EXISTS( SELECT
            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.

```sql
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);
```

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 p
on p.product_id = order_items.product_id
group by customers.customer_id
having count(distinct p.category_id) =
(select count(category_id) from categories);

```


INSPIRING PEOPLE TO RIDE, EXPERIENCE, AND EXPLORE.
FOLLOW FOR MORE