JENSON USA

Jenson Case Study

Presented by: Nikhil Take.

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

```sql
SELECT s.store_name, SUM(oi.quantity) AS total_products_sold
FROM jenkins.orders o
JOIN jenkins.order_items oi ON o.order_id = oi.order_id
JOIN jenkins.stores s ON o.store_id = s.store_id
GROUP BY s.store_name;
```

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

JENSON USA

🔍 Explanation:

1️⃣ SUM(oi.quantity) → Calculates the total number of products sold by summing up the quantity in order_items.

2️⃣ JOIN jenkins.orders o ON o.order_id = oi.order_id → Links orders with order items.

3️⃣ JOIN jenkins.stores s ON o.store_id = s.store_id → Links orders to stores to get store names.

4️⃣ GROUP BY s.store_name → Groups the data by store to get total sales for each store.

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

```sql
SELECT
    oi.product_id,
    p.product_name,
    o.order_date,
    SUM(oi.quantity) OVER (PARTITION BY oi.product_id ORDER BY o.order_date) AS cumulative_quantity_sold
FROM jenkins.order_items oi
JOIN jenkins.orders o ON oi.order_id = o.order_id
JOIN jenkins.products p ON oi.product_id = p.product_id
ORDER BY oi.product_id, o.order_date;
```

product_id	product_name	order_date	cumulative_quantity_sold
2	Ritchey Timberwolf Frameset - 2016	2016-01-03	2
2	Ritchey Timberwolf Frameset - 2016	2016-01-14	4
2	Ritchey Timberwolf Frameset - 2016	2016-01-18	5
2	Ritchey Timberwolf Frameset - 2016	2016-02-05	6
2	Ritchey Timberwolf Frameset - 2016	2016-02-09	7
2	Ritchey Timberwolf Frameset - 2016	2016-02-26	8
2	Ritchey Timberwolf Frameset - 2016	2016-02-28	10
2	Ritchey Timberwolf Frameset - 2016	2016-02-28	10
2	Ritchey Timberwolf Frameset - 2016	2016-03-08	11
2	Ritchey Timberwolf Frameset - 2016	2016-03-14	13
2	Ritchey Timberwolf Frameset - 2016	2016-03-20	17
2	Ritchey Timberwolf Frameset - 2016	2016-03-20	17
2	Ritchey Timberwolf Frameset - 2016	2016-03-21	18
2	Ritchey Timberwolf Frameset - 2016	2016-03-28	19
2	Ritchey Timberwolf Frameset - 2016	2016-04-08	20
2	Ritchey Timberwolf Frameset - 2016	2016-04-13	22
2	Ritchey Timberwolf Frameset - 2016	2016-04-15	24
2	Ritchey Timberwolf Frameset - 2016	2016-04-27	25
2	Ritchey Timberwolf Frameset - 2016	2016-04-30	27
2	Ritchey Timberwolf Frameset - 2016	2016-05-07	29

🔍 **Explanation:**

1️⃣ **SUM(oi.quantity) OVER (PARTITION BY oi.product_id ORDER BY o.order_date) → Computes the cumulative sum of the quantity sold for each product over time.**

2️⃣ **PARTITION BY oi.product_id → Ensures that the cumulative sum is calculated separately for each product.**

3️⃣ **ORDER BY o.order_date → Orders the sales chronologically for correct cumulative calculation.**

4️⃣ **JOIN jenkins.orders o → Links order_items to orders to get the order_date.**

5️⃣ **JOIN jenkins.products p → Links order_items to products to get the product_name.**

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

```sql
WITH ProductSales AS (
    SELECT
        p.product_id,
        p.product_name,
        c.category_id,
        c.category_name,
        SUM(oi.quantity * oi.list_price) AS total_sales
    FROM jenkins.order_items oi
    JOIN jenkins.products p ON oi.product_id = p.product_id
    JOIN jenkins.categories c ON p.category_id = c.category_id
    GROUP BY p.product_id, p.product_name, c.category_id, c.category_name
), RankedProducts AS (
    SELECT
        ps.*,
        RANK() OVER (PARTITION BY ps.category_id ORDER BY ps.total_sales DESC) AS rnk
    FROM ProductSales ps
)
SELECT
    product_id,
    product_name,
    category_name,
    total_sales
FROM RankedProducts
WHERE rnk = 1;
```

product_id	product_name	category_name	total_sales
23	Electra Girl's Hawaii 1 (20-inch) - 2015/2016	Children Bicycles	4619846.00
26	Electra Townie Original 7D EQ - 2016	Comfort Bicycles	8039866.00
16	Electra Townie Original 7D EQ - 2016	Cruisers Bicycles	9359844.00
11	Surly Straggler 650b - 2016	Cyclocross Bicycles	25382949.00
9	Trek Conduit+ - 2016	Electric Bikes	43499855.00
7	Trek Slash 8 275 - 2016	Mountain Bikes	61599846.00
56	Trek Domane SLR 6 Disc - 2017	Road Bikes	23649957.00

🔍 **Explanation:**

1️⃣ **ProductSales CTE → Calculates total sales (quantity * price) for each product, grouped by category.**

2️⃣ **RANK() OVER (PARTITION BY category ORDER BY total_sales DESC) → Assigns a rank to products within each category based on total sales.**

3️⃣ **WHERE rnk = 1 → Filters only the top-selling product from each category.**

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

JENSON USA

```sql
WITH CustomerSpending AS (
    SELECT
        c.customer_id,
        CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
        SUM(oi.quantity * oi.list_price) AS total_spent
    FROM jenkins.customers c
    JOIN jenkins.orders o ON c.customer_id = o.customer_id
    JOIN jenkins.order_items oi ON o.order_id = oi.order_id
    GROUP BY c.customer_id, customer_name
)
SELECT customer_id, customer_name, total_spent
FROM CustomerSpending
ORDER BY total_spent DESC
LIMIT 1;
```

customer_id	customer_name	total_spent
10	Pamelia Newman	3780184.00

🔍 Explanation:

1 SUM(oi.quantity * oi.list_price) → Calculates the total amount spent by each customer.

2 JOIN jenkins.orders o ON c.customer_id = o.customer_id → Links orders to customers.

3 JOIN jenkins.order_items oi ON o.order_id = oi.order_id → Links order items to get quantity and price.

4 GROUP BY c.customer_id, customer_name → Groups data by customer to calculate their total spending.

5 ORDER BY total_spent DESC LIMIT 1 → Finds the top-spending customer.

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

```sql
SELECT p.product_name, c.category_name, p.list_price
FROM products p
JOIN categories c ON p.category_id = c.category_id
WHERE p.list_price = (
    SELECT MAX(p2.list_price)
    FROM products p2
    WHERE p2.category_id = p.category_id
);
```

product_name	category_name	list_price
Electra Straight 8 3i (20-inch) - Boy's - 2017	Children Bicycles	48999.00
Electra Townie 3i EQ (20-inch) - Boys' - 2017	Children Bicycles	48999.00
Trek Superfly 24 - 2017/2018	Children Bicycles	48999.00
Electra Townie Go! 8i - 2017/2018	Comfort Bicycles	259999.00
Electra Townie Commute Go! - 2018	Cruisers Bicycles	299999.00
Electra Townie Commute Go! Ladies' - 2018	Cruisers Bicycles	299999.00
Trek Boone 7 Disc - 2018	Cyclocross Bicycles	399999.00
Trek Powerfly 8 FS Plus - 2017	Electric Bikes	499999.00
Trek Powerfly 7 FS - 2018	Electric Bikes	499999.00
Trek Super Commuter + 8S - 2018	Electric Bikes	499999.00
Trek Fuel EX 98 275 Plus - 2017	Mountain Bikes	529999.00
Trek Remedy 98 - 2017	Mountain Bikes	529999.00
Trek Domane SLR 9 Disc - 2018	Road Bikes	1199999.00

JENSON USA

🔍 Explanation:

1️⃣ JOIN categories c ON p.category_id = c.category_id → Connects products with categories to get category names.

2️⃣ WHERE p.list_price = (SELECT MAX(p2.list_price) ...) → Finds the highest-priced product in each category.

3️⃣ SELECT p.product_name, c.category_name, p.list_price → Displays the product name, category, and price of the most expensive product in each category.

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

```sql
SELECT c.customer_id, c.first_name, c.last_name, s.store_name, COUNT(o.order_id) AS total_orders
FROM orders o
JOIN customers c ON o.customer_id = c.customer_id
JOIN stores s ON o.store_id = s.store_id
GROUP BY c.customer_id, c.first_name, c.last_name, s.store_name
ORDER BY total_orders DESC;
```

customer_id	first_name	last_name	store_name	total_orders
46	Monika	Berg	Santa Cruz Bikes	3
32	Araceli	Golden	Santa Cruz Bikes	3
31	Williemae	Holloway	Santa Cruz Bikes	3
24	Corene	Wall	Santa Cruz Bikes	3
5	Charolette	Rice	Santa Cruz Bikes	3
30	Jamaal	Albert	Santa Cruz Bikes	3
53	Saturnina	Garner	Santa Cruz Bikes	3
33	Deloris	Burke	Santa Cruz Bikes	3
40	Ronna	Butler	Santa Cruz Bikes	3
2	Kasha	Todd	Santa Cruz Bikes	3
47	Bridgette	Guerra	Santa Cruz Bikes	3
3	Tameka	Fisher	Santa Cruz Bikes	3
9	Genoveva	Baldwin	Baldwin Bikes	3
7	Latasha	Hays	Baldwin Bikes	3
12	Robby	Sykes	Baldwin Bikes	3
17	Caren	Stephens	Baldwin Bikes	3
16	Emmitt	Sanchez	Baldwin Bikes	3
8	Jacquline	Duncan	Baldwin Bikes	3
1	Debra	Burks	Baldwin Bikes	3
18	Georgetta	Hardin	Baldwin Bikes	3

🔍 Explanation:

1️⃣ JOIN customers c ON o.customer_id = c.customer_id → Connects orders with customers to get customer details.

2️⃣ JOIN stores s ON o.store_id = s.store_id → Connects orders with stores to get store names.

3️⃣ COUNT(o.order_id) AS total_orders → Counts the total number of orders placed by each customer per store.

4️⃣ GROUP BY c.customer_id, c.first_name, c.last_name, s.store_name → Groups results by customer and store.

5️⃣ ORDER BY total_orders DESC → Sorts customers by the highest number of orders.

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


```sql
SELECT s.staff_id, s.first_name, s.last_name
FROM staffs s
LEFT JOIN orders o ON s.staff_id = o.staff_id
WHERE o.order_id IS NULL;
```

staff_id	first_name	last_name
1	Fabiola	Jackson
4	Virgie	Wiggins
5	Jannette	David
10	Bernardine	Houston

🔍 Explanation:

1️⃣ LEFT JOIN orders o ON s.staff_id = o.staff_id → Connects staff with orders to check which staff members have sales.

2️⃣ WHERE o.order_id IS NULL → Filters out staff members who have not made any sales (i.e., no matching records in orders).

3️⃣ SELECT s.staff_id, s.first_name, s.last_name → Retrieves staff ID and names of staff with zero sales.

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

```sql
SELECT p.product_name, SUM(od.quantity) AS total_quantity_sold
FROM order_items od
JOIN products p ON od.product_id = p.product_id
GROUP BY p.product_name
ORDER BY total_quantity_sold DESC
LIMIT 3;
```

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

🔍 Explanation:

1️⃣ JOIN products p ON od.product_id = p.product_id → Connects order_items with products to get product names.

2️⃣ SUM(od.quantity) AS total_quantity_sold → Calculates the total quantity sold for each product.

3️⃣ GROUP BY p.product_name → Groups results by product name to aggregate sales.

4️⃣ ORDER BY total_quantity_sold DESC → Sorts products in descending order based on total quantity sold.

5️⃣ LIMIT 3 → Retrieves only the top 3 most sold products.

JENSON USA

9.Find the median value of the price list.

```sql
SELECT AVG(list_price) AS median_price
FROM (
    SELECT list_price,
           ROW_NUMBER() OVER (ORDER BY list_price) AS row_num,
           COUNT(*) OVER () AS total_rows
    FROM products
) AS subquery
WHERE row_num IN (FLOOR((total_rows + 1) / 2), CEIL((total_rows + 1) / 2));
```

median_price
74999.000000

🔍 Explanation:

1️⃣ ROW_NUMBER() OVER (ORDER BY list_price) AS row_num → Assigns a row number to each price in ascending order.

2️⃣ COUNT(*) OVER () AS total_rows → Gets the total number of rows.

3️⃣ Handles even & odd cases:If the total count is odd, it picks the middle row.If the total count is even, it takes the average of the two middle values.

4️⃣ WHERE row_num IN (FLOOR((total_rows + 1) / 2), CEIL((total_rows + 1) / 2)) → Selects the middle value(s) based on row count.

5️⃣ AVG(list_price) AS median_price → Computes the median value.

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

JENSON USA

```sql
SELECT p.product_id, p.product_name
FROM products p
WHERE NOT EXISTS (
    SELECT 1
    FROM order_items oi
    WHERE oi.product_id = p.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
291	Electra Sweet Ride 1 (20-inch) - Girl's - 2018
316	Trek Checkpoint ALR 4 Women's - 2019
317	Trek Checkpoint ALR 5 - 2019
318	Trek Checkpoint ALR 5 Women's - 2019
319	Trek Checkpoint SL 5 Women's - 2019
320	Trek Checkpoint SL 6 - 2019
321	Trek Checkpoint ALR Frameset - 2019

🔍 Explanation:

1 WHERE NOT EXISTS (...) → Checks if a product does not exist in the order_items table.

2 SELECT 1 FROM order_items oi WHERE oi.product_id = p.product_id → Tries to find an entry for each product in order_items.

3 If no matching entry is found, the product is included in the result.

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

JENSON USA

```sql
SELECT s.staff_id, s.first_name, s.last_name, COUNT(o.order_id) AS total_sales
FROM staffs s
JOIN orders o ON s.staff_id = o.staff_id
GROUP BY s.staff_id, s.first_name, s.last_name
HAVING COUNT(o.order_id) > (
    SELECT AVG(sales_count)
    FROM (
        SELECT COUNT(order_id) AS sales_count
        FROM orders
        GROUP BY staff_id
    ) AS avg_sales
)
ORDER BY total_sales DESC;
```

staff_id	first_name	last_name	total_sales
6	Marcelene	Boyer	553
7	Venita	Daniel	540

🔍 Explanation: 1 JOIN orders o ON s.staff_id = o.staff_id → Connects staff with orders to count their total sales.
2 COUNT(o.order_id) AS total_sales → Calculates the total number of sales per staff member.
3 Subquery for average sales:SELECT COUNT(order_id) AS sales_count FROM orders GROUP BY staff_id → Gets sales count per staff.SELECT AVG(sales_count) FROM (...) → Computes the average sales per staff.
4 HAVING COUNT(o.order_id) > (subquery) → Filters staff members who have made more sales than the average.
5 ORDER BY total_sales DESC → Sorts results by highest sales first..

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

JENSON USA

```sql
SELECT c.customer_id, c.first_name, c.last_name
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
JOIN order_items oi ON o.order_id = oi.order_id
JOIN products p ON oi.product_id = p.product_id
JOIN categories cat ON p.category_id = cat.category_id
GROUP BY c.customer_id, c.first_name, c.last_name
HAVING COUNT(DISTINCT cat.category_id) = (SELECT COUNT(*) FROM categories);
```

customer_id	first_name	last_name
9	Genoveva	Baldwin

🔍 Explanation:

1 JOIN orders o ON c.customer_id = o.customer_id → Connects customers with orders to get their orders.

2 JOIN order_items oi ON o.order_id = oi.order_id → Links orders with ordered products.

3 JOIN products p ON oi.product_id = p.product_id → Retrieves product details.

4 JOIN categories cat ON p.category_id = cat.category_id → Maps products to their categories.

5 HAVING COUNT(DISTINCT cat.category_id) = (SELECT COUNT(*) FROM categories) →Counts the distinct product categories a customer has ordered.Compares it with the total number of categories in the categories table.Ensures the customer has ordered at least one product from every category.

Thank You for Viewing My Presentation!

JENSON USA