Salwa Niaz Preet

SQL Queries

Query No. 1:

Purpose: Product-Level Profitability

Script:

SELECT p.product_name,

ROUND(SUM(sod.quantity * (p.product_price * (1 + p.product_margin / 100))), 2) AS total_revenue,

ROUND(SUM(sod.quantity * p.product_price), 2) AS total_cost,

ROUND(SUM(sod.quantity * (p.product_price * (1 + p.product_margin / 100)) - sod.quantity * p.product_price), 2) AS total_profit

FROM Product p

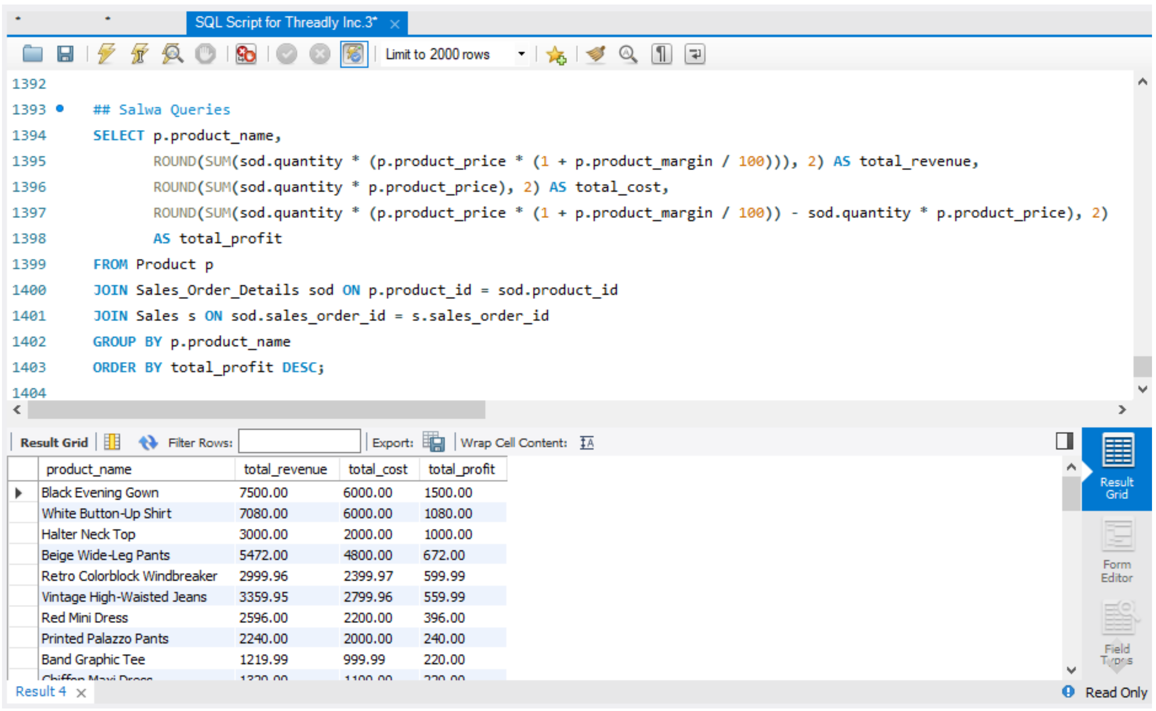
JOIN Sales_Order_Details sod ON p.product_id = sod.product_id

JOIN Sales s ON sod.sales_order_id = s.sales_order_id

GROUP BY p.product_name

ORDER BY total_profit DESC;

Execution:



```
## Salwa Queries
SELECT p.product_name,
       ROUND(SUM(sod.quantity * (p.product_price * (1 + p.product_margin / 100))), 2) AS total_revenue,
       ROUND(SUM(sod.quantity * p.product_price), 2) AS total_cost,
       ROUND(SUM(sod.quantity * (p.product_price * (1 + p.product_margin / 100)) - sod.quantity * p.product_price), 2)
       AS total_profit
FROM Product p
JOIN Sales_Order_Details sod ON p.product_id = sod.product_id
JOIN Sales s ON sod.sales_order_id = s.sales_order_id
GROUP BY p.product_name
ORDER BY total_profit DESC;
```

product_name	total_revenue	total_cost	total_profit
Black Evening Gown	7500.00	6000.00	1500.00
White Button-Up Shirt	7080.00	6000.00	1080.00
Halter Neck Top	3000.00	2000.00	1000.00
Beige Wide-Leg Pants	5472.00	4800.00	672.00
Retro Colorblock Windbreaker	2999.96	2399.97	599.99
Vintage High-Waisted Jeans	3359.95	2799.96	559.99
Red Mini Dress	2596.00	2200.00	396.00
Printed Palazzo Pants	2240.00	2000.00	240.00
Band Graphic Tee	1219.99	999.99	220.00
Chiffon Maxi Dress	1320.00	1100.00	220.00

Explanation:

The query calculates total revenue cost and profit for each product. Sale order details table has been joined with the products table to link product quantity (Sale order details table) and product name, price and margin (product table).

This query gives the list of all products along with their revenue, cost and profit. The output has been sorted such that profits appear in descending order. This will help us understand what product has high profitability – suggesting we ensure stock in the inventory and also market it more. The output will also help us analyze what product has most costs, helping us work out strategies to reduce the same.

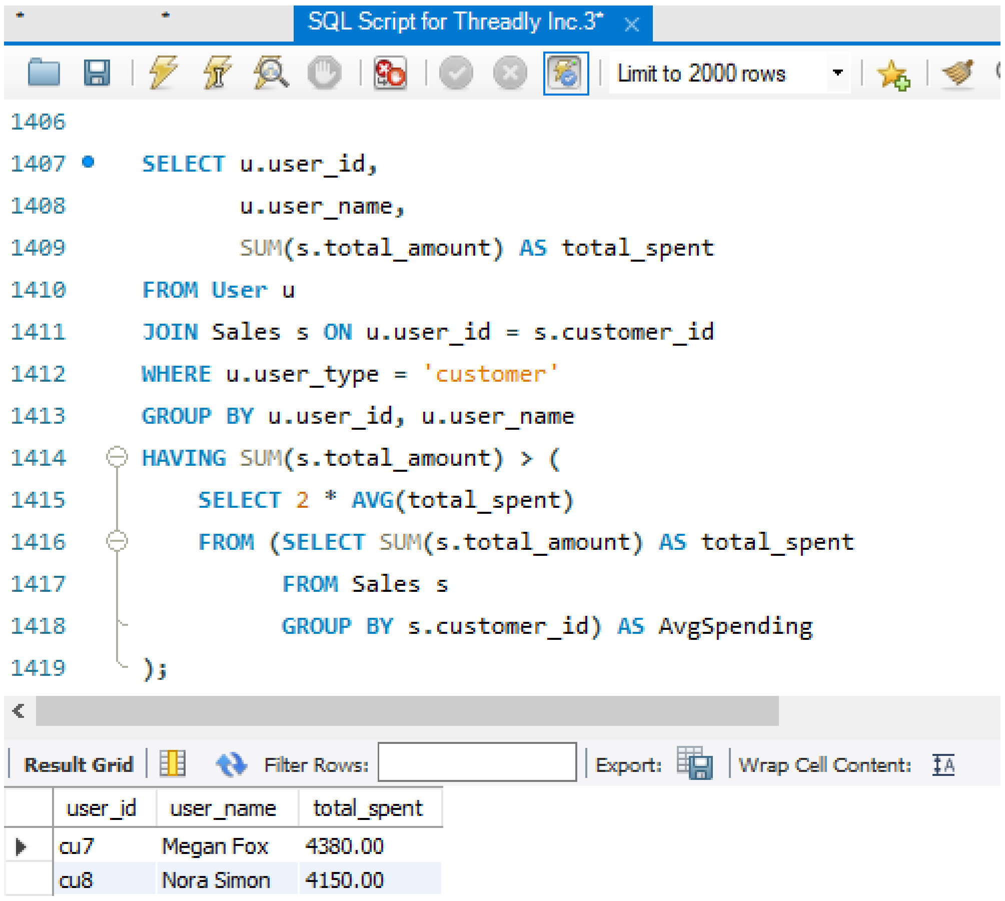
Query No. 2:

Purpose: Identifying high-potential customers:

Script:

```
SELECT u.user_id, u.user_name,
SUM(s.total_amount) AS total_spent
FROM User u
JOIN Sales s ON u.user_id = s.customer_id
WHERE u.user_type = 'customer'
GROUP BY u.user_id, u.user_name
HAVING SUM(s.total_amount) > ( SELECT 2 * AVG(total_spent)
FROM (SELECT SUM(s.total_amount) AS total_spent
FROM Sales s
GROUP BY s.customer_id) AS AvgSpending );
```

Execution:



```
SELECT u.user_id,
       u.user_name,
       SUM(s.total_amount) AS total_spent
FROM User u
JOIN Sales s ON u.user_id = s.customer_id
WHERE u.user_type = 'customer'
GROUP BY u.user_id, u.user_name
HAVING SUM(s.total_amount) > (
    SELECT 2 * AVG(total_spent)
    FROM (SELECT SUM(s.total_amount) AS total_spent
          FROM Sales s
          GROUP BY s.customer_id) AS AvgSpending
);
```

user_id	user_name	total_spent
cu7	Megan Fox	4380.00
cu8	Nora Simon	4150.00

Explanation:
This query returns customers who spend more than twice the average amount spent by all customers. This query also uses join to link the sales table to the user table so that we can get customer details.

This shows that these customers are frequent buyers and maybe even loyal to our brand. It is important to identify such customers because we need to retain them. We can increase their spending further by offering discounts, loyalty programs, and also by sending them promotional offers.

Query No. 3:

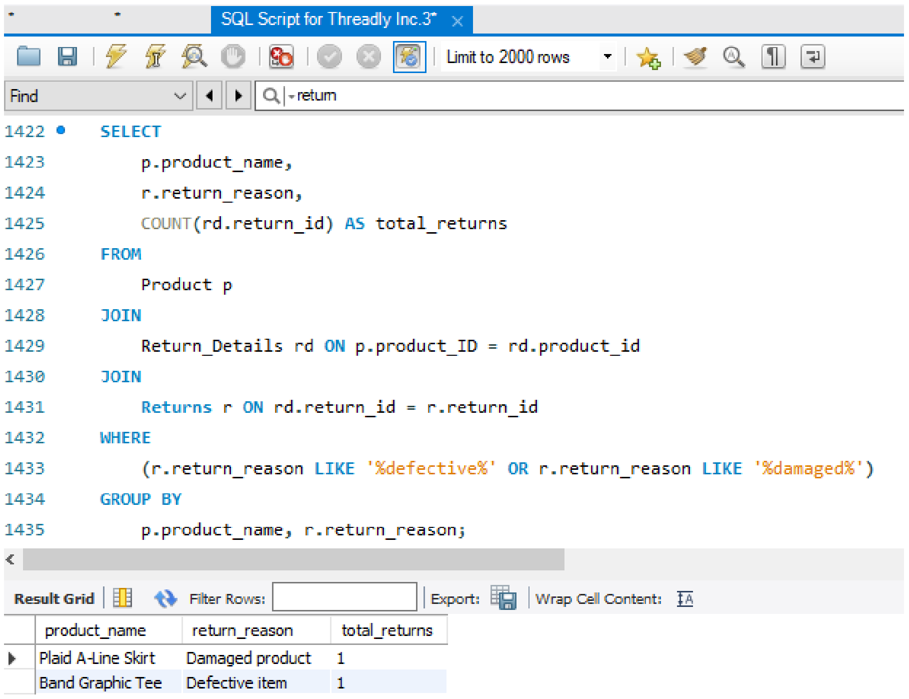
Purpose: Products returned because of defects:

Script:

```
SELECT
   p.product_name, r.return_reason,
   COUNT(rd.return_id) AS total_returns
FROM  Product p
JOIN  Return_Details rd ON p.product_ID = rd.product_id
JOIN Returns r ON rd.return_id = r.return_id
```

WHERE (r.return_reason LIKE '%defective%' OR r.return_reason LIKE '%damaged%')
GROUP BY p.product_name, r.return_reason;

Execution:



```
SELECT
    p.product_name,
    r.return_reason,
    COUNT(rd.return_id) AS total_returns
FROM
    Product p
JOIN
    Return_Details rd ON p.product_ID = rd.product_id
JOIN
    Returns r ON rd.return_id = r.return_id
WHERE
    (r.return_reason LIKE '%defective%' OR r.return_reason LIKE '%damaged%')
GROUP BY
    p.product_name, r.return_reason;
```

product_name	return_reason	total_returns
Plaid A-Line Skirt	Damaged product	1
Band Graphic Tee	Defective item	1

Explanation:

This query gives the list of products that were returned because of being damaged or defected. It also uses join to link return details to products so that we get product names based on product id appearing in the returns details table. It uses join to link returns table with return details table because we want the return reason of a specific product which is listed in the return table.

This will help us identify the products that are causing a possible loss to our business. With damaged products, not only will customers give negative reviews, impacting our brand image, but also that we will have to issue refunds, resulting in loss of revenue. The query will list products that were damaged and the number of times they were returned, helping us inspect that particular product to avoid future returns.

Query No. 4:

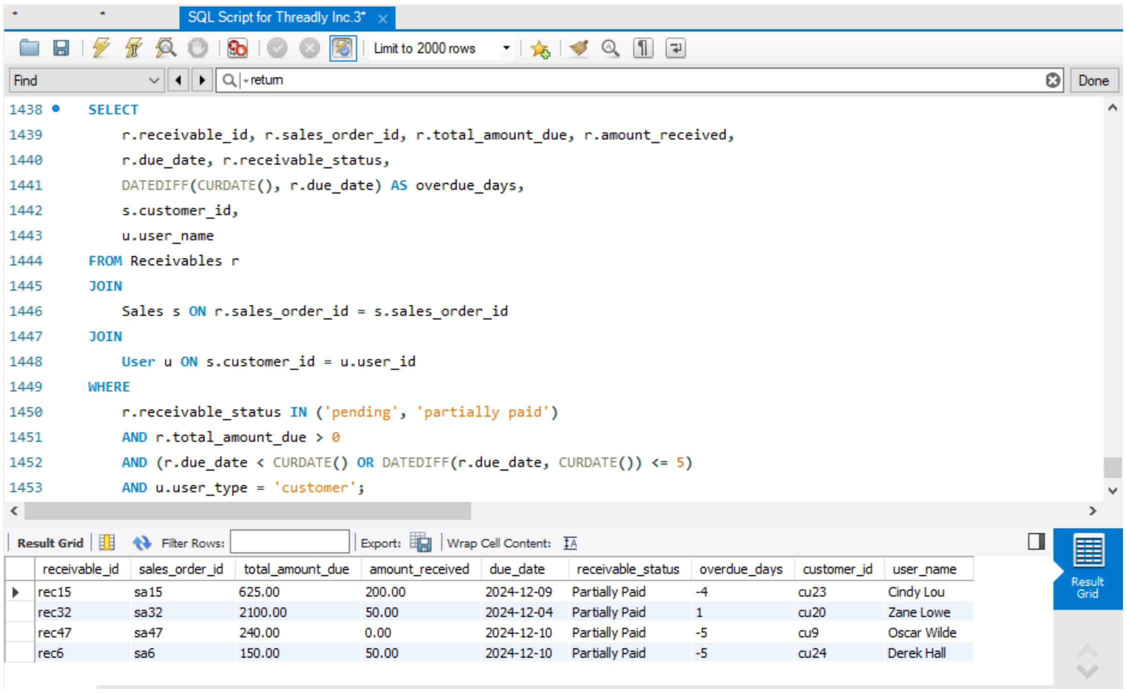
Purpose: Pending Receivables

Script:

SELECT
    r.receivable_id, r.sales_order_id, r.total_amount_due, r.amount_received,
    r.due_date, r.receivable_status,
    DATEDIFF(CURDATE(), r.due_date) AS overdue_days,
    s.customer_id,
    u.user_name
FROM Receivables r
JOIN

```
    Sales s ON r.sales_order_id = s.sales_order_id
JOIN
    User u ON s.customer_id = u.user_id
WHERE
    r.receivable_status IN ('pending', 'partially paid')
    AND r.total_amount_due > 0
    AND (r.due_date < CURDATE() OR DATEDIFF(r.due_date, CURDATE()) <= 5)
    AND u.user_type = 'customer';
```

Execution:



SQL Script for Threadly Inc.3*

Limit to 2000 rows

Find: - return

```
SELECT
    r.receivable_id, r.sales_order_id, r.total_amount_due, r.amount_received,
    r.due_date, r.receivable_status,
    DATEDIFF(CURDATE(), r.due_date) AS overdue_days,
    s.customer_id,
    u.user_name
FROM Receivables r
JOIN
    Sales s ON r.sales_order_id = s.sales_order_id
JOIN
    User u ON s.customer_id = u.user_id
WHERE
    r.receivable_status IN ('pending', 'partially paid')
    AND r.total_amount_due > 0
    AND (r.due_date < CURDATE() OR DATEDIFF(r.due_date, CURDATE()) <= 5)
    AND u.user_type = 'customer';
```

receivable_id	sales_order_id	total_amount_due	amount_received	due_date	receivable_status	overdue_days	customer_id	user_name
rec15	sa15	625.00	200.00	2024-12-09	Partially Paid	-4	cu23	Cindy Lou
rec32	sa32	2100.00	50.00	2024-12-04	Partially Paid	1	cu20	Zane Lowe
rec47	sa47	240.00	0.00	2024-12-10	Partially Paid	-5	cu9	Oscar Wilde
rec6	sa6	150.00	50.00	2024-12-10	Partially Paid	-5	cu24	Derek Hall

Explanation:

This query will return the list of customers who have either failed to pay within the due date or have 5 days remaining. The query links the sales table to the receivables table to get sales order id. It further joins the sales table to user table to get customer details.

Identifying late payments can be a serious problem as it can lead to cash flow disruptions, reduced profit margins and increased debt. This query will help us identify customers who pay late; we can strategize to either not give these customers the leverage to pay later or plan automatic payments. Additionally, the list will highlight customers who have 5 or lesser days remaining to pay. We can contact these customers to remind them to pay on time so that problems are avoided at our end.

NoSQL Queries

Query No. 1:

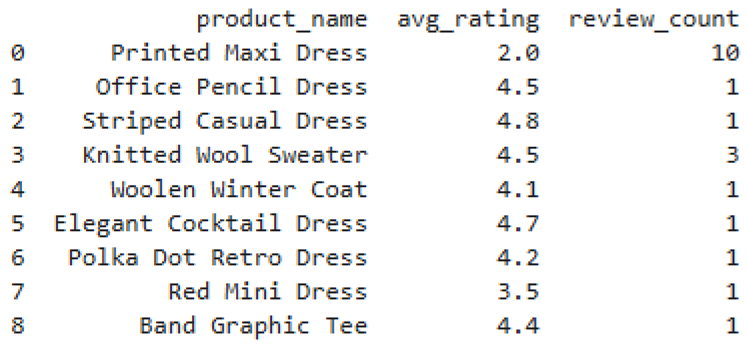
Purpose: Review Count and Average Rating Per Product

Script:

```
import pandas as pd
```

```
products_collection = db['Product']
reviews_collection = db['Reviews']
pipeline = [{ "$lookup": { "from": "Reviews", "localField": "product_ID", "foreignField":
"product_id", "as": "product_reviews" } },
            { "$unwind": "$product_reviews" },
            { "$group": { "_id": "$product_name", "avg_rating": { "$avg":
"$product_reviews.rating" }, "review_count": { "$sum": 1 } } },
            { "$project": { "product_name": "$_id", "avg_rating": 1, "review_count": 1, "_id": 0 } }]
result = list(products_collection.aggregate(pipeline))
df = pd.DataFrame(result)
df['avg_rating'] = df['avg_rating'].round(1)
df = df[['product_name', 'avg_rating', 'review_count']]
print(df)
```

Execution:



```
             product_name  avg_rating  review_count
0      Printed Maxi Dress         2.0            10
1     Office Pencil Dress         4.5             1
2    Striped Casual Dress         4.8             1
3    Knitted Wool Sweater         4.5             3
4      Woolen Winter Coat         4.1             1
5  Elegant Cocktail Dress         4.7             1
6   Polka Dot Retro Dress         4.2             1
7          Red Mini Dress         3.5             1
8         Band Graphic Tee         4.4             1
```

Explanation:

This query joins the product and reviews tables through “lookup” via the product_id. The “group” command groups the data by product_name, “avg” command calculates the average rating of all reviews for each product, and “sum” counts the number of reviews per product. “project” will return the output displaying only the product_name, avg_rating, and review_count. The round function will round off the average rating to 1 decimal place.

Result shows 10 negative reviews for the Printed Maxi Dress with an average rating of 2.0. The best products appear to be Striped Casual Dress with 1 review and average rating of 4.8 and Elegant Cocktail Dress with average rating of 4.7. This also shows us that customers are more likely to review a product when they have bad experiences. As business owners, we need to identify the problem with the Printed Maxi Dress and either remove it from inventory or try solving the issue. Further, the query result gives insights into the most trending products as well.