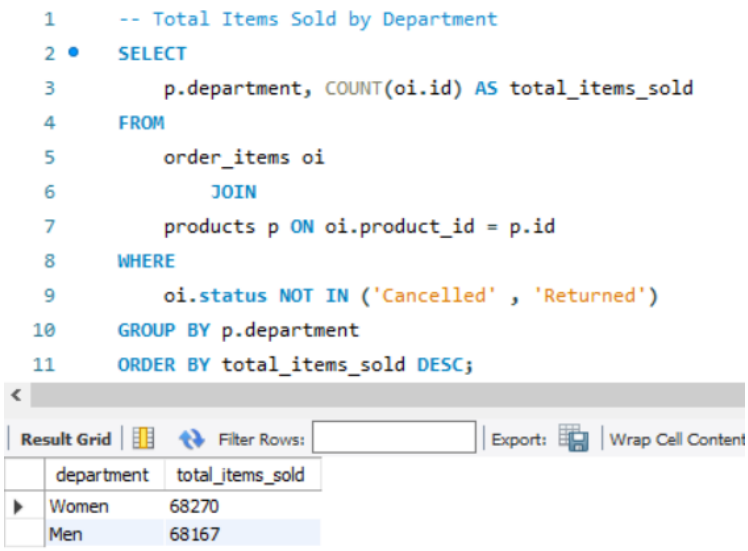
1.Quantity Sold

1.1. Per Product department (Women/Men)

```
-- Total Items Sold by Department
SELECT
    p.department, COUNT(oi.id) AS total_items_sold
FROM
    order_items oi
        JOIN
    products p ON oi.product_id = p.id
WHERE
    oi.status NOT IN ('Cancelled' , 'Returned')
GROUP BY p.department
ORDER BY total_items_sold DESC;
```



Function: This query calculates the total number of items sold from each department ('Men' and 'Women').

How it works: This query joins the order_items table (oi) with the products table (p) on product_id to link each sale to its department. It filters out any orders with a 'Cancelled' or 'Returned' status, then groups the results by department, and counts the number of items (COUNT(oi.id)) for each group, naming the result total_items_sold. Finally, it sorts these departments by their total_items_sold in descending order.

Output: A table showing two columns: department and total_items_sold.

1.2. Per Category

```
-- Total Items Sold by Category

SELECT

   p.category, COUNT(oi.id) AS total_items_sold

FROM

   order_items oi

      JOIN

   products p ON oi.product_id = p.id

WHERE

   oi.status NOT IN ('Cancelled' , 'Returned')

GROUP BY p.category

ORDER BY total_items_sold DESC

LIMIT 10;
```

```
13    -- Total Items Sold by Category
14    SELECT
15        p.category, COUNT(oi.id) AS total_items_sold
16    FROM
17        order_items oi
18            JOIN
19        products p ON oi.product_id = p.id
20    WHERE
21        oi.status NOT IN ('Cancelled' , 'Returned')
22    GROUP BY p.category
23    ORDER BY total_items_sold DESC
24    LIMIT 10;
```

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

category	total_items_sold
Intimates	10099
Jeans	9518
Tops & Tees	8965
Fashion Hoodies & Sweatshirts	8919
Swim	8564
Sweaters	8470
Sleep & Lounge	8366
Shorts	8249
Accessories	7277
Active	6798

Function: This query calculates the total number of items sold for each product category (e.g., 'Jeans', 'Sweaters') and lists the top 10 most popular.

How it works: This query joins the order_items table (oi) with the products table (p) on product_id. It filters out any rows where the status is 'Cancelled' or 'Returned', then groups the results by category. It counts the number of items for each category, aliasing it as total_items_sold, sorts the list in descending order, and uses LIMIT 10 to show only the top 10 results.

Output: A table showing the top 10 product categories and their corresponding total_items_sold.

1.3. Per brand

```
-- Total Items Sold by Brand
SELECT
    p.brand, COUNT(oi.id) AS total_items_sold
FROM
    order_items oi
        JOIN
    products p ON oi.product_id = p.id
WHERE
    oi.status NOT IN ('Cancelled' , 'Returned')
GROUP BY p.brand
ORDER BY total_items_sold DESC
LIMIT 10;
```

```
25  ------------------------------------------------------
26  -- Total Items Sold by Brand
27  SELECT
28      p.brand, COUNT(oi.id) AS total_items_sold
29  FROM
30      order_items oi
31          JOIN
32      products p ON oi.product_id = p.id
33  WHERE
34      oi.status NOT IN ('Cancelled' , 'Returned')
35  GROUP BY p.brand
36  ORDER BY total_items_sold DESC
37  LIMIT 10;
```

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

brand	total_items_sold
Allegra K	4718
Calvin Klein	2466
Carhartt	1991
Hanes	1386
Volcom	1375
Nautica	1345
Quiksilver	1326
Tommy Hilfiger	1275
Levi's	1164
Diesel	1155

Function: This query identifies the top 10 best-selling brands based on the total number of items sold.

How it works: This query joins the order_items table (oi) with the products table (p) on product_id. It filters out 'Cancelled' and 'Returned' items, then groups the results by brand. It counts the number of items for each brand, aliasing it as total_items_sold, sorts the list in descending order, and uses LIMIT 10 to show only the top 10 brands.

Output: A table showing the top 10 brands and their total_items_sold.

1.4. Per Distribution Center

```
-- Total Items Sold by Distribution Center
SELECT
    dc.name AS distribution_center_name,
    COUNT(oi.id) AS total_items_sold
FROM
    order_items oi
        JOIN
    products p ON oi.product_id = p.id
        JOIN
    distribution_centers dc ON p.distribution_center_id = dc.id
WHERE
    oi.status NOT IN ('Cancelled' , 'Returned')
GROUP BY dc.name
ORDER BY total_items_sold DESC;
```

```
39  -- Total Items Sold by Distribution Center
40  SELECT
41      dc.name AS distribution_center_name,
42      COUNT(oi.id) AS total_items_sold
43  FROM
44      order_items oi
45          JOIN
46      products p ON oi.product_id = p.id
47          JOIN
48      distribution_centers dc ON p.distribution_center_id = dc.id
49  WHERE
50      oi.status NOT IN ('Cancelled' , 'Returned')
51  GROUP BY dc.name
52  ORDER BY total_items_sold DESC;
```

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

distribution_center_name	total_items_sold
Memphis TN	18100
Chicago IL	17995
Houston TX	17016
Mobile AL	13992
Los Angeles CA	12976
Charleston SC	12729
Philadelphia PA	12694
Port Authority of New York/New Jersey NY/NJ	12128
New Orleans LA	9840
Savannah GA	8967

Function: This query calculates the total number of items fulfilled and shipped from each distribution center.

How it works: This query uses two joins: first linking order_items (oi) to products (p) on product_id, and second linking products (p) to distribution_centers (dc) on distribution_center_id. It filters out 'Cancelled' and 'Returned' items, groups the results by the distribution center's name, counts the number of items for each center as total_items_sold, and sorts the list in descending order.

Output: A table showing each distribution_center_name and the total_items_sold it has processed.

1.5. Top 20 Best-Selling Products by Name

```
-- Top 20 Best-Selling Products by Name
SELECT
    p.name, COUNT(oi.id) AS total_items_sold
FROM
    order_items oi
        JOIN
    products p ON oi.product_id = p.id
WHERE
    oi.status NOT IN ('Cancelled' , 'Returned')
GROUP BY p.name
ORDER BY total_items_sold DESC
LIMIT 20;
```

```
54    -- Top 20 Best-Selling Products by Name
55    SELECT
56        p.name, COUNT(oi.id) AS total_items_sold
57    FROM
58        order_items oi
59            JOIN
60        products p ON oi.product_id = p.id
61    WHERE
62        oi.status NOT IN ('Cancelled' , 'Returned')
63    GROUP BY p.name
64    ORDER BY total_items_sold DESC
65    LIMIT 20;
```

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

name	total_items_sold
Wrangler Men's Premium Performance Cowboy ...	50
7 For All Mankind Men's Standard Classic Straigh...	32
Puma Men's Socks	31
Thorlo Unisex Experia Running Sock	29
Kenneth Cole Men's Straight Leg Jean	28
True Religion Men's Ricky Straight Jean	27
Michael Kors Men's 3 Pack Brief	25
Mundi Big Fat Wallet	24
HUGO BOSS Men's Long Pant	24
Wrangler Men's Original Cowboy Cut Relaxed Fi...	24
Diesel Men's Shioner Skinny Straight Leg Jean	23

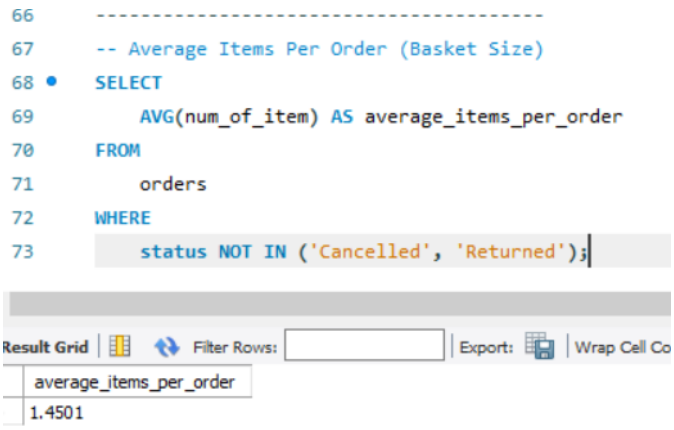
Result 23

Function: This query identifies the top 20 specific products based on the total number of items sold.

How it works: This query joins the order_items table (oi) with the products table (p) on product_id. It filters out any rows where the status is 'Cancelled' or 'Returned', then groups the results by the product's name. It counts the number of sales for each product, aliasing it as total_items_sold, sorts the list in descending order, and uses LIMIT 20 to show only the top 20 best-selling items.

Output: A table showing the top 20 product names and their corresponding total_items_sold.

1.6. Average Items Per Order (Basket Size)



-- Average Items Per Order (Basket Size)

SELECT

AVG(num_of_item) AS average_items_per_order

FROM

orders

WHERE

status NOT IN ('Cancelled', 'Returned');

Function: This query calculates the average number of items included in a single order across the entire store.

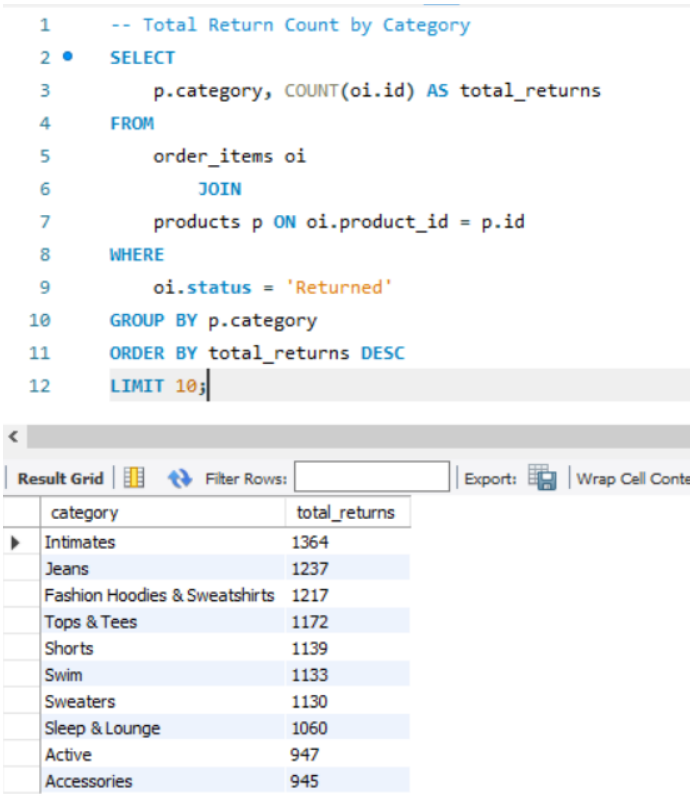
How it works: This query looks at the orders table. It filters out any orders that have a 'Cancelled' or 'Returned' status to focus on successful sales. It then uses the AVG function on the num_of_item column to calculate the overall average, which it returns as average_items_per_order.

Output: A single number representing the average number of items per order.

2.Quantity Returned

2.1. Per category

```
-- Total Return Count by Category
SELECT
    p.category, COUNT(oi.id) AS total_returns
FROM
    order_items oi
        JOIN
    products p ON oi.product_id = p.id
WHERE
    oi.status = 'Returned'
GROUP BY p.category
ORDER BY total_returns DESC
LIMIT 10;
```



```
1   -- Total Return Count by Category
2   SELECT
3       p.category, COUNT(oi.id) AS total_returns
4   FROM
5       order_items oi
6           JOIN
7       products p ON oi.product_id = p.id
8   WHERE
9       oi.status = 'Returned'
10  GROUP BY p.category
11  ORDER BY total_returns DESC
12  LIMIT 10;
```

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

category	total_returns
Intimates	1364
Jeans	1237
Fashion Hoodies & Sweatshirts	1217
Tops & Tees	1172
Shorts	1139
Swim	1133
Sweaters	1130
Sleep & Lounge	1060
Active	947
Accessories	945

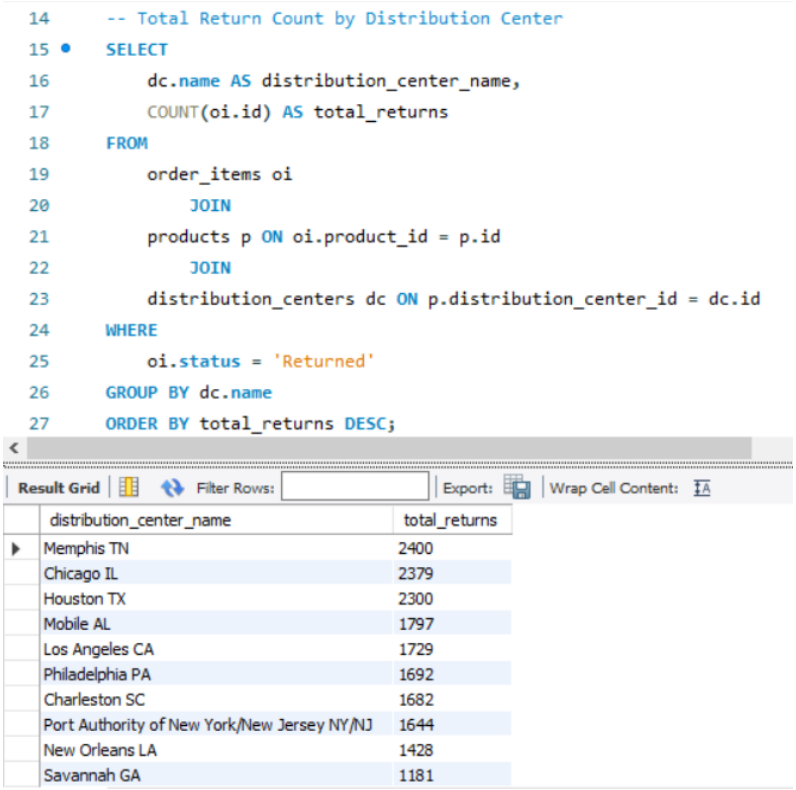
Function: This query identifies the top 10 product categories that have the highest *absolute number* of returned items.

How it works: This query joins the order_items table (oi) with the products table (p) on product_id. It filters the results to only include rows where the status is 'Returned', then groups these returns by category, counts the number of items in each group, and orders the list in descending order to show the top 10 most returned categories.

Output: A table showing the top 10 product categories and their corresponding total_returns.

2.2. Per distribution Center

```
-- Total Return Count by Distribution Center
SELECT
    dc.name AS distribution_center_name,
    COUNT(oi.id) AS total_returns
FROM
    order_items oi
        JOIN
    products p ON oi.product_id = p.id
        JOIN
    distribution_centers dc ON p.distribution_center_id = dc.id
WHERE
    oi.status = 'Returned'
GROUP BY dc.name
ORDER BY total_returns DESC;
```



```
14    -- Total Return Count by Distribution Center
15    SELECT
16        dc.name AS distribution_center_name,
17        COUNT(oi.id) AS total_returns
18    FROM
19        order_items oi
20            JOIN
21        products p ON oi.product_id = p.id
22            JOIN
23        distribution_centers dc ON p.distribution_center_id = dc.id
24    WHERE
25        oi.status = 'Returned'
26    GROUP BY dc.name
27    ORDER BY total_returns DESC;
```

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

distribution_center_name	total_returns
Memphis TN	2400
Chicago IL	2379
Houston TX	2300
Mobile AL	1797
Los Angeles CA	1729
Philadelphia PA	1692
Charleston SC	1682
Port Authority of New York/New Jersey NY/NJ	1644
New Orleans LA	1428
Savannah GA	1181

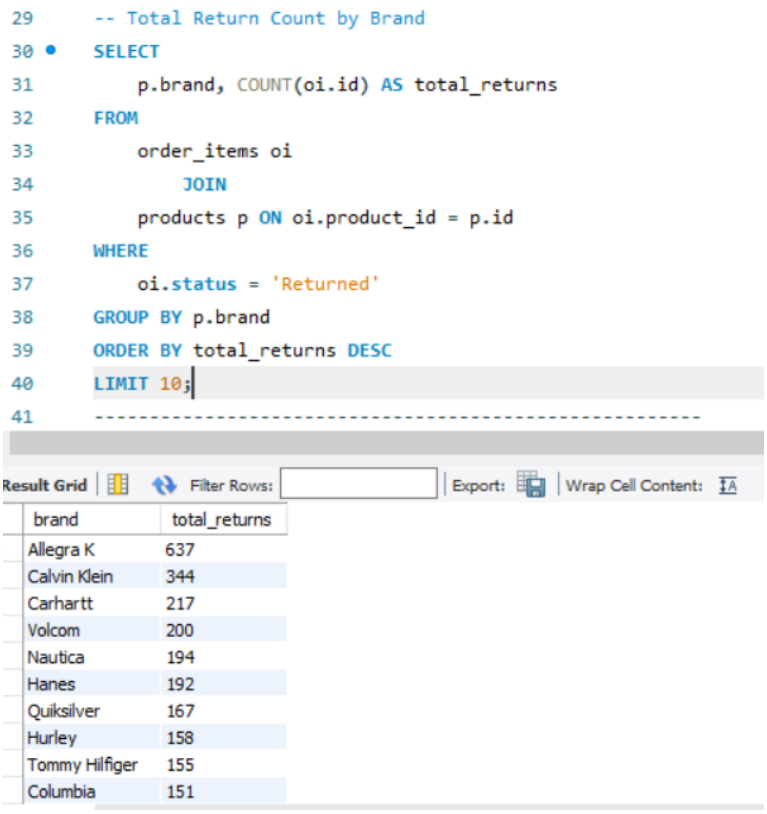
Function: This query calculates the total number of returned items associated with each distribution center.

How it works: This query first joins order_items (oi) to products (p) on product_id, and then joins that result to distribution_centers (dc) on distribution_center_id. It filters to only include rows where the oi.status is 'Returned', then groups these results by the distribution center's name. It counts the number of returns for each center, aliasing it as total_returns, and sorts the list in descending order.

Output: A table showing each distribution_center_name and the total_returns associated with it.

2.3. Per Brand

```
-- Total Return Count by Brand
SELECT
    p.brand, COUNT(oi.id) AS total_returns
FROM
    order_items oi
        JOIN
    products p ON oi.product_id = p.id
WHERE
    oi.status = 'Returned'
GROUP BY p.brand
ORDER BY total_returns DESC
LIMIT 10;
```



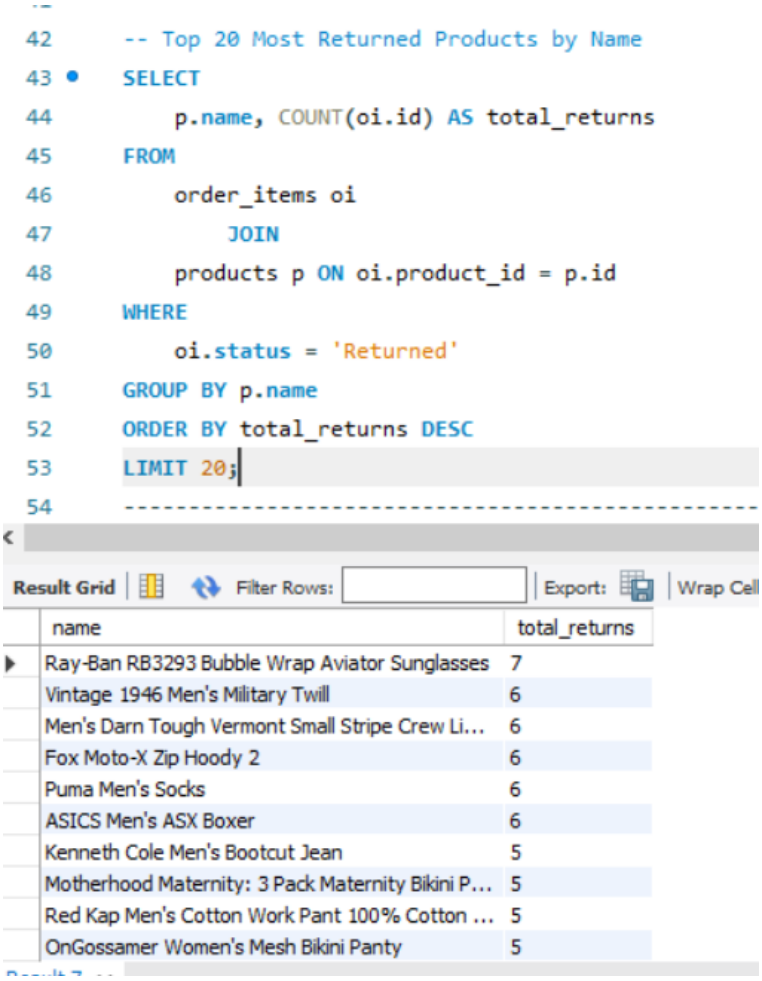
Function: This query identifies the top 10 brands that have the highest *absolute number* of returned items.

How it works: This query joins the order_items table (oi) with the products table (p) on product_id. It filters the results to only include rows where the status is 'Returned', then groups these returns by brand. It counts the number of items in each group, orders the list in descending order, and uses LIMIT 10 to show the top 10 brands with the most returns.

Output: A table showing the top 10 brands and their corresponding total_returns.

2.4. Top 20 Most Returned Products by Name

```
-- Top 20 Most Returned Products by Name
SELECT
    p.name, COUNT(oi.id) AS total_returns
FROM
    order_items oi
        JOIN
    products p ON oi.product_id = p.id
WHERE
    oi.status = 'Returned'
GROUP BY p.name
ORDER BY total_returns DESC
LIMIT 20;
```



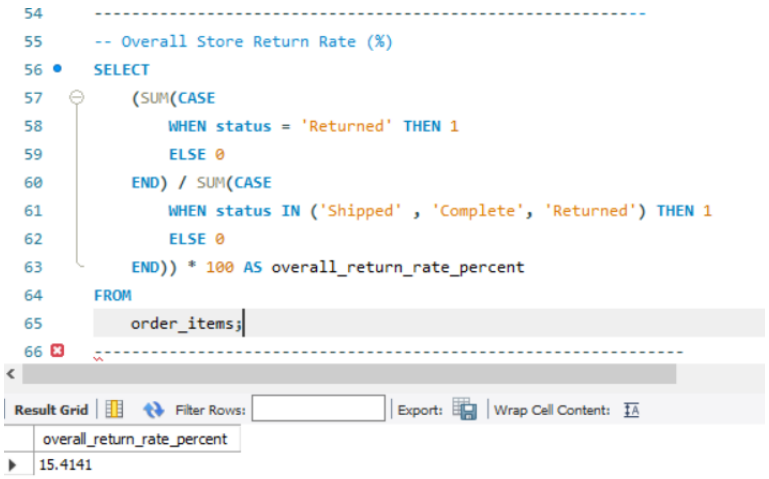
name	total_returns
Ray-Ban RB3293 Bubble Wrap Aviator Sunglasses	7
Vintage 1946 Men's Military Twill	6
Men's Darn Tough Vermont Small Stripe Crew Li...	6
Fox Moto-X Zip Hoody 2	6
Puma Men's Socks	6
ASICS Men's ASX Boxer	6
Kenneth Cole Men's Bootcut Jean	5
Motherhood Maternity: 3 Pack Maternity Bikini P...	5
Red Kap Men's Cotton Work Pant 100% Cotton ...	5
OnGossamer Women's Mesh Bikini Panty	5

Function: This query identifies the top 20 specific products that are returned most frequently, helping to pinpoint items with potential quality or description issues.

How it works: This query joins order_items (oi) with products (p) on product_id. It filters for 'Returned' items, then groups the results by the product name. It counts the number of returns for each specific product, orders the list from highest to lowest, and uses LIMIT 20 to show the biggest problem items.

Output: A table of the top 20 product names and their total_returns.

2.5. Overall Store Return Rate (%)




```
-- Overall Store Return Rate (%)
SELECT
    (SUM(CASE
        WHEN status = 'Returned' THEN 1
        ELSE 0
    END) / SUM(CASE
        WHEN status IN ('Shipped' , 'Complete', 'Returned') THEN 1
        ELSE 0
    END)) * 100 AS overall_return_rate_percent
FROM
    order_items;
```

Function: This is the main KPI for returns. It calculates the overall store-wide return rate as a percentage of all "returnable" items.

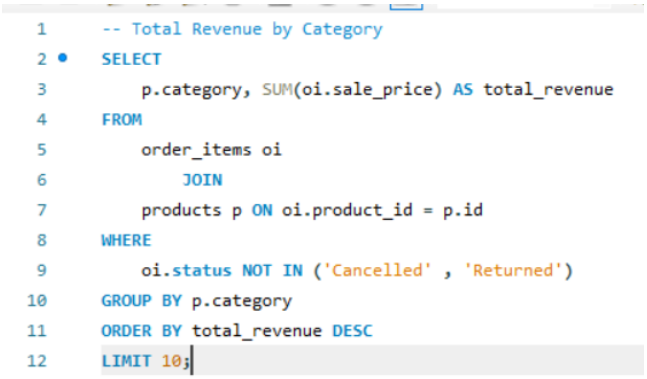
How it works: This query uses conditional aggregation in a single scan of the order_items table. It calculates a numerator (SUM(CASE WHEN status = 'Returned'...)) which is the total count of returned items. It divides this by a denominator (SUM(CASE WHEN status IN (...)) which is the total count of all items that *could* be returned (i.e., 'Shipped', 'Complete', or already 'Returned'). It then multiplies by 100 to get the final percentage.

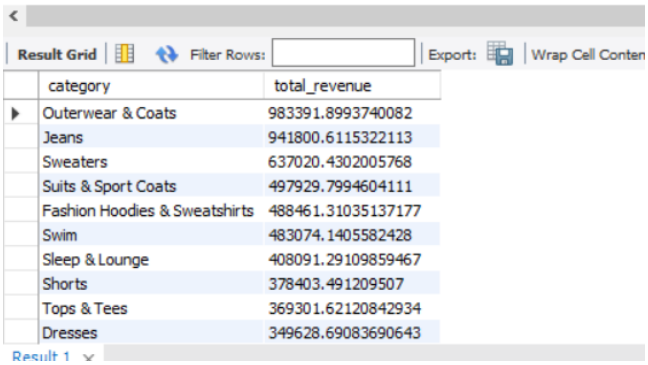
Output: A single percentage value named overall_return_rate_percent.

3.Revenue

3.1. Total Revenue by Category

```
-- Total Revenue by Category
SELECT
    p.category, SUM(oi.sale_price) AS total_revenue
FROM
    order_items oi
        JOIN
    products p ON oi.product_id = p.id
WHERE
    oi.status NOT IN ('Cancelled' , 'Returned')
GROUP BY p.category
ORDER BY total_revenue DESC
LIMIT 10;
```





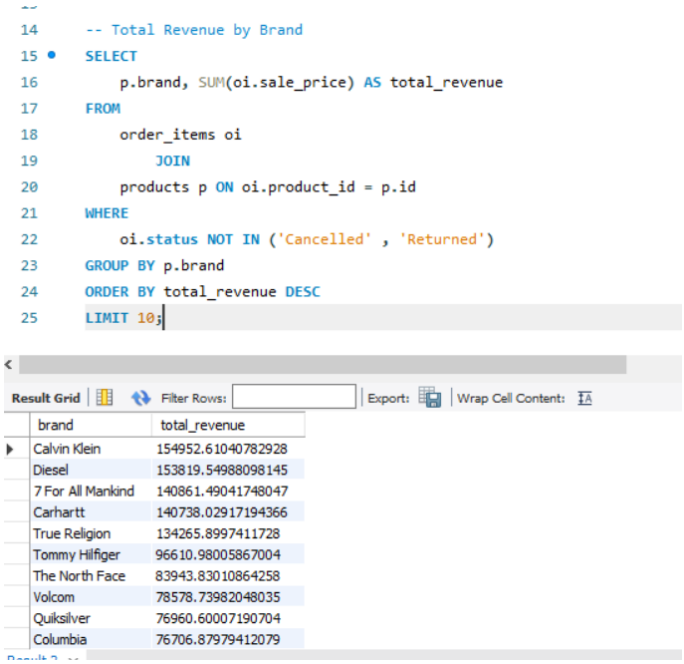
Function: This query calculates the total revenue generated by each product category (e.g., 'Jeans', 'Sweaters') and lists the top 10 most profitable.

How it works: This query joins the order_items table (oi) with the products table (p) on product_id. It filters out any rows where the status is 'Cancelled' or 'Returned', then groups the results by category. It calculates the sum of sale_price for each category, aliasing it as total_revenue, sorts the list in descending order, and uses LIMIT 10 to show only the top 10 results.

Output: A table showing the top 10 product categories and their corresponding total_revenue.

3.2. Total Revenue by Brand

```
-- Total Revenue by Brand
SELECT
    p.brand, SUM(oi.sale_price) AS total_revenue
FROM
    order_items oi
        JOIN
    products p ON oi.product_id = p.id
WHERE
    oi.status NOT IN ('Cancelled' , 'Returned')
GROUP BY p.brand
ORDER BY total_revenue DESC
LIMIT 10;
```



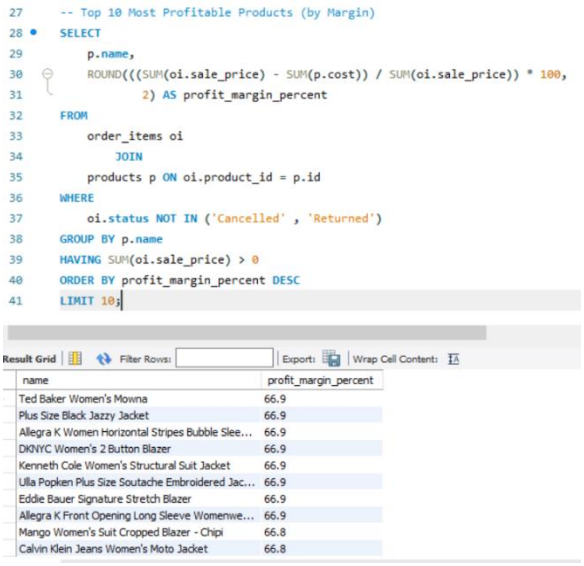
Function: This query identifies the top 10 most valuable brands based on the total revenue they generate.

How it works: This query joins the order_items table (oi) with the products table (p) on product_id. It filters out 'Cancelled' and 'Returned' items, then groups the results by brand. It calculates the sum of sale_price for each brand, aliasing it as total_revenue, sorts the list in descending order, and uses LIMIT 10 to show only the top 10 brands.

Output: A table showing the top 10 brands and their total_revenue.

5.Profit margin%

5.1. Top 10 Most Profitable Products (by Margin)



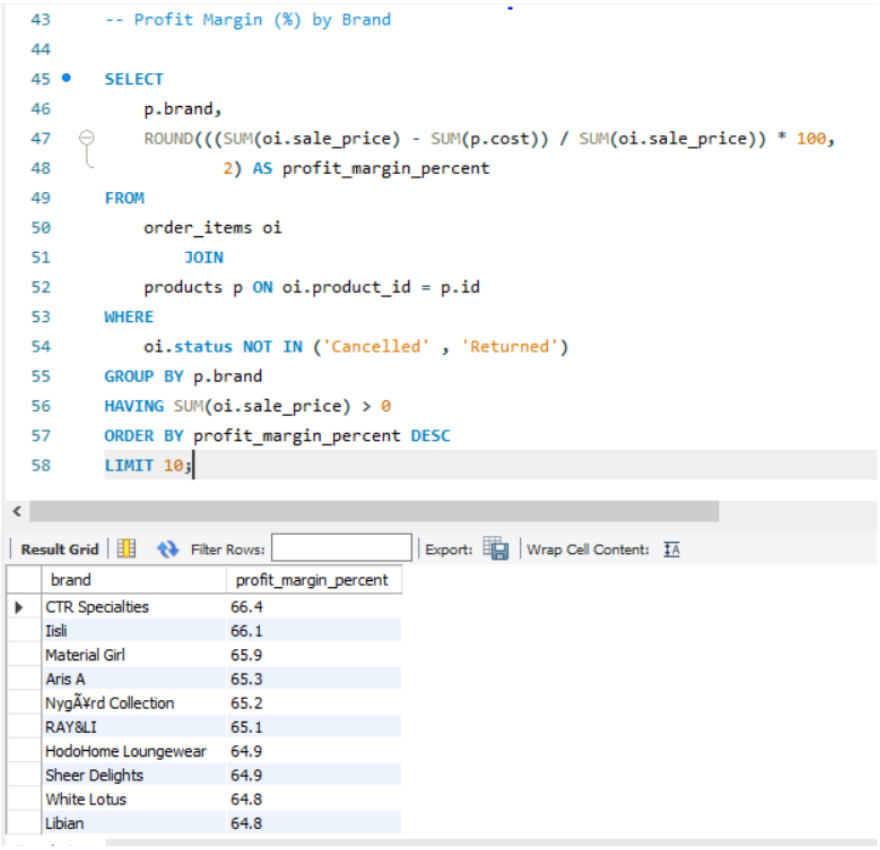
```
-- Top 10 Most Profitable Products (by Margin)
SELECT
    p.name,
    ROUND(((SUM(oi.sale_price) - SUM(p.cost)) / SUM(oi.sale_price)) * 100,
          2) AS profit_margin_percent
FROM
    order_items oi
        JOIN
    products p ON oi.product_id = p.id
WHERE
    oi.status NOT IN ('Cancelled' , 'Returned')
GROUP BY p.name
HAVING SUM(oi.sale_price) > 0
ORDER BY profit_margin_percent DESC
LIMIT 10;
```

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

name	profit_margin_percent
Ted Baker Women's Mowna	66.9
Plus Size Black Jazzy Jacket	66.9
Allegra K Women Horizontal Stripes Bubble Slee...	66.9
DKNYC Women's 2 Button Blazer	66.9
Kenneth Cole Women's Structural Suit Jacket	66.9
Ulla Popken Plus Size Soutache Embroidered Jac...	66.9
Eddie Bauer Signature Stretch Blazer	66.9
Allegra K Front Opening Long Sleeve Womenwe...	66.9
Mango Women's Suit Cropped Blazer - Chipi	66.8
Calvin Klein Jeans Women's Moto Jacket	66.8

5.2. **Profit Margin (%) by Brand**

```
-- Profit Margin (%) by Brand
SELECT
   p.brand,
   ROUND(((SUM(oi.sale_price) - SUM(p.cost)) / SUM(oi.sale_price)) * 100,
        2) AS profit_margin_percent
FROM
   order_items oi
      JOIN
   products p ON oi.product_id = p.id
WHERE
   oi.status NOT IN ('Cancelled' , 'Returned')
GROUP BY p.brand
HAVING
SUM(oi.sale_price) > 0
ORDER BY
profit_margin_percent
DESC
LIMIT 10;
```



```
43    -- Profit Margin (%) by Brand
44
45    SELECT
46        p.brand,
47        ROUND(((SUM(oi.sale_price) - SUM(p.cost)) / SUM(oi.sale_price)) * 100,
48              2) AS profit_margin_percent
49    FROM
50        order_items oi
51            JOIN
52        products p ON oi.product_id = p.id
53    WHERE
54        oi.status NOT IN ('Cancelled' , 'Returned')
55    GROUP BY p.brand
56    HAVING SUM(oi.sale_price) > 0
57    ORDER BY profit_margin_percent DESC
58    LIMIT 10;
```

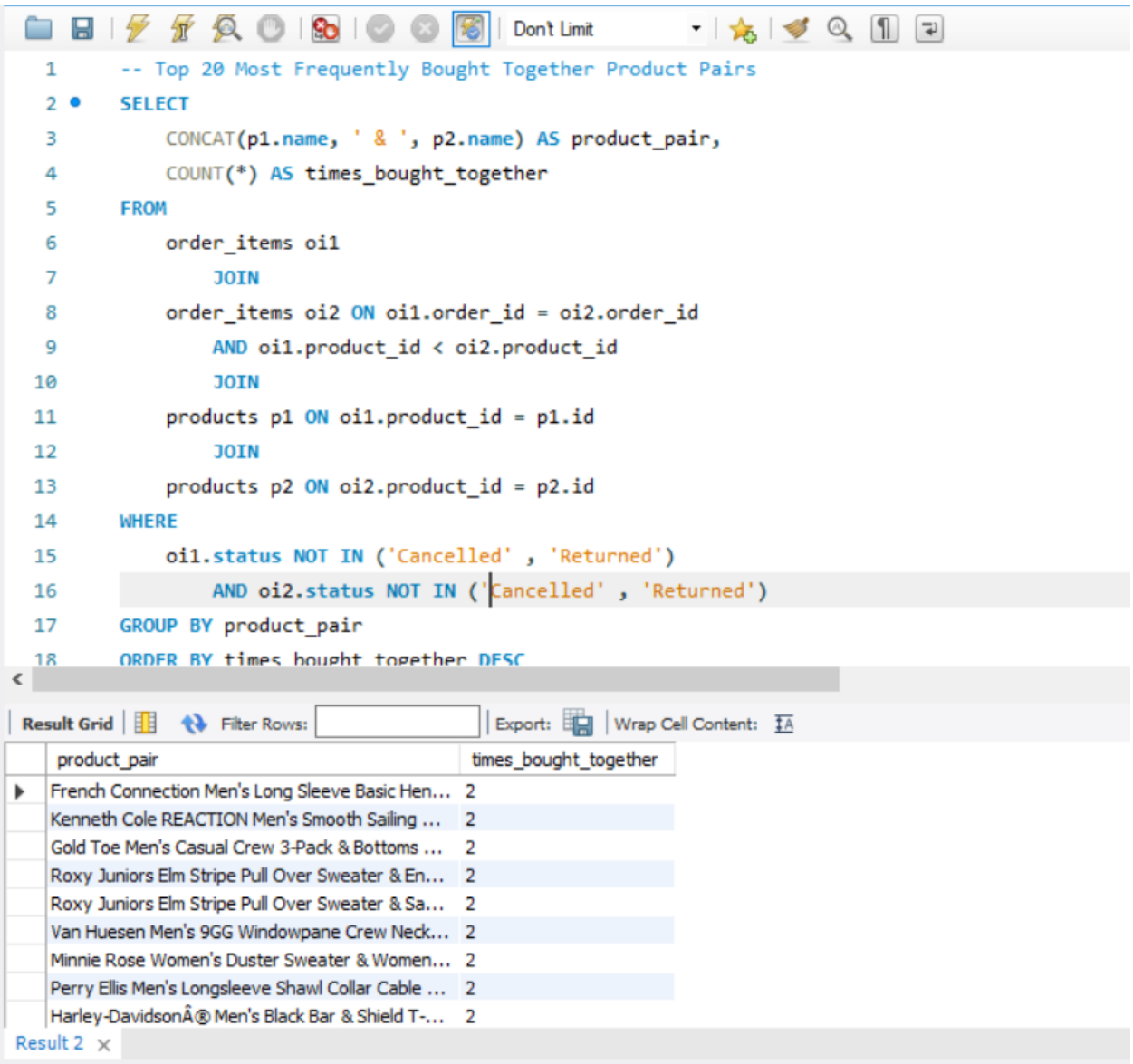
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

brand	profit_margin_percent
CTR Specialties	66.4
Iisli	66.1
Material Girl	65.9
Aris A	65.3
NygÃ¥rd Collection	65.2
RAY&LI	65.1
HodoHome Loungewear	64.9
Sheer Delights	64.9
White Lotus	64.8
Libian	64.8

6.Cross sell analysis

6.1. Top 20 Most Frequently Bought Together Product Pairs

```
-- Top 20 Most Frequently Bought Together Product Pairs
SELECT
   CONCAT(p1.name, ' & ', p2.name) AS product_pair,
   COUNT(*) AS times_bought_together
FROM
   order_items oi1
      JOIN
   order_items oi2 ON oi1.order_id = oi2.order_id
      AND oi1.product_id < oi2.product_id
      JOIN
   products p1 ON oi1.product_id = p1.id
      JOIN
   products p2 ON oi2.product_id = p2.id
WHERE
   oi1.status NOT IN ('Cancelled' , 'Returned')
      AND oi2.status NOT IN ('Cancelled' , 'Returned')
GROUP BY product_pair
ORDER BY times_bought_together DESC
LIMIT 10;
```



```sql
-- Top 20 Most Frequently Bought Together Product Pairs
SELECT
    CONCAT(p1.name, ' & ', p2.name) AS product_pair,
    COUNT(*) AS times_bought_together
FROM
    order_items oi1
        JOIN
    order_items oi2 ON oi1.order_id = oi2.order_id
        AND oi1.product_id < oi2.product_id
        JOIN
    products p1 ON oi1.product_id = p1.id
        JOIN
    products p2 ON oi2.product_id = p2.id
WHERE
    oi1.status NOT IN ('Cancelled' , 'Returned')
        AND oi2.status NOT IN ('Cancelled' , 'Returned')
GROUP BY product_pair
ORDER BY times_bought_together DESC
```

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

product_pair	times_bought_together
French Connection Men's Long Sleeve Basic Hen...	2
Kenneth Cole REACTION Men's Smooth Sailing ...	2
Gold Toe Men's Casual Crew 3-Pack & Bottoms ...	2
Roxy Juniors Elm Stripe Pull Over Sweater & En...	2
Roxy Juniors Elm Stripe Pull Over Sweater & Sa...	2
Van Huesen Men's 9GG Windowpane Crew Neck...	2
Minnie Rose Women's Duster Sweater & Women...	2
Perry Ellis Men's Longsleeve Shawl Collar Cable ...	2
Harley-DavidsonÂ® Men's Black Bar & Shield T-...	2

Result 2

Function: This query finds the top 20 pairs of products that are most frequently purchased together in the same order.

How it works: This query self-joins order_items to itself (as oi1 and oi2) on the same order_id to find all items on the same order. The oi1.product_id < oi2.product_id condition is the key best practice: it prevents duplicates (like pairing a product with itself) and reversed pairs (like (A,B) and (B,A)). It then joins to the products table twice (p1 and p2) to get the names for each ID, filters out unsuccessful items, concatenates the two names into a single product_pair, groups by this pair, and counts how many times each pair appears, sorting by the highest count.

Output: A table showing the product_pair (e.g., "Men's Jeans & Men's Classic Tee") and the times_bought_together**.**

But the result 2 is too small for the size of data meaning the data is probably not real

6.2. Top 5 Cross-Sell Recommendations for a *Specific* Product

```
-- Change the ‘product name’ to the required specific product
SELECT
   p_anchor.name AS anchor_product,
   p_rec.name AS recommended_product,
   COUNT(*) AS times_bought_together
FROM
   order_items oi_anchor
JOIN
   order_items oi_rec ON oi_anchor.order_id = oi_rec.order_id
            AND oi_anchor.product_id != oi_rec.product_id
JOIN
   products p_anchor ON oi_anchor.product_id = p_anchor.id
JOIN
   products p_rec ON oi_rec.product_id = p_rec.id
WHERE
   p_anchor.name = 'Product Name Here' -- <-- Change this
   AND oi_anchor.status NOT IN ('Cancelled', 'Returned')
   AND oi_rec.status NOT IN ('Cancelled', 'Returned')
GROUP BY
   p_anchor.name, p_rec.name
ORDER BY
   times_bought_together DESC
LIMIT 5;
```

Example for product “bailey 44 women’ ‘s undertow top”

```sql
SELECT
    p_anchor.name AS anchor_product,
    p_rec.name AS recommended_product,
    COUNT(*) AS times_bought_together
FROM
    order_items oi_anchor
        JOIN
    order_items oi_rec ON oi_anchor.order_id = oi_rec.order_id
        AND oi_anchor.product_id != oi_rec.product_id
        JOIN
    products p_anchor ON oi_anchor.product_id = p_anchor.id
        JOIN
    products p_rec ON oi_rec.product_id = p_rec.id
WHERE
    p_anchor.name = 'Bailey 44 Women''s Undertow Top'
        AND oi_anchor.status NOT IN ('Cancelled' , 'Returned')
        AND oi_rec.status NOT IN ('Cancelled' , 'Returned')
GROUP BY p_anchor.name , p_rec.name
ORDER BY times_bought_together DESC
LIMIT 5;
```

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

anchor_product	recommended_product	times_bought_together
Bailey 44 Women's Undertow Top	Sock It To Me Black Cat Knee High Womens Socks	1
Bailey 44 Women's Undertow Top	David Kahn Women's Nikki Boot Cut Osborne	1
Bailey 44 Women's Undertow Top	Bslingerie Womens Brocade Lace Up Back Under...	1
Bailey 44 Women's Undertow Top	Under Armour Surge Sport Sunglasses	1

Function: This query answers: "When a customer buys [Product X], what are the top 5 other products they are most likely to buy with it?"

How it works: This query self-joins order_items (as oi_anchor and oi_rec) on the same order_id. It joins to products twice to get the names, but this time it filters the WHERE clause to only find orders that contain a specific "anchor product" (e.g., 'Calvin Klein Men's Jeans'). It then counts and sorts all the *other* products (oi_rec) purchased on those same orders to find the most popular recommendations.

Output: A table showing the anchor_product and its top 5 recommended_products, ranked by times_bought_together.