SQL BUSSINESS REPORT

Sanjana K Venkatesh

08-09-2024

Table of Contents

List of Tables

1. WRITE A QUERY TO DISPLAY CUSTOMER FULL NAME WITH THEIR TITLE (MR/MS), BOTH FIRST NAME AND LAST NAME ARE IN UPPER CASE WITH

-- CUSTOMER EMAIL ID, CUSTOMER CREATIONDATE AND DISPLAY CUSTOMER'S CATEGORY AFTER APPLYING BELOW CATEGORIZATION RULES:

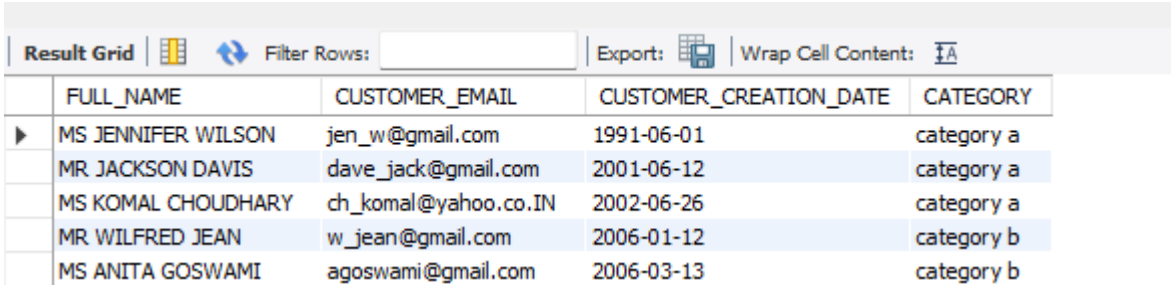
i.IF CUSTOMER CREATION DATE YEAR <2005 THEN CATEGORY A

ii.IF CUSTOMER CREATION DATE YEAR >=2005 AND <2011 THEN CATEGORY B

iii.IF CUSTOMER CREATION DATE YEAR>= 2011 THEN CATEGORY C

-- HINT: USE CASE STATEMENT, NO PERMANENT CHANGE IN TABLE REQUIRED. [NOTE: TABLES TO BE USED -ONLINE_CUSTOMER TABLE]

SOLUTION-QUERY1:


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

FULL_NAME	CUSTOMER_EMAIL	CUSTOMER_CREATION_DATE	CATEGORY
MS JENNIFER WILSON	jen_w@gmail.com	1991-06-01	category a
MR JACKSON DAVIS	dave_jack@gmail.com	2001-06-12	category a
MS KOMAL CHOUDHARY	ch_komal@yahoo.co.IN	2002-06-26	category a
MR WILFRED JEAN	w_jean@gmail.com	2006-01-12	category b
MS ANITA GOSWAMI	agoswami@gmail.com	2006-03-13	category b

EXPLANATION:

1. **CONCAT function:**
    a. Used to concatenate the title with the uppercased first and last names.
2. **UPPER function:**
    a. Converts first_name and last_name to uppercase.
3. **YEAR function:**
    a. Extracts the year from the creation_date.
4. **CASE statement:**
    a. Handles the logic for determining the title and category.

2. WRITE A QUERY TO DISPLAY THE FOLLOWING INFORMATION FOR THE PRODUCTS, WHICH HAVE NOT BEEN SOLD: PRODUCT_ID, PRODUCT_DESC,

-- PRODUCT_QUANTITY_AVAIL, PRODUCT_PRICE,INVENTORY VALUES(PRODUCT_QUANTITY_AVAIL*PRODUCT_PRICE), NEW_PRICE AFTER APPLYING DISCOUNT

-- AS PER BELOW CRITERIA. SORT THE OUTPUT WITH RESPECT TO DECREASING VALUE OF INVENTORY_VALUE.

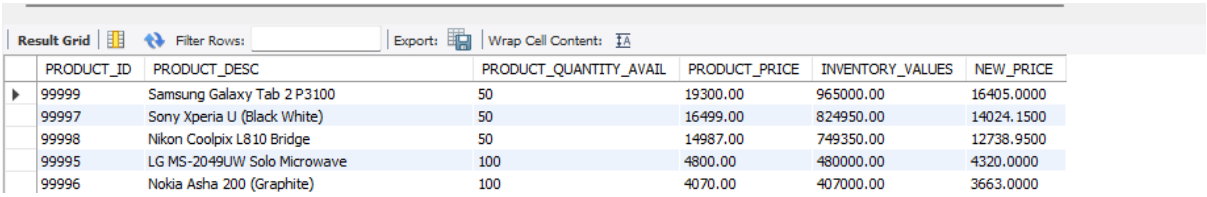
-- i.IF PRODUCT PRICE > 20,000 THEN APPLY 20% DISCOUNT

-- ii.IF PRODUCT PRICE > 10,000 THEN APPLY 15% DISCOUNT

-- iii.IF PRODUCT PRICE =< 10,000 THEN APPLY 10% DISCOUNT

-- HINT: USE CASE STATEMENT, NO PERMANENT CHANGE IN TABLE REQUIRED. [NOTE: TABLES TO BE USED -PRODUCT, ORDER_ITEMS TABLE]

SOLUTION- QUERY2:



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

PRODUCT_ID	PRODUCT_DESC	PRODUCT_QUANTITY_AVAIL	PRODUCT_PRICE	INVENTORY_VALUES	NEW_PRICE
99999	Samsung Galaxy Tab 2 P3100	50	19300.00	965000.00	16405.0000
99997	Sony Xperia U (Black White)	50	16499.00	824950.00	14024.1500
99998	Nikon Coolpix L810 Bridge	50	14987.00	749350.00	12738.9500
99995	LG MS-2049UW Solo Microwave	100	4800.00	480000.00	4320.0000
99996	Nokia Asha 200 (Graphite)	100	4070.00	407000.00	3663.0000

EXPLANTION:

- **LEFT JOIN between PRODUCT and ORDER_ITEMS:**
  - Joins the PRODUCT table with ORDER_ITEMS and keeps all records from PRODUCT. When PRODUCT_ID is NULL, it indicates that the product has not been sold.
- **INVENTORY_VALUE:**
  - Calculated by multiplying PRODUCT_QUANTITY_AVAIL by PRODUCT_PRICE.
- **CASE statement for NEW_PRICE:**
  - Applies discounts based on the price.
  - If PRODUCT_PRICE > 20000, a 20% discount is applied.
  - If PRODUCT_PRICE > 10000, a 15% discount is applied.
  - If PRODUCT_PRICE <= 10000, a 10% discount is applied.
- **ORDER BY INVENTORY_VALUE DESC:**

- Orders the results by INVENTORY_VALUE in descending order.

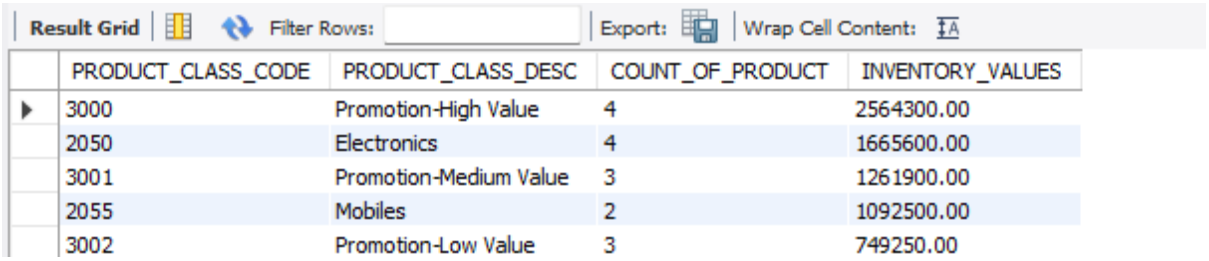
-- 3. WRITE A QUERY TO DISPLAY PRODUCT_CLASS_CODE, PRODUCT_CLASS_DESCRIPTION, COUNT OF PRODUCT TYPE IN EACH PRODUCT CLASS,

-- INVENTORY VALUE (P.PRODUCT_QUANTITY_AVAIL*P.PRODUCT_PRICE). INFORMATION SHOULD BE DISPLAYED FOR ONLY THOSE PRODUCT_CLASS_CODE

-- WHICH HAVE MORE THAN 1,00,000 INVENTORY VALUE. SORT THE OUTPUT WITH RESPECT TO DECREASING VALUE OF INVENTORY_VALUE.

-- [NOTE: TABLES TO BE USED -PRODUCT, PRODUCT_CLASS]

SOLUTION- QUERY3:



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

PRODUCT_CLASS_CODE	PRODUCT_CLASS_DESC	COUNT_OF_PRODUCT	INVENTORY_VALUES
3000	Promotion-High Value	4	2564300.00
2050	Electronics	4	1665600.00
3001	Promotion-Medium Value	3	1261900.00
2055	Mobiles	2	1092500.00
3002	Promotion-Low Value	3	749250.00

EXPLANATION:

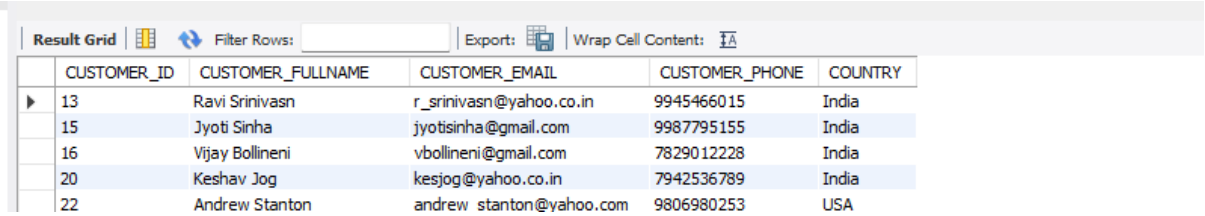
1. **COUNT() AS COUNT_OF_PRODUCT:**
   - Counts the number of products within each PRODUCT_CLASS_CODE.
2. **SUM() AS INVENTORY_VALUES:**
   - Aggregates the total inventory value for each product class.
3. **GROUP BY:**
   - Groups the result to get desired output.
4. **HAVING SUM(...) > 100000:**
   - Filters the groups to only include those where the total inventory value is greater than 100,000.
5. **ORDER BY INVENTORY_VALUES DESC:**
   - Orders the result set by the aggregated INVENTORY_VALUES in descending order.

-- 4. WRITE A QUERY TO DISPLAY CUSTOMER_ID, FULL NAME, CUSTOMER_EMAIL, CUSTOMER_PHONE AND COUNTRY OF CUSTOMERS WHO HAVE CANCELLED

-- ALL THE ORDERS PLACED BY THEM(USE SUB-QUERY)

-- [NOTE: TABLES TO BE USED - ONLINE_CUSTOMER, ADDRESSS, ORDER_HEADER]

SOLUTION- QUERY4:



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

CUSTOMER_ID	CUSTOMER_FULLNAME	CUSTOMER_EMAIL	CUSTOMER_PHONE	COUNTRY
13	Ravi Srinivasn	r_srinivasn@yahoo.co.in	9945466015	India
15	Jyoti Sinha	jyotisinha@gmail.com	9987795155	India
16	Vijay Bollineni	vbollineni@gmail.com	7829012228	India
20	Keshav Jog	kesjog@yahoo.co.in	7942536789	India
22	Andrew Stanton	andrew_stanton@yahoo.com	9806980253	USA

EXPLANTATION:

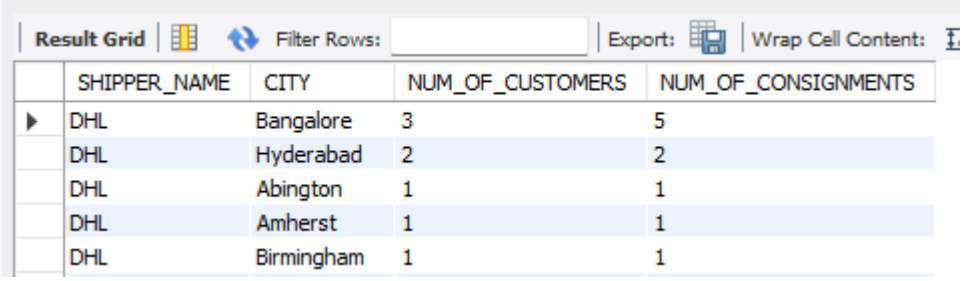
1. **Select Customer Information**:
    - We retrieve the same customer information: CUSTOMER_ID, full name, email, phone, and country from the online_customer and address tables.
2. **NOT EXISTS Subquery:**
    - The subquery checks for the existence of any orders for the customer that are **not cancelled**.
    - If such an order is found (i.e., an order with a status other than 'Cancelled'), then the NOT EXISTS condition will fail, and that customer will not be included in the results.
    - Conversely, if no orders exist with a status other than 'Cancelled', the customer will be selected.
3. **Join with address:**
    - The INNER JOIN ensures that we pull the associated COUNTRY from the address table for the selected customers.

-- 5. WRITE A QUERY TO DISPLAY SHIPPER NAME, CITY TO WHICH IT IS CATERING, NUMBER OF CUSTOMER CATERED BY THE SHIPPER IN THE CITY AND

-- NUMBER OF CONSIGNMENTS DELIVERED TO THAT CITY FOR SHIPPER DHL(9 ROWS)

-- [NOTE: TABLES TO BE USED -SHIPPER, ONLINE_CUSTOMER, ADDRESSS, ORDER_HEADER]

SOLUTION- QUERY5:



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

SHIPPER_NAME	CITY	NUM_OF_CUSTOMERS	NUM_OF_CONSIGNMENTS
DHL	Bangalore	3	5
DHL	Hyderabad	2	2
DHL	Abington	1	1
DHL	Amherst	1	1
DHL	Birmingham	1	1

EXPLANATION:

1. **Joins:**
    - shipper is joined with order_header using the SHIPPER_ID, since each order is handled by a specific shipper.
    - order_header is joined with online_customer on CUSTOMER_ID, which links the order to a customer.
    - address is joined on the ADDRESS_ID from the online_customer table to determine the city the customer is located in.
2. **Filter for Shipper 'DHL':**
    - We only want the results for the shipper **DHL**, so the WHERE clause ensures only records related to DHL are considered.
3. **Count of Unique Customers:**
    - The COUNT(DISTINCT) counts the **number of unique customers** catered by the shipper in each city.
4. **Count of Consignments:**
    - The counts the **number of consignments** (orders) delivered to each city by the shipper.
5. **Grouping and Ordering:**
    - The GROUP BY ensures the results are grouped by the shipper and the city.

- The ORDER BY orders the results by the number of consignments in descending order.

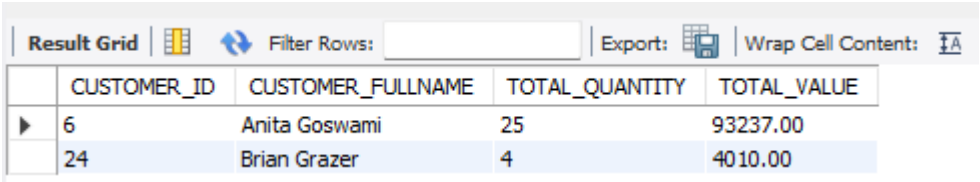
6. **Limit to 9 rows:**
    - The LIMIT 9 ensures that the query returns exactly 9 rowS.

-- 6. WRITE A QUERY TO DISPLAY CUSTOMER ID, CUSTOMER FULL NAME, TOTAL QUANTITY AND TOTAL VALUE (QUANTITY*PRICE) SHIPPED WHERE MODE

-- OF PAYMENT IS CASH AND CUSTOMER LAST NAME STARTS WITH 'G'

-- [NOTE: TABLES TO BE USED -ONLINE_CUSTOMER, ORDER_ITEMS, PRODUCT, ORDER_HEADER]

SOLUTION- QUERY6:


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

CUSTOMER_ID	CUSTOMER_FULLNAME	TOTAL_QUANTITY	TOTAL_VALUE
6	Anita Goswami	25	93237.00
24	Brian Grazer	4	4010.00

EXPLANATION:

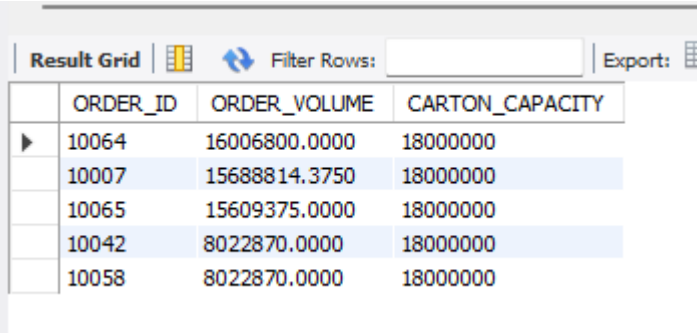
1. **CUSTOMER_ID and CUSTOMER_FULL_NAME:**
    - The query selects the CUSTOMER_ID and constructs the full name using CONCAT(oc.FIRST_NAME, ' ', oc.LAST_NAME).
2. **SUM()ON QUANTITY AS TOTAL_QUANTITY:**
    - This sums up the total quantity of products ordered by the customer.
3. **SUM() ON QUANTITY AND PRODUCT PRICE AS TOTAL_VALUE:**
    - This calculates the total value of all products ordered by multiplying the quantity by the product price and summing them up.
4. **INNER JOIN Clauses:**
    - The query joins the ONLINE_CUSTOMER, ORDER_HEADER, ORDER_ITEMS, and PRODUCT tables using the relevant foreign keys.
5. **WHERE Clause:**
    - Filters the results where the payment mode is CASH and the customer's last name starts with 'G'.

6. **GROUP BY Clause:**
    - Groups the results to aggregate the total quantity and value per customer.
7. **ORDER BY TOTAL_VALUE DESC:**
    - Orders the results in descending order based on the total value shipped.

-- 7. WRITE A QUERY TO DISPLAY ORDER_ID AND VOLUME OF BIGGEST ORDER (IN TERMS OF VOLUME) THAT CAN FIT IN CARTON ID 10

-- [NOTE: TABLES TO BE USED -CARTON, ORDER_ITEMS, PRODUCT]

SOLUTION- QUERY7:


Result Grid | Filter Rows: | Export:

ORDER_ID	ORDER_VOLUME	CARTON_CAPACITY
10064	16006800.0000	18000000
10007	15688814.3750	18000000
10065	15609375.0000	18000000
10042	8022870.0000	18000000
10058	8022870.0000	18000000

EXPLANATION:

1) **Inner Joins:**
    - We join order_items with the product table to get the **volume of each product** .
    - We also join the carton table with **Carton ID 10** to get the **capacity** of the carton .
2) **Volume Calculation:**
    - For each order, we calculate the **total volume**.
3) **Grouping and Summing:**
    - We group by ORDER_ID to get the total volume for each order.
4) **HAVING Clause:**
    - We use the HAVING clause to ensure that the **total volume of the order** is less than or equal to the carton capacity

5) **Order and Limit:**

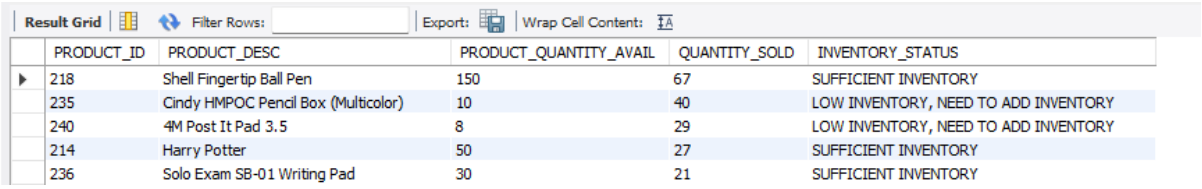
- We order the results by ORDER_VOLUME in descending order to find the largest order.
- The LIMIT 1 clause ensures we return only the **biggest order** that fits into Carton ID 10.

```
-- 8. WRITE A QUERY TO DISPLAY PRODUCT_ID, PRODUCT_DESC, PRODUCT_QUANTITY_AVAIL, QUANTITY SOLD, AND SHOW INVENTORY STATUS OF

-- PRODUCTS AS BELOW AS PER BELOW CONDITION:

        -- A.FOR ELECTRONICS AND COMPUTER CATEGORIES,

                -- i.IF SALES TILL DATE IS ZERO THEN SHOW 'NO SALES IN PAST, GIVE DISCOUNT TO REDUCE INVENTORY',

        -- ii.IF INVENTORY QUANTITY IS LESS THAN 10% OF QUANTITY SOLD, SHOW 'LOW INVENTORY, NEED TO ADD INVENTORY',

        -- iii.IF INVENTORY QUANTITY IS LESS THAN 50% OF QUANTITY SOLD, SHOW 'MEDIUM INVENTORY, NEED TO ADD SOME INVENTORY',

        -- iv.IF INVENTORY QUANTITY IS MORE OR EQUAL TO 50% OF QUANTITY SOLD, SHOW 'SUFFICIENT INVENTORY'

        -- B.FOR MOBILES AND WATCHES CATEGORIES,

                -- i.IF SALES TILL DATE IS ZERO THEN SHOW 'NO SALES IN PAST, GIVE DISCOUNT TO REDUCE INVENTORY',

        -- ii.IF INVENTORY QUANTITY IS LESS THAN 20% OF QUANTITY SOLD, SHOW 'LOW INVENTORY, NEED TO ADD INVENTORY',

        -- iii.IF INVENTORY QUANTITY IS LESS THAN 60% OF QUANTITY SOLD, SHOW 'MEDIUM INVENTORY, NEED TO ADD SOME INVENTORY',

        -- iv.IF INVENTORY QUANTITY IS MORE OR EQUAL TO 60% OF QUANTITY SOLD, SHOW 'SUFFICIENT INVENTORY'

        -- C.REST OF THE CATEGORIES,

                -- i.IF SALES TILL DATE IS ZERO THEN SHOW 'NO SALES IN PAST, GIVE DISCOUNT TO REDUCE INVENTORY',

        -- ii.IF INVENTORY QUANTITY IS LESS THAN 30% OF QUANTITY SOLD, SHOW 'LOW INVENTORY, NEED TO ADD INVENTORY',

        -- iii.IF INVENTORY QUANTITY IS LESS THAN 70% OF QUANTITY SOLD, SHOW 'MEDIUM INVENTORY, NEED TO ADD SOME INVENTORY',
```

-- iv. IF INVENTORY QUANTITY IS MORE OR EQUAL TO 70% OF QUANTITY SOLD, SHOW 'SUFFICIENT INVENTORY'

-- [NOTE: TABLES TO BE USED -PRODUCT, PRODUCT_CLASS, ORDER_ITEMS] (USE SUB-QUERY)

SOLUTION- QUERY8:



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

PRODUCT_ID	PRODUCT_DESC	PRODUCT_QUANTITY_AVAIL	QUANTITY_SOLD	INVENTORY_STATUS
218	Shell Fingertip Ball Pen	150	67	SUFFICIENT INVENTORY
235	Cindy HMPOC Pencil Box (Multicolor)	10	40	LOW INVENTORY, NEED TO ADD INVENTORY
240	4M Post It Pad 3.5	8	29	LOW INVENTORY, NEED TO ADD INVENTORY
214	Harry Potter	50	27	SUFFICIENT INVENTORY
236	Solo Exam SB-01 Writing Pad	30	21	SUFFICIENT INVENTORY

EXPLANATION:

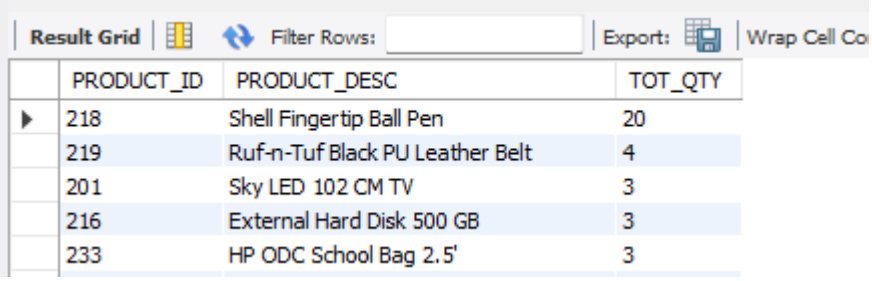
1. **Sales Calculation**:
    - We use SUM() to calculate the total quantity sold for each product.
    - We apply COALESCE(SUM() to handle cases where no sales have been made, and return 0 instead of NULL.
2. **Inventory Status**:
    - The CASE statement first checks which **product category** the product belongs to (Electronics, Computers, Mobiles, Watches, or Others).
    - Based on the product category, the **inventory status** is determined:
        - If sales are 0, display "NO SALES IN PAST, GIVE DISCOUNT TO REDUCE INVENTORY".
        - For different categories, different thresholds for inventory status (low, medium, sufficient) are applied using conditions like:
            - Inventory quantity < 10%, < 50%, etc., of the quantity sold.
3. **LEFT JOIN** with order_items table:
    - We use a LEFT JOIN on the order_items table so that products with no sales still appear in the result with 0 sales.
4. **GROUP BY**:
    - The query groups by the product fields.

-- 9. WRITE A QUERY TO DISPLAY PRODUCT_ID, PRODUCT_DESC AND TOTAL QUANTITY OF PRODUCTS WHICH ARE SOLD TOGETHER WITH PRODUCT ID 201

-- AND ARE NOT SHIPPED TO CITY BANGALORE AND NEW DELHI. DISPLAY THE OUTPUT IN DESCENDING ORDER WITH RESPECT TO TOT_QTY.(USE SUB-QUERY)

-- [NOTE: TABLES TO BE USED - ORDER_ITEMS,PRODUCT,ORDER_HEADER, ONLINE_CUSTOMER, ADDRESS]

SOLUTION- QUERY9:


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

PRODUCT_ID	PRODUCT_DESC	TOT_QTY
218	Shell Fingertip Ball Pen	20
219	Ruf-n-Tuf Black PU Leather Belt	4
201	Sky LED 102 CM TV	3
216	External Hard Disk 500 GB	3
233	HP ODC School Bag 2.5'	3

EXPLANATION:

1. **Inner Joins:**
    - We join the necessary tables:
        - order_items (oi) to get product details related to each order.
        - product (p) to retrieve the product description.
        - order_header (oh) to link orders to customers.
        - online_customer (oc) and address (a) to get the shipping address for each order.
2. **Subquery:**
    - This subquery retrieves all the ORDER_IDs where **Product ID 201** was sold. We use this to find all orders that included Product ID 201.
3. **Exclusion of Product ID 201 (AND p.PRODUCT_ID <> 201):**
    - After finding the orders with **Product ID 201**, we make sure that we are only interested in **other products** that were sold together with Product ID 201.
4. **Exclusion of Cities**
    - We filter out orders that were shipped to **Bangalore** or **New Delhi**.
5. **Grouping and Summing**

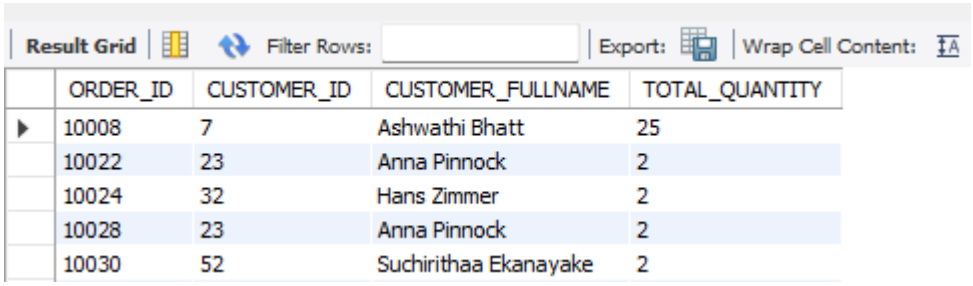
6. **Order by Total Quantity:**
    - We sort the results in descending order based on the total quantity of each product.

-- 10. WRITE A QUERY TO DISPLAY THE ORDER_ID,CUSTOMER_ID AND CUSTOMER FULLNAME AND TOTAL QUANTITY OF PRODUCTS SHIPPED FOR ORDER IDS

-- WHICH ARE EVENAND SHIPPED TO ADDRESS WHERE PINCODE IS NOT STARTING WITH "5"

-- [NOTE: TABLES TO BE USED - ONLINE_CUSTOMER,ORDER_HEADER, ORDER_ITEMS, ADDRESS]

SOLUTION- QUERY10:



ORDER_ID	CUSTOMER_ID	CUSTOMER_FULLNAME	TOTAL_QUANTITY
10008	7	Ashwathi Bhatt	25
10022	23	Anna Pinnock	2
10024	32	Hans Zimmer	2
10028	23	Anna Pinnock	2
10030	52	Suchirithaa Ekanayake	2

EXPLANATION:

1. **Inner Joins:**
    - We join the necessary tables:
        - order_items, product, order_header, online_customer.
2. **WHERE CLAUSE:**
    - For fetching even order id and pin code not starting with 5.
3. **Grouping**
    - For getting the desired result in the order.

NOTE: RESULT FROM THE QUERY CAN BE LIMITED TO 5 USING LIMIT 5. Just for the screenshot we have attached 5 top rows.