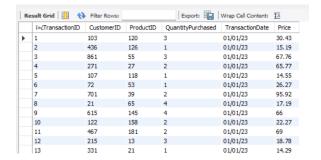
### Dataset Tables: -

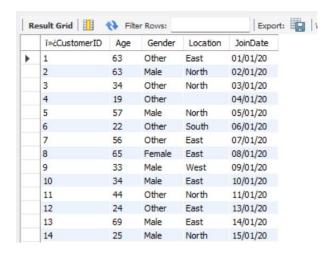
Sales transaction, customer profile, product inventory

## Create database retail analytics

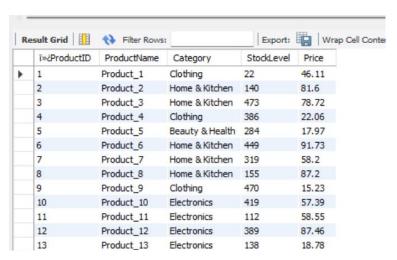
Select \* from sales\_transaction



Select \* from customer profile



Select \* from product\_inventory



1. Write a query to identify the number of duplicates in "sales\_transaction" table. Also, create a separate table containing the unique values and remove the original table from the databases and replace the name of the new table with the original name.

select TransactionID,count(\*)

from Sales\_transaction

group by TransactionID

having count(\*)>1;

create table Sales\_transaction\_copy

as select distinct \* from Sales\_transaction;

drop table Sales\_transaction;

alter table Sales\_transaction\_copy

rename to Sales\_transaction;

# select \* from Sales\_transaction;

#### Output

-					
<b></b>	++				
TransactionID	count(*)				
4999	++   2				
5000	2				
	++				
TransactionID	CustomerID		   QuantityPurchased	TransactionDate	Price
	+		·		++
1	103	120	3	2023-01-01	30.43
2	436	126	1	2023-01-01	15.19
3	861	55	3	2023-01-01	67.76
4	271	27	2	2023-01-01	65.77
5	107	118	1	2023-01-01	14.55
6	72	53	1	2023-01-01	26.27
7	701	39	2	2023-01-01	95.92
Q	j 21	65	4	2023-01-01	17 19 İ

2. Write a query to identify the discrepancies in the price of the same product in "sales\_transaction" and "product\_inventory" tables. Also, update those discrepancies to match the price in both the tables.

select s.Transactionid,s.price as Transactionprice,p.price as Inventoryprice

from sales\_transaction as s

join product\_inventory as p

on p.productid=s.productid

where s.price != p.price;

update sales\_transaction

set price=93.12

where productid=51;

# select \* from sales\_transaction;

+	<b></b>	<b>+</b>
Transactionid	Transactionprice	Inventoryprice
+	<b>+</b>	<b></b>
4968	9312	93.12
4754	9312	93.12
4532	9312	93.12
4408	9312	93.12
4221	9312	93.12
4158	9312	93.12
4148	9312	93.12
3962	9312	93.12
3959	9312	93.12

+	+	+	<u> </u>	+	++
TransactionID	CustomerID	ProductID	QuantityPurchased	TransactionDate	Price
+	+	++	H	+	++
1	103	120	3	2023-01-01	30.43
2	436	126	1	2023-01-01	15.19
3	861	55	3	2023-01-01	67.76
4	271	27	2	2023-01-01	65.77
5	107	118	1	2023-01-01	14.55
6	72	53	1	2023-01-01	26.27
7	701	39	2	2023-01-01	95.92
8	21	65	4	2023-01-01	17.19
9	615	145	4	2023-01-01	66
10	122	158	2	2023-01-01	22.27
11	467	181	2	2023-01-01	69
12	215	13	3	2023-01-01	18.78
j 13	331	21	1	2023-01-01	14.29

3. Write a SQL query to identify the null values in the dataset and replace those by "Unknown".

select count(\*) from customer\_profiles

where location is null;

update customer profiles set location="unknown"

where location is null;

select \* from customer\_profiles;

++				
count(*)				
++				
13				
1 12				
				l a . p . l
CustomerID	Age	Gender	Location	JoinDate
+	+	+	+	++
1	63	Other	East	01/01/20
2	63	Male	North	02/01/20
3	34	Other	North	03/01/20
4	19	Other	unknown	04/01/20
5	57	Male	North	05/01/20
6	22	Other	South	06/01/20
7	56	Other	East	07/01/20
8	65	Female	East	08/01/20
9	33	Male	West	09/01/20

4. Write a SQL query to clean the DATE column in the dataset.

create table Sales\_transaction\_copy as

select TransactionID, CustomerID, ProductID, QuantityPurchased,

str\_to\_date(transactiondate,"%Y-%m-%d") as TransactionDate,

Price, str to date(transactiondate, "%Y-%m-%d") as TransactionDate updated

from Sales\_transaction;

drop table Sales transaction;

alter table Sales\_transaction\_copy

rename to Sales\_transaction;

select \* from Sales\_transaction;

+	L		<b></b>	<b>.</b>	·	<b></b>
TransactionID	CustomerID	ProductID	QuantityPurchased	TransactionDate	Price	TransactionDate_updated
1	103	120	3	2023-01-01	30.43	2023-01-01
2	436	126	1	2023-01-01	15.19	2023-01-01
3	861	55	3	2023-01-01	67.76	2023-01-01
4	271	27	2	2023-01-01	65.77	2023-01-01
5	107	118	1	2023-01-01	14.55	2023-01-01
6	72	53	1	2023-01-01	26.27	2023-01-01
7	701	39	2	2023-01-01	95.92	2023-01-01
8	21	65	4	2023-01-01	17.19	2023-01-01
9	615	145	4	2023-01-01	66	2023-01-01
10	122	158	2	2023-01-01	22.27	2023-01-01

5. Write a SQL query to summarize the total sales and quantities sold per product by the company.

select ProductID,sum(QuantityPurchased) as TotalUnitsSold,

sum(QuantityPurchased \*price) as TotalSales

from Sales\_transaction

group by ProductID

order by sum(QuantityPurchased \*price) desc;

ProductID	TotalUnitsSold	TotalSales
17	100	9450
87	92	7817.239999999998
179	86	7388.259999999998
96	72	7132.32000000000015
54	86	7052.86000000000015
187	82	6915.8800000000003
156	76	6827.8400000000002
57	78	6622.199999999999
200	69	6479.7900000000001
127	68	6415.799999999999
28	69	6386.6400000000001
106	63	6262.829999999999
104	72	6230.1600000000001

6. Write a SQL query to count the number of transactions per customer to understand purchase frequency.

select CustomerID,count(TransactionID) as NumberOfTransactions

from Sales\_transaction

group by CustomerID

order by count(TransactionID) desc;

+	NumberOfTransactions
Cuscomer 10	Number of H alisaccions
664	14
958	12
99	12
113	12
929	12
936	12
670	12
39	12
277	11
476	11
776	11
727	11
648	11
613	11

7. Write a SQL query to evaluate the performance of the product categories based on the total sales which help us understand the product categories which needs to be promoted in the marketing campaigns.

select p.Category,sum(s.QuantityPurchased) as TotalUnitsSold,

sum(s.QuantityPurchased \* s.Price) as TotalSales

from Sales\_transaction s

join product\_inventory p

on s.productID=p.productID

group by p.Category

order by TotalSales desc;

+	<b></b>	++
Category	TotalUnitsSold	TotalSales
+	+	++
Home & Kitchen	3477	217755.93999999945
Electronics	3037	177548.4799999996
Clothing	2810	162874.210000000057
Beauty & Health	3001	143824.98999999947
+	+	++

8. Write a SQL query to find the top 10 products with the highest total sales revenue from the sales transactions. This will help the company to identify the High sales products which needs to be focused to increase the revenue of the company.

select ProductID,sum(QuantityPurchased \* Price) as TotalRevenue

from Sales\_transaction

group by ProductID

order by TotalRevenue desc

limit 10;

+    ProductID	TotalRevenue
17	9450
87	7817.23999999999
179	7388.25999999999
96	7132.32000000000015
54	7052.8600000000015
187	6915.880000000003
156	6827.840000000000
57	6622.199999999999
200	6479.7900000000001
127	6415.799999999999
+	++

9. Write a SQL query to identify the sales trend to understand the revenue pattern of the company.

select str\_to\_date(transactiondate,"%Y-%m-%d") as DATETRANS,

count(TransactionID) as Transaction\_count,sum(QuantityPurchased) as TotalUnitsSold,

sum(QuantityPurchased \* Price) as TotalSales

from sales\_transaction

group by str\_to\_date(transactiondate,"%Y-%m-%d")

### order by DATETRANS desc;

4		<b>.</b>	
DATETRANS	Transaction_count	TotalUnitsSold	TotalSales
+		+	·
2023-07-28	8	18	1158.86000000000001
2023-07-27	24	58	3065.809999999999
2023-07-26	24	58	3168.04000000000004
2023-07-25	24	54	2734.26
2023-07-24	24	63	3691.079999999999
2023-07-23	24	57	3578.58000000000004
2023-07-22	24	62	3350.8
2023-07-21	24	61	3443.72
2023-07-20	24	60	3216.57
2023-07-19	24	52	2068.50000000000005
2023-07-18	24	57	3251.0699999999997
2023-07-17	24	56	3051.5099999999998
2023-07-16	24	66	3145.46

10. Write a SQL query to understand the month on month growth rate of sales of the company which will help understand the growth trend of the company.

WITH monthly\_sales AS (

SELECT MONTH(transactiondate\_updated) AS month,

ROUND(SUM(QuantityPurchased \* Price), 2) AS total\_sales

FROM sales\_transaction

GROUP BY MONTH(transactiondate\_updated)),

growth\_calc AS (

SELECT month,total\_sales,

LAG(total\_sales) OVER (ORDER BY month) AS previous\_month\_sales

FROM monthly\_sales)

SELECT month,total\_sales,previous\_month\_sales,

ROUND(((total\_sales - previous\_month\_sales) / previous\_month\_sales) \* 100, 2) AS

mom\_growth\_percentage

FROM growth\_calc

### ORDER BY month;

++			·
month	total_sales	previous_month_sales	mom_growth_percentage
+			
1	104289.18	NULL	NULL
2	96690.99	104289.18	-7.29
3	103271.49	96690.99	6.81
4	101561.09	103271.49	-1.66
5	102998.84	101561.09	1.42
6	102210.28	102998.84	-0.77
7	90981.75	102210.28	-10.99
+			·

11. Write a SQL query that describes the number of transactions along with the total amount spent by each customer, which will help us understand the customers who are occasional customers or have low purchase frequency in the company.

select CustomerID, count(TransactionID) as NumberOfTransactions,

sum(QuantityPurchased \* Price) as TotalSpent

from Sales\_transaction

group by CustomerID

having count(TransactionID)<=2

order by

NumberOfTransactions asc,

### TotalSpent desc;

+		·
CustomerID	NumberOfTransactions	TotalSpent
+		·
94	1	360.64
181	1	298.23
979	1	265.16
317	1	257.73
479	1	254.91
799	1	254.700000000000000
45	1	241.350000000000000
110	1	236.16
169	1	230.37
706	1	224.49
965	1	215.72
212	1	203.9699999999999
333	1	189
603	1	171.56

12. Write a SQL query that describes the duration between the first and the last purchase of the customer in that particular company to understand the loyalty of the customer.

select CustomerID,min(TransactionDate\_updated) as FirstPurchase,

max(TransactionDate\_updated) as LastPurchase,

datediff(max(TransactionDate\_updated), min(TransactionDate\_updated)) as DaysBetweenPurchases

from Sales\_transaction

group by CustomerID

having DaysBetweenPurchases > 0

### order by DaysBetweenPurchases desc;

+	+	+	++
CustomerID	FirstPurchase	LastPurchase	DaysBetweenPurchases
+	+	+	++
215	2023-01-01	2023-07-28	208
414	2023-01-02	2023-07-26	205
664	2023-01-01	2023-07-24	204
701	2023-01-01	2023-07-23	203
277	2023-01-02	2023-07-24	203
22	2023-01-02	2023-07-24	203
976	2023-01-02	2023-07-24	203
647	2023-01-03	2023-07-25	203
162	2023-01-05	2023-07-27	203
806	2023-01-02	2023-07-23	202
511	2023-01-02	2023-07-23	202
703	2023-01-05	2023-07-26	202
188	2023-01-06	2023-07-27	202

13. Write an SQL query that segments customers based on the total quantity of products they have purchased. Also, count the number of customers in each segment which will help us target a particular segment for marketing.

with customertotalQuantity as (

select cp.CustomerID, sum(st.QuantityPurchased) as TotalQuantity

from sales\_transaction st

join customer\_profiles cp

on st.CustomerID=cp.CustomerID

group by cp.CustomerID),

Cust\_Seg as(

select CustomerID,

case

when TotalQuantity<=10 then "Low"

when TotalQuantity<=30 then "Med"

else "High"

end as CustomerSegment

from customertotalQuantity)

select CustomerSegment,count(\*)

from Cust\_Seg

group by CustomerSegment;

CustomerSegment	count(*)
Med	559
Low	423
High	7