Case Study Retail Analytics

* **Remove Duplicates**

**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 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\_Updated as

select distinct \*

from Sales\_transaction;

select \* from Sales\_transaction\_Updated;

drop table Sales\_transaction;

alter table Sales\_transaction\_Updated

rename to Sales\_transaction;

* **Fix Incorrect Price**

**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 st.TransactionId,

st.price as TransactionPrice,

pi.price as InventoryPrice

from sales\_transaction as st

join product\_inventory as pi

on st.ProductID = pi. ProductID

where st.Price!= pi.Price;

update sales\_transaction st

set Price = (

select p.price from

product\_inventory p

where p.ProductID =st.ProductID

)

where ProductID in(

select p.ProductID from product\_inventory p

where st.Price!= p.Price

);

select \* from sales\_transaction;

* **Fixing Null Values**

**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;

* **Cleaning Date**

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

create table Sales\_transaction\_update as

select \*,

CAST(TransactionDate as DATE) TransactionDate\_Updated

from Sales\_transaction;

select \* from Sales\_transaction\_update;

DROP table Sales\_transaction;

ALTER TABLE

Sales\_transaction\_update rename

TO Sales\_transaction;

**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;

* **Customer Purchase Frequency**

**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;

* **Product Categories Performance**

**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 pi.Category,

sum(st.QuantityPurchased) as TotalUnitsSold,

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

from Sales\_transaction st

join product\_inventory pi

on st.ProductID = pi.ProductID

group by pi.Category

ORDER BY sum(st.QuantityPurchased\*st.Price)

DESC;

* **High SalesProducts**

**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(Price\* QuantityPurchased) AS TotalRevenue

from Sales\_transaction

group by ProductID

ORDER BY TotalRevenue desc

limit 10;

* **Low SalesProducts**

**Write a SQL query to find the ten products with the least amount of units sold from the sales transactions,**

**provided that at least one unit was sold for those products.**

select ProductID,

sum(QuantityPurchased) as TotalUnitsSold

from Sales\_transaction

group by ProductID

HAVING sum(QuantityPurchased)>=1

order by TotalUnitsSold

limit 10;

* **Sales Trend**

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

select TransactionDate\_updated AS

DATETRANS,

count(TransactionID) as Transaction\_count,

SUM(QuantityPurchased) as TotaLUnitsSold,

SUM(QuantityPurchased \*Price) as TotalSales

from sales\_transaction

group by 1

order by DATETRANS desc;

* **Growth Rate of Sales**

**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 cte as(

select \*,

lag(total\_sales,1) over (order by month)

as previous\_month\_sales,

(total\_sales -

lag(total\_sales,1,0) over (order by month))/

lag(total\_sales,1,0) over (order by month)\*100

as mom\_growth\_percentage

from

(

select month(CAST(TransactionDate as DATE)) as month,

sum(QuantityPurchased\*Price) as total\_sales

from sales\_transaction

group by 1

) abc

)

select \* from cte;

* **High Purchase Frequency**

**Write a SQL query that describes the number of transaction along with the total amount spent by each customer which are on the higher side and will help us understand the customers who are the high frequency purchase customers in the company.**

select CustomerID,

count(TransactionId) as

NumberOfTransactions,

sum(QuantityPurchased\*Price) as TotalSpent

from sales\_transaction

group by 1

HAVING count(TransactionID) >10 AND

sum(QuantityPurchased\*Price)>1000

order by TotalSpent desc;

* **Occasional customers**

**Write a SQL query that describes the number of transaction along with the total amount spent by each customer, which will help us understand the customers who are occasional 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 NumberOfTransactions<=2

order by NumberOfTransactions,TotalSpent desc;

* **Repeat Purchase**

**Problem statement**

**Write a SQL query that describes the total number of purchases made by each customer against each productID to understand the repeat customers in the company.**

select customerId,

ProductID,

count(Transactionid) as TimesPurchased

from Sales\_transaction

group by 1,2

having count(Transactionid)>1

order by TimesPurchased desc;

**Problem statement**

**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(STR\_TO\_DATE(TransactionDate, '%Y-%m-%d')) AS FirstPurchase,

MAX(STR\_TO\_DATE(TransactionDate, '%Y-%m-%d')) AS LastPurchase,

DATEDIFF(

MAX(STR\_TO\_DATE(TransactionDate, '%Y-%m-%d')),

MIN(STR\_TO\_DATE(TransactionDate, '%Y-%m-%d'))

) AS DaysBetweenPurchases

FROM Sales\_transaction

GROUP BY CustomerID

HAVING DaysBetweenPurchases > 0

ORDER BY DaysBetweenPurchases DESC;

**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.**

Create Table Customer\_Segment as

select CustomerID,

CASE WHEN TotalQuantity>30 Then"High"

WHEN TotalQuantity Between 10 AND 30 Then "Med"

WHEN TotalQuantity Between 1 AND 10 Then "Low"

else "None"

END as CustomerSegment

FROM

(

select cp.CustomerID,

sum(st.QuantityPurchased) as TotalQuantity

from customer\_profiles cp

join sales\_transaction st

on cp.CustomerID = st.CustomerID

group by cp.CustomerID) a;

select CustomerSegment,count(\*)

from Customer\_Segment

group by CustomerSegment;