RETAIL ANALYTICS CASE STUDY

BY. SURAJIT BAL

INTRODUCING

SURAJIT BAL

Data Analyst with a proven track record of analyzing complex data sets, identifying trends, and providing actionable insights to drive business decisions. Skilled in statistical analysis, data visualization, and data mining techniques. Proficient in SQL, Python, and Excel. Strong problem-solving abilities and a keen attention to detail.

OVERVIEW

IN THE RAPIDLY EVOLVING RETAIL SECTOR, BUSINESSES CONTINUALLY SEEK INNOVATIVE STRATEGIES TO STAY AHEAD OF THE COMPETITION, IMPROVE CUSTOMER SATISFACTION, AND OPTIMIZE OPERATIONAL EFFICIENCY. LEVERAGING DATA ANALYTICS HAS BECOME A CORNERSTONE FOR ACHIEVING THESE OBJECTIVES. THIS CASE STUDY FOCUSES ON A RETAIL COMPANY THAT HAS ENCOUNTERED CHALLENGES IN UNDERSTANDING ITS SALES PERFORMANCE, CUSTOMER ENGAGEMENT, AND INVENTORY MANAGEMENT. THROUGH A COMPREHENSIVE DATA ANALYSIS APPROACH, THE COMPANY AIMS TO IDENTIFY HIGH OR LOW SALES PRODUCTS, EFFECTIVELY SEGMENT ITS CUSTOMER BASE, AND ANALYZE CUSTOMER BEHAVIOR TO ENHANCE MARKETING STRATEGIES, INVENTORY DECISIONS, AND OVERALL CUSTOMER EXPERIENCE.

BUSINESS PROBLEM

THE RETAIL COMPANY HAS OBSERVED STAGNANT GROWTH AND DECLINING CUSTOMER ENGAGEMENT METRICS OVER THE PAST QUARTERS. INITIAL ASSESSMENTS INDICATE POTENTIAL ISSUES IN PRODUCT PERFORMANCE VARIABILITY, INEFFECTIVE CUSTOMER SEGMENTATION, AND LACK OF INSIGHTS INTO CUSTOMER PURCHASING BEHAVIOR. THE COMPANY SEEKS TO LEVERAGE ITS SALES TRANSACTION DATA, CUSTOMER PROFILES, AND PRODUCT INVENTORY INFORMATION TO ADDRESS THE FOLLOWING KEY BUSINESS PROBLEMS:

- **PRODUCT PERFORMANCE VARIABILITY**: IDENTIFYING WHICH PRODUCTS ARE PERFORMING WELL IN TERMS OF SALES AND WHICH ARE NOT. THIS INSIGHT IS CRUCIAL FOR INVENTORY MANAGEMENT AND MARKETING FOCUS.

- **CUSTOMER SEGMENTATION**: THE COMPANY LACKS A CLEAR UNDERSTANDING OF ITS CUSTOMER BASE SEGMENTATION. EFFECTIVE SEGMENTATION IS ESSENTIAL FOR TARGETED MARKETING AND ENHANCING CUSTOMER SATISFACTION.

- CUSTOMER BEHAVIOUR ANALYSIS: UNDERSTANDING PATTERNS IN CUSTOMER BEHAVIOR, INCLUDING REPEAT PURCHASES AND LOYALTY INDICATORS, IS CRITICAL FOR TAILORING CUSTOMER ENGAGEMENT STRATEGIES AND IMPROVING RETENTION RATES.

OBJECTIVES

- TO UTILIZE SQL QUERIES FOR DATA CLEANING AND EXPLORATORY DATA ANALYSIS TO ENSURE DATA QUALITY AND GAIN INITIAL INSIGHTS.
- TO IDENTIFY HIGH AND LOW SALES PRODUCTS TO OPTIMIZE INVENTORY AND TAILOR MARKETING EFFORTS.
- TO SEGMENT CUSTOMERS BASED ON THEIR PURCHASING BEHAVIOR FOR TARGETED MARKETING CAMPAIGNS. CREATE CUSTOMER SEGMENTS BASED ON ‘TOTAL QUANTITY OF PRODUCT PURCHASED’.
- TO ANALYZE CUSTOMER BEHAVIOR FOR INSIGHTS ON REPEAT PURCHASES AND LOYALTY, INFORMING CUSTOMER RETENTION STRATEGIES.

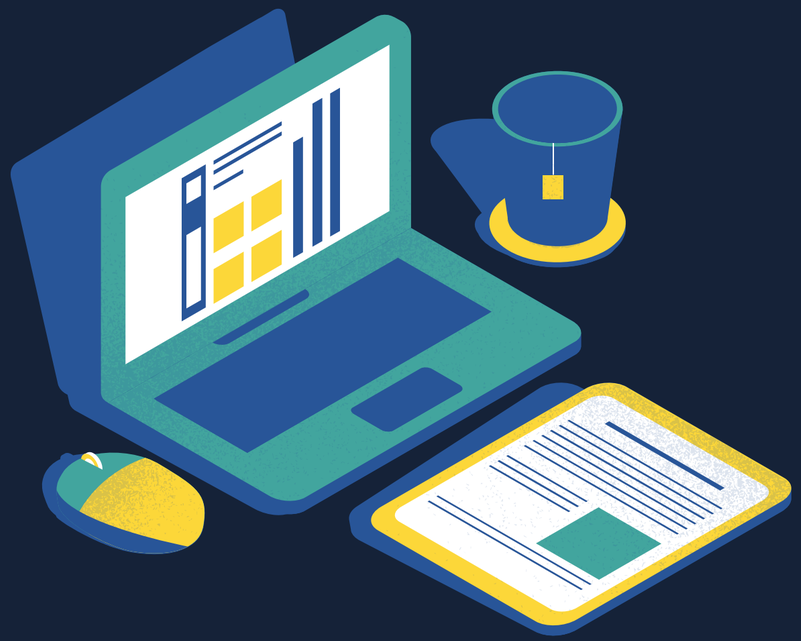
DATASETS TABLES

- **SALES TRANSACTION:** RECORDS OF SALES TRANSACTIONS, INCLUDING TRANSACTION ID, CUSTOMER ID, PRODUCT ID, QUANTITY PURCHASED, TRANSACTION DATE, AND PRICE.

- **CUSTOMER PROFILES:** INFORMATION ON CUSTOMERS, INCLUDING CUSTOMER ID, AGE, GENDER, LOCATION, AND JOIN DATE.

- **PRODUCT INVENTORY:** DATA ON PRODUCT INVENTORY, INCLUDING PRODUCT ID, PRODUCT NAME, CATEGORY, STOCK LEVEL, AND PRICE.

DATA CLEANING
&
ANALYSIS
IN
MYSQL



PROBLEM STATEMENT:

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.

```sql
select TransactionID, count(*) from sales_transaction group by 1 having count(*) > 1;
create table sales_transaction1 like sales_transaction;
insert into sales_transaction1 (select distinct * from sales_transaction);
drop table sales_transaction;
alter table sales_transaction1 rename to sales_transaction;
select * from sales_transaction;
```

TransactionID	count(*)
4999	2
5000	2

TransactionID	CustomerID	ProductID	QuantityPurchas...	TransactionDate	Price
1	103	120	3	01/01/23	30.43
2	436	126	1	01/01/23	15.19
3	861	55	3	01/01/23	67.76
4	271	27	2	01/01/23	65.77
5	107	118	1	01/01/23	14.55
6	72	53	1	01/01/23	26.27
7	701	39	2	01/01/23	95.92
8	21	65	4	01/01/23	17.19
9	615	145	4	01/01/23	66
10	122	158	2	01/01/23	22.27
11	467	181	2	01/01/23	69

PROBLEM STATEMENT:

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.

```sql
select TransactionID, t.price TransactionPrice, p.price InventoryPrice
from sales_transaction t INNER JOIN product_inventory p using(ProductID)
where t.price <> p.price;

update sales_transaction s INNER JOIN product_inventory p using(ProductID)
set s.price = p.price where s.price <> p.price;

select * from sales_transaction;
```

TransactionID	TransactionPrice	InventoryPrice
88	9312	93.12
236	9312	93.12
591	9312	93.12
1377	9312	93.12
1910	9312	93.12
2608	9312	93.12
2939	9312	93.12
3377	9312	93.12
3635	9312	93.12
3839	9312	93.12
3918	9312	93.12

TransactionID	CustomerID	ProductID	QuantityPurchas...	TransactionDate	Price
1	103	120	3	01/01/23	30.43
2	436	126	1	01/01/23	15.19
3	861	55	3	01/01/23	67.76
4	271	27	2	01/01/23	65.77
5	107	118	1	01/01/23	14.55
6	72	53	1	01/01/23	26.27
7	701	39	2	01/01/23	95.92
8	21	65	4	01/01/23	17.19
9	615	145	4	01/01/23	66
10	122	158	2	01/01/23	22.27
11	467	181	2	01/01/23	69

PROBLEM STATEMENT:

WRITE A SQL QUERY TO IDENTIFY THE BLANKS AND NULL VALUES IN THE LOCATION COLUMN AND REPLACE THOSE BY “UNKNOWN”.

```sql
SELECT count(*) from customer_profiles WHERE location IS NULL OR Location = '';
UPDATE customer_profiles
SET Location = "Unknown" WHERE Location IS NULL OR Location = '';

select * from customer_profiles;
```

count(*)
13

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
10	34	Male	East	10/01/20
11	44	Other	North	11/01/20

PROBLEM STATEMENT:

WRITE A SQL QUERY TO CLEAN THE DATE COLUMN IN THE DATASET.

```sql
alter table Sales_transaction add column TransactionDate_updated date;
update Sales_transaction
set TransactionDate_updated = str_to_date(TransactionDate, "%d/%m/%y");

select * from Sales_transaction;
```

TransactionID	CustomerID	ProductID	QuantityPurchas...	TransactionDate	Price	TransactionDate_updat...
1	103	120	3	01/01/23	30.43	2023-01-01
2	436	126	1	01/01/23	15.19	2023-01-01
3	861	55	3	01/01/23	67.76	2023-01-01
4	271	27	2	01/01/23	65.77	2023-01-01
5	107	118	1	01/01/23	14.55	2023-01-01
6	72	53	1	01/01/23	26.27	2023-01-01
7	701	39	2	01/01/23	95.92	2023-01-01
8	21	65	4	01/01/23	17.19	2023-01-01
9	615	145	4	01/01/23	66	2023-01-01

PROBLEM STATEMENT:
WRITE A SQL QUERY TO SUMMARIZE THE TOTAL SALES AND QUANTITIES SOLD PER PRODUCT BY THE COMPANY.

```sql
select ProductID, sum(QuantityPurchased) TotalUnitsSold,
sum(QuantityPurchased * Price) TotalSales
from Sales_transaction
Group by ProductID order by TotalSales desc;
```

ProductID	TotalUnitsSo...	TotalSales
17	100	9450
87	92	7817.239999999998
179	86	7388.259999999998
96	72	7132.320000000015
54	86	7052.860000000015
187	82	6915.880000000003
156	76	6827.840000000002
57	78	6622.199999999999
200	69	6479.790000000001

PROBLEM STATEMENT:
WRITE A SQL QUERY TO COUNT THE NUMBER OF TRANSACTIONS PER CUSTOMER TO UNDERSTAND PURCHASE FREQUENCY.

```sql
select CustomerID, count(TransactionID) NumberOfTransactions
from Sales_transaction
Group by CustomerID order by NumberOfTransactions desc;
```

CustomerID	NumberOfTransactio...
664	14
958	12
99	12
113	12
929	12
936	12
670	12
39	12
277	11

PROBLEM STATEMENT:

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.

```sql
select p.Category as Category, sum(s.QuantityPurchased) as TotalUnitsSold,
sum(s.QuantityPurchased * s.Price) as TotalSales
from product_inventory p INNER JOIN Sales_transaction s using(ProductID)
Group by p.Category Order by TotalSales desc;
```

Category	TotalUnitsSo...	TotalSales
Home & Kitchen	3477	217755.94000000026
Electronics	3037	177548.48000000007
Clothing	2810	162874.21000000005
Beauty & Health	3001	143824.98999999947

PROBLEM STATEMENT:

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.

```sql
select ProductID, sum(QuantityPurchased * Price) TotalRevenue
from Sales_transaction
Group by ProductID Order by TotalRevenue desc limit 10;
```

ProductID	TotalRevenue
17	9450
87	7817.239999999998
179	7388.259999999998
96	7132.3200000000015
54	7052.8600000000015
187	6915.880000000003
156	6827.840000000002
57	6622.199999999999
200	6479.790000000001

PROBLEM STATEMENT:
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.

```sql
select ProductID, sum(QuantityPurchased) TotalUnitsSold
from Sales_transaction
Group by ProductID Having TotalUnitsSold >= 1 Order by TotalUnitsSold  limit 10;
```

ProductID	TotalUnitsSo...
142	27
33	31
174	33
159	35
60	35
41	35
91	35
198	36
124	39

PROBLEM STATEMENT:

WRITE A SQL QUERY TO IDENTIFY THE SALES TREND TO UNDERSTAND THE REVENUE PATTERN OF THE COMPANY.

```sql
select TransactionDate_updated DATETRANS, count(TransactionID) Transaction_count,
sum(QuantityPurchased) TotalUnitsSold, sum(QuantityPurchased * Price) TotalSales
from sales_transaction
Group by DATETRANS order by DATETRANS desc;
```

DATETRANS	Transaction_cou...	TotalUnitsSo...	TotalSales
2023-07-28	8	18	1158.8600000000001
2023-07-27	24	58	3065.8099999999995
2023-07-26	24	58	3168.0400000000004
2023-07-25	24	54	2734.26
2023-07-24	24	63	3691.079999999999
2023-07-23	24	57	3578.5800000000004
2023-07-22	24	62	3350.8
2023-07-21	24	61	3443.72
2023-07-20	24	60	3216.57

PROBLEM STATEMENT:

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 month(TransactionDate) month, sum(QuantityPurchased * Price) total_sales,
lag(sum(QuantityPurchased * Price)) over(order by month(TransactionDate))
as previous_month_sales
from sales_transaction
group by month)
select *,
(((total_sales - previous_month_sales) / previous_month_sales)) * 100  mom_growth_percentage
from CTE;
```

month	total_sales	previous_month_sal...	mom_growth_percent...
1	104289.17999999993	NULL	NULL
2	96690.98999999995	104289.17999999993	-7.285693491884769
3	103271.49	96690.98999999995	6.805701337839299
4	101561.09000000014	103271.49	-1.656217025628141
5	102998.83999999995	101561.09000000014	1.4156504228142972
6	102210.28	102998.83999999995	-0.7656008553105592
7	90981.75000000004	102210.28	-10.985714939827927

PROBLEM STATEMENT:

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.

```sql
select CustomerID, count(TransactionID) NumberOfTransactions, sum(QuantityPurchased * Price) TotalSpent
from sales_transaction
Group by CustomerID
having NumberOfTransactions > 10 and TotalSpent > 1000
order by TotalSpent desc;
```

CustomerID	NumberOfTransactio...	TotalSpent
936	12	2834.4700000000003
664	14	2519.04
670	12	2432.15
39	12	2221.29
958	12	2104.71
75	11	1862.7299999999998
476	11	1821.4399999999998
929	12	1798.42
881	11	1713.2300000000002
704	11	1628.34
648	11	1572.9999999999998
776	11	1551.0100000000002

PROBLEM STATEMENT:

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 CUSTOMERS OR HAVE LOW PURCHASE FREQUENCY IN THE COMPANY.

```sql
select CustomerID, count(TransactionID) NumberOfTransactions, sum(QuantityPurchased * Price) TotalSpent
from Sales_transaction
Group by CustomerID having NumberOfTransactions <= 2
order by NumberOfTransactions asc, TotalSpent desc ;
```

CustomerID	NumberOfTransactio...	TotalSpent
94	1	360.64
181	1	298.23
979	1	265.16
317	1	257.73
479	1	254.91
799	1	254.70000000000002
45	1	241.35000000000002
110	1	236.16
169	1	230.37
706	1	224.49
965	1	215.72
212	1	203.96999999999997
333	1	189

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.

```sql
select CustomerID, ProductID, count(TransactionID) TimesPurchased
from Sales_transaction
Group by CustomerID, ProductID
having TimesPurchased > 1 order by TimesPurchased desc;
```

CustomerID	ProductID	TimesPurchased
685	192	3
215	13	2
492	74	2
242	172	2
822	165	2
296	196	2
613	44	2
225	75	2
710	156	2

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.

```sql
update Sales_transaction
set TransactionDate = str_to_date(TransactionDate, "%d/%m/%y");

select CustomerID,
min(TransactionDate) FirstPurchase, max(TransactionDate) LastPurchase,
datediff(max(TransactionDate), min(TransactionDate)) DaysBetweenPurchases
from Sales_transaction
Group by CustomerID
having DaysBetweenPurchases > 0
order by DaysBetweenPurchases desc;
```

CustomerID	FirstPurchase	LastPurchase	DaysBetweenPurchas...
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

PROBLEM STATEMENT:

WRITE A 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.

```sql
create table Cust_segment as
Select
    CASE WHEN qnt between 1 and 9 THEN "Low"
    WHEN qnt between 10 and 30 THEN "Med"
    WHEN qnt > 30 THEN "High"
    END as CustomerSegment, COUNT(*)
from (select c.CustomerID, sum(s.QuantityPurchased) as qnt
from Sales_transaction s INNER JOIN Customer_profiles c using(CustomerID)
Group by c.CustomerID) as xyz
Group by CustomerSegment;

select * from Cust_segment;
```

CustomerSegme...	COUNT(*)
Med	627
Low	355
High	7

THANK YOU!