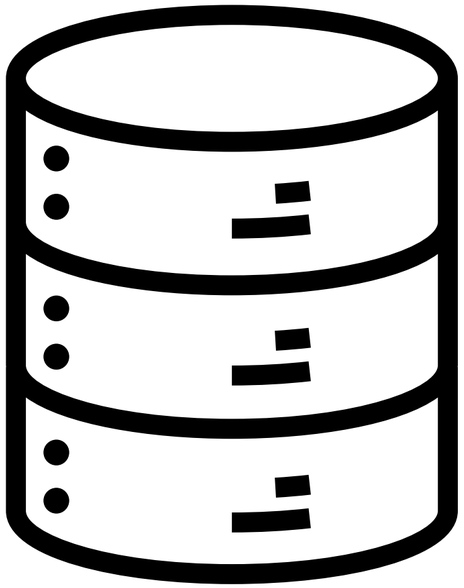
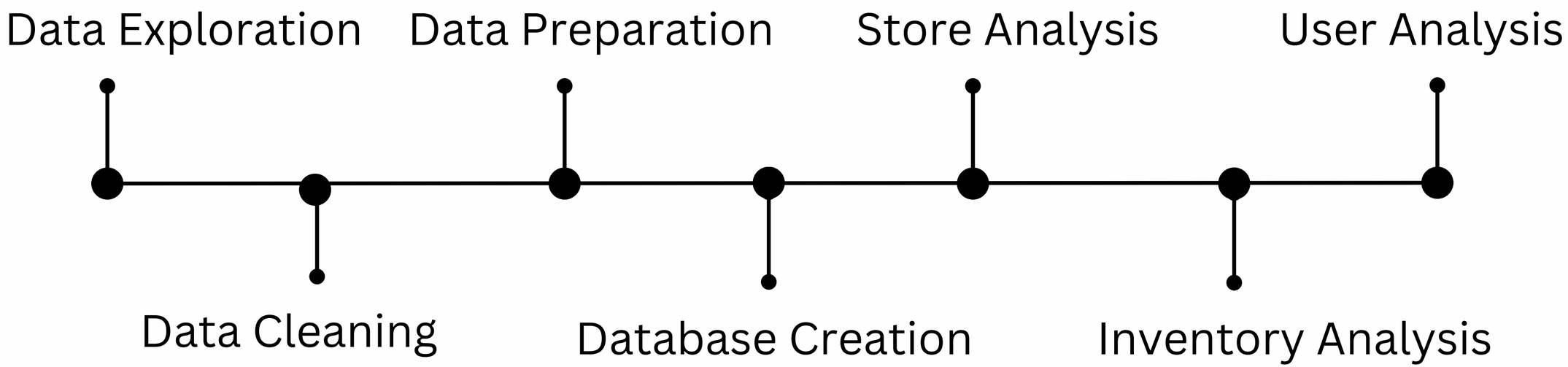


BUSINESS DATA MANAGEMENT

GROUP 2

SHIV AGARWAL JHALAK AGARWAL KYLLE CHANDRAN SNEH SMRITI

ANALYSIS STEPS



EXPLORING DATABASE INSIGHTS

- The analysis is based on a comprehensive retail database with tables including stores, inventory, bill details, user bills, and bill descriptions.
- The goal is to derive meaningful insights from the data to inform business strategies.

STRUCTURE

- The **store** table holds information about store names and their corresponding zones.
- The **inventory** table contains details about products, including product ID, description, category, color, size, and price.
- The **bill_details** table records transaction details such as bill ID, transaction date, store name, and bill value.
- The **user_bill** table establishes connections between users and their corresponding bills.
- The **bill_description** table links products to specific bills and includes line item details.

DATA CLEANING

- Tool Used: R was used for data cleaning.
- Null Rows: There were around 2,494 null rows in the dataset. After removing all the NULL rows, we had 263,284 data points.
- Specific Value removal: Rows containing "NOT-CAPTURED" in the "Colour" column were identified. These values had to be removed for better visualization of the data.
- After removing NULL and “NOT-CAPTURED” rows the dataset has 238,223 data points.

```r
library(data.table)
library(dplyr)
library(stringr)
str(data)
setDT(data)
#data
table(is.na(data))
str_detect(data,'NA')
str(data)
data[is.na(description),NROW(description)]
data[is.na(inventory_category),NROW(inventory_category)]
data_clean <- data[complete.cases(data), ]
table(is.na(data_clean))
data_clean[grep('NOT_CAPTURED',data_clean$colour)]
View(data_clean)
library(writexl)
write_xlsx(data_clean,"/home/sneh/Desktop/BDM/Project/data_clean.xls")
```

DATA PREPARATION

- The initial cleaned data had 238,223 rows and 12 columns.
- The data was transformed into the Third Normal Form (3NF) to organize the data more efficiently and reduce data redundancy.
- The original data was broken down into five tables namely Bill_Details, User_Bills, Store, Bill_Description, and Inventory.
- Tables "Bill Description" and "Inventory" lacked a unique column suitable for creating a primary key.
- The "product_id" and “line_item_id” columns were introduced to serve as a unique identifier for each data point in the inventory and bill_description table respectively.
- The product_id column was created by concatenating the description and price columns.
- The line_item_id is a combination of bill_id and the item serial number in the bill.

Bill_Details	
bill_id	Integer(PK)
transaction_date	Datetime
store_name	Varchar
bill_year	Integer
bill_value	Float
bill_discount	Float

User_Bills	
user_id	Integer(PK)
bill_id	Integer

Store	
store_name	Varchar(PK)
zone_name	Varchar

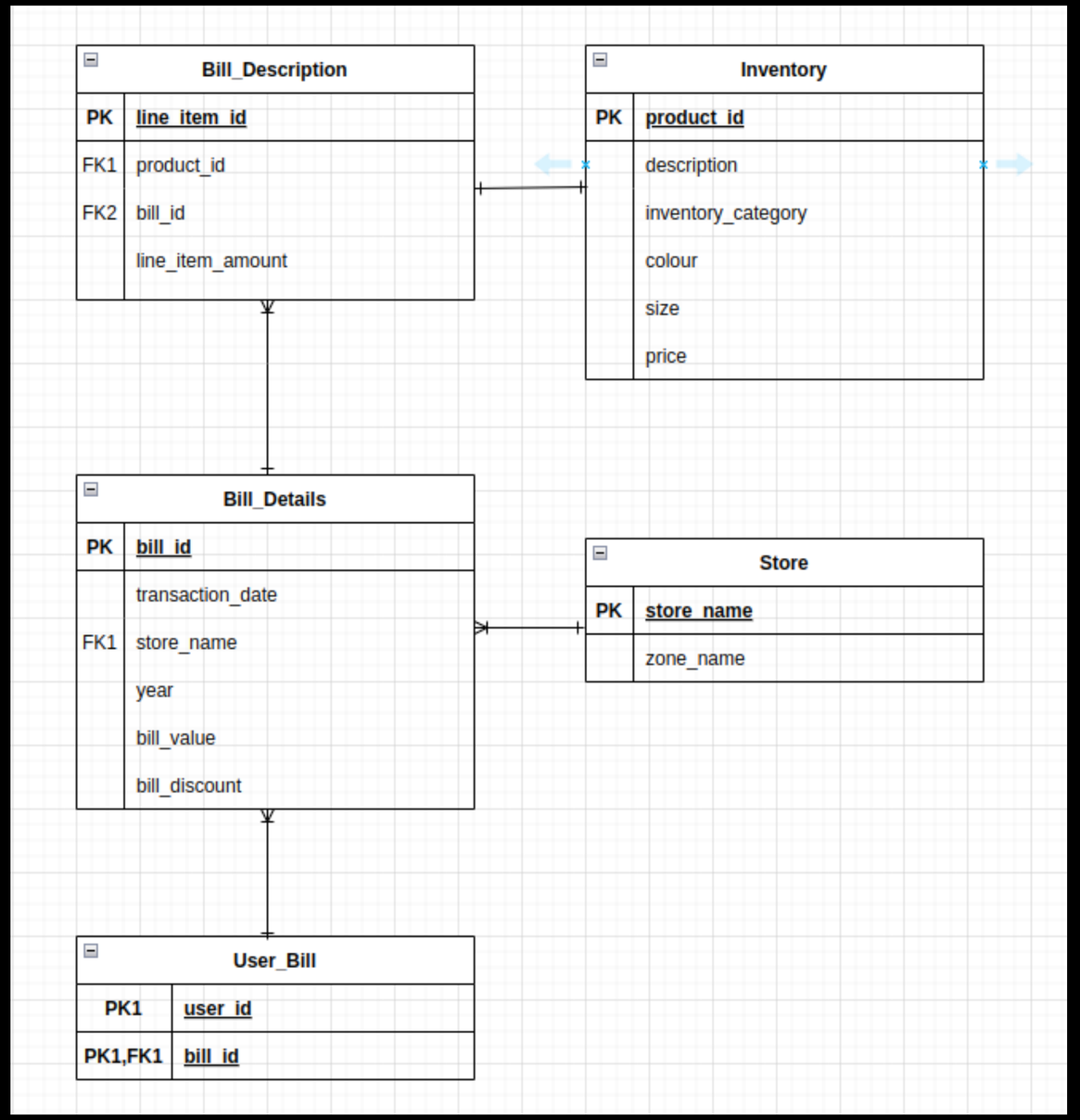
Bill_Description	
line_item_id	Varchar(PK)
Product_id	Varchar
bill_id	Integer
line_item_amount	Float

Inventory	
product_id	Varchar(PK)
description	Varchar
inventory_category	Varchar
colour	Varchar
size	Varchar
price	Float

TABLE CREATION

```
create Database BDM_FINAL_PROJECT_2

drop table if exists store,inventory, bill_details,user_bill,bill_description;

create table store (store_name varchar ,zone_name varchar,primary key(store_name))

create table inventory (product_id varchar,description varchar,inventory_category varchar,colour varchar,size varchar,
price float,primary key (product_id))

create table bill_details (bill_id int NOT NULL ,transaction_date datetime NOT NULL,store_name varchar,year int NOT nUll,
bill_value float,bill_discount float,primary key (bill_id),foreign key (store_name) references store (store_name))

create table user_bill (user_id int,bill_id int,primary key(user_id),foreign key (bill_id) references bill_details (bill_id))

create table bill_description (line_item_id int  not null,product_id varchar ,bill_id int ,lineitem_amount float,
primary key (line_item_id),foreign key (bill_id) references bill_details (bill_id),
foreign key (product_id) references inventory (product_id))
```

ER DIAGRAM

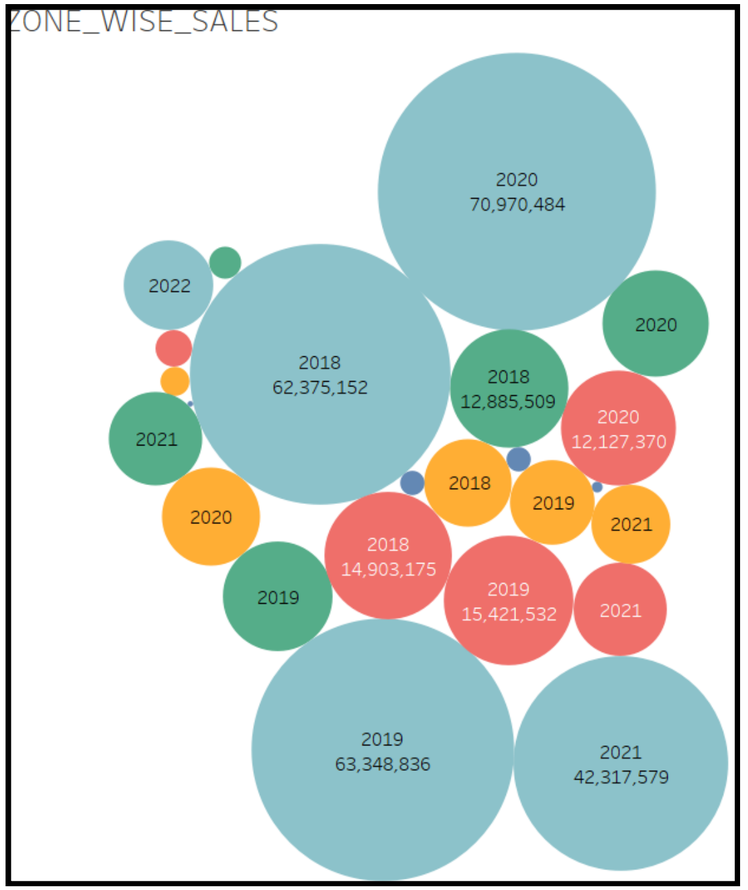

Bill_Description
PK line_item_id
FK1 product_id
FK2 bill_id
line_item_amount
Inventory
PK product_id
description
inventory_category
colour
size
price
Bill_Details
PK bill_id
transaction_date
FK1 store_name
year
bill_value
bill_discount
Store
PK store_name
zone_name
User_Bill
PK1 user_id
PK1,FK1 bill_id

STORE ANALYSIS

INFORMATION GATHERED

- In the store analysis, we leveraged SQL queries to extract relevant data from the bill_details table.
- We focused on visualizing and comparing the total sales values for each store, highlighting the top performers.
- We delved into the distribution of discounts, identifying specific stores,' that may require further attention regarding their discount strategies.

These analyses provide valuable insights into the performance and discounting strategies of individual stores, offering a foundation for strategic decision-making and optimization.



QUESTION :
LIST ALL STORE NAMES AND THEIR CORRESPONDING ZONE NAMES:

QUERY
SELECT STORE_NAME, ZONE_NAME FROM STORE;

store_name	zone_nam
Central_7209	Central
East_6037	East
East_6508	East
East_6509	East
East_6510	East
East_6511	East
East_7007	East
East_7024	East
East_7096	East
East_7157	East
East_7173	East
East_7190	East
East_7219	East
East_7258	East
East_7272	East
East_7273	East

QUESTION: FIND THE TOTAL SALES VALUE FOR EACH STORE:

QUERY

```
SELECT STORE_NAME,
SUM(BILL_VALUE) AS
TOTAL_SALES FROM
BILL_DETAILS GROUP BY
STORE_NAME;
```

	store_name	total_sales
1	North_7492	93746.6065673828
2	South_7408	2032931.16443062
3	North_7456	17403.6000976563
4	South_7469	2019.58984375
5	West_7043	4598
6	South_7407	2804624.54238224
7	South_7158	3458368.70875072
8	South_6014	329490.950561523
9	East_7173	1036564.85433483
10	East_7373	807310.071014404
11	South_7102	1526364.55926037
12	North_7417	2814184.48283434
13	South_7348	2211067.88603306
14	South_7170	1687593.98981762
15	South_7192	1374070.34212017
16	East_7463	3308.2607421875
17	West_7075	793632.639499664
18	East_7258	1376488.72364235
19	West_7466	161322.045349121
20	South_6017	195088.375305176
21	North_7013	2867826.49459076
22	West_7094	1504819.62333202
23	South_7421	190621.318084717
24	South_7439	421836.679885864

QUESTION :FIND THE AVG SALES VALUE BY STORE:

QUERY:

```
SELECT STORE_NAME,
AVG(BILL_VALUE) AS
AVERAGE_BILL_VALUE FROM
BILL_DETAILS GROUP BY
STORE_NAME;
```

	store_name	average_bill_value
1	North_7492	2083.25792371962
2	South_7408	2197.76342100607
3	North_7456	4350.90002441406
4	South_7469	2019.58984375
5	West_7043	4598
6	South_7407	2308.3329566932
7	South_7158	1915.99374446024
8	South_6014	2657.18508517358
9	East_7173	2012.74728997055
10	East_7373	2130.1057282702
11	South_7102	2119.95077675051
12	North_7417	2345.15373569528
13	South_7348	1993.74922094956
14	South_7170	2130.80049219396
15	South_7192	2020.69167958849
16	East_7463	1654.13037109375
17	West_7075	2307.07162645251
18	East_7258	2045.30270972117
19	West_7466	1967.3420164527
20	South_6017	1726.44579916085

QUESTION :FIND THE NUMBER OF SALES BILL GENERATED PER STORE IN A SPECIFIC YEAR (2018) ,ARRANGED BY NUMBER OF BILLS.

QUERY:

```
SELECT STORE_NAME, COUNT(*)
AS BILL_COUNT
FROM BILL_DETAILS
WHERE YEAR = 2018
GROUP BY STORE_NAME
ORDER BY BILL_COUNT DESC
```

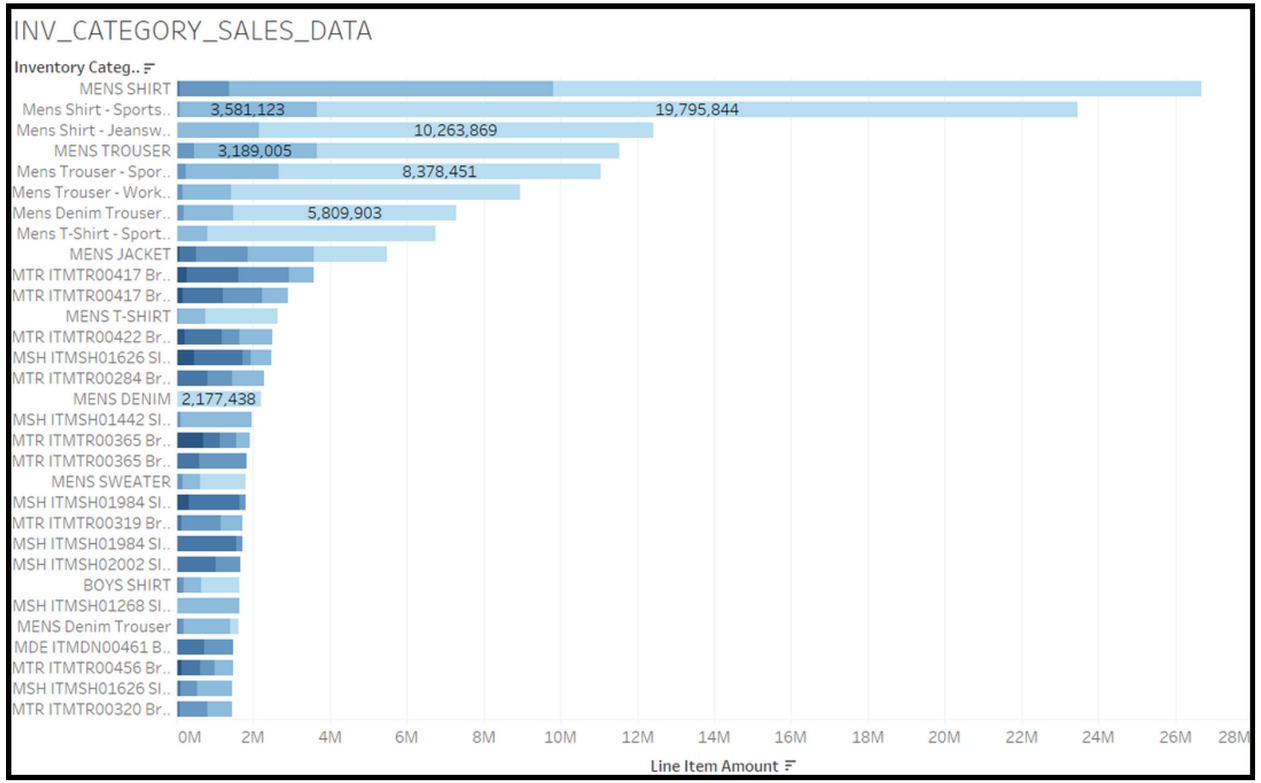
	store_name	bill_count
1	South_7001	1417
2	North_7283	1396
3	South_7038	1033
4	South_7489	975
5	South_7480	912
6	North_7155	870
7	South_7032	865
8	South_7077	848
9	South_7017	836
10	South_7267	816
11	South_7158	797
12	South_7005	787
13	South_7268	751
14	South_7467	726
15	South_7068	689
16	South_7464	677
17	West_7052	619
18	North_7039	591
19	South_7345	572
20	South_7232	571
21	South_7121	568
22	South_7120	561
23	South_7135	559
24	West_7266	544

INVENTORY ANALYSIS

INFORMATION GATHERED

- In inventory overview we delved into the distribution of products across different categories.
- We Identified the most and least populated categories, providing insights for inventory management.
- We explored the average price of products in each category, aiding in pricing strategies and product positioning.
- This analysis lays the foundation for making informed decisions regarding inventory stocking, pricing, and category management within the retail database.

.



QUESTION : IDENTIFY THE HIGHEST AND LOWEST-PRICED PRODUCTS IN THE INVENTORY

```
--LOWEST PRICED
SELECT TOP 1
   PRODUCT_ID, DESCRIPTION,PRICE
   FROM INVENTORY
   ORDER BY PRICE ASC;

 --HIGHEST PRICED
 SELECT TOP 1
   PRODUCT_ID, DESCRIPTION,PRICE
FROM INVENTORY
ORDER BY PRICE DESC;
```

	product_id	description	price
1	BTS ITBTS00396 Regular SS Aqua S 0	BTS ITBTS00396 Regular SS Aqua S	0

	product_id	description	price
1	MTS ITMTS00422 Stretch SS Black M 19026	MTS ITMTS00422 Stretch SS Black M	19026

QUESTION : PRODUCT CATEGORIES WITH MOST PRODUCTS

QUERY:

```
 SELECT
    INVENTORY_CATEGORY,
    COUNT(*) AS PRODUCT_COUNT
FROM
    INVENTORY
GROUP BY
    INVENTORY_CATEGORY
ORDER BY
    PRODUCT_COUNT DESC;
```

	inventory_category	product_count
1	MENS SHIRT	7729
2	Mens Shirt - Sportswear	6723
3	MENS TROUSER	3724
4	Mens Shirt - Jeanswear	3613
5	Mens Trouser - Sportswear	2486
6	Mens T-Shirt - Sportswear	1678
7	Mens Denim Trouser - Jeanswear	1495
8	Mens Trouser - Workwear	1423
9	MENS T-SHIRT	1084
10	BOYS SHIRT	721
11	MENS JACKET	573
12	MENS DENIM	572
13	MENS SWEATER	479
14	Mens T-Shirt - Jeanswear	400
15	MENS Denim Trouser	394
16	Mens Shorts - Sportswear	332
17	BOYS TROUSER	308
18	Mens T-Shirt - Workwear	277
19	MENS BELT	227
20	Boys Shirt - Sportswear	205
21	MTR ITMTR00254 Brooklyn FF ...	197

--QUESTION : RETRIEVE THE TOTAL SALES FOR A SPECIFIC CATEGORY OF PRODUCTS.

QUERY:

```
SELECT INVENTORY_CATEGORY,
SUM(BDH.LINE_ITEM_AMOUNT) AS
TOTAL_SALES
FROM BILL_DESCRIPTION BDH
JOIN INVENTORY I ON BDH.PRODUCT_ID =
I.PRODUCT_ID
GROUP BY INVENTORY_CATEGORY
```

	inventory_category	total_sales
1	BSH ITBSH00269 Regular LS Blue S	21133.8215332031
2	BSH ITBSH00351 Regular LS Blue M	3515.70648193359
3	BSH ITBSH00311 Regular LS Indigo EEL	50916.1322631836
4	MSW ITMSW00105 Regular LS Rust 2XL	65552.2640380859
5	MSH ITMTR00549 Brooklyn FF Cement 30	10307.9613037109
6	MTS ITMTS00521 Regular SS Red S	5611.96502685547
7	MTR ITMTR00546 Brooklyn FF Sap 30	6332.03344726563
8	MTS ITMTS00477 Regular SS Pine 2XL	14183.6876831055
9	MSH ITMSH01800 Slim SS Peacoat 4XL	5464.88256835938
10	MSH ITMTS00709 Regular SS Red 4XL	1214.19946289063
11	BSH ITBSH00540 Regular LS Ochre ES	4301.19390869141
12	Boys Denim Trouser	341081.504516602
13	BSH ITBSH00344 Regular LS Fir Green L	47727.4963989258
14	MSH ITMSH01632 Slim SS Aqua XL	180899.817382813
15	MSH ITMSH01466 Slim LS Cobalt L	36675.9294433594
16	MSO ITMSO00124 Slim FF Sky 38	40144.9498291016
17	BSH ITBSH00509 Regular LS Indigo EL	65305.1881713867
18	MSH ITMSH02963 Slim LS Olive 3XL	29152.7194824219

```
--QUESTION: RETRIEVE THE TOP 5 SELLING
PRODUCTS (BASED ON QUANTITY) FOR
NORTH_7420

SELECT TOP(5) BDC.PRODUCT_ID, COUNT(*)
AS QUANTITY_SOLD
FROM BILL_DESCRIPTION BDC
JOIN BILL_DETAILS BD ON BD.BILL_ID =
BDC.BILL_ID
JOIN INVENTORY I ON BDC.PRODUCT_ID =
I.PRODUCT_ID
WHERE BD.STORE_NAME = 'NORTH_7420'
GROUP BY BDC.PRODUCT_ID
ORDER BY QUANTITY_SOLD DESC
```

Results | Messages

	product_id	quantity_sold
1	ITMTS00002 1299	8
2	MSH ITMSH00847 Chiseled LS Olive S 1899	7
3	ITMSH00177 1999	6
4	MSH ITMSH01626 Slim LS Pale Pink L 2799	5
5	ITMTR00105 2399	4

QUESTION : CALCULATE THE AVERAGE PRICE OF PRODUCTS IN EACH INVENTORY CATEGORY.

QUERY:

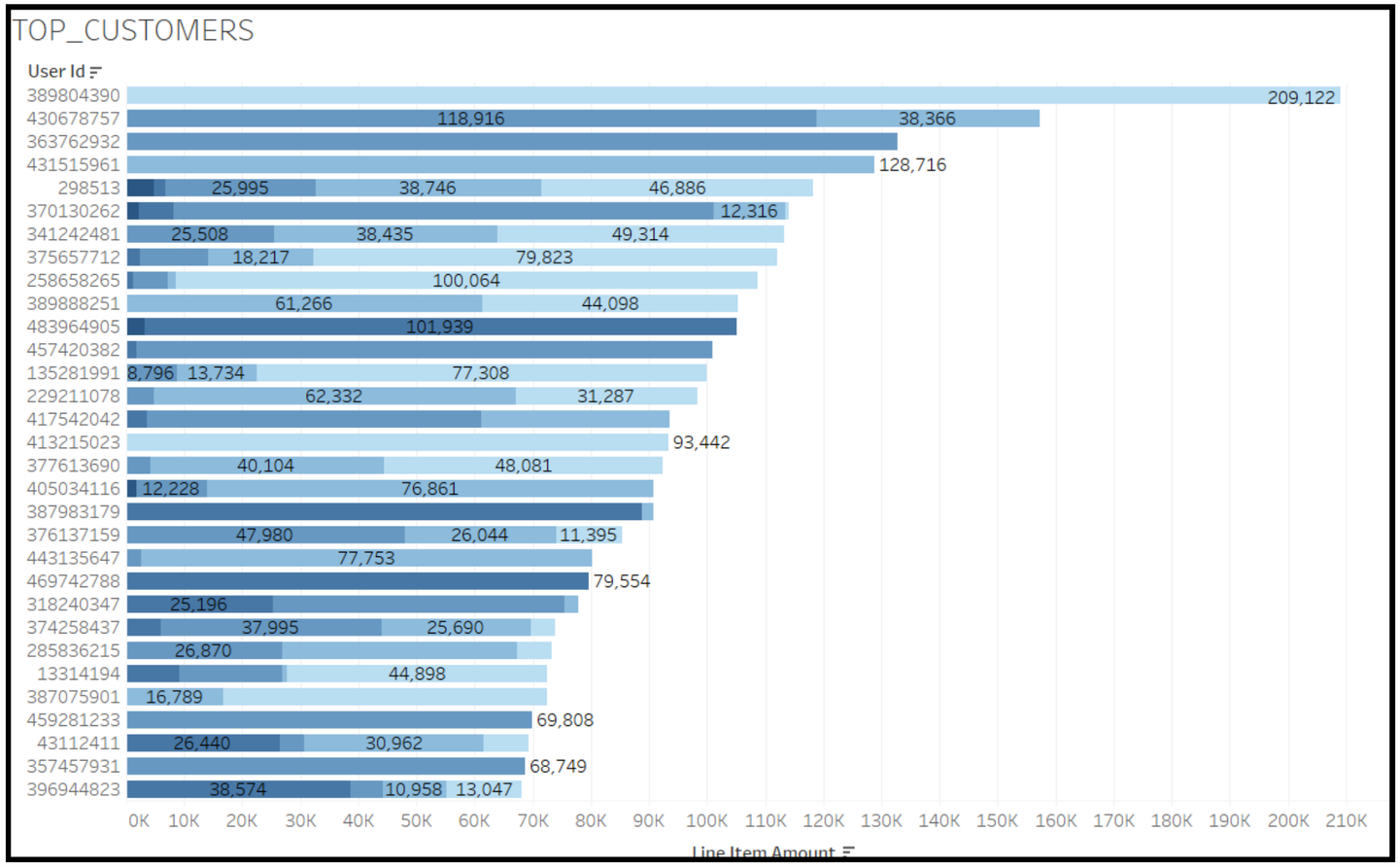
```
SELECT INVENTORY_CATEGORY, AVG(PRICE)
AS AVERAGE_PRICE FROM INVENTORY GROUP
BY INVENTORY_CATEGORY;
```

	inventory_category	average_price
1	BSH ITBSH00351 Regular LS Blue M	593
2	MSH ITMTR00549 Brooklyn FF Cement 30	1288
3	BSH ITBSH00540 Regular LS Ochre ES	716
4	MSH ITMTS00709 Regular SS Red 4XL	1214
5	MTS ITMTS00521 Regular SS Red S	1123
6	BSH ITBSH00344 Regular LS Fir Green L	876
7	MSH ITMSH01632 Slim SS Aqua XL	2012
8	MSH ITMSH02963 Slim LS Olive 3XL	1629
9	MSO ITMSO00124 Slim FF Sky 38	1254
10	BSO ITBSO00104 Regular 5P Lt Khaki M	526
11	BSO ITBSO00121 Regular FF Olive S	661
12	MTR ITMTR00608 Brooklyn FF Black 32	2170
13	MSH ITMSH01997 Slim LS Lt Indigo 2XL	1665
14	MSW ITMSW00116 Regular SL Lt Blue M. XL	2257
15	BSS ITBSS00051 Regular LS Cobalt EEL	1063
16	Mens Belt - Jeanswear	886
17	MSH ITMSH02063 Slim LS Rust 4XL	2034
18	BSH ITBSH00289 Regular LS Fir Green ES	611

USER ANALYSIS

INFORMATION GATHERED

- The user analysis aimed to recognize and acknowledge the most valuable customers in terms of total purchases.
- We utilized SQL queries to aggregate and analyze transaction data from the user bill table.
- We identified key users for potential recognition, loyalty programs, or targeted marketing strategies.
- This analysis provides insights into user behavior and allows for tailored strategies to enhance customer satisfaction and retention within the retail database.



QUESTION-IDENTIFY THE USERS WHO MADE THE HIGHEST TOTAL PURCHASES

QUERY

```
SELECT UB.USER_ID, SUM(BD.BILL_VALUE) AS TOTAL_PURCHASE
FROM USER_BILL UB
JOIN BILL_DETAILS BD ON UB.BILL_ID = BD.BILL_ID
GROUP BY UB.USER_ID
ORDER BY TOTAL_PURCHASE DESC;
```

	user_id	total_purchase
1	389804390	209122.203491211
2	430678757	157282.332275391
3	363762932	132846
4	431515961	128716
5	298513	118348.811035156
6	370130262	114063.647109985
7	341242481	113257.291259766
8	375657712	112128.080322266
9	258658265	108683.789794922
10	389888251	105364.081542969
11	483964905	105045.510986328
12	457420382	100831.56640625
13	135281991	99837.4011230469
14	229211078	98415.962890625
15	417542042	93518.1715087891
16	413215023	93441.6821289063
17	377613690	92383.6633300781
18	405034116	90859.4366455078
19	387983179	90683.3052978516
20	376137159	85419.1240234375
21	443135647	80252.2751464844
22	469742788	79553.609375

QUESTION-IDENTIFY HIGH-VALUE USERS

QUERY

```
SELECT
    U.USER_ID,
    SUM(BD.BILL_VALUE) AS
TOTAL_PURCHASE_AMOUNT
FROM
    USER_BILL U
    JOIN BILL_DETAILS BD ON U.BILL_ID =
BD.BILL_ID
GROUP BY
    U.USER_ID
HAVING
    SUM(BD.BILL_VALUE) > 1000
ORDER BY
    TOTAL_PURCHASE_AMOUNT DESC;
```

	user_id	total_purchase_amount
1	389804390	209122.203491211
2	430678757	157282.332275391
3	363762932	132846
4	431515961	128716
5	298513	118348.811035156
6	370130262	114063.647109985
7	341242481	113257.291259766
8	375657712	112128.080322266
9	258658265	108683.789794922
10	389888251	105364.081542969
11	483964905	105045.510986328
12	457420382	100831.56640625
13	135281991	99837.4011230469
14	229211078	98415.962890625
15	417542042	93518.1715087891
16	413215023	93441.6821289063
17	377613690	92383.6633300781
18	405034116	90859.4366455078
19	387983179	90683.3052978516
20	376137159	85419.1240234375
21	443135647	80252.2751464844
22	469742788	79553.609375
23	318240347	77849.6584472656
24	374258437	73780.2513427734
25	285836215	73307.1349487305
26	13314194	72505.8365478516

QUESTION---IDENTIFY USER LOYALTY

QUERY

```
SELECT
    U.USER_ID,
    COUNT(DISTINCT BD.STORE_NAME) AS
UNIQUE_STORES_VISITED,
    COUNT(DISTINCT BD.BILL_ID) AS
TOTAL_TRANSACTIONS
FROM
    USER_BILL U
    JOIN BILL_DETAILS BD ON U.BILL_ID =
BD.BILL_ID
GROUP BY
    U.USER_ID
ORDER BY
    TOTAL_TRANSACTIONS DESC;
```

	user_id	unique_stores_visited	total_transactions
1	377613690	1	29
2	341242481	4	26
3	367876232	2	26
4	318240347	1	22
5	306978414	1	22
6	352716981	4	21
7	375657712	4	20
8	389092191	2	19
9	393068013	2	19
10	366204811	3	18
11	277468490	3	18
12	374643603	1	18
13	396944823	1	18
14	330382981	2	17
15	370130262	3	17
16	366811488	1	17
17	338348001	2	17
18	392768482	1	16
19	399210817	1	16
20	367511686	2	16
21	350996828	2	15
22	281873	1	15
23	329670096	1	15
24	305896136	2	15
25	408121042	1	15
26	390078507	1	15

QUESTION---ZONE WISE USER COUNT

QUERY:

```
 SELECT
    S.ZONE_NAME,
    COUNT(DISTINCT U.USER_ID) AS
USER_COUNT
FROM
    USER_BILL U
    JOIN BILL_DETAILS BD ON U.BILL_ID =
BD.BILL_ID
    JOIN STORE S ON BD.STORE_NAME =
S.STORE_NAME
GROUP BY
    S.ZONE_NAME
ORDER BY
    USER_COUNT DESC;
```

	zone_name	user_count
1	South	83858
2	North	21728
3	West	16464
4	East	11697
5	Central	526

QUESTION -IDENTIFY TOP 3 CONSUMERS BY ZONE

QUERY:

```
WITH USERPURCHASES AS (
   SELECT
      U.USER_ID,
      S.ZONE_NAME,
      SUM(BD.BILL_VALUE) AS TOTAL_PURCHASES
   FROM
      USER_BILL U
      JOIN BILL_DETAILS BD ON U.BILL_ID = BD.BILL_ID
      JOIN STORE S ON BD.STORE_NAME = S.STORE_NAME
   GROUP BY
      U.USER_ID, S.ZONE_NAME
)

SELECT
   ZONE_NAME, USER_ID, TOTAL_PURCHASES
FROM (
   SELECT
      ZONE_NAME, USER_ID, TOTAL_PURCHASES,
      ROW_NUMBER() OVER (PARTITION BY ZONE_NAME
ORDER BY TOTAL_PURCHASES DESC) AS RANK
   FROM USERPURCHASES) RANKED USERPURCHASES
WHERE RANK <= 3;
```

	zone_name	user_id	total_purchases
1	Central	323843511	30578.75390625
2	Central	391931650	19016.1982421875
3	Central	321889058	13498.2652587891
4	East	469742788	79553.609375
5	East	372394990	66342.3178710938
6	East	384401356	63546.8306884766
7	North	363762932	132846
8	North	457420382	100831.56640625
9	North	443135647	80252.2751464844
10	South	431515961	128716
11	South	298513	118348.811035156
12	South	370130262	114063.647109985
13	West	389804390	207967.728515625
14	West	430678757	157282.332275391
15	West	417542042	93518.1715087891

QUESTION:GIVE THE DETAILS OF SALES DATA MONTH WISE IN DESCENDING ORDER ACROSS DIFFERENT YEAR

```
SELECT
   U.USER_ID,
   YEAR(BD.TRANSACTION_DATE) AS YEAR,
   MONTH(BD.TRANSACTION_DATE) AS MONTH,
   SUM(BD.BILL_VALUE) AS MONTHLY_SPENDING
FROM
   USER_BILL U
   JOIN BILL_DETAILS BD ON U.BILL_ID = BD.BILL_ID
GROUP BY
   U.USER_ID, YEAR(BD.TRANSACTION_DATE),
MONTH(BD.TRANSACTION_DATE)
ORDER BY
   MONTHLY_SPENDING DESC;
```

	user_id	year	month	monthly_spending
1	389804390	2018	1	209122.203491211
2	363762932	2020	1	132846
3	431515961	2019	9	128716
4	430678757	2020	1	111937
5	457420382	2020	9	99107.31640625
6	258658265	2018	1	95766.2587890625
7	413215023	2018	12	93441.6821289063
8	483964905	2021	3	89886.451171875
9	387983179	2021	1	88824.5322265625
10	469742788	2021	1	79553.609375
11	370130262	2020	12	79200
12	443135647	2019	12	77753.2751464844
13	459281233	2020	10	69807.5991210938
14	357457931	2020	9	68748.5512695313

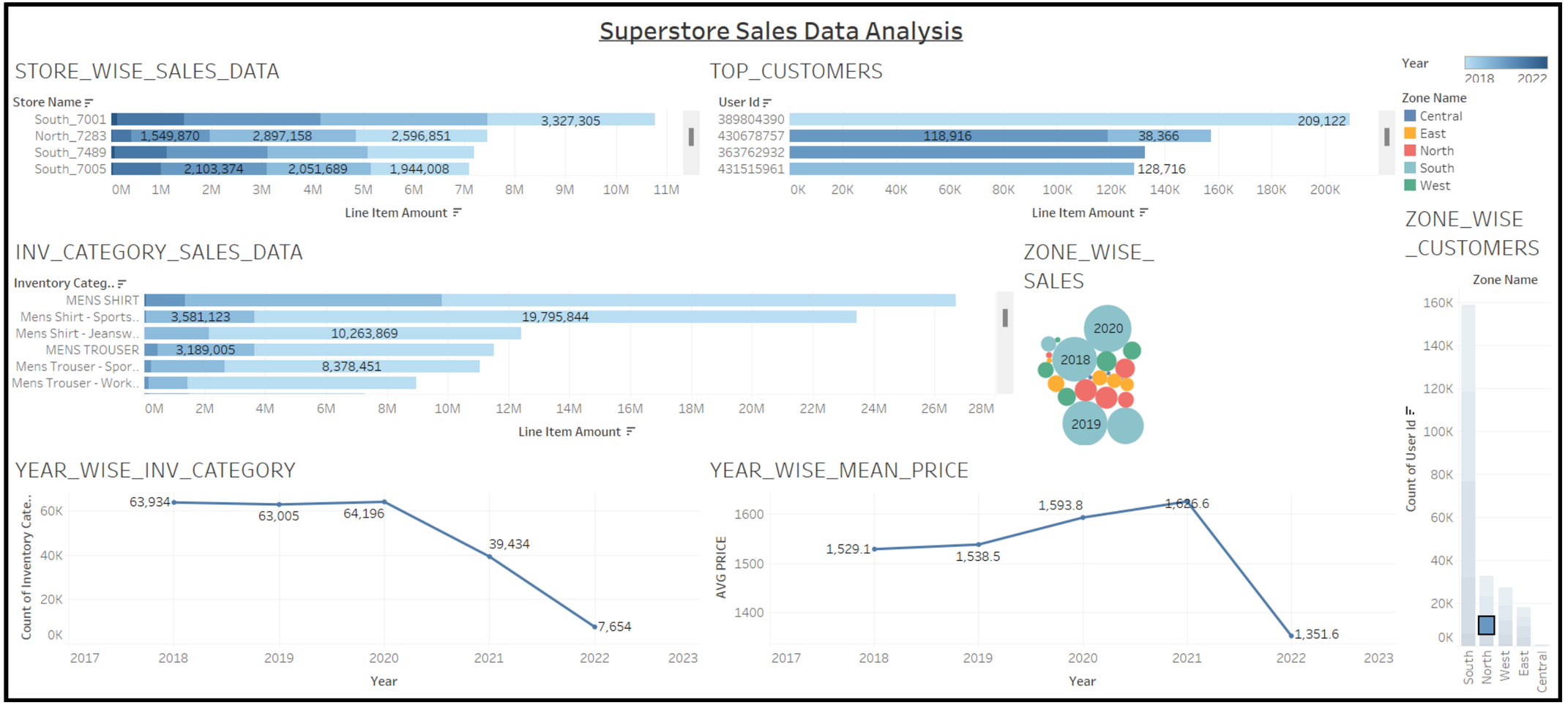
QUESTION -USERS WITH HIGH PURCHASE FREQUENCY

QUERY:

```
SELECT U.USER_ID, COUNT(BD.BILL_ID) AS
TRANSACTION_COUNT FROM USER_BILL U
JOIN BILL_DETAILS BD ON U.BILL_ID =
BD.BILL_ID GROUP BY U.USER_ID
HAVINGCOUNT(BD.BILL_ID) > 10 -- SET YOUR
THRESHOLD FOR HIGH-FREQUENCY
USERSORDER BY TRANSACTION_COUNT DESC;
```

	user_id	transaction_count
1	377613690	29
2	367876232	26
3	341242481	26
4	306978414	22
5	318240347	22
6	352716981	21
7	375657712	20
8	389092191	19
9	393068013	19
10	396944823	18
11	374643603	18
12	366204811	18
13	277468490	18
14	366811488	17
15	370130262	17
16	338348001	17
17	330382981	17
18	367511686	16
19	399210817	16
20	392768482	16
21	390078507	15
22	408121042	15

DATA VISUALISATION DASHBOARD



REFERENCES

- Data source: Kaggle: https://www.kaggle.com/datasets/nishchay331/retail-store

Thank You