Atm Optimization

INTRODUCTION

The DATAKLIQ TEAM analyzed and compiled the ATM Optimization project to formulate data-driven strategies focused on improving ATM performance for enhanced customer retention. Welcome to the realm of modern banking, where customer loyalty serves as the driving force for financial success. Recognizing the ATM as more than a mere cash dispenser, we acknowledge its critical role as a touchpoint capable of shaping customer relationships. Our mission is deeply rooted in a comprehensive dataset collected from five diverse states—**Enugu, Lagos, Kano, Rivers,** and the **FCT**. Empowered by this wealth of information, we are embarking on a groundbreaking Business Intelligence and Data Analytics initiative.

OBJECTIVE:

1. **Uncover Smart, Data-Driven Strategies:**
   ◦ We're diving into a Business Intelligence and Data Analytics project with the aim of revealing intelligent, data-driven strategies.
   . They're designed to create a more profound connection with our customers.
2. **Decode Customer Behavior Across Dynamic Regions:**

◦ Leveraging the power of data analytics, our goal is to decipher customer behavior across diverse regions—Enugu, Lagos, Kano, Rivers, and the FCT.

◦ This insight into regional nuances allows us to tailor our approach and offerings to meet the unique needs of each area.

3.

Boost Operational Efficiency:

◦ Data analytics isn't just about understanding customers; it's also about enhancing operational efficiency.

◦ By leveraging the insights gained, we aim to streamline operations, ensuring a more seamless and efficient banking experience.

4.

Craft Relationships That Stand the Test of Time:

◦ Beyond transactions, we're committed to building lasting relationships.

◦ The insights from our project empower us to craft connections that endure, standing strong through changing times.

TOOLS USED:

- **PostgreSQL**
- **Microsoft Power bi**

Database was created on my PostgreSQL server, naming it **Advanced Project.** The dataset is in a CSV format and was then imported into the SQL server, To import into the SQL server A table was created in SQL to facilitate the importation of various CSV files.

- Table 1 - ATM_LOCATION_LOOKUP

```
CREATE TABLE atm_location_lookup (
LocationID VARCHAR(100),
"Location Name" VARCHAR(100),
"No of ATMs" VARCHAR(100),
City VARCHAR(100),
State VARCHAR(100),
```

```
Country VARCHAR(100)
);
```

The code above was used to create a table designed to store information about ATM locations. The utilization of the VARCHAR datatype enhances the flexibility when importing data .

To copy data from a CSV file into the specified table a permission needs to be granted and then retrieve all the data from that table. The `COPY` command is a common way to import data from an external file into a PostgreSQL table

```
COPY atm_location_lookup
FROM'C:\Users\HP\Documents\New folder\atm_location lookup.csv'DE
```

The created Table was previewed use the SELECT command

```
SELECT*
FROM atm_location_lookup
```

After every steps click on RUN to run the query .

- Table 2 ATM_MAINTENANCE_SCHEDULE_LOOKUPS

```
CREATE TABLE ATM_MAINTENANCE_SCHEDULE_LOOKUPS
 "S/N" VARCHAR(100),
"ATM ID" VARCHAR(100),
"Maintenance Timestamp" TIMESTAMP,
"ATM Uptime/Downtime" VARCHAR(100),
"MTBF (hours)" TIMESTAMP
"MTTR (hours)" TIMESTAMP
"Cash Availability" VARCHAR(100),
"Cash Replenishment Cycle" VARCHAR(100),
"Transaction Success Rate (Percent)" DECIMAL,
"Error Rate (%)" DECIMAL(100),
"Service Response Time (minutes)" TIMESTAMP
```

```
"ATM Utilization" VARCHAR,
"ATM Dwell Time (seconds)"VARCHAR(100)
"Third Party Maintenance Costs ($)" VARCHAR(100)
);
```

```
COPY ATM_MAINTENANCE_SCHEDULE_LOOKUPS
FROM 'C:\Users\HP\Documents\New folder\ATM_Maintenance_Schedule_
DELIMITER ',' CSV HEADER
NULL 'N/A';
```

```
SELECT*
FROM ATM_MAINTENANCE_SCHEDULE_LOOKUPS
```

- Table 3 CALENDAR_LOOKUP

```
CREATE TABLE calendar_lookup (
  date_column DATE PRIMARY KEY,
  year INTEGER,
  month_name VARCHAR(50),
  month INTEGER,
  quarter VARCHAR(50),
  week_of_year INTEGER,
  end_of_week DATE,
  day_of_week INTEGER,
  day_name VARCHAR(50),
  is_holiday BOOLEAN
);
```

```
COPY  calendar_lookup
FROM'C:\Users\HP\Documents\New folder\calendar lookup.csv'DELIM
```

```
SELECT*
FROM calendar_lookup
```

- Table 4 CUSTOMER_LOOKUP_INFO

```
CREATE TABLE customer_lookup_info (
CardholderID VARCHAR(100),
First_Name VARCHAR(100),
Last_Name VARCHAR(100),
Gender VARCHAR(100),
LOCATIONID VARCHAR(100),
Birth_Date DATE,
Occupation VARCHAR(100),
AccountType VARCHAR(100),
IsDaKliq_Bank VARCHAR(100),
Profession_Category VARCHAR(100)
);
```

```
COPY customer_lookup_info
FROM'C:\Users\HP\Documents\New folder\customers_lookup_Info.csv
```

```
SELECT*
FROM customer_lookup_info
```

- Table 5 HOUR_LOOKUP

```
CREATE TABLE hourlookup(
Hour_Key VARCHAR(100),
Hour_Start_Time Timestamp
Hour_End_Time Timestamp
);
```

```
COPY hourlookup
FROM'C:\Users\HP\Documents\New folder\hour lookup.csv'DELIMITER
```

```
SELECT*
FROM hourlookup
```

- Table 6 Transaction_type_lookup

```
CREATE TABLE transaction_type_lookup(
TransactionTypeID VARCHAR(100),
TransactionTypeName VARCHAR(100)
);
```

```
COPY transaction_type_lookup
FROM 'C:\Users\HP\Documents\New folder\transaction_type lookup.c
```

```
SELECT*
FROM transaction_type_lookup
```

- Table 7

This involves merging tables from all states to generate a unified table referred to as the **Transactiontable**. After importing tables, similar to the process I followed for other tables previously, this consolidated table is created.

```
CREATE VIEW TransactionTable AS
SELECT *
FROM Enugu_state
UNION ALL
```

```sql
SELECT *
FROM Fct_s
UNION ALL
SELECT *
FROM Kano_state
UNION ALL
SELECT *
FROM Lagos_State
UNION ALL
SELECT *
FROM River_State;
```

This table was created as a view in PostgreSQL. Creating a table as a view means creating a virtual table that displays data from multiple tables. This provided SQL code combines data from Enugu_state, Fct_s, Kano_state, Lagos_State, and River_State using the UNION ALL operator to form the **TransactionTable view.**

Creating Calendar Dimension Table :A calander dimension table was created from the previous calander table because it has only has information of that of 2019.

```sql
CREATE TABLE calendar_dimensions_2020 (
date_column DATE PRIMARY KEY,
year INTEGER,
month_name VARCHAR(50),
month INTEGER,
quarter VARCHAR(50),
week_of_year INTEGER,
end_of_week DATE,
day_of_week INTEGER,
day_name VARCHAR(50),
is_holiday BOOLEAN
);
```

```
INSERT INTO calendar_dimensions_2020 (
date_column, year, month_name, month, quarter, week_of_year, end
)
SELECT
date_column AS date_column,
EXTRACT(YEAR FROM date_column) AS year,
TO_CHAR(date_column, 'Month') AS month_name,
EXTRACT(MONTH FROM date_column) AS month,
'Q' || TO_CHAR(date_column, 'Q') AS quarter,
EXTRACT(WEEK FROM date_column) AS week_of_year,
date_column + INTERVAL '6 days' - INTERVAL '1 day' * EXTRACT(DOW
EXTRACT(DOW FROM date_column) AS day_of_week,
TO_CHAR(date_column, 'Day') AS day_name,
CASE
WHEN EXTRACT(MONTH FROM date_column) = 1 AND EXTRACT(DAY FROM da
WHEN EXTRACT(MONTH FROM date_column) = 5 AND EXTRACT(DAY FROM da
-- Add more holiday conditions as needed
ELSE FALSE
END AS is_holiday
FROM
generate_series('2020-01-01'::date, '2020-12-31'::date, '1 day'
```

```
SELECT * FROM calendar_dimensions_2020;
```

This was carried out for the years 2021 and 2022, respectively, in order to populate the Calendar_dimension_table.

— Having tables named calendar_dimension_2019, calendar_dimension_2020, and calendar_dimension_2021.

- - Create a fresh table named "**calendar_dimension_Table"** to store the combination of data from all years.

- - Union all the data from individual calendar dimension tables into the new table

```
INSERT INTO calendar_dimension_Table
SELECT * FROM calendar_dimensions
UNION ALL
SELECT * FROM calendar_dimensions_2020
UNION ALL
SELECT * FROM calendar_dimensions_2021
UNION ALL
SELECT * FROM calendar_dimensions_2022
;
```

- - View the contents of the combined table.

```
SELECT * FROM calendar_dimension_Table;
```

As mentioned before, creating a table as a view involves constructing a virtual table that exhibits data from various tables.

DATA TRANSFORMATION :

Data transformation was done using SQL,

The ATM_MAINTENANCE_SCHEDULE_LOOKUPS includes columns with both NULL values and empty entries, would influencing the analysis. Therefore, I utilized the update syntax to modify specific columns where changes were necessary.

```
UPDATE ATM_MAINTENANCE_SCHEDULE_LOOKUP
SET "MTTR (hours)" = COALESCE("MTTR (hours)", '0 hours'),
"MTBF (hours)" = COALESCE("MTBF (hours)", '0 hours');
```

```
UPDATE ATM_MAINTENANCE_SCHEDULE_LOOKUP
```

```
SET "Cash Availability" = COALESCE("Cash Availability", 'Low'),
"Cash Replenishment Cycle" = COALESCE("Cash Replenishment Cycle'
```

```
UPDATE ATM_MAINTENANCE_SCHEDULE_LOOKUP
SET
"Error Rate (%)" = COALESCE("Error Rate (%)", '1.8'),
"Transaction Success Rate (Percent)" = COALESCE("Transaction Suc
```

To obtain the average of cells containing NULL values, I employed the following query

```
SELECT AVG(COALESCE(column_name, 0)) AS average_value
FROM your_table;
```

This is so beacause,The `COALESCE` function is used to handle NULL values,The `COALESCE` function returns the first non-NULL argument, so you can replace NULL or empty values with a default value (e.g., zero) before calculating the average.

There are multiple approaches to accomplish this, but the method I used is this abov

```
UPDATE ATM_MAINTENANCE_SCHEDULE_LOOKUPS
SET Service_Response_Time_Minutes = 0
WHERE Service_Response_Time_Minutes = 'N/A';
```

```
UPDATE ATM_MAINTENANCE_SCHEDULE_LOOKUPS
SET ATM_Utilization  = 0
WHERE ATM_Utilization  = 'N/A';
```

```
UPDATE ATM_MAINTENANCE_SCHEDULE_LOOKUPS
SET ATM_Dwell_Time_Seconds = 0
WHERE  ATM_Dwell_Time_Seconds = 'N/A';
```

I utilized the `ALTER` statement to modify the *data type* of certain columns, ensuring that it doesn't affect my analysis.

```
ALTER TABLE ATM_MAINTENANCE_SCHEDULE_LOOKUPS
ALTER COLUMN ATM_Dwell_Time_Seconds  TYPE NUMERIC USING CAST(ATM
```

```
ALTER TABLE ATM_MAINTENANCE_SCHEDULE_LOOKUPS
ALTER COLUMN   Service_Response_Time_Minutes TYPE NUMERIC USING
```

```
ALTER TABLE ATM_MAINTENANCE_SCHEDULE_LOOKUPS
ALTER COLUMN TRANSACTION_SUCCESS_RATE_PERCENTAGE TYPE float USIN
```

```
ALTER TABLE ATM_MAINTENANCE_SCHEDULE_LOOKUPS
ALTER COLUMN Error_Rate_Percentage  TYPE float USING CAST(Error_
```

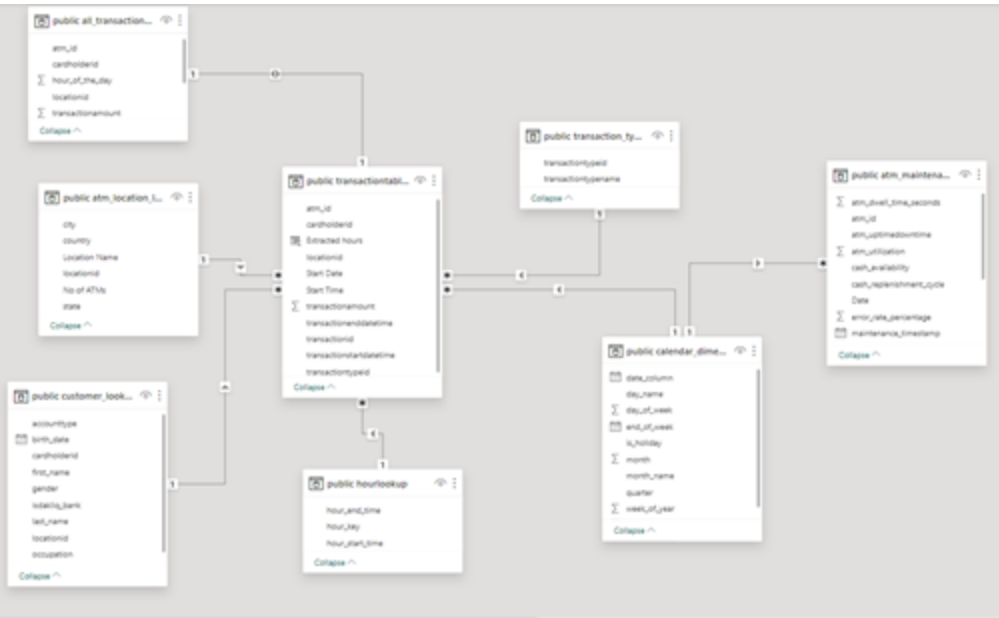
Leveraging on the tool: **Power BI,**”

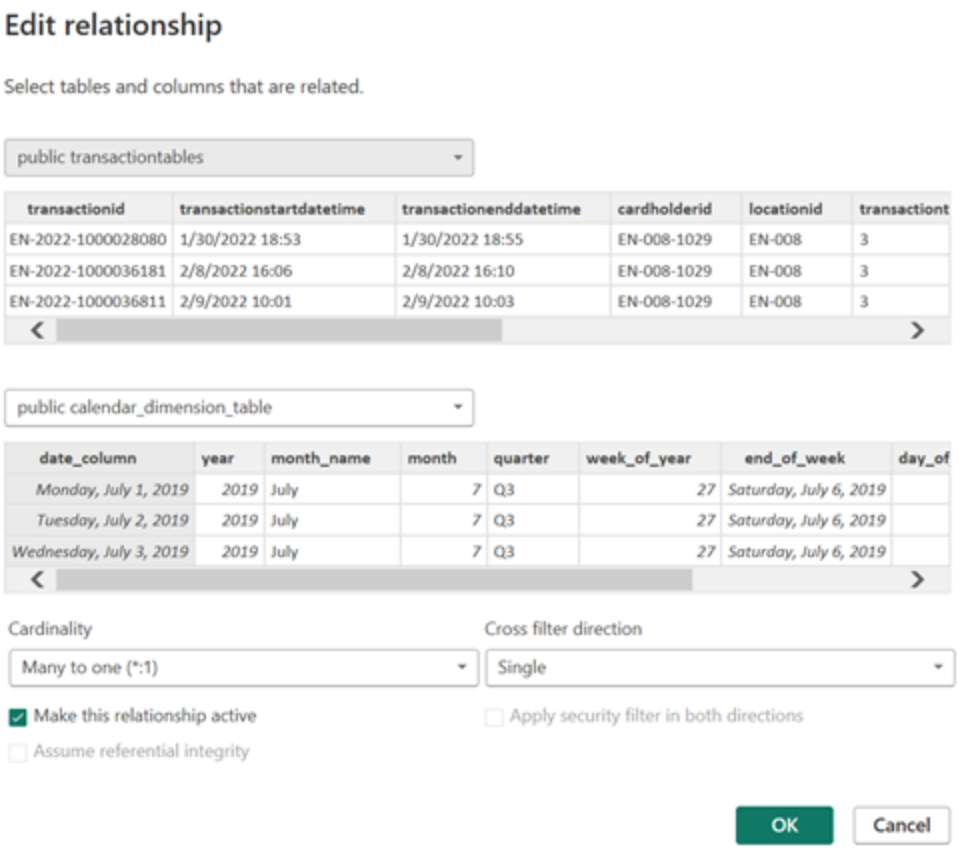
- I established a link from SQL to Power BI using the "Get Data" tab. Upon selecting "More" in the Get Data option, I navigated to the database section and chose PostgresSQL Database.

Input your server name and specify the database name, which I created from scratch as "ADVANCED PROJECT." Choose the import mode and click OK.

DATA MODELLING:

I applied the snowflake model to interconnect my facts and dimensions seamlessly. I opted for the import storage mode to bring my data into Power BI, establishing relationships. Most of these relationships exhibit a many to one cardinality, and a singular filter direction was enforced, as depicted in the provided screenshot.





DAX (Data Analysis Expression)

I have created a set of DAX formulas to carry out calculations and analysis on the data within my reports. The use of DAX is instrumental in developing dynamic and insightful reports, tailored to meet the specific needs outlined in the objectives of our project.

DAX MEASURES

On the Transaction (Fact Table)

- Total Amount = SUM('public transaction_fact_table'[transactionamount])
- Total Records = COUNT('public transaction_fact_table'[transactionid])

On the customer’s Table

- No of Customers = COUNT('public customer_lookup_info'[cardholderid])

On the Atm_Location Table

- ```
  Location Rank =
  RANKX(
      ALL('public atm_location_lookup'[Location Name]),   -- Consider all locations
      CALCULATE(
          COUNTROWS('public all_transactions_table') -- Count transactions for each location
      ),
      ,
      ASC
  )
  ```
- `Total City = DISTINCTCOUNT('public atm_location_lookup'[city])`

On Atm Machine Table

- No of ATMs = SUM('public atm_location_lookup'[no_of_atms])

On Atm_Maintance_lookup_Table

- Error Rate = AVERAGE('public atm_maintenance_schedule_lookups'[error_rate_percentage])
- Service_Response_Time =
  CALCULATE(
  AVERAGE('public atm_maintenance_lookup_table'[service_response_time_in_minutes]),
  ALLEXCEPT('public atm_maintenance_lookup_table', 'public atm_maintenance_lookup_table'[atm_id])
)
- MTBF for Location =
  CALCULATE(AVERAGE('public atm_maintenance_lookup_table'[mtbf_in_hours]),
  FILTER('public atm_location_lookup', 'public atm_location_lookup'[location_name] = "Location Name")
)
- Error Rate = AVERAGE('public atm_maintenance_lookup_table'[ERROR*RATE*(%)])
- Machine failure interval =
  VAR average_ = AVERAGE('public atm_maintenance_schedule_lookups'[MTBF convert to days])
  VAR formattedAverage = FORMAT(average_, "0")
  RETURN
  formattedAverage & " days"
  ```

- Rating =
  VAR conditionsMet =
  IF([Error Rate] < 1.75, 1, 0) +
  IF([Success Rate] > 92.3, 1, 0) +
  IF([user per machine]> 287.60, 1, 0) +
  IF([service_response _target] < 55, 1, 0) +
  IF([use_wait_time_simplified] < 1.87, 1, 0) +
  IF([Repair_time_2] < 3.57, 1, 0) +
  IF([machine failure interval_2] > 81, 1, 0)
  RETURN
  IF(conditionsMet >= 1, conditionsMet, 0)

- Star_Rating =
  VAR conditionsMet =
  IF([Error Rate] < 1.78, 1, 0) +
  IF([Success Rate]> 92.3, 1, 0) +
  IF([user per machine] > 286.60, 1, 0) +
  IF([service_response _target] < 55.49, 1, 0) +
  IF([use_wait_time_simplified] < 1.87, 1, 0) +
  IF([Repair_time_2] < 2.5, 1, 0) +
  IF([machine failure interval_2] > 67, 1, 0)
  RETURN
  SWITCH(
  conditionsMet,
  0, "No Star Rating",
  1, "⭐",
  2, "⭐⭐",
  3, "⭐⭐⭐",
  4, "⭐⭐⭐⭐",
  5, "⭐⭐⭐⭐⭐",
  6, "⭐⭐⭐⭐⭐⭐",
  7,"⭐⭐⭐⭐⭐⭐⭐"
)

- Use_wait_time_simplified =
  DIVIDE(
  CALCULATE(
  AVERAGEX(
  FILTER(
  'public atm_maintenance_schedule_lookups',
  'public atm_maintenance_schedule_lookups'[atm_dwell_time_seconds]
  <> 0
  ),
  'public atm_maintenance_schedule_lookups'[atm_dwell_time_seconds]
)
  ),
  60
)
- User per machine = AVERAGE('public atm_maintenance_schedule_lookups'[atm_utilization])
- Total_Transaction_Count =
  Total_Transaction_Count = CALCULATE(COUNTROWS('public all_transactions_table'),ALLEXCEPT('public all_transactions_table','public transaction_type_lookup'[transactiontypename])

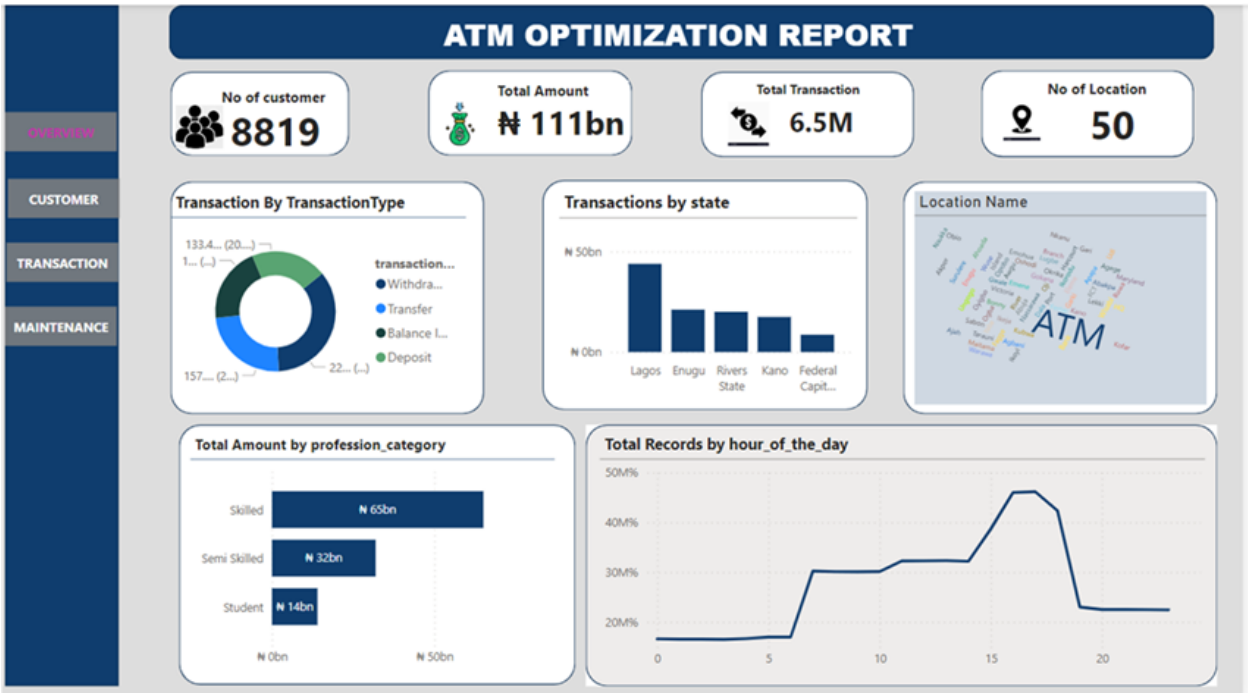
ANALYSIS AND VISUALIZATION

- The analysis of ATM optimization, spanning from 2019 to 2022, reveals a total transaction count of 6.5 million, with March 2022 setting highest records.
- Over the 4-year span of ATM optimization, four Transaction Types were recorded with the following percentages: Withdrawal at 34.99%, Transfer at 24.1%, Balance Inquiries at 20.46%, and Deposit at 20.45%.
- The transactions were categorized based on professions into three groups: the skilled, the semi-skilled, and the students.

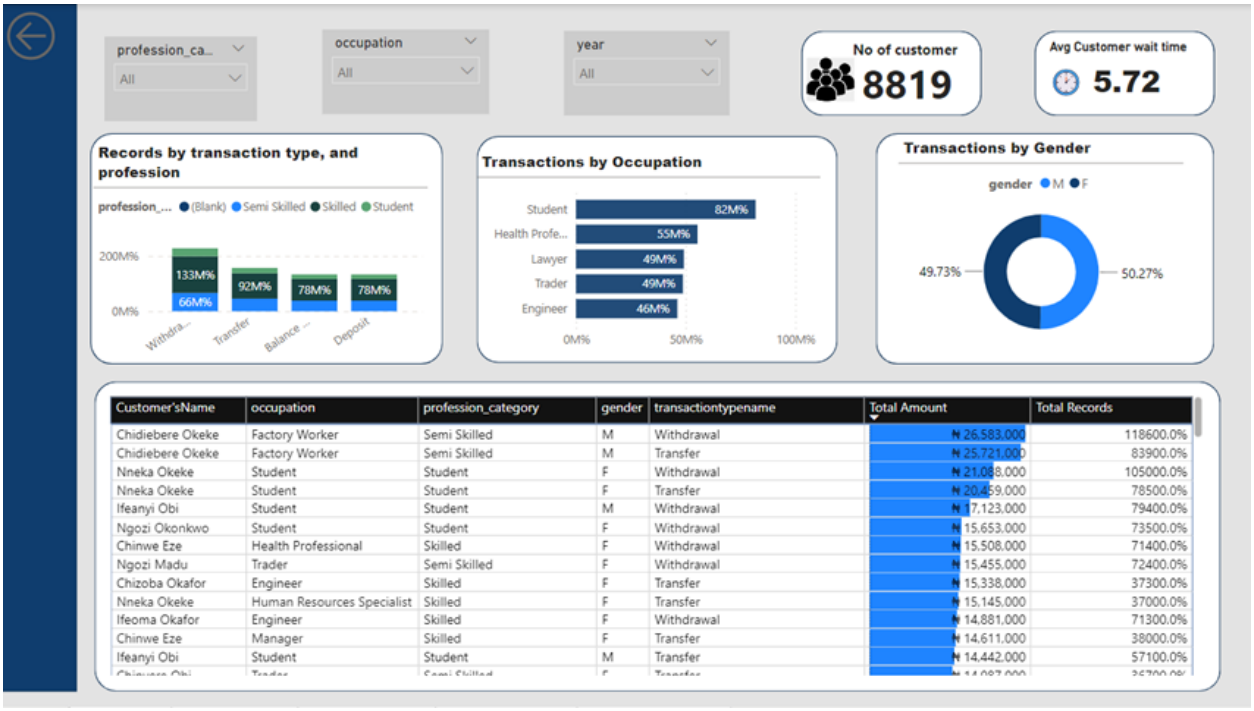
- These transactions covered a comprehensive record across five states, with Lagos leading the merger at 43 billion, followed by Enugu with 21 billion, Rivers states with 20 billion, Kano states with 17 billion, and the FCT with 8 billion.
- During the study period, the peak hours for transactions across the states occurred between 15:00 and 18:00 (African Time).
- Throughout the years of this optimization, approximately 50 locations were included, with a total of 131 ATM machines spread across these locations.
- In the maintenance analysis of the year 2022, the best-maintained ATM was located in Rivers states with location IDs RI-9-002, RI-9-001, and RI-8-002. Conversely, the worst-maintained ATM was found in Enugu states with location IDs EN-1-002, EN-1-002, and EN-1-003. The average wait time for users was recorded at 1.87 minutes, while the time taken to repair the ATM averaged 2.57 hours.
- The goal was to leverage the power of data analytics and a customer-centric approach to empower financial institutions, which enables in making informed decisions that would enhance customer retention and loyalty.

DASHBOARD

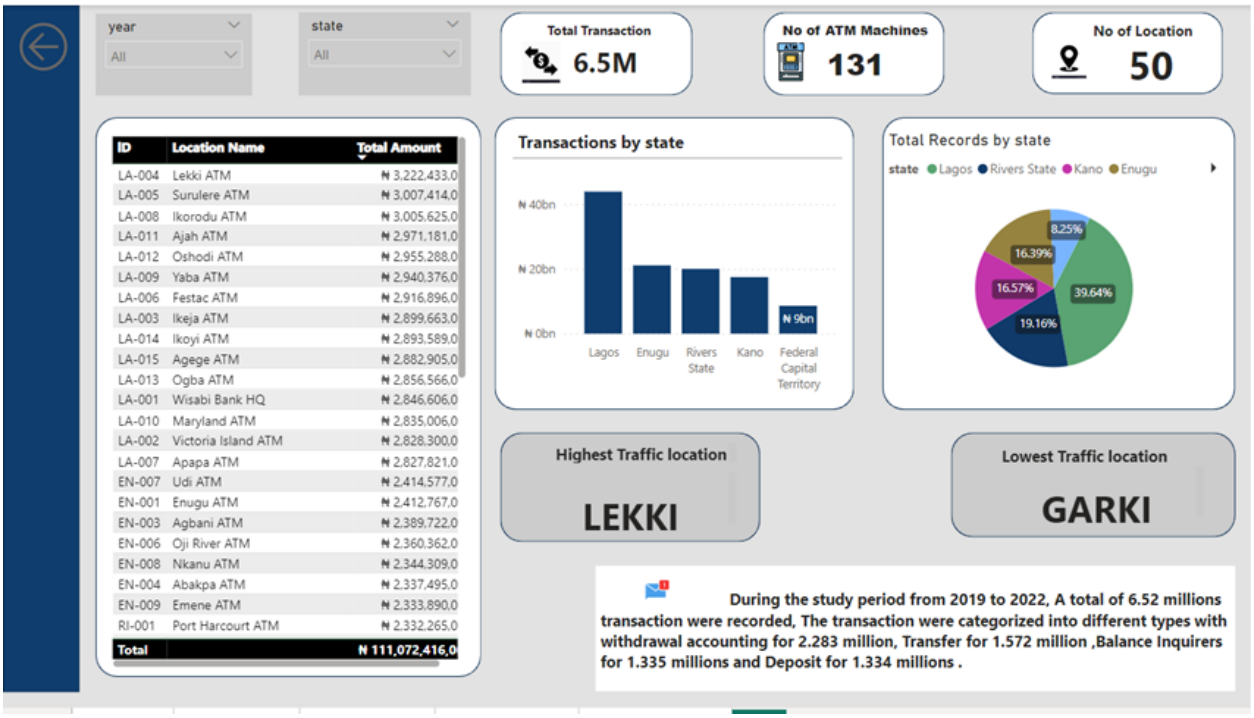
OVERVIEW



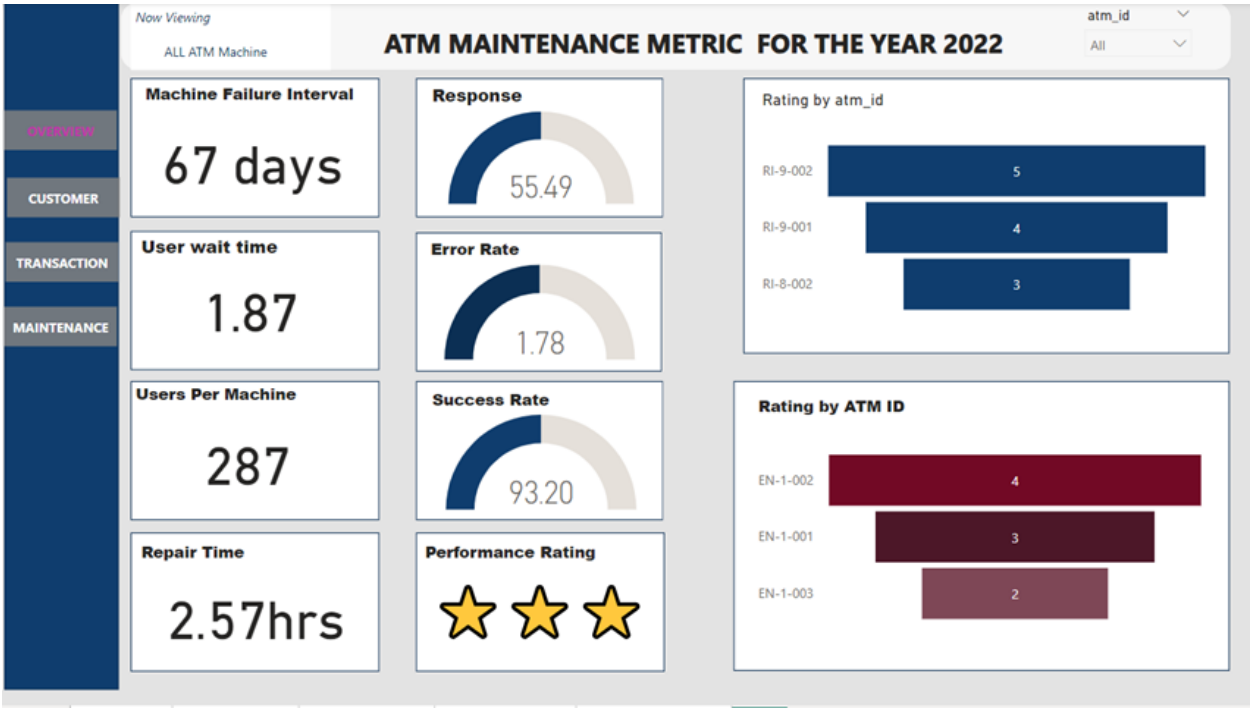
CUSTOMER:



TRANSACTIONS:



MAINTENANCE:



CONCLUSION:

To conclude, it is essential to regularly monitor and maintain ATMs for optimal efficiency. Swift repair times are crucial, given that the report highlights daily usage of ATMs. Locations in close proximity to student areas and rated worst ATM should undergo frequent checks. The report additionally provides insights into how banks can enhance customer's retaintion and loyalty.

Project Dataset : Here

Power bi : Here