**Abstract:**

This project aims to create a robust customer analytics system capable of generating insights for customer segmentation, churn prediction, and lifecycle value estimation. Data is sourced from multiple CRM and operational systems, aggregated into CSV files, and processed using a custom ETL pipeline. The data is then stored in a star schema-based PostgreSQL data warehouse. An analysis layer performs advanced computations and exports the results to CSV for visualization. Tableau dashboards provide intuitive visualizations for actionable insights. This system demonstrates the seamless integration of data engineering, analysis, and visualization to enhance customer understanding and drive data-driven business decisions.

**Introduction :**

**Problem Statement**

Understanding customer behavior is a cornerstone for businesses seeking to enhance customer retention and drive growth. Despite its importance, many organizations lack a systematic approach to analyze customer data effectively, which prevents them from gaining actionable insights and tailoring their strategies.

**Relevance**

Adopting data-driven methodologies allows businesses to segment customers accurately, anticipate churn risks, and compute customer lifecycle value. By employing advanced ETL pipelines and visualization tools, organizations can transform raw data into precise and actionable insights, ultimately enhancing decision-making processes and strategic planning.

**Scope**

This project aims to build an end-to-end system that converts raw customer data into meaningful insights. It involves designing ETL pipelines for data processing, implementing a star schema-based data warehouse for efficient storage and querying, performing advanced analyses, and leveraging Tableau for intuitive data visualizations. Together, these components provide a comprehensive framework for customer analytics.

**Objectives**

* Develop an ETL pipeline to transform raw CSV data into a clean, structured format.
* Implement a star schema data warehouse in PostgreSQL for efficient query performance.
* Perform advanced analysis for customer segmentation, churn prediction, and lifecycle value computation.
* Create dashboards using Tableau to visualize insights effectively.

**System Architecture**

The system consists of four main components:

* **ETL Layer**: Processes raw data collected from multiple CRM and operational systems and loads it into the data warehouse.
* **Data Warehouse**: Stores processed data in a star schema for efficient querying and analysis.
* **Analysis Layer**: Extracts data from the warehouse for advanced computations.
* **Visualization Layer**: Uses Tableau to present the analysis results through interactive dashboards.

**Architecture Diagram**

(Include a high-level diagram showing data flow from CRM systems and CSV files to Tableau dashboards.)

**Methodology**

**Data Collection**

Data for this project is sourced from various Customer Relationship Management (CRM) platforms and operational systems. These systems provide rich datasets, including customer demographics, transaction history, and engagement metrics. The data is exported in CSV format, which serves as the input for the ETL process.

**Data Transformation**

The extracted CSV files undergo rigorous cleaning and transformation to ensure consistency and accuracy. This includes handling missing values, resolving inconsistencies, and normalizing data formats. For example:

* Standardizing date formats across all datasets.
* Converting categorical fields like customer\_tier into numerical representations for better analysis.

**Data Integration**

Transformed data from the CSV files is integrated into a star schema in the PostgreSQL data warehouse. The schema design prioritizes query performance and analytical efficiency, aligning with business intelligence best practices.

**Analysis Techniques**

The analysis layer retrieves data from the warehouse and applies advanced statistical and machine learning models to derive insights such as:

* Customer segmentation based on demographic and transactional behaviors.
* Predicting churn likelihood using historical engagement patterns.
* Estimating customer lifecycle value through transactional trends.

**Visualization**

Results from the analysis layer are exported into structured CSV files. Tableau consumes these outputs to create dynamic and interactive dashboards. These dashboards allow users to:

* Explore segmentation clusters visually.
* Track churn prediction trends.
* Analyze lifecycle value distributions across customer cohorts

**Implementation Details**

**ETL Process**

The ETL (Extract, Transform, Load) pipeline is the backbone of this project. It consists of:

1. Extraction: Reading raw data from CSV files provided by CRM and operational systems.
2. Transformation: Cleaning and enriching the data, including:
   * Removing duplicates.
   * Filling missing values with appropriate statistical imputation.
   * Encoding categorical variables for downstream processing.
3. Loading: Populating the star schema in the PostgreSQL database.

The ETL process is implemented using Python, leveraging libraries like Pandas for data manipulation and SQLAlchemy for database interaction.

**Data Warehouse**

The PostgreSQL database is designed with a star schema structure. Key details include:

* Fact Tables: transactions and engagements contain the core transactional and behavioral data.
* Dimension Tables: customers and products store descriptive details supporting the fact tables.
* Indexes and Constraints: Implemented to optimize query performance and maintain data integrity.

**Analysis Layer**

The analysis layer uses Python-based tools, including Pandas and NumPy, to perform complex calculations. For instance:

* Customer segmentation is achieved through k-means clustering.
* Churn prediction leverages logistic regression models.
* Lifecycle value computation uses cumulative revenue metrics.

**Visualization** **Layer**

Interactive dashboards built in Tableau provide actionable insights. Key features include:

* Cluster analysis visualization to distinguish customer groups.
* Predictive dashboards highlighting churn risks.
* Lifecycle value heatmaps for strategic planning

**Source Code:**

**DB Schema:**

**Data Warehouse Star Schema**

CREATE TABLE customer\_tiers (

    tier\_id SERIAL PRIMARY KEY,

    tier\_name VARCHAR(20),

    discount\_rate DECIMAL(5, 2)

);

CREATE TABLE product\_categories (

    category\_id SERIAL PRIMARY KEY,

    category\_name VARCHAR(50)

);

CREATE TABLE payment\_methods (

    payment\_method\_id SERIAL PRIMARY KEY,

    payment\_method\_name VARCHAR(50)

);

CREATE TABLE customers (

    customer\_id SERIAL PRIMARY KEY,

    first\_name VARCHAR(50),

    last\_name VARCHAR(50),

    email VARCHAR(100),

    phone\_number VARCHAR(15),

    gender VARCHAR(10),

    dob DATE,

    age INTEGER,

    city VARCHAR(50),

    state VARCHAR(50),

    country VARCHAR(50),

    signup\_date DATE,

    is\_active BOOLEAN,

    customer\_tier INTEGER REFERENCES customer\_tiers(tier\_id)

);

CREATE TABLE transactions (

    transaction\_id SERIAL PRIMARY KEY,

    customer\_id INTEGER REFERENCES customers(customer\_id),

    transaction\_date TIMESTAMP,

    amount DECIMAL(10, 2),

    payment\_method INTEGER REFERENCES payment\_methods(payment\_method\_id),

    product\_id VARCHAR(50),

    product\_category INTEGER REFERENCES product\_categories(category\_id),

    quantity INTEGER,

    discount\_applied BOOLEAN,

    transaction\_status VARCHAR(20)

);

CREATE TABLE engagements (

    engagement\_id SERIAL PRIMARY KEY,

    customer\_id INTEGER REFERENCES customers(customer\_id),

    engagement\_date DATE,

    login\_frequency INTEGER,

    time\_spent DECIMAL(5, 2),

    pages\_visited INTEGER,

    purchase\_clicks INTEGER,

    feedback\_score INTEGER,

    email\_open\_rate DECIMAL(5, 2),

    promo\_redemptions INTEGER

);

**Analysis View**

CREATE VIEW customer\_summary AS

SELECT

    c.customer\_id,

    c.first\_name,

    c.last\_name,

    MAX(t.transaction\_date) AS last\_transaction\_date,

    COUNT(t.transaction\_id) AS total\_transactions,

    SUM(t.amount) AS total\_spent,

    AVG(t.amount) AS avg\_transaction\_amount,

    EXTRACT(DAY FROM '2021-12-31' - MAX(t.transaction\_date)) AS recency,

    CASE

        WHEN EXTRACT(DAY FROM '2021-12-31' - MAX(t.transaction\_date)) > 180 THEN TRUE

        ELSE FALSE

    END AS churn\_flag

FROM

    customers c

LEFT JOIN

    transactions t ON c.customer\_id = t.customer\_id

GROUP BY

    c.customer\_id, c.first\_name, c.last\_name;

**DB Models:**

from sqlalchemy import Column, Integer, String, Numeric, Boolean, Date, DateTime, ForeignKey

from sqlalchemy.orm import relationship

from sqlalchemy.ext.declarative import declarative\_base

Base = declarative\_base()

class CustomerTier(Base):

    \_\_tablename\_\_ = 'customer\_tiers'

    tier\_id = Column(Integer, primary\_key=True, autoincrement=True)

    tier\_name = Column(String(20), nullable=False, unique=True)

    discount\_rate = Column(Numeric(5, 2), nullable=True)

    customers = relationship("Customer", back\_populates="tier")

class ProductCategory(Base):

    \_\_tablename\_\_ = 'product\_categories'

    category\_id = Column(Integer, primary\_key=True, autoincrement=True)

    category\_name = Column(String(50), nullable=False, unique=True)

    transactions = relationship("Transaction", back\_populates="category")

class PaymentMethod(Base):

    \_\_tablename\_\_ = 'payment\_methods'

    payment\_method\_id = Column(Integer, primary\_key=True, autoincrement=True)

    payment\_method\_name = Column(String(50), nullable=False, unique=True)

    transactions = relationship(

        "Transaction",

        back\_populates="payment\_method\_rel",

        foreign\_keys="[Transaction.payment\_method]"

    )

class Customer(Base):

    \_\_tablename\_\_ = 'customers'

    customer\_id = Column(Integer, primary\_key=True, autoincrement=True)

    first\_name = Column(String(50), nullable=False)

    last\_name = Column(String(50), nullable=False)

    email = Column(String(100), unique=True)

    phone\_number = Column(String(15))

    gender = Column(String(10))

    dob = Column(Date)

    age = Column(Integer)

    city = Column(String(50))

    state = Column(String(50))

    country = Column(String(50))

    signup\_date = Column(Date)

    is\_active = Column(Boolean, default=True)

    customer\_tier = Column(Integer, ForeignKey('customer\_tiers.tier\_id'))

    tier = relationship("CustomerTier", back\_populates="customers")

    transactions = relationship("Transaction", back\_populates="customer")

    engagements = relationship("Engagement", back\_populates="customer")

class Transaction(Base):

    \_\_tablename\_\_ = 'transactions'

    transaction\_id = Column(Integer, primary\_key=True, autoincrement=True)

    customer\_id = Column(Integer, ForeignKey('customers.customer\_id'))

    payment\_method = Column(Integer, ForeignKey('payment\_methods.payment\_method\_id'))

    product\_category = Column(Integer, ForeignKey('product\_categories.category\_id'))

    transaction\_date = Column(DateTime)

    amount = Column(Numeric(10, 2))

    product\_id = Column(String(50))

    quantity = Column(Integer)

    discount\_applied = Column(Boolean)

    transaction\_status = Column(String(20))

    customer = relationship("Customer", back\_populates="transactions")

    payment\_method\_rel = relationship("PaymentMethod", back\_populates="transactions")

    category = relationship("ProductCategory", back\_populates="transactions")

class Engagement(Base):

    \_\_tablename\_\_ = 'engagements'

    engagement\_id = Column(Integer, primary\_key=True, autoincrement=True)

    customer\_id = Column(Integer, ForeignKey('customers.customer\_id'))

    engagement\_date = Column(Date)

    login\_frequency = Column(Integer)

    time\_spent = Column(Numeric(5, 2))

    pages\_visited = Column(Integer)

    purchase\_clicks = Column(Integer)

    feedback\_score = Column(Numeric(3, 1))

    email\_open\_rate = Column(Numeric(5, 2))

    promo\_redemptions = Column(Integer)

    customer = relationship("Customer", back\_populates="engagements")

**ETL Layer:**

import pandas as pd

from sqlalchemy import create\_engine

from sqlalchemy.orm import sessionmaker

from sqlalchemy.dialects.postgresql import insert

from ..db\_connection import DB\_USER, DB\_PASSWORD, DB\_HOST, DB\_PORT, DB\_NAME

from ..models import (

    Base,

    CustomerTier,

    PaymentMethod,

    ProductCategory,

    Customer,

    Transaction,

    Engagement

)

def create\_db\_engine():

    try:

        connection\_string = f'postgresql+psycopg2://{DB\_USER}:{DB\_PASSWORD}@{DB\_HOST}:{DB\_PORT}/{DB\_NAME}'

        engine = create\_engine(connection\_string)

        return engine

    except Exception as error:

        print(f"Error creating database engine: {error}")

        return None

def get\_or\_create\_reference\_data(session, model, name\_column, value):

    existing = session.query(model).filter(

        getattr(model, name\_column) == value

    ).first()

    if existing:

        return existing

    new\_record = model(\*\*{name\_column: value})

    session.add(new\_record)

    session.commit()

    return new\_record

def process\_customers\_data(session, df):

    for \_, row in df.iterrows():

        tier = get\_or\_create\_reference\_data(

            session,

            CustomerTier,

            'tier\_name',

            row['customer\_tier']

        )

        customer = Customer(

            first\_name=row['first\_name'],

            last\_name=row['last\_name'],

            email=row['email'],

            phone\_number=row['phone\_number'],

            gender=row['gender'],

            dob=row['dob'],

            age=row['age'],

            city=row['city'],

            state=row['state'],

            country=row['country'],

            signup\_date=row['signup\_date'],

            is\_active=row['is\_active'],

            customer\_tier=tier.tier\_id

        )

        session.add(customer)

    session.commit()

def process\_transactions\_data(session, df):

    for \_, row in df.iterrows():

        payment\_method = get\_or\_create\_reference\_data(

            session,

            PaymentMethod,

            'payment\_method\_name',

            row['payment\_method']

        )

        product\_category = get\_or\_create\_reference\_data(

            session,

            ProductCategory,

            'category\_name',

            row['product\_category']

        )

        transaction = Transaction(

            customer\_id=row['customer\_id'],

            transaction\_date=row['transaction\_date'],

            amount=row['amount'],

            payment\_method=payment\_method.payment\_method\_id,

            product\_id=row['product\_id'],

            product\_category=product\_category.category\_id,

            quantity=row['quantity'],

            discount\_applied=row['discount\_applied'],

            transaction\_status=row['transaction\_status']

        )

        session.add(transaction)

    session.commit()

def process\_engagements\_data(session, df):

    for \_, row in df.iterrows():

        customer = session.query(Customer).filter\_by(

            customer\_id=row['customer\_id']

        ).first()

        engagement = Engagement(

            customer\_id=row['customer\_id'],

            engagement\_date=row['engagement\_date'],

            login\_frequency=row['login\_frequency'],

            time\_spent=row['time\_spent'],

            pages\_visited=row['pages\_visited'],

            purchase\_clicks=row['purchase\_clicks'],

            feedback\_score=row['feedback\_score'],

            email\_open\_rate=row['email\_open\_rate'],

            promo\_redemptions=row['promo\_redemptions']

        )

        session.add(engagement)

    session.commit()

def etl\_process(csv\_file, process\_function):

    try:

        engine = create\_db\_engine()

        if not engine:

            return

        Base.metadata.create\_all(engine)

        Session = sessionmaker(bind=engine)

        session = Session()

                df = pd.read\_csv(csv\_file)

        process\_function(session, df)

        session.close()

        print(f"Data loaded successfully from {csv\_file}")

    except Exception as error:

        print(f"Error in ETL process: {error}")

def main():

    customers\_file = r'E:\Programs\CustomerSegmentation\_DW\data\processed\cleaned\_customers.csv'

    transactions\_file = r'E:\Programs\CustomerSegmentation\_DW\data\processed\cleaned\_transactions.csv'

    engagements\_file = r'E:\Programs\CustomerSegmentation\_DW\data\processed\cleaned\_engagements.csv'

    etl\_process(customers\_file, process\_customers\_data)

    etl\_process(transactions\_file, process\_transactions\_data)

    etl\_process(engagements\_file, process\_engagements\_data)

if \_\_name\_\_ == '\_\_main\_\_':

    main()

**Analysis Layer:**  
import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

from sklearn.impute import SimpleImputer

from sqlalchemy import create\_engine

from ..db\_connection import DB\_USER, DB\_PASSWORD, DB\_HOST, DB\_PORT, DB\_NAME

engine = create\_engine(f'postgresql+psycopg2://{DB\_USER}:{DB\_PASSWORD}@{DB\_HOST}:{DB\_PORT}/{DB\_NAME}')

df = pd.read\_sql("SELECT \* FROM customer\_summary", engine)

df['R'] = pd.qcut(df['recency'], 4, labels=[3, 2, 1, 0], duplicates="drop")  *# Higher recency = lower score*

df['F'] = pd.qcut(df['total\_transactions'], 4, labels=[0, 1, 2], duplicates="drop")  *# Higher frequency = higher score*

df['M'] = pd.qcut(df['total\_spent'], 4, labels=[0, 1, 2, 3], duplicates="drop")  *# Higher monetary = higher score*

df['RFM\_Score'] = df['R'].astype(str) + df['F'].astype(str) + df['M'].astype(str)

def rfm\_level(df):

    if df['RFM\_Score'][1:] in ['33', '23', '13', '03']:

        return 'Champions'

    elif df['RFM\_Score'][1:] in ['32', '22', '12', '02']:

        return 'Potential Loyalists'

    elif df['RFM\_Score'][1:] in ['31', '21', '11', '01']:

        return 'At Risk Customers'

    elif df['RFM\_Score'][1:] in ['30', '20', '10', '00']:

        return 'Hibernating'

    else:

        return 'Other'

df['Customer\_Segment'] = df.apply(rfm\_level, axis=1)

X = df[['recency', 'total\_transactions', 'total\_spent']]

y = df['churn\_flag']

imputer = SimpleImputer(strategy='mean')  *# You can also use 'median' or 'most\_frequent'*

X = imputer.fit\_transform(X)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = LogisticRegression()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Churn Prediction Model Accuracy: {accuracy \* 100:.2f}%")

df['churn\_prediction'] = model.predict(X)

df['predicted\_clv'] = df['total\_spent'] \* (df['total\_transactions'] / (df['recency'] + 1))

df = df.dropna()

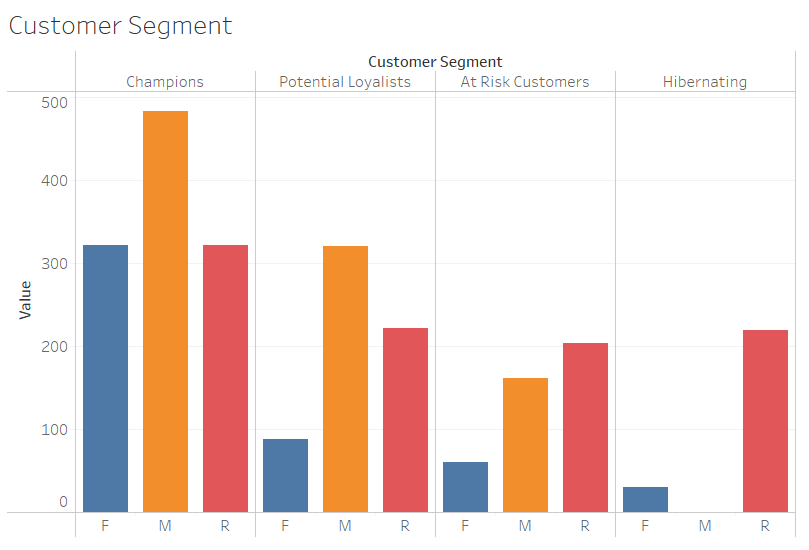
clv\_csv\_path = r'E:\Programs\CustomerSegmentation\_DW\src\visualizer\exports\clv\_estimations.csv'

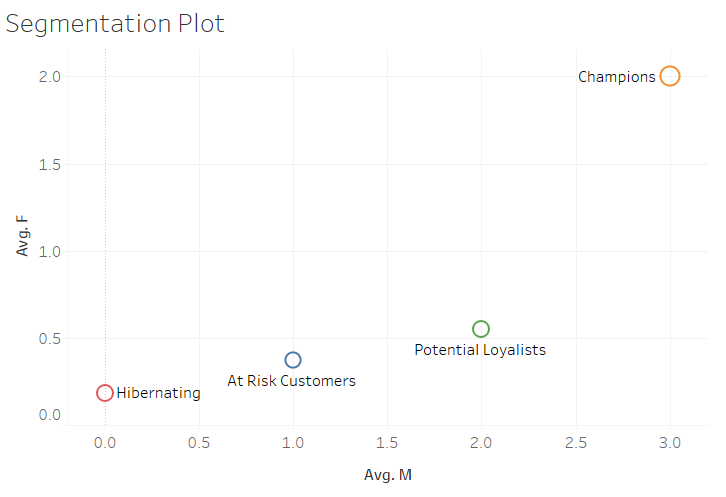
df.to\_csv(clv\_csv\_path, index=False)

print(f"CLV estimations saved to {clv\_csv\_path}")

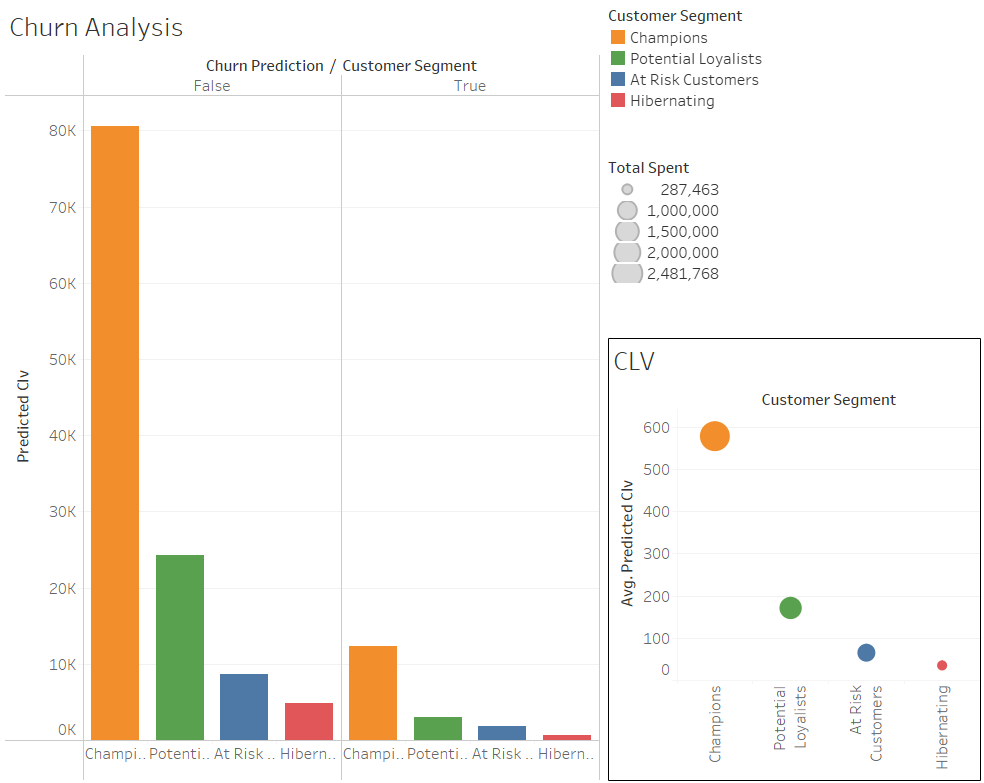
**Visualizations:**

**Customer Segments:**

****

****

**Churn Prediction and CLV :**

****

**Results and Visualizations**

**Segmentation Insights**

Tableau dashboards reveal distinct customer clusters based on demographics and transaction behavior. For example:

* High-value customers with frequent transactions.
* Low-engagement customers with sporadic purchases.

**Churn Prediction**

Predictive models achieve high accuracy in identifying churn risks. Dashboards provide a visual representation of churn likelihood across customer segments, aiding in targeted retention strategies.

**Lifecycle Value Estimation**

Visualizations highlight trends in customer lifetime value (CLV), enabling businesses to identify high-value customers and allocate resources effectively.

**Challenges and Limitations**

**Challenges**

* Data Quality Issues: Handling missing and inconsistent data required extensive preprocessing.
* Schema Design: Optimizing the star schema for diverse analytical queries was non-trivial.
* ETL Automation: Ensuring the robustness of the ETL process for large datasets posed challenges.

**Limitations**

* The system relies on periodic CSV exports, which might not reflect real-time data changes.
* Predictive models require continuous refinement for accuracy improvements.

**Future Scope**

* Real-time Data Integration: Implementing APIs for real-time data updates from CRM systems.
* Advanced Analytics: Incorporating deep learning models for enhanced predictive capabilities.
* Scalability: Transitioning to cloud-based architectures for handling larger datasets and concurrent users.

**Appendix:**

**Github Source**: https://github.com/I-am-Aswin/Customer\_Segmentation\_and\_Analysis\_DW.git