VALLEY BANK - FCC ANALYTICS ENGINEER ->TAKE HOME ASSESSMENT
-SURYARAJ MACHANI-

Customer segmentation problem based on transaction behaviour and demographics.

Data Sources:

- Transaction DataMart: Contains information about transactions, including transaction amount, timestamp, Customer ID, and transaction type.
- Customer Data: Includes demographic information such as age, gender, profession, work experience and family size.
- Occupational Employment and Wage Statistics: The US Bureau of Labor Statistics (BLS) data released in May 2022. Mainly focused on the Annual Median Wage Column.

Tasks:

1. **Data Ingestion and Cleaning:**

   Utilised MySQL workbench as RDBMS tool and ingested transaction data and customer data into a MySQL database. Name the database schema as 'project'.

   1. Created customer_datamart table and imported the data from excel file and added a constraint that Customer_ID is the primary key . No duplicate values found .The data looks clean and has no missing value.

Customer_ID	Gender	Age	Profession_Code	Work_Experience	Family_Size
1000	Male	19	53-0000	1	3
1001	Female	31	25-3031	6	2
1002	Male	23	41-0000	1	2
1003	Female	35	15-1244	9	4
1004	Female	24	53-7000	2	1
1005	Male	23	51-0000	3	2

Customer_ID	Gender	Age	Profession_Code	Work_Experience	Family_Size
10999	Female	37	11-9000	9	4
10998	Female	45	41-4010	18	4
10997	Female	30	31-1131	5	4
10996	Male	37	27-0000	9	4
10995	Male	22	43-4051	3	2

2. Created transaction_datamart table and imported the data from excel file . Found that this table doesn't have an unique identifier(PK). It is a good practice to include an unique column identifying the rows. So added a column by concatenating customer_ID,Amount and Transaction_Type which could be used to uniquely identify rows in the table. Made this column (PK)

ind	Customer_ID	Timestamp	Amount	Transaction_Type
76286783927376Deposit	7628	2023-09-28 01:32:00	67839.27376	Deposit
94036656738017Withdrawal	9403	2023-12-30 17:25:00	665.6738017	Withdrawal
41533881963898Deposit	4153	2023-10-24 17:58:00	38819.63898	Deposit
84492771222971Deposit	8449	2024-02-11 01:05:00	27712.22971	Deposit
13201738976438Withdrawal	1320	2024-01-12 11:40:00	1738.976438	Withdrawal

ind	Customer_ID	Timestamp	Amount	Transaction_Type
99997779077195Withdrawal	9999	2023-11-22 03:48:00	777.9077195	Withdrawal
99997432725287Transfer	9999	2024-02-01 05:21:00	743.2725287	Transfer
99996495675465Card	9999	2023-09-30 08:56:00	64.95675465	Card
99994536832801Check	9999	2024-02-24 18:14:00	4536.832801	Check
99994416921534Deposit	9999	2024-01-25 06:43:00	44169.21534	Deposit

3. Obtaining the Annual Median Wages from the US Bureau of Labor Statistics (BLS) and ingesting it into the customer table (customer).

   - Cleaned and removed all the unnecessary columns from national_M2022_dl. Removed Area,Area_Title, Area_Type, Prim_State, NAICS, NAICS_TITLE, I_GROUP, JOBS_1000,LOC_QUOTIENT,PCT_TOTAL,PCT_RPT,ANNUAL and HOURLY in excel. Ingested this data into the project database as 'bls_data'. Added Columns Annual_Median and Occupation_Name into customer_datamart table.
   - Updated Annual_Median and Occupation_Name in customer_datamart table by extracting annual_median and occupation_name from bls_data. Simply performed a left join. Assumed Profession_Code to be same as OCC_code.

Customer_ID	Gender	Age	Profession_Code	Work_Experience	Family_Size	Annual_Median	Occupation_Name
1000	Male	19	53-0000	1	3	37,940	Transportation and Material Moving Occupations
1001	Female	31	25-3031	6	2	35,250	Substitute Teachers, Short-Term
1002	Male	23	41-0000	1	2	35,290	Sales and Related Occupations
1003	Female	35	15-1244	9	4	90,520	Network and Computer Systems Administrators
1004	Female	24	53-7000	2	1	35,670	Material Moving Workers

Customer_ID	Gender	Age	Profession_Code	Work_Experience	Family_Size	Annual_Median	Occupation_Name
10999	Female	37	11-9000	9	4	99,740	Other Management Occupations
10998	Female	45	41-4010	18	4	67,750	Sales Representatives, Wholesale and Manufac...
10997	Female	30	31-1131	5	4	35,760	Nursing Assistants
10996	Male	37	27-0000	9	4	58,030	Arts, Design, Entertainment, Sports, and Media...
10995	Male	22	43-4051	3	2	37,780	Customer Service Representatives

4. Data Preprocessing and Cleaning:

- Checked for null values - No null values found.
- Replaced ‘,’ in Annual_Median column in customer_datamart with empty ‘’.

Customer_ID	Gender	Age	Profession_Code	Work_Experience	Family_Size	Annual_Median	Occupation_Name
1000	Male	19	53-0000	1	3	37940	Transportation and Material Moving Occupations
1001	Female	31	25-3031	6	2	35250	Substitute Teachers, Short-Term
1002	Male	23	41-0000	1	2	35290	Sales and Related Occupations
1003	Female	35	15-1244	9	4	90520	Network and Computer Systems Administrators
1004	[illegible]	24	53-7000	2	1	35670	Material Moving Workers

- Since '*' in the 'annual_median' column signifies an unavailable wage estimate and ‘#’ includes wages greater than $239,200 per year, deleting these rows would be a better option as we lack the true estimate of their salary. - Just 5 rows affected out of 10000 records.

  If provided with an exact wage , we could have included it in the dataset. # could be any value ranging from ‘$239,200’ to anything. So it is better to drop those values as this may affect the clustering algorithm analysis.

- In the transaction_datamart table the timestamp column is converted into a format that would be recognized and interpreted by mysql . That is converting it into '%m/%d/%Y %H:%i’ format.

Customer_ID	Timestamp	Amount	Transaction_Type	ind
10000	2023-10-01 08:03:00	[illegible]	Transfer	100001381886017Transfer
10000	2023-10-19 0[illegible]	[illegible]	Deposit	100001559259521Deposit
10000	2023-09-03 12:57:00	2285.379455	Check	100002285379455Check
10000	2023-09-17 19:11:00	27130.39672	Loan Payment	100002713039672Loan Payment
10000	2023-10-06 16:01:00	27263.79146	Loan Payment	100002726379146Loan Payment

2. Stored Procedure Development:

1. Created a table as per the requirements.(Account_profile).
2. Created a stored procedure and performed the aggregations tasks as required. Named the procedure as AccountProfileProcedure and called it. The below account_table table was generated .

Customer_ID	Card_Avg	Check_Avg	Deposit_Avg	Loan_Payment_Avg	Transfer_Avg	Withdrawal_Avg	Card_Count	Check_Count	Deposit_Count	Loan_Payment_Count	Transfer_Count	Withdrawal_Count
1000	29.70	3323.04	53392.84	22722.13	NULL	793.09	1	2	1	1	NULL	2
1001	52.77	7874.51	52811.10	16223.55	2030.74	1539.00	4	6	1	2	2	1
1002	NULL	4051.47	26184.28	NULL	NULL	NULL	NULL	1	2	NULL	NULL	NULL
1003	19.48	3339.02	NULL	25074.55	903.15	1329.38	1	2	NULL	3	1	2
1004	NULL	5802.39	18112.69	NULL	1515.54	583.86	NULL	4	1	NULL	3	1

3. Implemented an after insert trigger . When someone inserts a new row into the transaction_datamart, the trigger updates the Account_profile table accordingly.
4. Check the performance of the Stored Procedure. Inserted a new row into the transaction_datamart . The trigger works well and was able to update the account profile table.
   The only issue is . It is taking 4 s in average to run the query . Which is not desired in real world scenarios. Optimised the Stored Procedure implementation with few changes and created an Advanced Stored procedure as below:

5. Optimized Stored Procedure:
   The main issue causing longer procedure run times was initially related to users adding new rows to the customer_datamart. The AccountProfileProcedure was defined to operate on the entire dataset, thus leading to increased processing time.

   This issue was subsequently addressed by optimizing the procedure. A new procedure, named AdvancedAccountProfileProcedure, was introduced. This procedure accepts customer_ID as input, and when invoked, it retrieves only the information related to the specified customer_ID from the customer_datamart and updates it accordingly. As a result of this optimization, a significant reduction in procedure run time was achieved. On average, the processing time decreased to 63ms per update, representing a 96% decrease compared to the initial processing time.

   (The Idea was mine but I must admit I have utilized chatgpt to guide me through the programming hurdles. This is the only part where I referenced ChatGPT to help me. 🙂)

3. <u>Customer Segmentation:</u>

   1. Created a customer_transaction table and merged customer_datamart and transaction_datamart based on Customer_ID. Use this table in visual_insights.ipynb notebook . Where I did few supporting visualisations and found few key insights from the visualizations.

Customer_ID	Timestamp	Amount	Transaction_Type	Gender	Age	Profession_Code	Work_Experience	Family_Size	Annual_Median	Occupation_Name
10000	2023-10-01 08:03:00	1381.886017	Transfer	Male	36	27-3031	11	5	67440	Public Relations Specialists
10000	2023-10-19 09:49:00	15592.59521	Deposit	Male	36	27-3031	11	5	67440	Public Relations Specialists
10000	2023-09-03 12:57:00	2285.379455	Check	Male	36	27-3031	11	5	67440	Public Relations Specialists
10000	2023-09-17 19:11:00	27130.39672	Loan Payment	Male	36	27-3031	11	5	67440	Public Relations Specialists
10000	2023-10-06 16:01:00	27263.79146	Loan Payment	Male	36	27-3031	11	5	67440	Public Relations Specialists

2. Created a customer_segmentation table which includes transaction behaviour from account_profile table , customer demographics and annual_median wage.This table is further exported and used in customer_segmentation.ipynb notebook . Performed the clustering analysis here.

Field	Type	Null	Key
Customer_ID	int	NO	PRI
Gender	varchar(45)	YES	
Age	int	YES	
Profession_Code	varchar(45)	YES	
Work_Experience	int	YES	
Family_Size	int	YES	
Annual_Median	varchar(45)	YES	
Occupation_Name	varchar(255)	YES	
Card_Avg	double	YES	
Check_Avg	double	YES	
Deposit_Avg	double	YES	
Loan_Payment_Avg	double	YES	
Transfer_Avg	double	YES	
Withdrawal_Avg	double	YES	
Card_Count	int	YES	
Check_Count	int	YES	
Deposit_Count	int	YES	
Loan_Payment_C...	int	YES	
Transfer_Count	int	YES	
Withdrawal_Count	int	YES	

Please refer to Jupyter Notebooks for detailed analysis. I have explained the process by adding markdown text in the notebook. After reviewing customer_segmentation.ipynb notebook please visit the documentation again for inference.

INFERENCE:

From these plots we can compare each cluster and look at their distributions. Key Findings:

- Clusters 4 and 1 are characterized by customers with higher levels of work experience and older age compared to other clusters. These clusters also exhibit higher annual median salaries compared to others. Moreover, customers in these clusters tend to engage more in deposit activities and have lower instances of loan payments compared to other customers.

- Clusters 4 and 1 have a lower customer population compared to other clusters. This observation indicates that the segments represented by these clusters constitute a smaller portion of the overall customer base.

Key Insights for Valley Bank:

- With the understanding that customers in Clusters 4 and 1 are likely to be older,have higher incomes, and engage more in deposit activities, the Valley Bank can tailor marketing campaigns to cater to the needs and preferences of these segments. For example, targeted promotions for retirement planning services or higher-value deposit products could be implemented.

- Understanding that customers in these clusters have lower instances of loan payments could indicate lower credit risk. However, it's essential to conduct further analysis to assess the underlying factors contributing to this behavior and to ensure prudent risk management practices are maintained.

- Analyzing the risk profiles of customers within each cluster can aid in assessing credit risk and implementing appropriate risk mitigation measures. For instance, customers in Cluster 3, with their diverse characteristics, may require more risk assessment techniques to ensure responsible lending practices.

CHALLENGES:

1. When I was working with procedures and triggers, I caught an error: Explicit or implicit commit is not allowed in stored functions or triggers, which I solved by ensuring that my stored procedure or trigger does not contain any statements that would perform a commit operation. Instead,I designed my procedures and triggers to only perform data manipulation and other operations that are allowed within the context of a transaction.

2. Since the customer_segmentation data contained numerous missing values, it was not advisable to feed such data into K-means clustering. Therefore, I initially filled the missing values with 0 and formed clusters. Using 0 made sense because the missing values signify that the customer hasn’t made any transactions. However, this approach resulted in a very low Silhouette Score. My model treated 0's as outliers, which proved to be harmful.
   To address this issue, I implemented a better solution by imputing the missing values of the columns with the mean and median of the respective column. This approach yielded a better score of 0.60 and resulted in the formation of 5 clusters.

<u>FUTURE WORK:</u>

1. In the transaction table indexes could be added to improve the performance of data retrieval and manipulation operations . The main reason why I haven’t included

indexes is that we perform updates on the table monthly, and indexes are not a preferred option for this solution. It again depends on the application .

2. Better clustering algorithms, such as DBSCAN and hierarchical clustering, could have been implemented for improved analysis. However, due to limited time, I was not able to explore them.

3. Jupyter notebook can be directly connected to mysql database using sqllite3 package however for simplicity sake I have exported the data table and read the file in the notebook.

APPENDIX: