TERADATA CERTIFICATION PROJECT REPORT –EDUREKA

Manoj K V

05/13/2017

Problem Statement

The project should be able to segment the banking customers according to the business rules defined. Customer segmentation is the process of dividing the customers into groups who have similar characteristics in terms of their balance in various products.

Business Rules:

1. The project should segment the customers who hold Saving Accounts, Credit Card accounts and Mortgage accounts.

2. The project should exclude other accounts.

3. Customers should be segmented as following categories.

Low-Networth, Medium-Networth or High-Networth customers (Based on their balance held in the accounts)

4. The project should build a target table that contains customers and their corresponding segments.

Criteria for Segmentation:

1. If the customer meets *one of the below criteria*, then the segment of the customer will be

High-Networth.

Saving accounts balance > 5, 00,000

Credit Card Balance > 3, 00,000

Mortgage account amount > 50, 00,000

2. If the customer meets *one of the below criteria*, then the segment of the customer will be

Medium-Networth

Saving accounts balance > 2, 00,000

Credit Card Balance > 1, 00,000

Mortgage account amount > 10, 00,000

3. If the customer meets *one of the below criteria*, then the segment of the customer will be

Low-Networth

Saving accounts balance < 2, 00,000

Credit Card Balance < 1, 00,000

Mortgage account amount < 10, 00,000

Source system

Source system would provide below data:

- Customer - All customers of the bank
- Accounts - All open accounts
- Account Type – Reference table for accounts like Savings, Credit Card, Mortgage, and PPF.

DDLs:

Created the staging and target tables as per structure provided.

```
--Customer_Stg;

CREATE MULTISET TABLE edureka.Customer_Stg ,NO FALLBACK ,
     NO BEFORE JOURNAL,
     NO AFTER JOURNAL,
     CHECKSUM = DEFAULT,
     DEFAULT MERGEBLOCKRATIO
     (
      Customer_ID INTEGER,
      Customer_FName CHAR(15) CHARACTER SET LATIN NOT CASESPECIFIC,
      Customer_LName CHAR(15) CHARACTER SET LATIN NOT CASESPECIFIC,
      Customer_Dob DATE FORMAT 'yyyy-mm-dd',
      Customer_City CHAR(15) CHARACTER SET LATIN NOT CASESPECIFIC,
      Customer_State CHAR(2) CHARACTER SET LATIN NOT CASESPECIFIC,
      Customer_Phone CHAR(10) CHARACTER SET LATIN NOT CASESPECIFIC)
UNIQUE PRIMARY INDEX ( Customer_ID );


--Customer;

CREATE SET TABLE edureka.Customer ,NO FALLBACK ,
     NO BEFORE JOURNAL,
     NO AFTER JOURNAL,
     CHECKSUM = DEFAULT,
     DEFAULT MERGEBLOCKRATIO
     (
      Customer_ID INTEGER,
      Customer_FName CHAR(15) CHARACTER SET LATIN NOT CASESPECIFIC,
      Customer_LName CHAR(15) CHARACTER SET LATIN NOT CASESPECIFIC,
      Customer_Dob DATE FORMAT 'yyyy-mm-dd',
      Customer_City CHAR(15) CHARACTER SET LATIN NOT CASESPECIFIC,
      Customer_State CHAR(2) CHARACTER SET LATIN NOT CASESPECIFIC,
      Customer_Phone CHAR(10) CHARACTER SET LATIN NOT CASESPECIFIC)
UNIQUE PRIMARY INDEX ( Customer_ID );


--Accounts_Stg;

CREATE MULTISET TABLE edureka.Accounts_Stg ,NO FALLBACK ,
     NO BEFORE JOURNAL,
     NO AFTER JOURNAL,
     CHECKSUM = DEFAULT,
     DEFAULT MERGEBLOCKRATIO
     (
      Customer_Id INTEGER,
      Account_ID INTEGER,
      Account_Open_Dt DATE FORMAT 'yyyy-mm-dd',
      Account_Close_Dt DATE FORMAT 'yyyy-mm-dd',
      Account_Type_Cd BYTEINT,
      Account_Balance INTEGER)
UNIQUE PRIMARY INDEX ( Account_ID );
```

```sql
--Accounts;

CREATE SET TABLE edureka.Accounts ,NO FALLBACK ,
     NO BEFORE JOURNAL,
     NO AFTER JOURNAL,
     CHECKSUM = DEFAULT,
     DEFAULT MERGEBLOCKRATIO
     (
      Customer_Id INTEGER,
      Account_ID INTEGER,
      Account_Open_Dt DATE FORMAT 'yyyy-mm-dd',
      Account_Close_Dt DATE FORMAT 'yyyy-mm-dd',
      Account_Type_Cd BYTEINT,
      Account_Balance INTEGER)
UNIQUE PRIMARY INDEX ( Account_ID )
INDEX ( Customer_Id );


--Account_Type;

CREATE SET TABLE edureka.Account_Type ,NO FALLBACK ,
     NO BEFORE JOURNAL,
     NO AFTER JOURNAL,
     CHECKSUM = DEFAULT,
     DEFAULT MERGEBLOCKRATIO
     (
      Account_Type_Cd BYTEINT,
      Account_Type_Desc CHAR(10) CHARACTER SET LATIN NOT CASESPECIFIC)
UNIQUE PRIMARY INDEX ( Account_Type_Cd );


--Customer_First_Account;

CREATE SET TABLE edureka.Customer_First_Account ,NO FALLBACK ,
     NO BEFORE JOURNAL,
     NO AFTER JOURNAL,
     CHECKSUM = DEFAULT,
     DEFAULT MERGEBLOCKRATIO
     (
      Customer_Id INTEGER,
      First_Account_Dt DATE FORMAT 'yyyy-mm-dd')
UNIQUE PRIMARY INDEX ( Customer_Id );


--Customer_Accounts;

CREATE SET TABLE edureka.Customer_Accounts ,NO FALLBACK ,
     NO BEFORE JOURNAL,
     NO AFTER JOURNAL,
     CHECKSUM = DEFAULT,
     DEFAULT MERGEBLOCKRATIO
     (
      Customer_Id INTEGER,
      Savings_Balance INTEGER,
      Credit_Balance INTEGER,
      Mortgage_Balance INTEGER)
UNIQUE PRIMARY INDEX ( Customer_Id );


--Customer_Segment;

CREATE SET TABLE edureka.Customer_Segment ,NO FALLBACK ,
     NO BEFORE JOURNAL,
     NO AFTER JOURNAL,
     CHECKSUM = DEFAULT,
     DEFAULT MERGEBLOCKRATIO
     (
      Customer_Id INTEGER,
      Customer_FName CHAR(15) CHARACTER SET LATIN NOT CASESPECIFIC,
      Customer_LName CHAR(15) CHARACTER SET LATIN NOT CASESPECIFIC,
      Customer_Type CHAR(1) CHARACTER SET LATIN NOT CASESPECIFIC,
      Customer_Segment_Value CHAR(15) CHARACTER SET LATIN NOT CASESPECIFIC,
      Segment_Start_Dt DATE FORMAT 'yyyy-mm-dd',
      Segment_End_Dt DATE FORMAT 'yyyy-mm-dd')
UNIQUE PRIMARY INDEX ( Customer_Id );
```

Scripts:

1. Written a fastload script named as ***customerload.fl*** to load the given ***Customer.txt*** into staging table *Customer_Stg*.

```
fastload <<EOF
sessions 16;

logon 127.0.0.1/dbc,dbc;

DROP TABLE edureka.UV_Customer_Stg;
DROP TABLE edureka.ET_Customer_Stg;

delete from edureka.Customer_Stg;

set record vartext ",";

define
Customer_ID (varchar(15)),
Customer_FName (varchar(15)),
Customer_LName (varchar(15)),
Customer_Dob (varchar(15)),
Customer_City (varchar(15)),
Customer_State (varchar(2)),
Customer_Phone (varchar(10)),

file=Customer.txt
;

begin loading edureka.Customer_Stg
errorfiles edureka.ET_Customer_Stg, edureka.UV_Customer_Stg
;

insert into edureka.Customer_Stg
(
:Customer_ID,
:Customer_FName,
:Customer_LName,
:Customer_Dob,
:Customer_City,
:Customer_State,
:Customer_Phone
);

end loading;

logoff;

EOF
```

2. Written a fastload script named as ***accountsload.fl*** to load the below ***Accounts.txt*** into staging table ***Accounts_Stg***;

```
fastload <<EOF
sessions 16;

logon 127.0.0.1/dbc,dbc;

DROP TABLE edureka.UV_Accounts_Stg;
DROP TABLE edureka.ET_Accounts_Stg;

delete from edureka.Accounts_Stg;

set record vartext ",";

define
Customer_ID (varchar(15)),
Account_ID (varchar(15)),
Account_Open_Dt (varchar(15)),
Account_Close_Dt (varchar(15)),
Account_Type_Cd (varchar(15)),
Account_Balance (varchar(15)),

file=Accounts.txt
;
begin loading edureka.Accounts_Stg
errorfiles edureka.ET_Accounts_Stg, edureka.UV_Accounts_Stg
;

insert into edureka.Accounts_Stg
(
:Customer_Id,
:Account_ID,
:Account_Open_Dt,
:Account_Close_Dt,
:Account_Type_Cd,
:Account_Balance

);

end loading;

logoff;

EOF
```

3. Written insert queries to load the given records into ***Account_type*** table

```
insert into edureka.Account_type (Account_Type_Cd,Account_Type_Desc ) values
(01,'Savings');
insert into edureka.Account_type (Account_Type_Cd,Account_Type_Desc ) values
(02,'Credit Card');
insert into edureka.Account_type (Account_Type_Cd,Account_Type_Desc ) values
(03,'Mortgage');
insert into edureka.Account_type (Account_Type_Cd,Account_Type_Desc ) values
(04,'PPF');

select * from edureka.Account_type;
/*
Account_Type_Cd         Account_Type_Desc
        1             Savings
        2             Credit Car
        3             Mortgage
        4             PPF
*/
```

4.Written a BTEQ script named as ***CustomerAccountsLoad.bteq*** to load the ***Customer*** and ***Accounts*** table from their corresponding staging tables ***Customer_Stg*** and ***Accounts_Stg***; (BTEQ script also have the delete queries to delete the existing records from ***Customer*** and ***Accounts*** tables and load from ***Customer_Stg*** and ***Accounts_Stg***);

```
bteq <<EOF

.LOGON 127.0.0.1/dbc,dbc;

.MAXERROR 1

--Customer Table
DELETE FROM edureka.Customer;
INSERT INTO edureka.Customer
(
    Customer_ID,
    Customer_FName,
    Customer_LName,
    Customer_Dob,
    Customer_City,
    Customer_State,
    Customer_Phone
)
select
    STG.Customer_ID,
    STG.Customer_FName,
    STG.Customer_LName,
    STG.Customer_Dob,
    STG.Customer_City,
    STG.Customer_State,
    STG.Customer_Phone
FROM edureka.Customer_Stg STG
;

--Accounts Table
DELETE FROM edureka.Accounts ;
INSERT INTO edureka.Accounts
(
    Customer_Id,
    Account_ID,
    Account_Open_Dt,
    Account_Close_Dt,
    Account_Type_Cd,
    Account_Balance
)
select
    STG.Customer_Id,
    STG.Account_ID,
    STG.Account_Open_Dt,
    STG.Account_Close_Dt,
    STG.Account_Type_Cd,
    STG.Account_Balance
FROM edureka.Accounts_Stg STG
;

.LOGOFF

.QUIT 0;

EOF
```

5. Written a BTEQ script named as ***CustomerSegment.bteq***. This BTEQ script has below steps.

```
bteq <<EOF

.LOGON 127.0.0.1/dbc,dbc;

.MAXERROR 1

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@
Write a BTEQ script named as CustomerSegment.bteq. This BTEQ script should have below
steps.
/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@
1.Create a Volatile table named as "Customer_First_Account" with columns Customer_Id and
First_Account_Dt.
Write an Insert query to identify the first account open date for each customer from
Accounts table and
insert into "Customer_First_Account" table.
If the customer has multiple accounts, then the oldest Account_Open_Dt should be
considered.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@*/

CREATE MULTISET VOLATILE TABLE Customer_First_Account
     (
      Customer_Id INTEGER,
      First_Account_Dt DATE FORMAT 'yyyy-mm-dd'
      )
UNIQUE PRIMARY INDEX ( Customer_Id )
ON COMMIT PRESERVE ROWS;

INSERT INTO Customer_First_Account
(
    Customer_Id,
    First_Account_Dt
)
SELECT
    accnt.Customer_Id,
    accnt.Account_Open_Dt as First_Account_Dt

FROM edureka.Accounts accnt

QUALIFY ROW_NUMBER () OVER (partition by Customer_Id order by Account_Open_Dt)=1 --first
account open date/customer
;

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
2.Create a Volatile table named as "Customer_Accounts" with columns Customer_Id,
Savings_Balance, Credit_Balance and Mortgage_Balance.
Write an insert query to calculate the savings account balance, credit card balance and
mortgage balance
from Accounts table for each customer and insert into "Customer_Accounts" table.
For each customer, there should be only one record in this table which contains their
savings account balance, credit card balance and mortgage balance.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/

CREATE MULTISET VOLATILE TABLE Customer_Accounts
    (
      Customer_Id INTEGER,
      Savings_Balance INTEGER,
      Credit_Balance INTEGER,
      Mortgage_Balance INTEGER)
UNIQUE PRIMARY INDEX ( Customer_Id )
```

```
ON COMMIT PRESERVE ROWS;

INSERT INTO Customer_Accounts
(
    Customer_Id,
    Savings_Balance,
    Credit_Balance,
    Mortgage_Balance
)
SELECT
    bal.Customer_Id,
    bal.Savings_Balance,
    bal.Credit_Balance,
    bal.Mortgage_Balance
FROM
(
    SELECT
        Customer_Id,
        sum(coalesce( case when Account_Type_Cd=1 then Account_Balance end, 0)) as
Savings_Balance,
        sum(coalesce( case when Account_Type_Cd=2 then Account_Balance end, 0)) as
Credit_Balance,
        sum(coalesce( case when Account_Type_Cd=3 then Account_Balance end, 0)) as
Mortgage_Balance,
        sum(coalesce( case when Account_Type_Cd=4 then Account_Balance end, 0)) as
PPF_Balance
    FROM edureka.Accounts accnt

    WHERE Account_Type_Cd <> 4   --Excluding PPF
    GROUP BY 1
) bal
;
/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@
3.Write a delete query to delete existing data from "customer_segment" table
Insert into "Customer_Segment" table from Customer, "Customer_Accounts" and
"Customer_First_Account" tables. Criteria for each column is given below.
Column Customer_Type is set to N, if the first account opened date (First_Account_Dt from
"Customer_First_Account" table) of the customer is within last 3 months, otherwise it is
set to E.
Column "Customer_Segment"ation_Value is calculated based on the segmentation criteria
provided in problem statement.
Compare the Savings_Balance, Credit_Balance and Mortgage_Balance against the criteria
provided and assign "Customer_Segment"ation_value as High_Networth, Medium_Networth or
Low_Networth.
Column Segmentation_start_dt should be set to current date and segmentation_end_dt should
be set to 9999-12-31.
@@@@@@@@@@@@@@@@@@@@@@@@@@22@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@*/
DELETE FROM edureka.customer_segment;
INSERT INTO edureka.customer_segment
(
    Customer_Id,
    Customer_FName,
    Customer_LName,
    Customer_Type,
    Customer_Segment_Value,
    Segment_Start_Dt,
    Segment_End_Dt
)
SELECT
    cust.Customer_Id,
    cust.Customer_FName,
    cust.Customer_LName,
    case
        when cfa.First_Account_Dt between (CURRENT_DATE- INTERVAL '3' MONTH) and
(CURRENT_DATE) then 'N'
        else 'E'
    end as Customer_Type,
    CASE
        WHEN ca.Savings_Balance > 500000 OR ca.Credit_Balance > 300000 OR
ca.Mortgage_Balance > 5000000 THEN 'High-Networth'
        WHEN ca.Savings_Balance > 200000 OR ca.Credit_Balance > 100000 OR
ca.Mortgage_Balance > 1000000 THEN 'Medium-Networth'
        WHEN ca.Savings_Balance < 200000 OR ca.Credit_Balance < 100000 OR
ca.Mortgage_Balance < 1000000 THEN 'Low-Networth'
        ELSE 'Others'
    END as Customer_Segment_Value,
    CURRENT_DATE as Segment_Start_Dt,
    cast('9999-12-31' as date) as Segment_End_Dt

FROM edureka.Customer cust

JOIN Customer_Accounts  ca
ON cust.Customer_Id = ca.Customer_Id

JOIN Customer_First_Account cfa
ON cust.Customer_Id = cfa.Customer_Id
;

.LOGOFF

.QUIT 0;

EOF
```

Output

```
SELECT * FROM edureka.customer_segment ORDER BY Customer_Id;
```

Customer_Id	Customer_FName	Customer_LName	Customer_Type	Customer_Segment_Value	Segment_Start_Dt	Segment_End_Dt
100123	Deepak	Sharma	E	High-Networth	2017-05-12	9999-12-31
103256	Ramesh	Kumar	E	High-Networth	2017-05-12	9999-12-31
109345	Ram	Kumar	E	Low-Networth	2017-05-12	9999-12-31
119834	Anand	Sharma	E	Low-Networth	2017-05-12	9999-12-31
125783	Dilip	Mehta	E	Medium-Networth	2017-05-12	9999-12-31
146784	Mohan	Kanna	E	Low-Networth	2017-05-12	9999-12-31
157345	Siva	Kannan	E	Low-Networth	2017-05-12	9999-12-31
191289	Anand	Kannan	E	Low-Networth	2017-05-12	9999-12-31
210923	Umesh	Yadav	E	Medium-Networth	2017-05-12	9999-12-31

Appendix

Customer.txt   Accounts.txt   customerload.fl   accountsload.fl   Insert_stmt_for_Account_type.txt

CustomerAccountsLoad.bteq   CustomerSegment.bteq

Thank you!