**SQL**

**TASK-1:**

TO CALCULATE THE TOTAL COST OF EACH PRODUCT

* CREATE TABLE STRUCTURE IN SNOWFLAKE BY CREATING YOUR OWN WAREHOUSE.
* INSERT SOME 10 ROWS USING INSERT COMMAND
* TRY DIFFERENT VALUES FOR ALL THE COLUMNS AND THEN CHECK USING SELECT \*
* CREATE A TABLE CONSISTING OF THE PRODUCT NAME, NUMBER OF ITEMS BOUGHT AND PRICE OF A SINGLE ITEM.
* PRODUCTS MAY APPEAR MULTIPLE TIMES WITH DIFFERENT PRICES.
* WRITE AN SQL QUERY THAT RETURNS THE TOTAL AMOUNT OF MONEY SPENT ON EACH PRODUCT IN DESCENDING ALPHABETICAL ORDER BY PRODUCT

CREATE WAREHOUSE M\_WAREHOUSE;

CREATE DATABASE M\_DATABASE;

USE DATABASE M\_DATABASE;

--TO CALCULATE THE TOTAL COST OF EACH PRODUCT

/\*CREATE A TABLE CONSISTING OF THE PRODUCT NAME, NUMBER OF ITEMS BOUGHT AND PRICE OF A SINGLE ITEM. PRODUCTS MAY APPEAR MULTIPLE TIMES WITH DIFFERENT PRICES. \*/

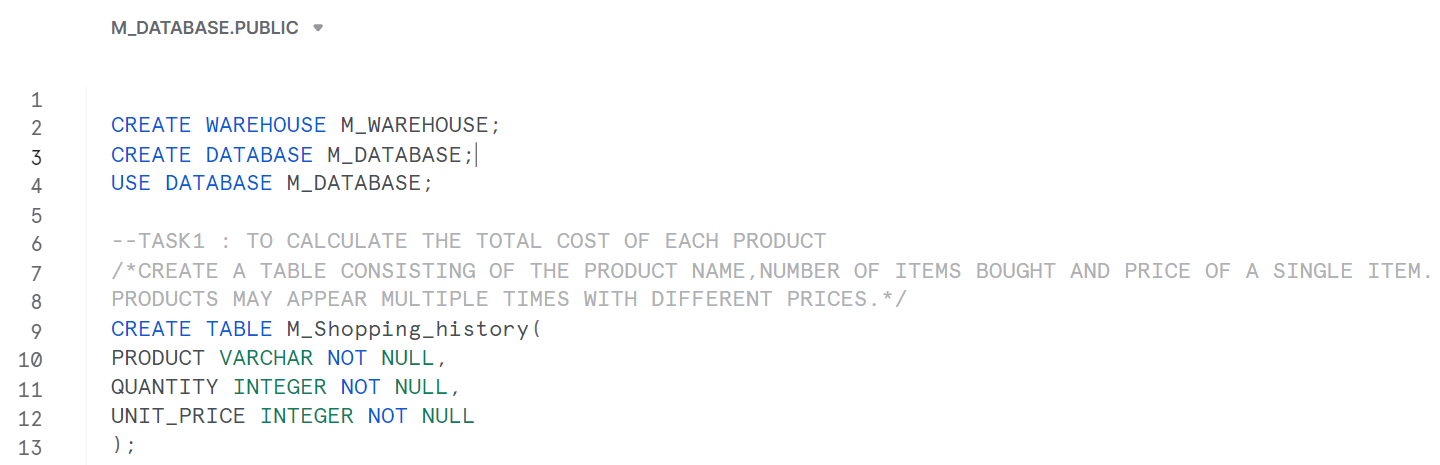
CREATE TABLE M\_Shopping\_history(

PRODUCT VARCHAR NOT NULL,

QUANTITY INTEGER NOT NULL,

UNIT\_PRICE INTEGER NOT NULL

);



INSERT INTO M\_Shopping\_history VALUES('Milk',3,10);

INSERT INTO M\_Shopping\_history VALUES('Bread',7,3);

INSERT INTO M\_Shopping\_history VALUES('Bread',5,2);

INSERT INTO M\_Shopping\_history VALUES('Milk',6,12);

INSERT INTO M\_Shopping\_history VALUES('Cheese',3,8);

INSERT INTO M\_Shopping\_history VALUES('Milk',2,8);

INSERT INTO M\_Shopping\_history VALUES('Bread',1,5);

INSERT INTO M\_Shopping\_history VALUES('Cheese',2,3);

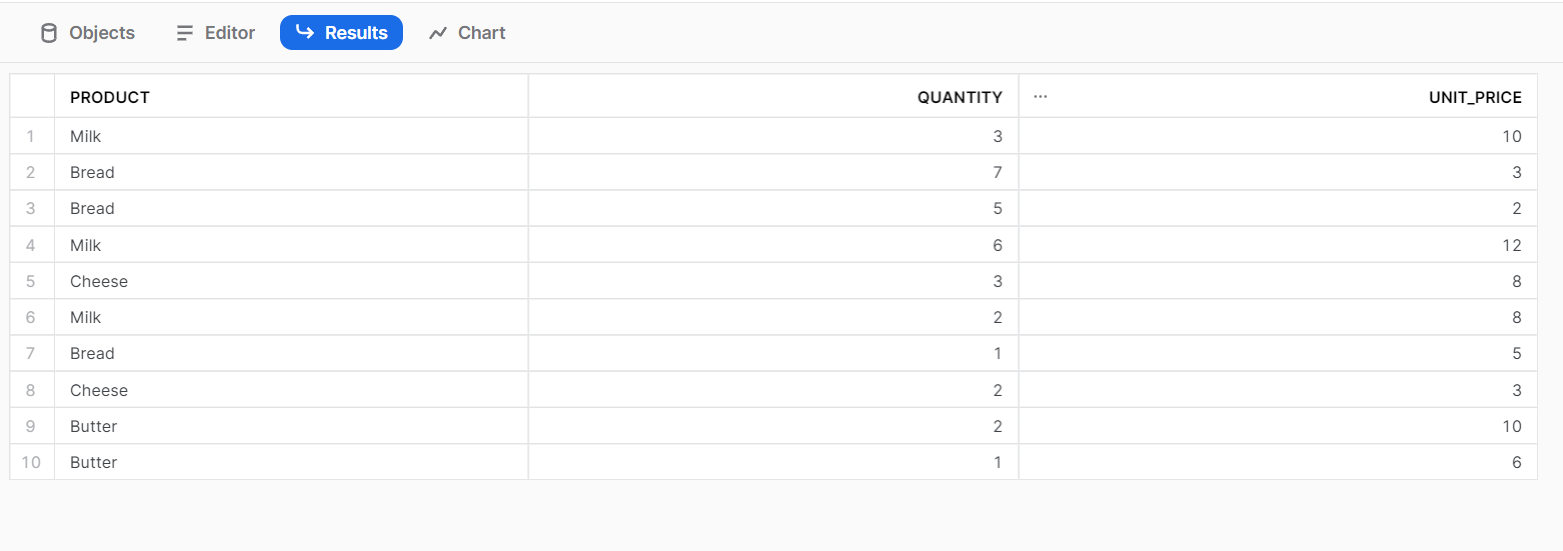
INSERT INTO M\_Shopping\_history VALUES('Butter',2,10);

INSERT INTO M\_Shopping\_history VALUES('Butter',1,6);



SELECT \* FROM M\_Shopping\_history;

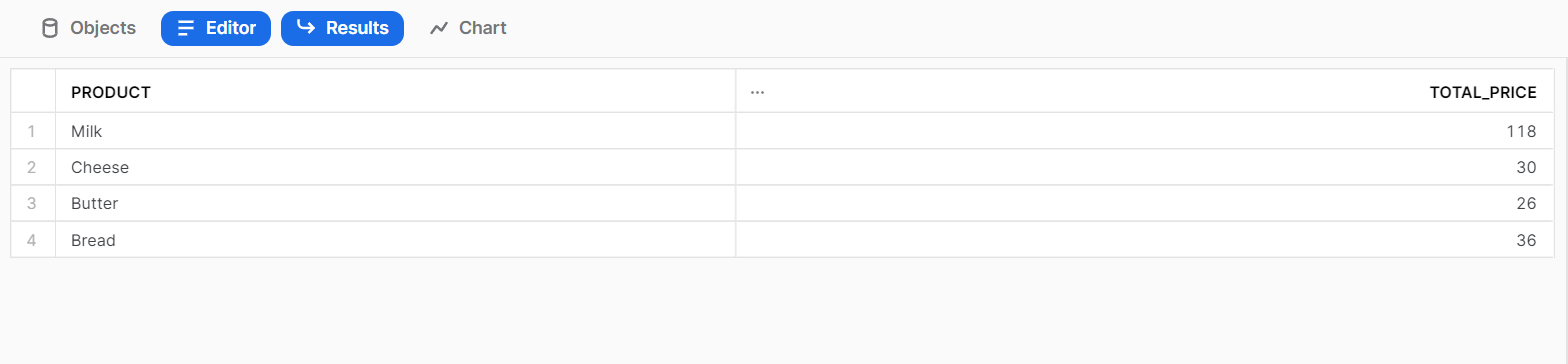




--RETURNS THE TOTAL AMOUNT OF MONEY SPENT ON EACH PRODUCT IN DESCENDING ALPHABETICAL ORDER BY PRODUCT

SELECT PRODUCT, SUM(QUANTITY\*UNIT\_PRICE) AS Total\_price FROM M\_Shopping\_history group by(PRODUCT) ORDER BY PRODUCT DESC;





**TASK-2:**

TO FIND NAME OF ALL THE CLIENTS WHO TALKED FOR ATLEAST 10 MINUTES IN TOTAL.

* CREATE A TABLE PHONES WHICH HAS THE INFORMATION ABOUT THE CLIENT’S NAME AND PHONE\_NUMBER.
* EVERY CLIENT HAS ONLY ONE NUMBER.
* CREATE A TABLE CALLS WHERE ID HAS THE INFORMATION ABOUT SINGLE CALL, PHONE NUMBER OF THE CALLER, PHONE NUMBER OF THE CALLEE AND THE DURATION OF CALL.
* WRITE AN SQL QUERY THAT RETURNS THE NAMES OF ALL THE CLIENTS WHO TALKED FOR ATLEAST 10 MINUTES IN TOTAL

--CREATE A TABLE PHONES WHICH HAS THE INFORMATION ABOUT THE CLIENT’S NAME AND PHONE\_NUMBER.

--EVERY CLIENT HAS ONLY ONE NUMBER

CREATE TABLE Phones(

NAME VARCHAR(20) NOT NULL UNIQUE,

PHONE\_NUMBER INTEGER NOT NULL UNIQUE

);

--CREATE A TABLE CALLS WHERE ID HAS THE INFORMATION ABOUT SINGLE CALL,PHONE NUMBER OF THE CALLER ,PHONE NUMBER OF THE CALLEE AND THE DURATION OF CALL.

CREATE TABLE Calls(

ID INTEGER NOT NULL,

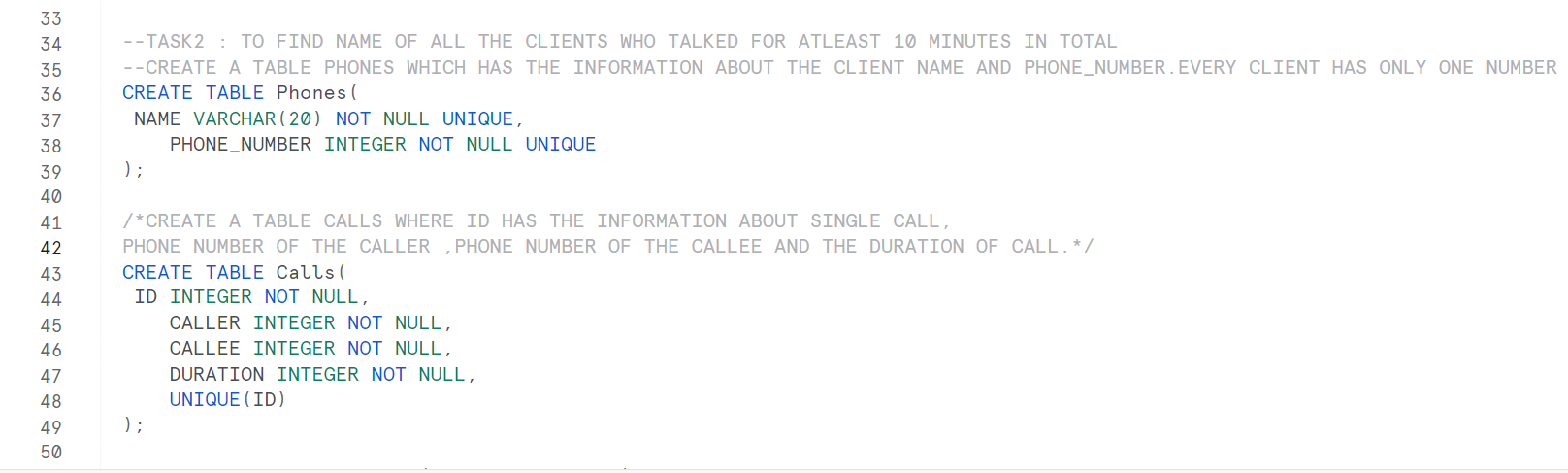
CALLER INTEGER NOT NULL,

CALLEE INTEGER NOT NULL,

DURATION INTEGER NOT NULL,

UNIQUE(ID)

);



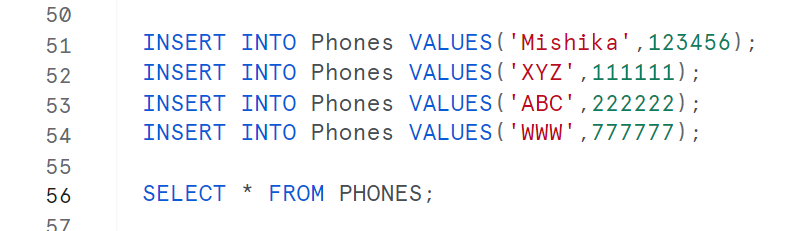
INSERT INTO Phones VALUES('Mishika',123456);

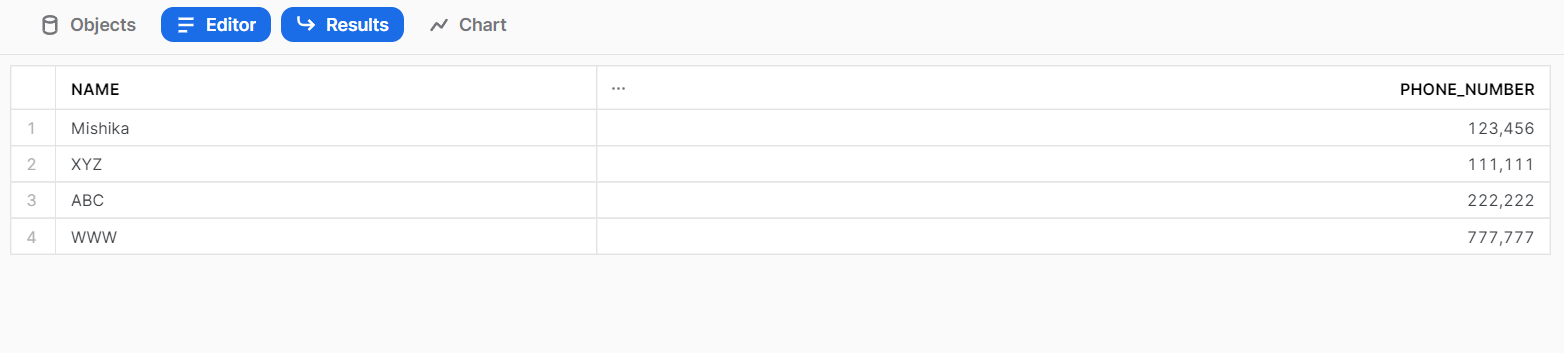
INSERT INTO Phones VALUES('XYZ',111111);

INSERT INTO Phones VALUES('ABC',222222);

INSERT INTO Phones VALUES('WWW',777777);

SELECT \* FROM Phones;





INSERT INTO Calls VALUES(1,123456,111111,8);

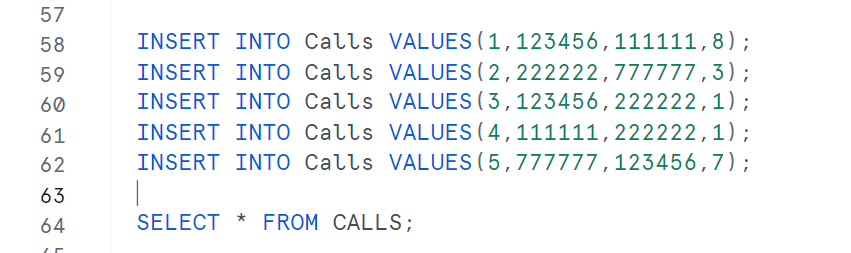
INSERT INTO Calls VALUES(2,222222,777777,3);

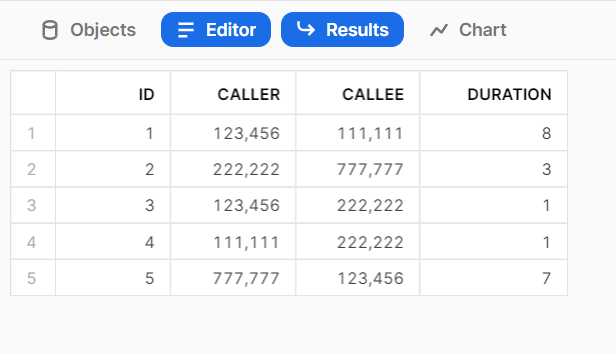
INSERT INTO Calls VALUES(3,123456,222222,1);

INSERT INTO Calls VALUES(4,111111,222222,1);

INSERT INTO Calls VALUES(5,777777,123456,7);

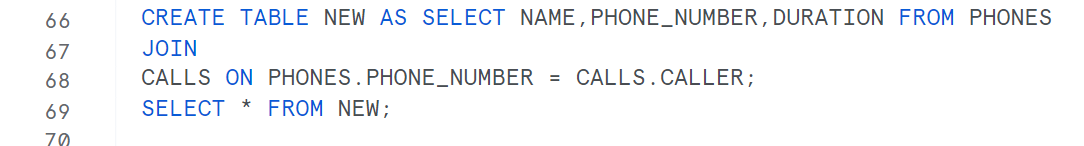
SELECT \* FROM CALLS;

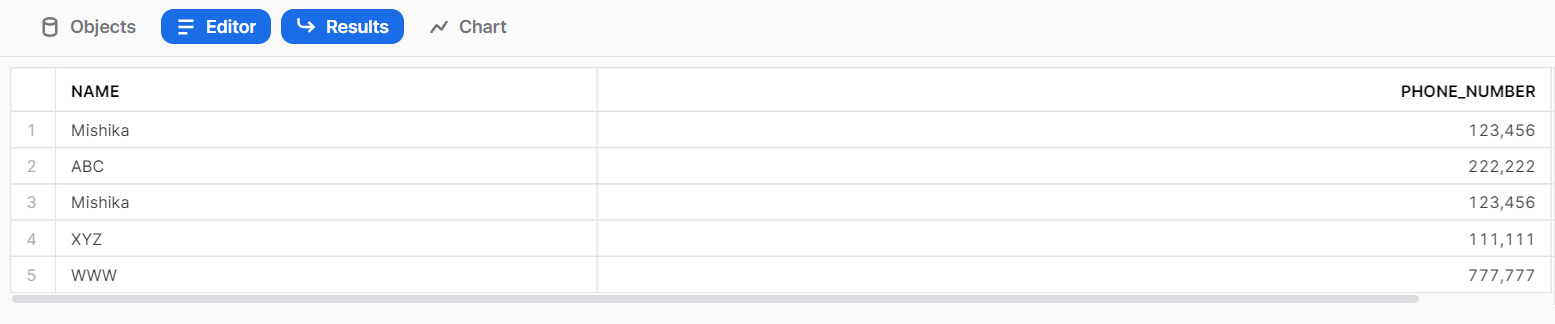




CREATE TABLE NEW AS SELECT NAME, PHONE\_NUMBER, DURATION FROM PHONES JOIN CALLS ON PHONES.PHONE\_NUMBER = CALLS.CALLER;

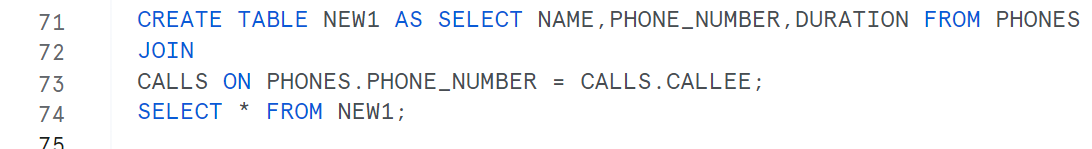
SELECT \* FROM NEW;

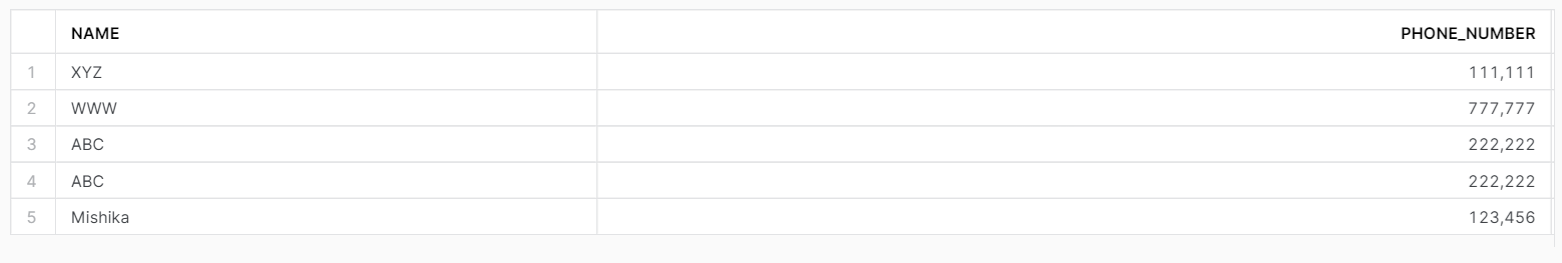




CREATE TABLE NEW1 AS SELECT NAME, PHONE\_NUMBER, DURATION FROM PHONES JOIN CALLS ON PHONES.PHONE\_NUMBER = CALLS.CALLEE;

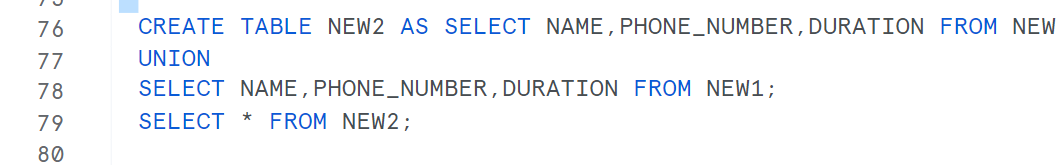
SELECT \* FROM NEW1;





CREATE TABLE NEW2 AS SELECT NAME, PHONE\_NUMBER, DURATION FROM NEW UNION SELECT NAME, PHONE\_NUMBER, DURATION FROM NEW1;

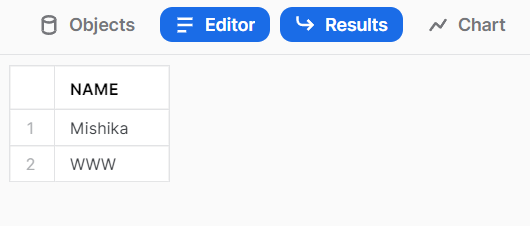
SELECT \* FROM NEW2;





--IT RETURNS THE NAMES OF ALL THE CLIENTS WHO TALKED FOR AT LEAST 10 MINUTES IN TOTAL

SELECT NAME FROM NEW2 GROUP BY NAME HAVING SUM(DURATION)>=10;



**TASK-3** :

SQL QUERY THAT RETURNS A TABLE CONTAINING ONLY BALANCE COLUMN WHICH HAS THE TOTAL BALANCE OF YOUR ACCOUNT AT THE END OF THE YEAR INCLUDING ALL THE CREDIT CARD PAYMENTS, INCOMING TRANSFER AND THE FEE FOR HOLDING THE CARD

* THE CREDIT CARD COST CHARGED EACH MONTH IS 5.IT IS NOT CHARGED IF ATLEAST 3 CREDIT CARD PAYMENTS FOR TOTAL COST OF ATLEAST 100 IS DONE WIHTIN THAT MONTH.
* CREATE TABLE TRANSACTIONS WHICH HAS THE INFORMATION ABOUT THE SINGLE TRANSACTION CONTAINING THE AMOUNT OF MONEY AND THE DATE OF THE TRANSACTION
* IF THE AMOUNT IS NEGATIVE, IT IS THE CREDIT CARD PAYMENT
* THERE IS NO TRANSACTION WITH AN AMOUNT OF 0
* IT RETURNS BALANCE COLUMN WHICH CONTAINS THE TOTAL BALANCE OF THE ACCOUNT AT THE END OF THE YEAR.

--CREATE TABLE TRANSACTIONS WHICH HAS THE INFORMATION ABOUT THE SINGLE TRANSACTION CONTAINING THE AMOUNT OF MONEY AND THE DATE OF THE TRANSACTION

CREATE TABLE TRANSACTIONS(

AMOUNT INTEGER NOT NULL,

DATE DATE NOT NULL

);

--IF THE AMOUNT IS NEGATIVE, IT IS THE CREDIT CARD PAYMENT OTHERWISE AN INCOMING TRANSFER.

--THERE IS NO TRANSACTION WITH AN AMOUNT OF 0

INSERT INTO TRANSACTIONS VALUES(1000,'2020-01-06');

INSERT INTO TRANSACTIONS VALUES(-10,'2020-01-14');

INSERT INTO TRANSACTIONS VALUES(-75,'2020-01-20');

INSERT INTO TRANSACTIONS VALUES(-5,'2020-01-25');

INSERT INTO TRANSACTIONS VALUES(-4,'2020-01-29');

INSERT INTO TRANSACTIONS VALUES(2000,'2020-03-10');

INSERT INTO TRANSACTIONS VALUES(-75,'2020-03-12');

INSERT INTO TRANSACTIONS VALUES(-20,'2020-03-15');

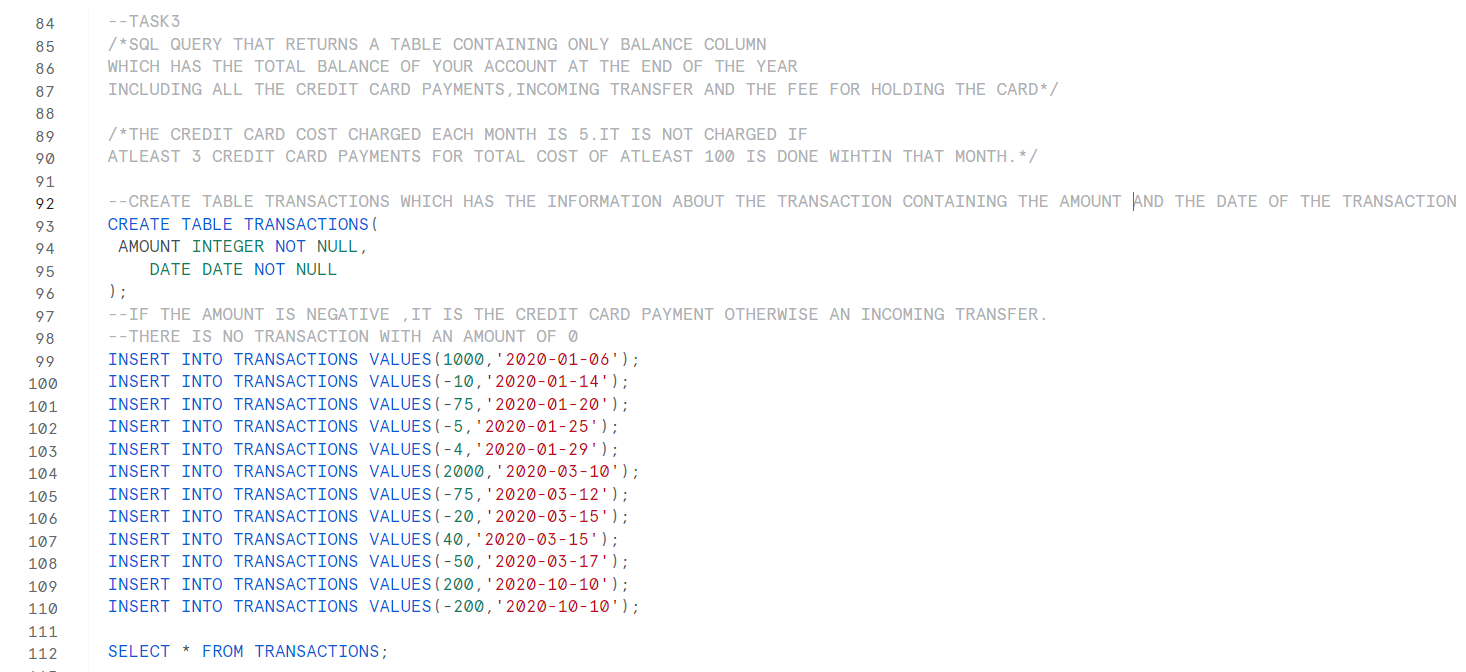
INSERT INTO TRANSACTIONS VALUES(40,'2020-03-15');

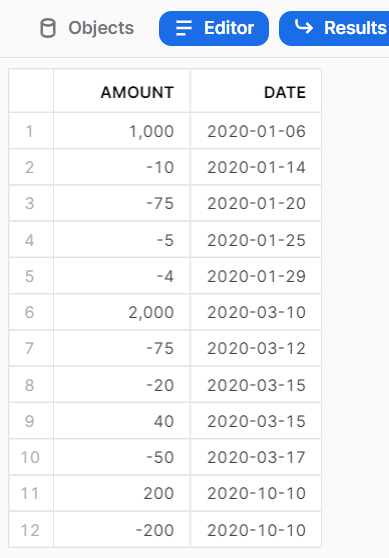
INSERT INTO TRANSACTIONS VALUES(-50,'2020-03-17');

INSERT INTO TRANSACTIONS VALUES(200,'2020-10-10');

INSERT INTO TRANSACTIONS VALUES(-200,'2020-10-10');

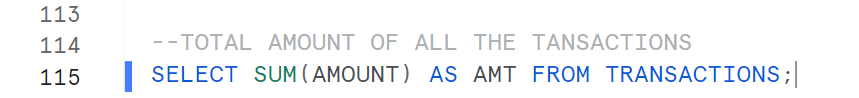
SELECT \* FROM TRANSACTIONS;

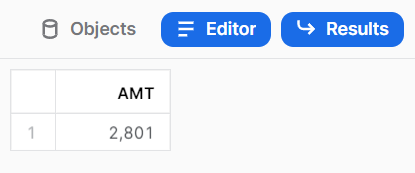




--TOTAL AMOUNT OF ALL THE TANSACTIONS

SELECT SUM(AMOUNT) AS AMT FROM TRANSACTIONS;



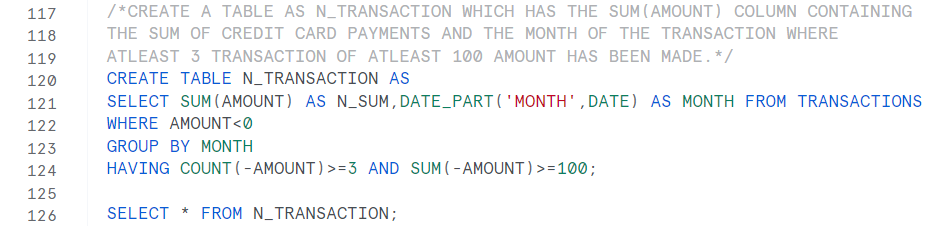


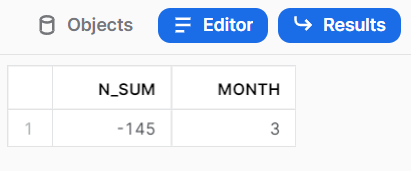
--CREATE A TABLE AS N\_TRANSACTION WHICH HAS THE SUM(AMOUNT) COLUMN CONTAINING THE SUM OF CREDIT CARD PAYMENTS AND THE MONTH OF THE TRANSACTION WHERE ATLEAST 3 TRANSACTION OF ATLEAST 100 AMOUNT HAS BEEN MADE.

CREATE TABLE N\_TRANSACTION AS

SELECT SUM(AMOUNT) AS N\_SUM, DATE\_PART('MONTH',DATE) AS MONTH FROM TRANSACTIONS WHERE AMOUNT<0 GROUP BY MONTH HAVING COUNT(-AMOUNT)>=3 AND SUM(-AMOUNT)>=100;

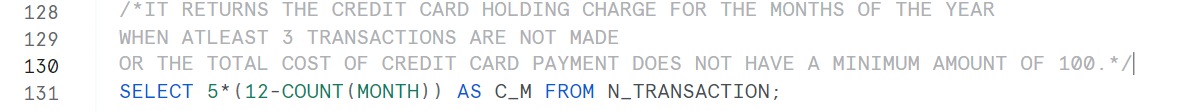
SELECT \* FROM N\_TRANSACTION;

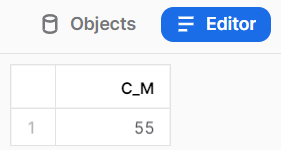




--IT RETURNS THE CREDIT CARD HOLDING CHARGE FOR THE MONTHS OF THE YEAR WHEN ATLEAST 3 TRANSACTIONS ARE NOT MADE OR THE TOTAL COST OF CREDIT CARD PAYMENT DOES NOT HAVE A MINIMUM AMOUNT OF 100.

SELECT 5\*(12-COUNT(MONTH)) AS C\_M FROM N\_TRANSACTION;





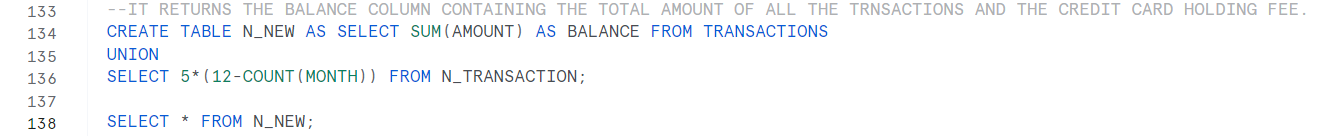
--IT RETURNS THE BALANCE COLUMN CONTAINING THE TOTAL AMOUNT OF ALL THE TRNSACTIONS AND THE CREDIT CARD HOLDING FEE.

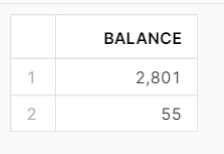
CREATE TABLE N\_NEW AS SELECT SUM(AMOUNT) AS BALANCE FROM TRANSACTIONS

UNION

SELECT 5\*(12-COUNT(MONTH)) FROM N\_TRANSACTION;

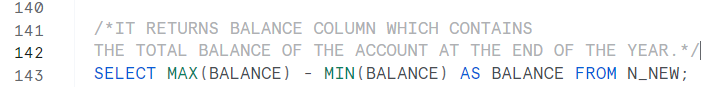
SELECT \* FROM N\_NEW;

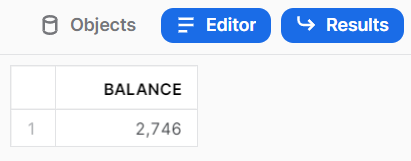




--IT RETURNS BALANCE COLUMN WHICH CONTAINS THE TOTAL BALANCE OF THE ACCOUNT AT THE END OF THE YEAR.

SELECT MAX(BALANCE) - MIN(BALANCE) AS BALANCE FROM N\_NEW;





\*\*\*\*\*\*