**SNOWFLAKE TASKS-**

**Task 1- You are given a table shopping\_history with the following structure:**

**create table shopping history (**

**product varchar not null,**

**quantity integer not null,**

**unit\_price integer not null**

**);**

**It represents a list of shopping transactions, where each transaction consists of the product name, the number of items bought and the price of a single item. Notice that some products may appear multiple times, sometimes with different prices. You are asked to calculate the total cost of each product.**

**Write an SQL query that, for each "product", returns the total amount of money spent on it. Rows should be ordered in descending alphabetical order by "product".**

**CODE-**

// CRREATING DATABASE-

CREATE DATABASE "SNOWFLAKE\_ASSIGNMENTS";

use SNOWFLAKE\_ASSIGNMENTS;

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* TASK NO.1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// CRREATING TABLE-

CREATE table shopping\_history(

PRODUCT varchar(30) not null,

QUANTITY integer not null,

UNIT\_PRICE integer not null

);

// LOADING DATA INTO SHOPPING\_HISTORY TABLE-

INSERT INTO shopping\_history values ('BUTTER' , 2 , 20),

('MILK' , 2 , 65),

('RICE' , 5 , 90),

('OLIVE OIL' , 3 , 70),

('PEANUT BUTTER' , 3 , 500),

('FLOUR' , 1 , 250),

('TOOTHPASTE' , 3 , 90),

('BREAD' , 10 , 50),

('DRAWING BOOK' , 5 , 320),

('PAINT BRUSHES' , 1 , 170);

// TOTAL AMOUNT OF MONEY SPENT ON PRODUCTS(BY DESCENDING ALPHABETICAL ORDER)-

select PRODUCT, (QUANTITY \* UNIT\_PRICE) as TOTAL\_PRICE from shopping\_history order by PRODUCT desc;

**Task 2 - A telecommunications company decided to find which of their clients talked for at least 10 minutes on the phone in total and offer them a new contract.**

**You are given two tables, phones and calls, with the following structure:**

**create table phones (**

**name varchar(20) not null unique,**

**phone\_number integer not null unique**

**);**

**create table calls**

**( id integer not null,**

**caller integer not null,**

**callee integer not null,**

**duration integer not null,**

**unique(id)**

**);**

**Each row of the table phones contains Information about a client: name (name) and phone number (phone\_number). Each client has only one phone number. Each row of the table calls contains Information about a single call: Id (id), phone number of the caller (caller), phone number of the callee (callee) and duration of the call in minutes (duration).**

**Write an SQL query that finds all clients who talked for at least 10 minutes in total. The table of results should contain one column: the name of the client (name). Rows should be sorted alphabetically,**

**CODE-**

**// \*\*\*\*\*\*\*\*\*\*\* TASK NO. 2 \*\*\*\*\*\*\*\*\*\*\*\***

// CRREATING TABLE-

CREATE TABLE PHONES(

NAME varchar(50) not null unique,

PHONE\_NUMBER integer not null unique

);

// LOADING DATA INTO PHONES TABLE-

INSERT INTO PHONES VALUES ('NISHA' , 123456),

('SHIVI' , 111111),

('VISHAL' , 222222),

('ANSHU' , 333333),

('ANSHIKA' , 444444),

('HARSHITA' , 555555),

('SWAPNIL' , 666666),

('DIVYANSH' , 777777),

('RATNESH' , 888888),

('SRAJAN' , 999999)

;

// CRREATING TABLE-

CREATE TABLE if not exists CALLS(

ID integer not null unique,

CALLER integer not null,

CALLEE integer not null,

DURATION integer not null

);

// LOADING DATA INTO CALLS TABLE-

INSERT INTO CALLS VALUES (2 , 123456 , 222222 , 8),

(37 , 123456 , 333333 , 6),

(208 , 444444 , 777777 , 5),

(72 , 111111 , 222222 , 9),

(9 , 123456 , 333333 , 2),

(106 , 123456 , 666666 , 8),

(65 , 111111 , 999999 , 6),

(50 , 888888 , 222222 , 7),

(18 , 666666 , 222222 , 4),

(150 , 555555 , 888888 , 9);

// NAMES OF CLIENTS WHO TALKED FOR MORE THAN 10 MINUTES-

SELECT PHONES.NAME FROM PHONES JOIN CALLS ON PHONES.PHONE\_NUMBER = CALLS.CALLER

UNION

SELECT PHONES.NAME FROM PHONES JOIN CALLS ON PHONES.PHONE\_NUMBER = CALLS.CALLEE

GROUP BY NAME

HAVING SUM(DURATION) >= 10;

**TASK 3- You are given a history of your bank account transactions for the year 2020. Each transaction was either a credit card payment or an incoming transfer.**

**There is a fee for holding a credit card which you have to pay every month. The cost you are charged each month is 5. However, you are not charged for a given month if you made at least three credit card payments for a total cost of at least 100 within that month. Note that this fee is not included in the supplied history of transactions.**

**At the beginning of the year, the balance of your account was 0. Your**

**task is to compute the balance at the end of the year.**

**You are given a table transactions with the following structure:**

**create table transactions (**

**amount integer not null,**

**date date not null**

**);**

**Each row of the table contains information about a single transaction: the amount of money (amount) and the date when the transaction happened (date). If the amount value is negative, it is a credit card payment. Otherwise, it is an incoming transfer. There are no transactions with an amount of 0.**

**Write an SQL query that returns a table containing one column, balance. The table should contain one row with the total balance of your account at the end of the year, including the fee for holding a credit**

**// \*\*\*\*\*\*\*\*\*\*\* TASK NO. 3 \*\*\*\*\*\*\*\*\*\*\*\***

// CRREATING TABLE TRANSACTIONS-

CREATE TABLE if not exists TRANSACTIONS(

AMOUNT INTEGER NOT NULL,

DATE DATE NOT NULL

);

// LOADING DATA INTO TABLE-

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

(-10, '2020-01-14'),

(-75, '2020-01-20'),

(-5, '2020-01-25'),

(-4, '2020-01-29'),

(2000, '2020-03-10'),

(-75, '2020-03-12'),

(-20, '2020-03-15'),

(40, '2020-03-15'),

(-50, '2020-03-17'),

(200, '2020-10-10'),

(200, '2020-10-10');

// BALANCE FOR MONTHLY CREADITED TRANSACTIONS-

WITH monthly\_credited\_transactions

AS (SELECT Date\_part('month', date) AS cred\_month,

Sum(CASE

WHEN amount < 0 THEN Abs(amount)

ELSE 0

END) AS credited\_amount,

Sum(CASE

WHEN amount < 0 THEN 1

ELSE 0

END) AS credited\_cnt

FROM transactions

GROUP BY 1),

credit\_fee

AS (SELECT ( 12 - Count(1) ) \* 5 AS fee,

1 AS id

FROM monthly\_credited\_transactions

WHERE credited\_amount >= 100

AND credited\_cnt >= 3),

trans

AS (SELECT Sum(amount) AS amount,

1 AS id

FROM transactions)

SELECT amount - fee AS balance

FROM trans a

LEFT JOIN credit\_fee b

ON a.id = b.id