Graphical user interface, text, application

Description automatically generated

CREATE DATABASE PROJECT\_DATABASE;

USE PROJECT\_DATABASE;

TASK 1

CREATE TABLE SHOPPING\_HISTORY (

PRODUCT VARCHAR NOT NULL,

QUANTITY INTEGER NOT NULL,

UNIT\_PRICE INTEGER NOT NULL

);

INSERT INTO SHOPPING\_HISTORY VALUES ('MILK', 3, 10);

INSERT INTO SHOPPING\_HISTORY VALUES ('BREAD', 7, 3);

INSERT INTO SHOPPING\_HISTORY VALUES ('BREAD', 5, 2);

SELECT \* FROM SHOPPING\_HISTORY.

**TASK1: QUERY: -**

SELECT PRODUCT, SUM(QUANTITY \* UNIT\_PRICE) AS TOTAL\_PRICE

FROM SHOPPING\_HISTORY

GROUP BY PRODUCT

ORDER BY PRODUCT DESC;

**--1. GIVEN:-**

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,

COLLEE INTEGER NOT NULL,

DURATION INTEGER NOT NULL,

UNIQUE(ID) );

INSERT INTO PHONES VALUES ('JACK', 1234),('LENA', 3333), ('MARK', 9999),('ANNA', 7582);

SELECT \* FROM PHONES.

INSERT INTO CALLS VALUES

(25, 1234, 7582, 8),

(7, 9999, 7582, 1),

(18, 9999, 3333, 4),

(2, 7582, 3333, 3),

(3, 3333, 1234, 1),

(21, 3333, 1234, 1);

SELECT \* FROM CALLS;

**TASK2: STATEMENT 1:**

Write a 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.

**QUERY 1:-**

SELECT NAME FROM

(SELECT DISTINCT P.NAME, SUM (C. DURATION) AS TOTAL\_DURATION

FROM PHONES P

JOIN CALLS C

WHERE P. PHONE\_NUMBER IN (C. CALLER, C. COLLEE)

GROUP BY 1 HAVING TOTAL\_DURATION >= 10

ORDER BY 1);

**2.GIVEN**

CREATE OR REPLACE TABLE PHONE (

NAME STRING (50),

PHONE\_NUMBER INTEGER NOT NULL,

PRIMARY KEY (PHONE\_NUMBER)

);

CREATE OR REPLACE TABLE CALL (

ID INTEGER NOT NULL,

CALLER INTEGER FOREIGN KEY REFERENCES PHONE(PHONE\_NUMBER),

COLLEE INTEGER FOREIGN KEY REFERENCES PHONE(PHONE\_NUMBER),

DURATION INTEGER NOT NULL);

INSERT INTO PHONE VALUES ('JOHN', 6356),

('ADDITION', 4315),

('KATE', 8003),

('GINNY', 9831);

SELECT \* FROM PHONE ;

INSERT INTO CALL VALUES (65, 8003, 9831, 7),

(100, 9831, 8003, 3),

(145, 4315, 9831, 18);

SELECT \* FROM CALL ;

**TASK2: STATEMENT 2:**

**QUERY 2:-**

SELECT NAME FROM

(SELECT DISTINCT P.NAME, SUM(C.DURATION) AS TOTAL\_DURATION

FROM PHONE P

JOIN

CALL C

WHERE P.PHONE\_NUMBER IN (C.CALLER, C.COLLEE)

GROUP BY 1 HAVING TOTAL\_DURATION >= 10

ORDER BY 1);

**TASK 3**

**TASK 3.1**

create table transactions (

amount integer not null,

date DATE not null

);

insert into transactions(amount,date)

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');

select \* from transactions;

WITH CREDITS AS

(SELECT M,SUM(AMOUNT)AS TOTAL FROM (SELECT AMOUNT, YEAR(DATE) AS Y, MONTH(DATE) AS M FROM TRANSACTIONS WHERE AMOUNT<0 ORDER BY Y,M)

GROUP BY M,Y

HAVING TOTAL<-100

ORDER BY M,Y)

SELECT SUM(AMOUNT)-((12-(SELECT COUNT(\*) AS NON\_CHARGES FROM CREDITS))\*5) AS BALANCE FROM TRANSACTIONS;

/\*

STEP 1: TASK TO FIND OUT CREDITS

\*/

WITH CREDITS AS (SELECT AMOUNT, YEAR(DATE) AS Y, MONTH(DATE) AS M FROM TRANSACTIONS WHERE AMOUNT<0 ORDER BY Y,M);

/\*

STEP 2:TASK TO FIND OUT SUM OF CREDITS MONTHLY

\*/

SELECT M ,SUM(AMOUNT)AS TOTAL FROM transactions

GROUP BY M,Y;

/\*

ALTERNATE METHOD WHEN CRITERIA IS GREATER THAN 3 TRANSACTION AND SUM OF CREDIT IS GREATER THAN 100

\*/

/\* STEP 1: \*/

WITH CRED\_TABLE AS (SELECT AMOUNT, YEAR(DATE) AS Y, MONTH(DATE) AS M FROM TRANSACTIONS WHERE AMOUNT<0 ORDER BY Y,M)

SELECT COUNT(AMOUNT) AS TOTAL\_TRANS, SUM(AMOUNT) AS TOTAL\_CREDIT FROM CRED\_TABLE GROUP BY Y,M;

/\* STEP 2: \*/

WITH M\_CRED AS (WITH CRED\_TABLE AS (SELECT AMOUNT, YEAR(DATE) AS Y, MONTH(DATE) AS M FROM TRANSACTIONS WHERE AMOUNT<0 ORDER BY Y,M)

SELECT COUNT(AMOUNT) AS TOTAL\_TRANS, SUM(AMOUNT) AS TOTAL\_CREDIT FROM CRED\_TABLE GROUP BY Y,M HAVING TOTAL\_TRANS>=3 AND TOTAL\_CREDIT<=-100)

SELECT SUM(AMOUNT)-(5\*(12-(SELECT COUNT(TOTAL\_TRANS) FROM M\_CRED))) AS BALANCE FROM TRANSACTIONS;

/\* ONE MORE \*/

WITH DESIRED\_TABLE AS (SELECT SUM(AMOUNT) AS TOTAL\_TRANS,COUNT(AMOUNT) AS NO\_OF\_TRANS

FROM TRANSACTIONS

WHERE AMOUNT<0

GROUP BY YEAR(DATE),MONTH(DATE)

HAVING TOTAL\_TRANS<=-100 AND NO\_OF\_TRANS>=3)

SELECT SUM(AMOUNT)-(5\*(12-(SELECT COUNT(\*) FROM DESIRED\_TABLE))) AS BALANCE FROM TRANSACTIONS;

**TASK 3.2**

TRUNCATE TRANSACTIONS;

INSERT INTO TRANSACTIONS VALUES (1,'2020-06-29'),

(35,'2020-02-20'),

(-50,'2020-02-03'),

(-1,'2020-02-26'),

(-200,'2020-08-01'),

(-44,'2020-02-07'),

(-5,'2020-02-25'),

(1,'2020-06-29'),

(1,'2020-06-29'),

(-100,'2020-12-29'),

(-100,'2020-12-30'),

(-100,'2020-12-31');

WITH DESIRED\_TABLE AS (SELECT SUM(AMOUNT) AS TOTAL\_TRANS,COUNT(AMOUNT) AS NO\_OF\_TRANS

FROM TRANSACTIONS

WHERE AMOUNT<0

GROUP BY YEAR(DATE),MONTH(DATE)

HAVING TOTAL\_TRANS<=-100 AND NO\_OF\_TRANS>=3)

SELECT SUM(AMOUNT)-(5\*(12-(SELECT COUNT(\*) FROM DESIRED\_TABLE))) AS BALANCE FROM TRANSACTIONS;

**TASK 3.3**

TRUNCATE TRANSACTIONS;

INSERT INTO TRANSACTIONS VALUES (6000,'2020-04-03'),

(5000,'2020-04-02'),

(4000,'2020-04-01'),

(3000,'2020-03-01'),

(2000,'2020-02-01'),

(1000,'2020-01-01');

WITH DESIRED\_TABLE AS (SELECT SUM(AMOUNT) AS TOTAL\_TRANS,COUNT(AMOUNT) AS NO\_OF\_TRANS

FROM TRANSACTIONS

WHERE AMOUNT<0

GROUP BY YEAR(DATE),MONTH(DATE)

HAVING TOTAL\_TRANS<=-100 AND NO\_OF\_TRANS>=3)

SELECT SUM(AMOUNT)-(5\*(12-(SELECT COUNT(\*) FROM DESIRED\_TABLE))) AS BALANCE FROM TRANSACTIONS;