**ЛАБОРАТОРНАЯ РАБОТА №3.5**  
Язык SQL

**Выполнил:**

студент 3 курса 13 группы кафедры ТП.

Петров Андрей Александрович

# ЗАДАЧИ РАБОТЫ:

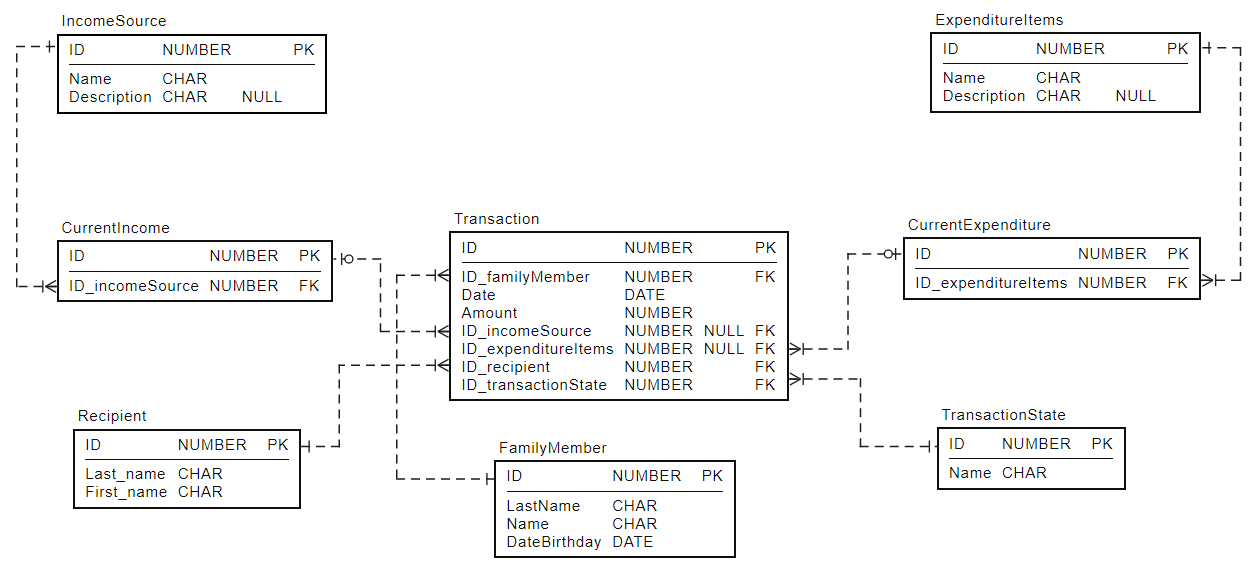
При выполнении данной лабораторной работы требуется использовать базу данных, созданную студентом в лабораторной работе №2.

1. Составьте на выборку данных с использованием рефлексивного соединения для таблицы из задания 5 лабораторной работы №2

Составьте запросы на выборку данных с использованием следующих операторов, конструкций функций языка SQL:

1. простого оператора CASE;
2. поискового оператора CASE;
3. оператора WITH;
4. встроенного представления;
5. некоррелированного запроса;
6. коррелированного запроса;
7. функции NULLIF;
8. функции NVL2;
9. TOP-N анализа;
10. функции ROLLUP.
11. Составьте запрос на использование оператора MERGE языка манипулирования данными.

# ХОД РАБОТЫ:



1. Составим запрос на выборку данных с использованием рефлексивного соединения.

*Получение сведений обо всех расходах семьи.*

SELECT T1.*"ID\_familyMember"*, T1.*"Date"*, T2.*"ID\_recipient"*FROM "Transaction" T1  
 JOIN "Transaction" T2 ON T1.*"ID\_familyMember"* = T2.*"ID\_familyMember"* AND T1.*"ID\_expenditureItems"* IS NOT NULL;

2. Составим запрос на выборку данных с использованием простого оператора CASE.

*Вывод всех транзакций с указанием типа транзакции.*

SELECT *"ID\_familyMember"*, *“Date”*  
 CASE  
 WHEN *"ID\_incomeSource"* IS NOT NULL  
 THEN *"Income"* ELSE *"Expenditure"* END AS "Type transaction",  
 *"Amount"*FROM "Transaction";

3. Составим запрос на выборку данных с использованием поискового оператора CASE.

*Вывод расходов с указанием суммы только в транзакциях после ‘2021-01-01’*

SELECT CASE  
 WHEN *"Date"* >= '2021-01-01' AND *"Date"* <= *SYSDATE*   
 THEN *"Amount"* END AS "Expenditure"  
FROM "Transaction" WHERE *"ID\_expenditureItems"* IS NOT NULL;

4. Составим запрос на выборку данных с использованием оператора WITH.

*Вывод суммы расходов на каждого члена семьи.*

WITH SUM AS (SELECT *"ID\_familyMember"*, *SUM*(*"Amount"*) SP  
 FROM "Transaction"  
 WHERE *"ID\_expenditureItems"* IS NOT NULL  
 GROUP BY *"ID\_familyMember"*)  
SELECT "ID\_familyMember", "LastName", "Name"  
FROM "FamilyMember"  
 JOIN SUM ON "FamilyMember".ID = SUM."ID\_familyMember";

5. Составим запрос на выборку данных с использованием встроенного представления.

*Вывод транзакций с указанием полных сведений о совершивших их.*

SELECT MEMBER."LastName", MEMBER."Name", "Transaction"."Date", "Transaction"."Amount"  
FROM "Transaction"  
 JOIN (SELECT *"ID"*, *"LastName"*, *"Name"* FROM "FamilyMember") MEMBER  
 ON "Transaction"."ID\_familyMember" = MEMBER."ID";

6. Составим запрос на выборку данных с использованием некоррелированного запроса.

*Вывести все транзакции расходов выше средних.*

SELECT *\** FROM Transaction  
WHERE *Amount* > (SELECT *AVG*(*Amount*) FROM "Transaction" WHERE *"ID\_expenditureItems"* IS NOT NULL);

7. Составим запрос на выборку данных с использованием коррелированного запроса.

*Вывести все транзакции, добавив сведения о свершившем ее человеке.*

SELECT *"Date"*, *"Amount"*,  
 (SELECT *"LastName"*, *"Name"* FROM "FamilyMember"  
 WHERE "FamilyMember".ID = "Transaction"."ID\_familyMember") AS "FamilyMember"  
FROM "Transaction";

8. Составим запрос на выборку данных с использованием функции NULLIF.

SELECT *"Name"*, *"Description"*, *NULLIF*(*"Description"*, *"Something income source"*)  
FROM "IncomeSource";

9. Составим запрос на выборку данных с использованием функции NVL2.

*Получение списка транзакций с указанием типа.*

SELECT *"ID\_familyMember"*, *"Date"*, *"Amount"*,

*NVL2*(*"ID\_expenditureItems"*, *"Expenditure"*, *"Income"*) AS "TransactionType"  
FROM "Transaction";

10. Составим запрос на выборку данных с использованием TOP-N анализа.

*Вывод 10 самых больших транзакций.*

SELECT *"ID\_familyMember"*, *"Amount"*FROM (SELECT *\** FROM "Transaction" ORDER BY *"Amount"* DESC)  
WHERE *ROWNUM* <= 10

11. Составим запрос на выборку данных с использованием функции ROLLUP.

*Вывод суммы всех расходов каждого члена семьи*

SELECT *"ID\_familyMember"*, *SUM*(“Amount”)  
FROM "Transaction"  
WHERE *"ID\_expenditureItems"* IS NOT NULL  
GROUP BY ROLLUP (*"ID\_familyMember"*);

12. Составим запрос на выборку данных с использованием оператора MERGE.

По условию объединения обновляем данные, а если таких данных нет, то добавляем их.

CREATE TABLE "TestFamilyMember"  
(  
 "ID" NUMBER NOT NULL,  
 "LastName" CHAR NOT NULL,  
 "Name" CHAR NOT NULL,  
 "DateBirthday" DATE NOT NULL,  
 PRIMARY KEY ("ID")  
);

MERGE "TestFamilyMember" AS T\_Base *--Целевая таблица*USING "FamilyMember" AS T\_Source *--Таблица источник*ON (T\_Base."ID" = T\_Source."ID") *--Условие объединения*WHEN MATCHED THEN *--Если истина (UPDATE)* UPDATE   
 SET "LastName" = T\_Source."LastName",   
 "Name" = T\_Source."Name",   
 "DateBirthday" = T\_Source."DateBirthday"  
WHEN NOT MATCHED THEN *--Если НЕ истина (INSERT)* INSERT ("ID", "LastName", "Name", "DateBirthday")   
 VALUES (T\_Source."ID", T\_Source."LastName", T\_Source."Name", T\_Source."DateBirthday")  
*--Посмотрим, что мы сделали*OUTPUT $action AS [Операция], Inserted."ID",  
 Inserted."LastName" AS LastNameNEW,  
 Inserted."Name" AS NameNEW,  
 Inserted."DateBirthday" AS DateBirthdayNEW,  
 Deleted."LastName" AS LastNameOLD,  
 Deleted."Name" AS NameOLD;  
 Deleted."DateBirthday" AS DateBirthdayOLD;