КАФЕДРА №

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

| должность, уч. степень, звание | подпись, дата | инициалы, фамилия |
|---|---|---|

# ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №7

Хранимые процедуры

по дисциплине: Проектирование баз данных

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. _____ _____ _____

подпись, дата | инициалы, фамилия

Санкт-Петербург
2023

**Цель работы** - По аналогии с примерами, приведенными в п. 1 реализовать запросы г) .. ж), указанные в варианте задания.

Один из запросов на максимум/минимум реализовать с помощью директивы all

Запрос на «все» (реляционное деление) реализовать с помощью 2 not exists

Запросы на разность реализовать в 3 вариантах: Not in,except(MySQL не поддерживает, поэтому только синтаксис), с использованием левого/правого соединения

**Задание (13 вариант):**

калькулятор бюджета семьи: категория дохода (продажа, зарплата), категория расхода (еда, счета за КУ, здоровье ...), статьи дохода и расхода, дата расхода/дохода. Категория- более общее понятие чем статья. Например категория- еда, а статьи в ней мясо, рыба, вкусное к чаю, а конкретный расход «печенье «курабье» к чаю 11.09»

а. расходы всех категорий, которые содержат часть слова «транспорт», но не заканчиваются на него

б. месяц, в котором были статьи дохода от работы и от дарения

в. пользователь без доходов

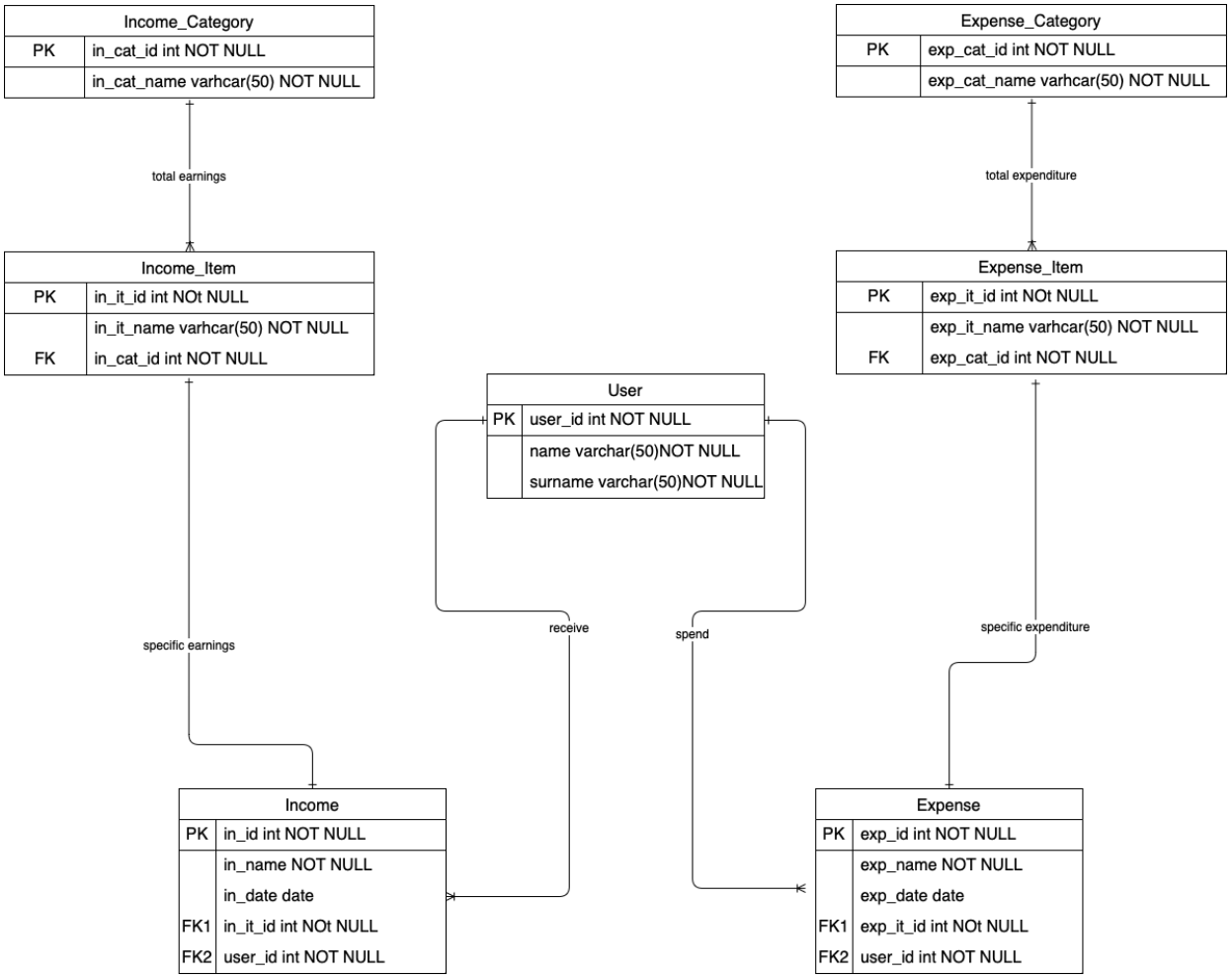
г. категория, по которой были расходы в этом году максимальные по сумме

д. категория, по которой не было расходов в январе, но были в мае

е. категория расхода, по которой траты были у всех членов семьи

ж. месяц, в котором были расходы максимального количества статей

# Физическая модель предметной области:



— вставку с пополнением справочников (получаем ссылку на внешний ключ по значению данных из родительской таблицы, если данных нет- добавляем в родительскую, затем вставляем в дочернюю);

При вставке данных необходимо получить ссылку на внешний ключ из родительской таблицы по значению данных. Если соответствующих данных в родительской таблице нет, они должны быть добавлены в родительскую таблицу, после чего осуществляется вставка в дочернюю таблицу.

Хранимая процедура InsertIncomeWithReference предназначена для добавления записей о доходе в базу данных. Она проверяет наличие и при необходимости добавляет нового пользователя, категорию и статью дохода в соответствующие таблицы, а затем вставляет информацию о доходе в таблицу Income.

```
DELIMITER $$


CREATE PROCEDURE InsertIncomeWithReference(
    IN _user_name VARCHAR(255),
    IN _surname VARCHAR(255),
    IN _in_cat_name VARCHAR(255),
    IN _in_it_name VARCHAR(255),
    IN _in_name VARCHAR(255),
    IN _in_date DATE,
    IN _in_money DECIMAL(10,2)
)
BEGIN
    DECLARE _user_id INT;
    DECLARE _in_cat_id INT;
    DECLARE _in_it_id INT;


    -- Проверяем существует ли пользователь, если нет - вставляем и получаем ID
```

```
    SELECT user_id INTO _user_id FROM User WHERE name = _user_name AND surname = _surname;

    IF _user_id IS NULL THEN

        INSERT INTO User(name, surname) VALUES (_user_name, _surname);

        SET _user_id = LAST_INSERT_ID();

    END IF;


    -- Проверяем существует ли категория дохода, если нет - вставляем и получаем ID

    SELECT in_cat_id INTO _in_cat_id FROM Income_Category WHERE in_cat_name = _in_cat_name;

    IF _in_cat_id IS NULL THEN

        INSERT INTO Income_Category(in_cat_name) VALUES (_in_cat_name);

        SET _in_cat_id = LAST_INSERT_ID();

    END IF;


    -- Проверяем существует ли статья дохода, если нет - вставляем и получаем ID

    SELECT in_it_id INTO _in_it_id FROM Income_Item WHERE in_it_name = _in_it_name AND in_cat_id = _in_cat_id;

    IF _in_it_id IS NULL THEN

        INSERT INTO Income_Item(in_it_name, in_cat_id) VALUES (_in_it_name, _in_cat_id);

        SET _in_it_id = LAST_INSERT_ID();

    END IF;


    -- Теперь вставляем запись в таблицу Income

    INSERT INTO Income(in_name, in_date, in_money, in_it_id, user_id) VALUES (_in_name, _in_date, _in_money, _in_it_id, _user_id);

END$$


DELIMITER ;
```

CALL InsertIncomeWithReference('Имя', 'Фамилия', 'КатегорияДохода', 'СтатьяДохода', 'Название дохода', '2023-01-01', 10000.00);

До:

| user_id | name | surname |
|---|---|---|
| 1 | Александр | Иванов |
| 2 | Елена | Иванова |
| 3 | Михаил | Иванов |
| 4 | Анастасия | Иванова |
| NULL | NULL | NULL |

| in_cat_id | in_cat_name |
|---|---|
| 1 | Зарплата |
| 2 | Подарок |
| 3 | Фриланс |
| NULL | NULL |

| in_it_id | in_it_name | in_cat_id |
|---|---|---|
| 1 | Работа Александра | 1 |
| 2 | День рождения Михаила | 2 |
| 3 | Фриланс Елены | 3 |
| 4 | Фриланс Александра | 3 |
| NULL | NULL | NULL |

| in_id | in_name | in_date | in_money | in_it_id | user_id |
|---|---|---|---|---|---|
| 1 | Зарплата Александра | 2023-01-15 | 60000.00 | 1 | 1 |
| 2 | Подарок Михаилу | 2023-02-20 | 5000.00 | 2 | 3 |
| 3 | Фриланс Елены | 2023-03-05 | 15000.00 | 3 | 2 |
| 4 | Фриланс Александра | 2023-02-20 | 10000.00 | 4 | 1 |
| 5 | Другой подарок Михаилу | 2023-01-20 | 3000.00 | 2 | 3 |
| NULL | NULL | NULL | NULL | NULL | NULL |

После:

| user_id | name | surname |
|---|---|---|
| 1 | Александр | Иванов |
| 2 | Елена | Иванова |
| 3 | Михаил | Иванов |
| 4 | Анастасия | Иванова |
| 5 | Имя | Фамилия |
| NULL | NULL | NULL |

| in_cat_id | in_cat_name |
|---|---|
| 1 | Зарплата |
| 2 | Подарок |
| 3 | Фриланс |
| 4 | КатегорияДохода |
| NULL | NULL |

| in_it_id | in_it_name | in_cat_id |
|---|---|---|
| 1 | Работа Александра | 1 |
| 2 | День рождения Михаила | 2 |
| 3 | Фриланс Елены | 3 |
| 4 | Фриланс Александра | 3 |
| 5 | СтатьяДохода | 4 |
| NULL | NULL | NULL |

| in_id | in_name | in_date | in_money | in_it_id | user_id |
|---|---|---|---|---|---|
| 1 | Зарплата Александра | 2023-01-15 | 60000.00 | 1 | 1 |
| 2 | Подарок Михаилу | 2023-02-20 | 5000.00 | 2 | 3 |
| 3 | Фриланс Елены | 2023-03-05 | 15000.00 | 3 | 2 |
| 4 | Фриланс Александра | 2023-02-20 | 10000.00 | 4 | 1 |
| 5 | Другой подарок Михаилу | 2023-01-20 | 3000.00 | 2 | 3 |
| 6 | Название дохода | 2023-01-01 | 10000.00 | 5 | 5 |
| NULL | NULL | NULL | NULL | NULL | NULL |

CALL InsertIncomeWithReference('Имя', 'Фамилия', 'КатегорияДохода', 'СтатьяДохода', 'Название дохода', '2023-01-01', 10000.00);

CALL InsertIncomeWithReference('Алексей', 'Смирнов', 'Инвестиции', 'Дивиденды', 'Дивиденды от акций', '2023-06-01', 15000.00);

CALL InsertIncomeWithReference('Иван', 'Петров', 'Инвестиции', 'Дивиденды', 'Дивиденды от акций', '2023-06-30', 80000.00);

— удаление с очисткой справочников – удаление данных из родительской таблицы, если после удаления данных из дочерней у строки родительской больше нет зависимых (удаляется информация о студенте, если в его группе нет больше студентов, запись удаляется из таблицы с перечнем групп);

Процедура будет удалять запись из родительской таблицы income_category, если после удаления записи из дочерней таблицы income_item не останется связанных записей в income_category.

Процедура DeleteIncomeItemWithCategoryCleanup удаляет запись из таблицы Income_Item и, если после этого в соответствующей категории дохода (Income_Category) не остается других записей, удаляет и эту категорию дохода.

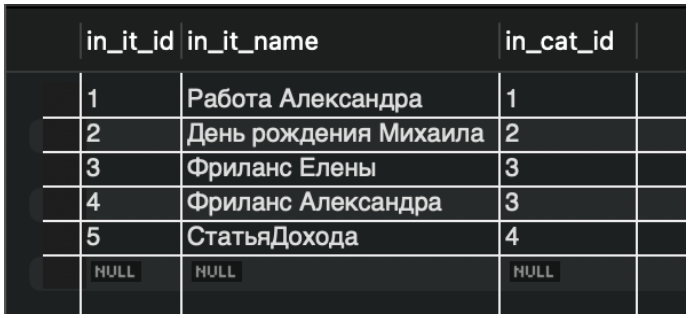
```
DELIMITER $$


CREATE PROCEDURE DeleteIncomeItemWithCategoryCleanup(
    IN _in_it_id INT
)
BEGIN
    DECLARE _in_cat_id INT;


    -- Получаем in_cat_id для удаляемого income_item
    SELECT in_cat_id INTO _in_cat_id FROM Income_Item WHERE in_it_id = _in_it_id;


    -- Удаляем запись из income_item
    DELETE FROM Income_Item WHERE in_it_id = _in_it_id;


    -- Проверяем, остались ли связанные записи в income_category
    IF NOT EXISTS (SELECT 1 FROM Income_Item WHERE in_cat_id = _in_cat_id) THEN
-- Если связанные записи отсутствуют, удаляем категорию
        DELETE FROM Income_Category WHERE in_cat_id = _in_cat_id;
    END IF;
END$$
```

DELIMITER ;

CALL DeleteIncomeItemWithCategoryCleanup(5);

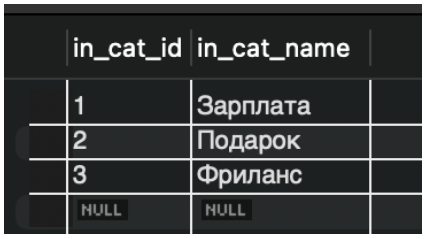
До:

| in_cat_id | in_cat_name |
|---|---|
| 1 | Зарплата |
| 2 | Подарок |
| 3 | Фриланс |
| 4 | КатегорияДохода |
| NULL | NULL |

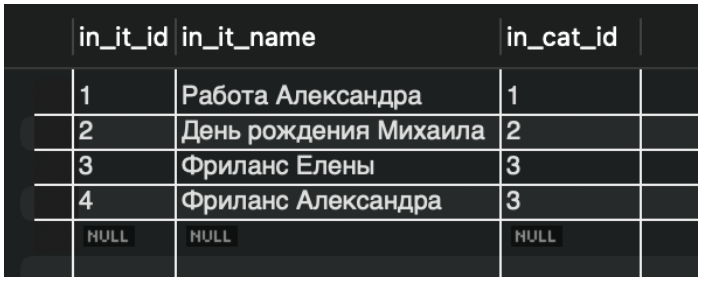


| in_it_id | in_it_name | in_cat_id |
|---|---|---|
| 1 | Работа Александра | 1 |
| 2 | День рождения Михаила | 2 |
| 3 | Фриланс Елены | 3 |
| 4 | Фриланс Александра | 3 |
| 5 | СтатьяДохода | 4 |
| NULL | NULL | NULL |

После:



| in_cat_id | in_cat_name |
|---|---|
| 1 | Зарплата |
| 2 | Подарок |
| 3 | Фриланс |
| NULL | NULL |



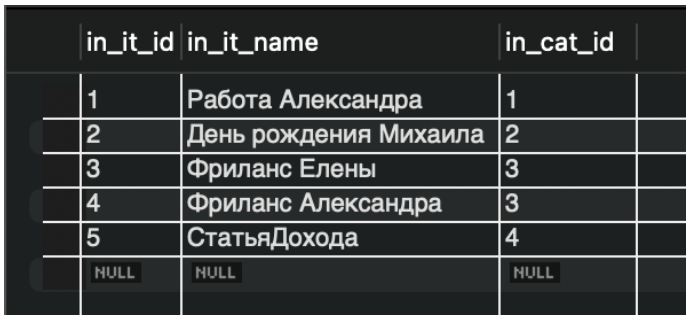
| in_it_id | in_it_name | in_cat_id |
|---|---|---|
| 1 | Работа Александра | 1 |
| 2 | День рождения Михаила | 2 |
| 3 | Фриланс Елены | 3 |
| 4 | Фриланс Александра | 3 |
| NULL | NULL | NULL |

— каскадное удаление (удаление всех зависимых данных);

Процедура CascadeDeleteIncomeItem удаляет запись из таблицы Income_Item и все связанные с ней записи из таблицы Income по заданному идентификатору in_it_id.

```
DELIMITER $$

CREATE PROCEDURE CascadeDeleteIncomeItem(
    IN _in_it_id INT
)
BEGIN
    -- Удаляем связанные данные из дочерней таблицы income
    DELETE FROM Income WHERE in_it_id = _in_it_id;

    -- Удаляем запись из родительской таблицы income_item
    DELETE FROM Income_Item WHERE in_it_id = _in_it_id;
END$$

DELIMITER ;

CALL CascadeDeleteIncomeItem(5);
```

До:



| in_it_id | in_it_name | in_cat_id |
|---|---|---|
| 1 | Работа Александра | 1 |
| 2 | День рождения Михаила | 2 |
| 3 | Фриланс Елены | 3 |
| 4 | Фриланс Александра | 3 |
| 5 | СтатьяДохода | 4 |
| NULL | NULL | NULL |

| in_id | in_name | in_date | in_money | in_it_id | user_id |
|---|---|---|---|---|---|
| 1 | Зарплата Александра | 2023-01-15 | 60000.00 | 1 | 1 |
| 2 | Подарок Михаилу | 2023-02-20 | 5000.00 | 2 | 3 |
| 3 | Фриланс Елены | 2023-03-05 | 15000.00 | 3 | 2 |
| 4 | Фриланс Александра | 2023-02-20 | 10000.00 | 4 | 1 |
| 5 | Другой подарок Михаилу | 2023-01-20 | 3000.00 | 2 | 3 |
| 6 | Название дохода | 2023-01-01 | 10000.00 | 5 | 5 |
| NULL | NULL | NULL | NULL | NULL | NULL |

После:

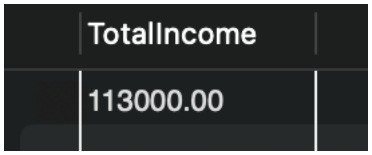
| in_it_id | in_it_name | in_cat_id |
|---|---|---|
| 1 | Работа Александра | 1 |
| 2 | День рождения Михаила | 2 |
| 3 | Фриланс Елены | 3 |
| 4 | Фриланс Александра | 3 |
| NULL | NULL | NULL |

| in_id | in_name | in_date | in_money | in_it_id | user_id |
|---|---|---|---|---|---|
| 1 | Зарплата Александра | 2023-01-15 | 60000.00 | 1 | 1 |
| 2 | Подарок Михаилу | 2023-02-20 | 5000.00 | 2 | 3 |
| 3 | Фриланс Елены | 2023-03-05 | 15000.00 | 3 | 2 |
| 4 | Фриланс Александра | 2023-02-20 | 10000.00 | 4 | 1 |
| 5 | Другой подарок Михаилу | 2023-01-20 | 3000.00 | 2 | 3 |
| NULL | NULL | NULL | NULL | NULL | NULL |

— вычисление и возврат значения агрегатной функции (т.к. агрегатная функция дает единственный результат) (задача- вернуть данные из процедуры/функции);

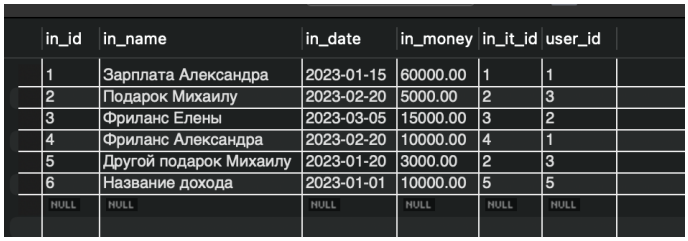
Процедура CalculateTotalIncome вычисляет и возвращает общую сумму доходов из таблицы Income.

```
DELIMITER $$

CREATE PROCEDURE CalculateTotalIncome(
    OUT total_income DECIMAL(10,2)
)
BEGIN
    SELECT SUM(in_money) INTO total_income FROM Income;
END$$

DELIMITER ;

CALL CalculateTotalIncome(@total);
SELECT @total AS TotalIncome;
```



| TotalIncome |
| --- |
| 113000.00 |

Проверка:



| in_id | in_name | in_date | in_money | in_it_id | user_id |
| --- | --- | --- | --- | --- | --- |
| 1 | Зарплата Александра | 2023-01-15 | 60000.00 | 1 | 1 |
| 2 | Подарок Михаилу | 2023-02-20 | 5000.00 | 2 | 3 |
| 3 | Фриланс Елены | 2023-03-05 | 15000.00 | 3 | 2 |
| 4 | Фриланс Александра | 2023-02-20 | 10000.00 | 4 | 1 |
| 5 | Другой подарок Михаилу | 2023-01-20 | 3000.00 | 2 | 3 |
| 6 | Название дохода | 2023-01-01 | 10000.00 | 5 | 5 |
| NULL | NULL | NULL | NULL | NULL | NULL |

— формирование статистики во временной таблице. (задача- работа с временными таблицами).

процедура создает временную таблицу, заполняет её агрегированными данными о сумме доходов по каждой категории и выводит результат. После вывода результатов временная таблица удаляется. Это обеспечивает работу со временными таблицами и формирование статистики по заданным критериям

Эта процедура создает временную таблицу и заполняет её детализированной статистикой по категориям дохода, включая количество записей о доходах, общую и среднюю сумму доходов в каждой категории.

Процедура GenerateDetailedIncomeStatistics создает временную таблицу и заполняет её детальной статистикой по категориям дохода, включая количество записей, общую и среднюю сумму доходов для каждой категории.

```
DROP PROCEDURE IF EXISTS GenerateDetailedIncomeStatistics;



DELIMITER $$


CREATE PROCEDURE GenerateDetailedIncomeStatistics()
BEGIN
    -- Создание временной таблицы для статистики
    CREATE TEMPORARY TABLE IF NOT EXISTS TempDetailedIncomeStatistics (
        CategoryName VARCHAR(255),
        TotalRecords INT,
        TotalAmount DECIMAL(10,2),
        AverageAmount DECIMAL(10,2)
    );


    -- Заполнение временной таблицы данными
```

```
    INSERT INTO TempDetailedIncomeStatistics (CategoryName, TotalRecords, TotalAmount,
AverageAmount)
    SELECT IC.in_cat_name,
        COUNT(I.in_id),
        SUM(I.in_money),
        AVG(I.in_money)
    FROM Income I
    JOIN Income_Item II ON I.in_it_id = II.in_it_id
    JOIN Income_Category IC ON II.in_cat_id = IC.in_cat_id
    GROUP BY IC.in_cat_name;

    -- Вывод данных из временной таблицы
    SELECT * FROM TempDetailedIncomeStatistics;

    -- Удаляем временную таблицу
    DROP TEMPORARY TABLE IF EXISTS TempDetailedIncomeStatistics;
END$$

DELIMITER ;

CALL GenerateDetailedIncomeStatistics();
```

| CategoryName | TotalRecords | TotalAmount | AverageAmount |
|---|---|---|---|
| Зарплата | 1 | 60000.00 | 60000.00 |
| Подарок | 2 | 8000.00 | 4000.00 |
| Фриланс | 2 | 25000.00 | 12500.00 |
| КатегорияДохода | 1 | 10000.00 | 10000.00 |