# **Добрый день.**

# **Для решения задач я использовала MySQL Community Server 8.0.27.**

**Задание №1. Подготовьте DDL-скрипты создания объектов для приведённой модели: создание таблиц, первичных, уникальных, внешних ключей и т.д.**

**Создаем базу данных CFT:**

CREATE SCHEMA cft DEFAULT CHARACTER SET utf8;

**Создаем таблицу CLIENTS:**

CREATE TABLE cft.clients (

ID INT NOT NULL AUTO\_INCREMENT,

NAME VARCHAR(1000) NOT NULL,

PLACE\_OF\_BIRTH VARCHAR(1000) NOT NULL,

DATE\_OF\_BIRTH DATE NOT NULL,

ADDRESS VARCHAR(1000) NULL,

PASSPORT VARCHAR(100) NOT NULL,

PRIMARY KEY (ID),

UNIQUE INDEX ID\_UNIQUE (ID ASC) VISIBLE,

UNIQUE INDEX PASSPORT\_UNIQUE (PASSPORT ASC) VISIBLE)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

**Создаем таблицу PRODUCTS:**

CREATE TABLE cft.products (

ID INT NOT NULL AUTO\_INCREMENT,

PRODUCT\_TYPE\_ID INT NOT NULL,

NAME VARCHAR(100) NOT NULL,

CLIENT\_REF INT NOT NULL,

OPEN\_DATE DATE NOT NULL,

CLOSE\_DATE DATE NULL,

PRIMARY KEY (ID),

UNIQUE INDEX ID\_UNIQUE (ID ASC) VISIBLE)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

**Добавляем настройку для вторичного ключа CLIENT\_REF таблицы PRODUCTS к полю ID таблицы CLIENTS:**

ALTER TABLE cft.products

ADD INDEX ‘products.CLIENT\_REF\_idx’ (CLIENT\_REF ASC) VISIBLE;

ALTER TABLE cft.products

ADD CONSTRAINT ‘products.CLIENT\_REF’

ADD FOREIGN KEY (CLIENT\_REF)

REFERENCES cft.clients (ID)

ON DELETE RESTRICT

ON UPDATE RESTRICT;

**Создаем таблицу PRODUCT\_TYPE:**

CREATE TABLE cft.product\_type (

ID INT NOT NULL AUTO\_INCREMENT,

NAME VARCHAR(100) NOT NULL,

BEGIN\_DATE DATE NOT NULL,

END\_DATE DATE NULL,

TARIF\_REF INT NOT NULL,

PRIMARY KEY (ID),

UNIQUE INDEX ID\_UNIQUE (ID ASC) VISIBLE)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

**Добавляем настройку для вторичного ключа PRODUCT\_TYPE\_ID таблицы PRODUCTS к полю ID таблицы PRODUCT\_TYPE:**

ALTER TABLE cft.products

ADD INDEX ‘products.PRODUCT\_TYPE\_ID\_idx’ (PRODUCT\_TYPE\_ID ASC) VISIBLE;

ALTER TABLE cft.products

ADD CONSTRAINT ‘products.PRODUCT\_TYPE\_ID’

ALTER TABLE cft.products

ADD FOREIGN KEY (PRODUCT\_TYPE\_ID)

REFERENCES cft.product\_type (ID)

ON DELETE RESTRICT

ON UPDATE RESTRICT;

**Создаем таблицу TARIFS:**

CREATE TABLE cft.tarifs (

ID INT NOT NULL AUTO\_INCREMENT,

NAME VARCHAR(100) NOT NULL,

COST FLOAT NOT NULL DEFAULT 0.00,

PRIMARY KEY (ID),

UNIQUE INDEX ID\_UNIQUE (ID ASC) VISIBLE)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

**Добавляем настройку для вторичного ключа TARIF\_REF таблицы PRODUCT\_TYPE к полю ID таблицы TARIFS:**

ALTER TABLE cft.product\_type

ADD INDEX ‘product\_type.TARIF\_REF\_idx’ (TARIF\_REF ASC) VISIBLE;

ALTER TABLE cft.product\_type

ADD CONSTRAINT ‘product\_type.TARIF\_REF’

FOREIGN KEY (TARIF\_REF)

REFERENCES cft.tarifs (ID)

ON DELETE RESTRICT

ON UPDATE CASCADE;

**Создаем таблицу ACCOUNTS:**

CREATE TABLE cft.accounts (

ID INT NOT NULL AUTO\_INCREMENT,

NAME VARCHAR(100) NOT NULL,

SALDO FLOAT NOT NULL DEFAULT 0.00,

CLIENT\_REF INT NOT NULL,

OPEN\_DATE DATE NOT NULL,

CLOSE\_DATE DATE NULL,

PRODUCT\_REF INT NOT NULL,

ACC\_NUM VARCHAR(25) NOT NULL,

PRIMARY KEY (ID))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

**Добавляем настройку для вторичного ключа CLIENT\_REF таблицы ACCOUNTS к полю ID таблицы CLIENTS, и для вторичного ключа PRODUCT\_REF таблицы ACCOUNTS к полю ID таблицы PRODUCTS:**

ALTER TABLE cft.accounts

ADD INDEX ‘accounts.CLIENT\_REF\_idx’ (CLIENT\_REF ASC) VISIBLE,

ADD INDEX ‘accounts.PRODUCT\_REF\_idx’ (PRODUCT\_REF ASC) VISIBLE;

ALTER TABLE cft.accounts

ADD CONSTRAINT `accounts.CLIENT\_REF`

FOREIGN KEY (CLIENT\_REF)

REFERENCES cft.clients (ID)

ON DELETE RESTRICT

ON UPDATE RESTRICT,

ADD CONSTRAINT `accounts.PRODUCT\_REF`

FOREIGN KEY (PRODUCT\_REF)

REFERENCES cft.products (ID)

ON DELETE RESTRICT

ON UPDATE RESTRICT;

**Создание таблицы RECORDS:**

CREATE TABLE cft.records (

ID INT NOT NULL AUTO\_INCREMENT,

DT INT NOT NULL,

SUM FLOAT NOT NULL DEFAULT 0,

ACC\_REF INT NOT NULL,

OPER\_DATE DATE NOT NULL,

PRIMARY KEY (ID),

UNIQUE INDEX ID\_UNIQUE (ID ASC) VISIBLE,

CONSTRAINT `records\_chk\_1` CHECK ((DT >= 0) and (DT <= 1)))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

**Добавляем настройку для вторичного ключа ACC\_REF таблицы RECORDS к полю ID таблицы ACCOUNTS:**

ALTER TABLE cft.records

ADD INDEX `records.ACC\_REF\_idx` (ACC\_REF ASC) VISIBLE;

ALTER TABLE cft.records

ADD CONSTRAINT `records.ACC\_REF`

FOREIGN KEY (ACC\_REF)

REFERENCES cft.accounts (ID)

ON DELETE RESTRICT

ON UPDATE RESTRICT;

**Задание №3. Подготовьте скрипты заполнения таблиц тестовыми данными, достаточными для выполнения заданий ниже.**

**Добавим еще 3х клиентов:**

INSERT INTO clients (ID, NAME, PLACE\_OF\_BIRTH, DATE\_OF\_BIRTH, ADDRESS, PASSPORT)

VALUES (4, 'Краснова Светлана Петровна', 'Россия, Кемеровская область, г. Новокузнецк', str\_to\_date('12.05.1999','%d.%m.%Y'), 'Россия, Кемеровская область, г. Новокузнецк, ул. Пущкина, д. 5', '3215 555555, выдан МВД г. Новокузнецка, 10.01.2015');

INSERT INTO clients (ID, NAME, PLACE\_OF\_BIRTH, DATE\_OF\_BIRTH, ADDRESS, PASSPORT)

VALUES (5, 'Ежиков Сергей Иванович', 'Россия, Кемеровская область, г. Новокузнецк', str\_to\_date('19.06.1995','%d.%m.%Y'), 'Россия, Кемеровская область, г. Новокузнецк, ул. Кирова, д. 75', '3215 666666, выдан МВД г. Новокузнецка, 10.01.2015');

INSERT INTO clients (ID, NAME, PLACE\_OF\_BIRTH, DATE\_OF\_BIRTH, ADDRESS, PASSPORT)

VALUES (6, 'Сергеев Константин Викторович', 'Россия, Кемеровская область, г. Новокузнецк', str\_to\_date('29.07.1993','%d.%m.%Y'), 'Россия, Кемеровская область, г. Новокузнецк, пр. Металлургов, д. 9', '3215 888888, выдан МВД г. Новокузнецка, 10.01.2015');

**Добавим еще 4 продукта:**

INSERT INTO products

SELECT 4 AS ID, 2 AS PRODUCT\_TYPE\_ID, 'Депозитный договор с Красновой С.П.' AS NAME, 4 AS CLIENT\_REF, str\_to\_date('01.08.2017','%d.%m.%Y') AS OPEN\_DATE, null AS CLOSE\_DATE;

INSERT INTO products

SELECT 5 AS ID, 2 AS PRODUCT\_TYPE\_ID, 'Депозитный договор с Ежиковым С.И.' AS NAME, 5 AS CLIENT\_REF, str\_to\_date('01.08.2017','%d.%m.%Y') AS OPEN\_DATE, null AS CLOSE\_DATE;

INSERT INTO products

SELECT 6 AS ID, 1 AS PRODUCT\_TYPE\_ID, 'Кредитный договор с Ежиковым С.И.' AS NAME, 5 AS CLIENT\_REF, str\_to\_date('01.06.2015','%d.%m.%Y') AS OPEN\_DATE, null AS CLOSE\_DATE;

INSERT INTO products

SELECT 7 AS ID, 1 AS PRODUCT\_TYPE\_ID, 'Кредитный договор с Сергеевым К.В.' AS NAME, 6 AS CLIENT\_REF, str\_to\_date('01.01.2022','%d.%m.%Y') AS OPEN\_DATE, null AS CLOSE\_DATE;

**Добавим еще 4 счета:**

insert into accounts values (4, 'Депозитный счет для Красновой С.П.', 10000, 4, str\_to\_date('01.08.2017','%d.%m.%Y'), null, 4, '42301810400000000002');

insert into accounts values (5, 'Депозитный счет для Ежикова С.И.', 15000, 5, str\_to\_date('01.08.2017','%d.%m.%Y'), null, 5, '42301810400000000003');

insert into accounts values (6, 'Кредитный счет для Ежикова С.И.', -5000, 5, str\_to\_date('01.06.2015','%d.%m.%Y'), null, 6, '45502810401020000023');

insert into accounts values (7, 'Кредитный счет для Сергеева К.В.', -5000, 6, str\_to\_date('01.01.2022','%d.%m.%Y'), null, 7, '45502810401020000024');

**Добавим операцию и изменение остатка по счету на основании данной операции:**

INSERT INTO records VALUES (1, 1, 5000, 1, str\_to\_date('01.06.2015','%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO - (SELECT sum FROM records WHERE ID=1) WHERE accounts.ID= 1

INSERT INTO records VALUES (2, 0, 1000, 1, str\_to\_date('01.07.2015', '%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO + (SELECT sum FROM records WHERE ID=2) WHERE accounts.ID= 1

INSERT INTO records VALUES (3, 0, 2000, 1, str\_to\_date('01.08.2015', '%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO + (SELECT sum FROM records WHERE ID=3) WHERE accounts.ID= 1

INSERT INTO records VALUES (4, 0, 3000, 1, str\_to\_date('01.09.2015', '%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO + (SELECT sum FROM records WHERE ID=4) WHERE accounts.ID= 1

INSERT INTO records VALUES (5, 1, 5000, 1, str\_to\_date('01.10.2015', '%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO - (SELECT sum FROM records WHERE ID=5) WHERE accounts.ID= 1

INSERT INTO records VALUES (6, 0, 3000, 1, str\_to\_date('01.10.2015', '%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO + (SELECT sum FROM records WHERE ID=6) WHERE accounts.ID= 1

INSERT INTO records VALUES (7, 0, 10000, 2, str\_to\_date('01.08.2017', '%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO + (SELECT sum FROM records WHERE ID=7) WHERE accounts.ID= 2

INSERT INTO records VALUES (8, 1, 1000, 2, str\_to\_date('05.08.2017', '%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO - (SELECT sum FROM records WHERE ID=8) WHERE accounts.ID= 2

INSERT INTO records VALUES (9, 1, 2000, 2, str\_to\_date('21.09.2017', '%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO - (SELECT sum FROM records WHERE ID=9) WHERE accounts.ID= 2

INSERT INTO records VALUES (10, 1, 5000, 2, str\_to\_date('24.10.2017', '%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO - (SELECT sum FROM records WHERE ID=10) WHERE accounts.ID= 2

INSERT INTO records VALUES (11, 0, 6000, 2, str\_to\_date('26.11.2017', '%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO + (SELECT sum FROM records WHERE ID=11) WHERE accounts.ID= 2

INSERT INTO records VALUES (12, 0, 120000, 3, str\_to\_date('08.09.2017', '%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO + (SELECT sum FROM records WHERE ID=12) WHERE accounts.ID= 3

INSERT INTO records VALUES (13, 1, 1000, 3, str\_to\_date('05.10.2017', '%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO - (SELECT sum FROM records WHERE ID=13) WHERE accounts.ID= 3

INSERT INTO records VALUES (14, 1, 2000, 3, str\_to\_date('21.10.2017', '%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO - (SELECT sum FROM records WHERE ID=14) WHERE accounts.ID= 3

INSERT INTO records VALUES (15, 1, 5000, 3, str\_to\_date('24.10.2017', '%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO - (SELECT sum FROM records WHERE ID=15) WHERE accounts.ID= 3

INSERT INTO records VALUES (16, 0, 500, 4, str\_to\_date('01.06.2015','%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO + (SELECT sum FROM records WHERE ID=16) WHERE accounts.ID= 4

INSERT INTO records VALUES (17, 0, 1500, 4, str\_to\_date('01.06.2015','%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO + (SELECT sum FROM records WHERE ID=17) WHERE accounts.ID= 4

INSERT INTO records VALUES (18, 0, 4000, 4, str\_to\_date('01.06.2015','%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO + (SELECT sum FROM records WHERE ID=18) WHERE accounts.ID= 4

INSERT INTO records VALUES (19, 1, 500, 6, str\_to\_date('01.06.2015','%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO - (SELECT sum FROM records WHERE ID=19) WHERE accounts.ID= 6

INSERT INTO records VALUES (20, 1, 2000, 6, str\_to\_date('01.06.2015','%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO - (SELECT sum FROM records WHERE ID=20) WHERE accounts.ID= 6

INSERT INTO records VALUES (21, 0, 1000, 6, str\_to\_date('01.06.2015','%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO + (SELECT sum FROM records WHERE ID=21) WHERE accounts.ID= 6

INSERT INTO records VALUES (22, 0, 1000, 5, str\_to\_date('01.06.2015','%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO + (SELECT sum FROM records WHERE ID=22) WHERE accounts.ID= 5

INSERT INTO records VALUES (23, 0, 2000, 5, str\_to\_date('01.06.2015','%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO + (SELECT sum FROM records WHERE ID=23) WHERE accounts.ID= 5

INSERT INTO records VALUES (24, 0, 200, 5, str\_to\_date('01.06.2015','%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO + (SELECT sum FROM records WHERE ID=24) WHERE accounts.ID= 5

INSERT INTO records VALUES (25, 0, 1000, 1, str\_to\_date('20.01.2022','%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO + (SELECT sum FROM records WHERE ID=25) WHERE accounts.ID= 1

INSERT INTO records VALUES (26, 0, 500, 1, str\_to\_date('21.01.2022','%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO + (SELECT sum FROM records WHERE ID=26) WHERE accounts.ID= 1

INSERT INTO records VALUES (27, 0, 500, 1, str\_to\_date('21.01.2022','%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO + (SELECT sum FROM records WHERE ID=27) WHERE accounts.ID= 1

INSERT INTO records VALUES (28, 0, 1000, 2, str\_to\_date('20.01.2022','%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO + (SELECT sum FROM records WHERE ID=28) WHERE accounts.ID= 2

INSERT INTO records VALUES (29, 0, 2000, 2, str\_to\_date('22.01.2022','%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO + (SELECT sum FROM records WHERE ID=29) WHERE accounts.ID= 2

INSERT INTO records VALUES (30, 1, 3000, 3, str\_to\_date('20.01.2022','%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO - (SELECT sum FROM records WHERE ID=30) WHERE accounts.ID= 3

INSERT INTO records VALUES (31, 0, 4000, 3, str\_to\_date('22.01.2022','%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO + (SELECT sum FROM records WHERE ID=31) WHERE accounts.ID= 3

INSERT INTO records VALUES (32, 1, 1000, 3, str\_to\_date('22.01.2022','%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO - (SELECT sum FROM records WHERE ID=32) WHERE accounts.ID= 3

INSERT INTO records VALUES (33, 0, 1000, 1, str\_to\_date('09.02.2022','%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO + (SELECT sum FROM records WHERE ID=33) WHERE accounts.ID= 1

INSERT INTO records VALUES (34, 0, 6500, 6, str\_to\_date('01.06.2015','%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO + (SELECT sum FROM records WHERE ID=34) WHERE accounts.ID= 6

INSERT INTO records VALUES (35, 0, 5000, 7, str\_to\_date('03.01.2022','%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO + (SELECT sum FROM records WHERE ID=35) WHERE accounts.ID= 7

INSERT INTO records VALUES (36, 1, 5000, 1, str\_to\_date('01.02.2022','%d.%m.%Y'));

UPDATE accounts SET SALDO = SALDO - (SELECT sum FROM records WHERE ID=36) WHERE accounts.ID= 1

**Задание 4. Сформируйте отчет, который содержит все счета, относящиеся к продуктам типа ДЕПОЗИТ, принадлежащих клиентам, у которых нет открытых продуктов типа КРЕДИТ.**

SELECT accounts.NAME, accounts.ACC\_NUM

FROM accounts INNER JOIN products ON accounts.PRODUCT\_REF=products.ID

WHERE accounts.CLIENT\_REF NOT IN

(SELECT products.CLIENT\_REF FROM products WHERE PRODUCT\_TYPE\_ID=1)

AND products.PRODUCT\_TYPE\_ID=2;

**Задание 5. Сформируйте выборку, которая содержит средние движения по счетам в рамках одного произвольного дня, в разрезе типа продукта.**

**Создаем представление для объединения двух таблиц (счета и продукты):**

CREATE VIEW accountsID\_productsID AS

SELECT accounts.ID AS AccID, accounts.name AS NameAccount, products.PRODUCT\_TYPE\_ID AS prodTypeID, products.name AS NameProduct

FROM accounts INNER JOIN products ON accounts.PRODUCT\_REF=products.ID;

**Создаем выборку:**

SELECT (

SELECT product\_type.NAME FROM product\_type

WHERE product\_type.ID= accountsID\_productsID.prodTypeID)

AS 'Тип продукта',

AVG(RECORDS.SUM) AS 'Средние движения по счетам',

records.OPER\_DATE

FROM records INNER JOIN accountsID\_productsID ON records.ACC\_REF=accountsID\_productsID.AccID

WHERE records.OPER\_DATE='2015.06.01' GROUP BY prodTypeID;

**ЗАДАНИЕ 6. Сформируйте выборку, в который попадут клиенты, у которых были операции по счетам за прошедший месяц от текущей даты. Выведите клиента и сумму операций за день в разрезе даты.**

SELECT (

SELECT (

SELECT clients.NAME FROM clients WHERE clients.ID=accounts.CLIENT\_REF)

FROM accounts WHERE accounts.ID=records.ACC\_REF) AS 'client',

sum ,

records.OPER\_DATE AS date

FROM records

WHERE records.OPER\_DATE >= (SELECT CURRENT\_DATE - INTERVAL 30 DAY)

GROUP BY client, date;

**ЗАДАНИЕ 7. В результате сбоя в базе данных разъехалась информация между остатками и операциями по счетам. Напишите нормализацию (процедуру выравнивающую данные), которая найдет такие счета и восстановит остатки по счету.**

**Я думаю, что при сбое в конкретный день, мы бы остаток по счету на утро, прибавили/отняли операции и сравнили с текущим остатком. Или проверяли прошла ли каждая транзакция по счету, и, если нет, проводили транзакцию.**

**Но т.к. у нас нет такой информации, я взяла остаток на момент открытия счета initialBalance, отняла от него сумму всех операций по дебиту DTs, прибавила все операции по кредиту KT, и сравнила с текущим остатком currentBalance.**

**Создаем хранимую процедуру:**

DELIMITER $$

USE cft $$

CREATE PROCEDURE `procedure2`(IN AccID INT, initialBalance int)

BEGIN

DECLARE firstBalance, DTs, KT, totallBalance, currentBalance INT;

SET @ firstBalance = initialBalance;

SET @DTs = (SELECT sum(sum) FROM records WHERE acc\_ref = AccID AND DT = 1);

SET @KT = (SELECT sum(sum) FROM records WHERE acc\_ref = AccID AND DT = 0);

SET @ totallBalance = @ firstBalance -@DTs + @KT;

SET @currentBalance = (SELECT saldo FROM accounts WHERE id = AccID);

IF @currentBalance != @ totallBalance THEN

Update accounts SET saldo = @ totallBalance WHERE id = AccID;

END IF;

END;

**Вызываем хранимую процедуру для каждого счета:**

call cft.procedure2(1, -2000);

call cft.procedure2(2, 6000);

call cft.procedure2(3, 8000);

call cft.procedure2(4, 10000);

call cft.procedure2(5, 15000);

call cft.procedure2(6, -5000);

call cft.procedure2(7, -5000);

**ЗАДАНИЕ 8. Сформируйте выборку, которая содержит информацию о клиентах, которые полностью погасили кредит, но при этом не закрыли продукт и пользуются им дальше (по продукту есть операция новой выдачи кредита).**

**Делаем выборку по клиентам, у которых есть кредитный счет (PRODUCT\_REF = 1). Добавляем условия: счет не закрыт, остаток не равен 0, есть операции повторной выдачи кредита.**

SELECT \* FROM accounts WHERE

PRODUCT\_REF = 1

AND CLOSE\_DATE IS NULL

AND SALDO !=0

AND 0 < (SELECT count(DT) FROM records WHERE DT=1 AND records.ACC\_REF = accounts.ID );

**ЗАДАНИЕ 9. Закройте продукты (установите дату закрытия равную текущей) типа КРЕДИТ, у которых произошло полное погашение, но при этом не было повторной выдачи.**

**Создаем запрос, который устанавливает дату закрытия счета клиента типа “кредит”, остаток по счету которого равен 0, т.е повторной выдачи кредита на конец дня не произошло.**

UPDATE accounts SET CLOSE\_DATE = current\_date() WHERE PRODUCT\_REF = 1 AND SALDO =0;

**ЗАДАНИЕ 10. Закройте возможность открытия (установите дату окончания действия) для типов продуктов, по счетам продуктов которых, не было движений более одного месяца.**

**Воспользуемся представлением “accountsID\_productsID” из задания 5.**

**Создаем запрос на обновление даты окончания типа продукта, по которому не было движения по счетам клиентов более 30 дней:**

UPDATE product\_type SET END\_DATE = current\_date()

WHERE product\_type.ID NOT IN (

SELECT distinct prodTypeID FROM accountID\_productsID

WHERE accID IN (

SELECT DISTINCT records.ACC\_REF FROM records

WHERE records.OPER\_DATE >=

(SELECT CURRENT\_DATE - INTERVAL 30 DAY)));

**ЗАДАНИЕ 11. В модель данных добавьте сумму договора по продукту. Заполните поле для всех продуктов суммой максимальной дебетовой операции по счету для продукта типа КРЕДИТ, и суммой максимальной кредитовой операции по счету продукта для продукта типа ДЕПОЗИТ или КАРТА.**

**Добавим столбец в таблицу product\_type:**

ALTER TABLE cft.product\_type

ADD COLUMN `SUM\_CONTRACT` INT NOT NULL DEFAULT 0 AFTER `NAME`;

**Заполним столбцы используя уже созданное представление из задания 5 “accountID\_productsID”:**

**Для кредита:**

UPDATE product\_type SET SUM\_CONTRACT = (

SELECT sum(sum) FROM records

WHERE DT=1 and ACC\_REF in (

SELECT AccID FROM accountID\_productsID

WHERE prodTypeID =1))

WHERE product\_type.ID=1;

**Для депозита:**

UPDATE product\_type SET SUM\_CONTRACT = (

SELECT sum(sum) FROM records

WHERE DT=0 and ACC\_REF in (

SELECT AccID FROM accountID\_productsID2

WHERE prodTypeID =2))

WHERE product\_type.ID=2;

**Для карт:**

UPDATE product\_type SET SUM\_CONTRACT = (

SELECT sum(sum) FROM records

WHERE DT=0 and ACC\_REF in (

SELECT AccID FROM accountID\_productsID2

WHERE prodTypeID =3))

WHERE product\_type.ID=3;