DB Final Project Report

prepared by:

Ayana Anartbekova

Ernur Kurmangali

Kazakh British Technical University

2020-2021 spring semester

Task given:

We are a small business, a shop selling goods to customers:

- We stock goods from several buyers paying specific price
- We offer goods to customers by selling items without own price
- We track order details, such as addresses, names, quantity, purchasing date, etc,as well as whether customers payed for their order.
- We should calculate our monthly profit

Normalization

The normalization process is one of the most important initial steps for creating an effective database with minimal redundancy. In this project, normalization was chunked into three stages:

1) 1st normalization was executed to group all similar attributes from all tables. Then, these attributes were separated into distinct tables to prevent repetition and further data reading confusion.
2) 2nd normalization was done by comparing data which had compound and partial keys and creating a separate table for them. These steps are crucial to determine the relationship between tables and see correspondence between rows.
3) The last 3rd normalization compared the non-primary-key attributes. In other words, if the attributes had a relationship they were separated into more tables.

0NF

acc_id
acc_name
wealth
last_upd
stock_id
stock_name
stock_quantity
sell_price
buyer_id
buyer_name
buyer_address
quantity
buy_price
customer_id
customer_name
customer_address
cust_login
cust_password
admin_login
admin_password
emp_id
emp_wage
defaulter_id
defaulter_name
defaulter_address
order_id
date_ordered
date_delivered
total_price
paid (paid or pending)

status
status_desc
num_ordered
bonuses
open_date
last_bonus
present_id
present_name
present_price
month
profit
purchase_date

1NF

Stock table

stock_id
stock_name
sell_price
stock_q
discount

Buyers table

buyer_id
stock_id
buyer_name
buyer_address
quantity
buy_price

Customers table

customer_id
customer_name
customer_address
login
password

bonuses
open_date
last_bonus

Order table
order_id
stock_id
date_ordered
date_delivered
customer_id
number_ordered
total_price
paid (pending or paid)
status
status_desc

Defaulters table
defaulter_id
order_id

Purchase table
buyer_id
stock_id
purchase_date
quantity
stock_to_buy_id
stock_to_buy_name

Profit table
month
profit

Presents table
customer_id
present_id
present_name

present_price

Admins table
login
password

Employees table
emp_id
emp_wage

Accounts table
acc_id
acc_name
wealth
last_upd

2NF

Stock table
stock_id
stock_name
sell_price
stock_q
discount

Buyer_stock table
buyer_id
stock_id
quantity
buy_price

Buyer_info table
buyer_id

buyer_name
buyer_address

Customers table
customer_id
customer_name
customer_address
login
password

Bonus cards table
customer_id
bonuses
open_date
last_bonus

Detail_order table
order_id
date_ordered
date_delivered
customer_id
total_price
paid (pending or paid)
status

Status table
status
status_desc

Order table
order_id
stock_id

number_ordered

Defaulters table
defaulter_id
order_id

Purchase_date
buyer_id
stock_id
purchase_date
quantity

To_buy table
stock_id
stock_name

Profit table
month/year
profit

Presents table
customer_id
present_id
present_name
present_price

Admins table
login
password

Employees table
emp_id
emp_wage

Accounts table

acc_id
acc_name
wealth
last_upd

3NF

In_stock table

stock_id
stock_name
sell_price
stock_q

Discounts table

stock_id
discount
last_date

Buyer_stock table

buyer_id
stock_id
quantity
buy_price

Buyer_info

buyer_id
buyer_name
buyer_address

Customers table

customer_id
customer_name
customer_address

Custs_login table
customer_id
login
password

Bonus cards table
customer_id
bonuses
open_date
last_bonus

Status table
status
status_desc

Detail_order table
order_id
date_ordered
date_delivered
customer_id
total_price
paid (pending or paid)
status

Order table
order_id
stock_id
number_ordered

Purchase_date
buyer_id
stock_id
purchase_date
quantity

Defaulters table
defaulter_id
order_id

To_buy table
stock_id
stock_name

Profit table
month/year
profit

Items_month table
stock_id
num_ordered

Presents table
present_id
present_name
present_price

Presents_bonus table
operation_id
customer_id
present_id

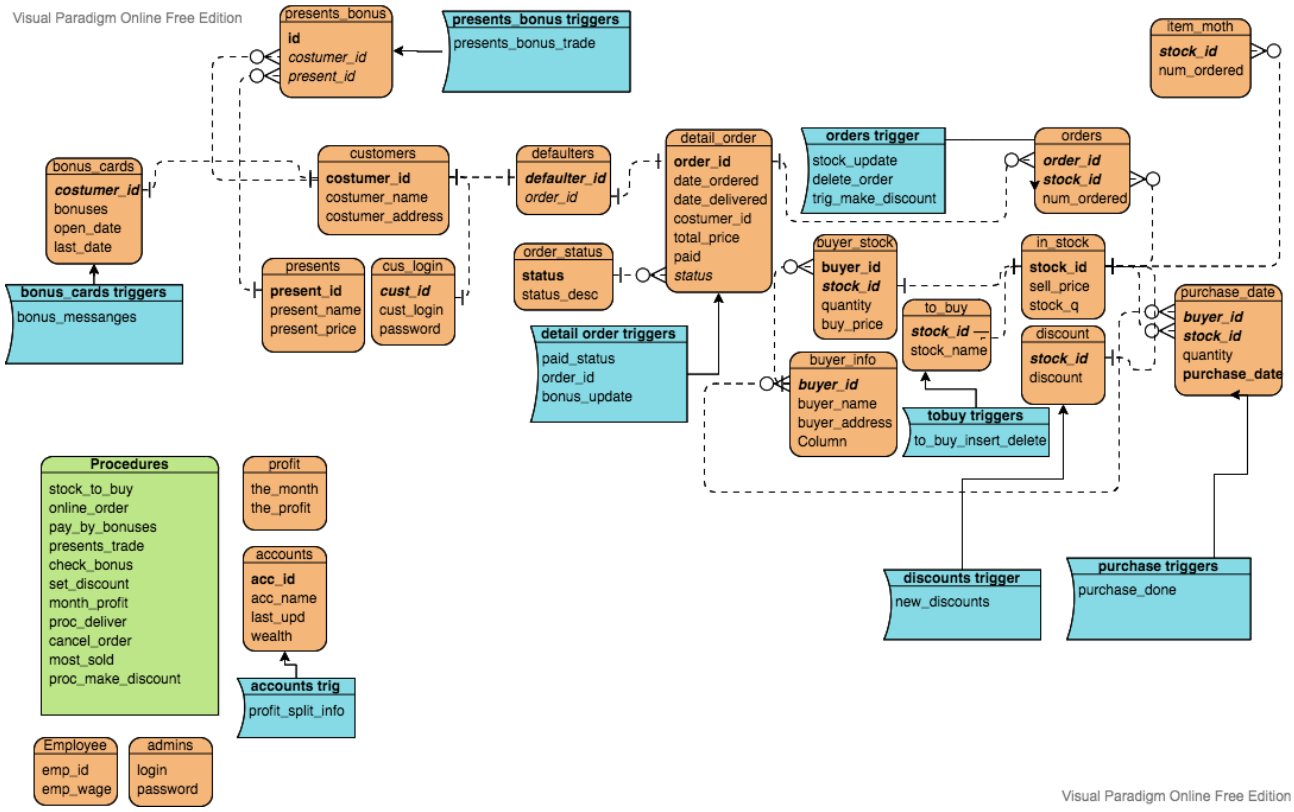
Admins table
login

password

Employees table
emp_id
emp_wage

Accounts table
acc_id
acc_name
wealth
last_upd

ERD


Visual Paradigm Online Free Edition
presents_bonus
id
costumer_id
present_id
presents_bonus triggers
presents_bonus_trade
item_moth
stock_id
num_ordered
bonus_cards
costumer_id
bonuses
open_date
last_date
customers
costumer_id
costumer_name
costumer_address
defaulters
defaulter_id
order_id
detail_order
order_id
date_ordered
date_delivered
costumer_id
total_price
paid
status
orders trigger
stock_update
delete_order
trig_make_discount
orders
order_id
stock_id
num_ordered
bonus_cards triggers
bonus_messanges
presents
present_id
present_name
present_price
cus_login
cust_id
cust_login
password
order_status
status
status_desc
buyer_stock
buyer_id
stock_id
quantity
buy_price
in_stock
stock_id
sell_price
stock_q
purchase_date
buyer_id
stock_id
quantity
purchase_date
detail order triggers
paid_status
order_id
bonus_update
to_buy
stock_id
stock_name
discount
stock_id
discount
buyer_info
buyer_id
buyer_name
buyer_address
Column
tobuy triggers
to_buy_insert_delete
Procedures
stock_to_buy
online_order
pay_by_bonuses
presents_trade
check_bonus
set_discount
month_profit
proc_deliver
cancel_order
most_sold
proc_make_discount
profit
the_month
the_profit
accounts
acc_id
acc_name
last_upd
wealth
accounts trig
profit_split_info
discounts trigger
new_discounts
purchase triggers
purchase_done
Employee
emp_id
emp_wage
admins
login
password
Visual Paradigm Online Free Edition

The process of creating a database with the tables

1) Creating tables for each variable.

```
create database shop
use shop

create table in_stock (
    stock_id integer identity(1,1),
    constraint pk_stock primary key(stock_id),
    sell_price integer,
    stock_q integer
)

create table stock_decs (
    stock_id integer identity(1,1),
    constraint pk_sd primary key(stock_id),
    stock_name varchar(10)
)

create table buyer_stock (
    buyer_id integer,
    stock_id integer,
    constraint pk_bs primary key(buyer_id, stock_id),
    quantity integer,
    buy_price integer,
)

create table buyer_info (
    buyer_id integer identity(1,1),
    constraint pk_bi primary key(buyer_id),
    buyer_name varchar(15),
    buyer_address varchar(30),
)
```

```
create table costumers (
    costumer_id integer identity(1,1),
    constraint pk_costumers primary key(costumer_id),
    costumer_name varchar(20),
    costumer_address varchar(30)
)

create table detail_order (
    order_id integer identity(1,1),
    constraint pk_do primary key(order_id),
    date_ordered date,
    date_delivered date,
    costumer_id integer,
    total_price integer,
    paid varchar(1)
)

create table orders (
    order_id integer,
    stock_id integer,
    constraint pk_order primary key(order_id, stock_id),
    num_ordered integer,
)

create table defaulters (
    defaulter_id integer identity(1,1),
    order_id integer,
    constraint pk_defaulters primary key(defaulter_id)
)
```

```
create table profit (
    the_month varchar(10),
    constraint pk_profit primary key(the_month),
    the_profit integer
)

create table purchase_date (
    buyer_id integer,
    stock_id integer,
    constraint pk_purchase primary key(buyer_id, stock_id),
    quantity integer,
    purchase_date date
)
```

```
create table to_buy(
    stock_id integer,
    constraint pk_to_buy primary key(stock_id),
    stock_name varchar(10)
)

create table order_status(
    status varchar(5),
    constraint pk_status primary key(status),
    status_desc varchar(100)
)
```

```
create table bonus_cards(
    costumer_id integer,
    constraint pk_bonus primary key(costumer_id),
    constraint fk_bonus foreign key(costumer_id) references costumers(costumer_id),
    bonuses integer,
    open_date date,
    last_bonus date
)

create table presents(
    present_id integer identity(1,1),
    constraint pk_presents primary key(present_id),
    present_name varchar(50),
    present_price integer
)

create table presents_bonus(
    id integer identity(1,1),
    costumer_id integer,
    constraint pk_present_bonus primary key(id),
    constraint fk_present_bonus foreign key(costumer_id) references costumers(costumer_id),
    present_id integer,
    constraint fk_bonus_present foreign key(present_id) references presents(present_id)
)
GO

create table discounts(
    stock_id integer,
    constraint fk_discounts foreign key(stock_id) references in_stock(stock_id),
    discount integer
)
GO

create table accounts(
    acc_id int,
    acc_name VARCHAR(25),
    wealth int,
    last_upd VARCHAR(25),
    CONSTRAINT pk_accounts PRIMARY KEY(acc_id)
)

create table employees(
    emp_id int IDENTITY(1,1),
    emp_wage int,
    CONSTRAINT pk_employees PRIMARY KEY(emp_id)
)
```

```
create table items_month(
    stock_id integer,
    constraint fk_items_month foreign key(stock_id) references in_stock(stock_id),
    num_ordered integer
)
GO

create table admins(
    login VARCHAR(20),
    PASSWORD VARCHAR(30)
)
```

2) Inserting values into the table.

```
insert into in_stock values (12000, 9000)
insert into in_stock values (120000, 19000)
insert into in_stock values (15000, 90000)
insert into in_stock values (10900, 1000)
insert into in_stock values (2000, 8000)
insert into in_stock values (12090, 10000)
insert into in_stock values (50000, 9500)
insert into in_stock values (47200, 100)
insert into in_stock values (37500, 9500)
insert into in_stock values (8900, 5000)

insert into stock_decs values ('Monitor')
insert into stock_decs values ('Computer')
insert into stock_decs values ('Keyboard')
insert into stock_decs values ('Mouse')
insert into stock_decs values ('HeadPhones')
insert into stock_decs values ('PS5')
insert into stock_decs values ('Printer')
insert into stock_decs values ('iPhone12')
insert into stock_decs values ('LapTop')
insert into stock_decs values ('Flascard')

insert into buyer_info values ('Uni-Care', 'Japan, Tokyo')
insert into buyer_info values ('Greentest', 'Kazakhstan, Almaty')
insert into buyer_info values ('Technics', 'Russia, Moscow')
insert into buyer_info values ('Digitals', 'Korea, Seoul')
insert into buyer_info values ('Vestidots', 'Beijing, China')
insert into buyer_info values ('OptMob', 'Berlin, Germany')
```

```
insert into buyer_stock values (1, 1, 1000, 8000)
insert into buyer_stock values (1, 5, 1000, 1200)
insert into buyer_stock values (2, 10, 20000, 7000)
insert into buyer_stock values (3, 6, 5000, 9990)
insert into buyer_stock values (3, 7, 5000, 38000)
insert into buyer_stock values (3, 9, 5000, 20000)
insert into buyer_stock values (4, 8, 1500, 35000)
insert into buyer_stock values (4, 2, 1500, 100000)
insert into buyer_stock values (5, 4, 7000, 8500)
insert into buyer_stock values (6, 3, 3000, 11000)

insert into costumers values ('Kenneth Hampton', 'Kazakhstan, Almaty')
insert into costumers values ('Bonnie Morton', 'Kazakhstan, Nur-Sultan')
insert into costumers values ('Vivien Glenn', 'Ukrain, Kyiv')
insert into costumers values ('Joseph Ross', 'Belarus, Minsk')
insert into costumers values ('Alice Paul', 'Russia, Moscow')

insert into detail_order values ('01.01.2021', '11.01.2021', 1, 0, 'T')
insert into detail_order values ('02.01.2021', '12.01.2021', 2, 0, 'T')
insert into detail_order values ('03.01.2021', null, 3, 0, 'F')
insert into detail_order values ('04.01.2021', '14.01.2021', 4, 0, 'T')
insert into detail_order values ('05.01.2021', '15.01.2021', 4, 0, 'T')
insert into detail_order values ('06.01.2021', '16.01.2021', 5, 0, 'T')
insert into detail_order values ('07.01.2021', null, 2, 0, 'F')
insert into detail_order values ('11.02.2021', '21.02.2021', 1, 0, 'T')
insert into detail_order values ('12.02.2021', '22.02.2021', 2, 0, 'T')
insert into detail_order values ('04.03.2021', '14.03.2021', 4, 0, 'T')
insert into detail_order values ('05.03.2021', '15.03.2021', 4, 0, 'T')
insert into detail_order values ('06.03.2021', '16.03.2021', 5, 0, 'T')
```

```
insert into orders values (1, 1, 2)
insert into orders values (1, 10, 1)
insert into orders values (2, 7, 1)
insert into orders values (3, 5, 1)
insert into orders values (3, 2, 3)
insert into orders values (4, 4, 4)
insert into orders values (5, 9, 2)
insert into orders values (5, 8, 4)
insert into orders values (6, 6, 1)
insert into orders values (7, 3, 2)
insert into orders values (7, 9, 5)
insert into orders values (8, 5, 1)
insert into orders values (8, 9, 2)
insert into orders values (9, 3, 4)
insert into orders values (10, 7, 1)
insert into orders values (11, 2, 4)
insert into orders values (12, 10, 5)
```

```
declare @a int
set @a = 0
while @a <= 12
begin
    update detail_order set total_price = (select sum(o.num_ordered*s.sell_price)
    from orders o join in_stock s on o.stock_id = s.stock_id
    where o.order_id = @a) where order_id = @a
    set @a = @a+1
end

insert into defaulters (order_id) select order_id from detail_order where paid like 'F'
and order_id not in (select order_id from defaulters)

insert into purchase_date VALUES (1,3,1000,'01.01.2021')
insert into purchase_date VALUES (2,10,2000,'03.01.2021')
insert into purchase_date VALUES (1,5,3000,'05.01.2021')

insert into profit values ('Январь', 0)
insert into profit values ('Февраль', 0)
insert into profit values ('Март', 0)
update profit set the_profit = (select sum((i.sell_price - b.buy_price)*o.num_ordered)
from in_stock i join buyer_stock b on i.stock_id = b.stock_id
join orders o on i.stock_id = o.stock_id
join detail_order d on d.order_id = o.order_id
where DateName( month , DateAdd( month , month(d.date_ordered) , 0 ) - 1 ) = the_month)

select*from profit
```

```
alter table stock_decs add constraint fk_stock foreign key(stock_id) references in_stock(stock_id)
alter table buyer_stock add constraint fk_bs foreign key(stock_id) references stock_decs(stock_id)
alter table buyer_stock add constraint fk_buyerstock foreign key(buyer_id) references buyer_info(buyer_id)
alter table detail_order add constraint fk_do foreign key(costumer_id) references costumers(costumer_id)
alter table orders add constraint fk_orders foreign key(order_id) references detail_order(order_id)
alter table defaulters add constraint fk_defaulters foreign key(order_id) references detail_order(order_id)
alter table purchase_date add constraint fk_purchase foreign key(buyer_id) references buyer_info(buyer_id)
alter table purchase_date add constraint fk_pd foreign key(stock_id) references stock_decs(stock_id)
```

Writing queries

```
select s.stock_name, sum(o.num_ordered) as orders
from stock_decs s join orders o
on s.stock_id = o.stock_id
group by s.stock_name order by orders desc
```

100 %

Результаты | Сообщения

	stock_name	orders
1	LapTop	9
2	Computer	7
3	Flascard	6
4	Keyboard	6
5	iPhone12	4
6	Mouse	4
7	Printer	2
8	Monitor	2
9	HeadPhones	2
10	PS5	1

```
select c.costumer_name as Costumer, count(d.order_id) as orders
from costumers c join detail_order d
on c.costumer_id = d.costumer_id
group by c.costumer_name order by orders desc
```

100 %

Результаты | Сообщения

	Costumer	orders
1	Joseph Ross	4
2	Bonnie Morton	3
3	Alice Paul	2
4	Kenneth Hampton	2
5	Vivien Glenn	1

```
select s.stock_name, (i.sell_price - b.buy_price) as profit
from stock_decs s join in_stock i on s.stock_id = i.stock_id
join buyer_stock b on b.stock_id = i.stock_id
order by profit desc
```

100 %

Результаты | Сообщения

	stock_name	profit
1	Computer	20000
2	LapTop	17500
3	iPhone12	12200
4	Printer	12000
5	Monitor	4000
6	Keyboard	4000
7	Mouse	2400
8	PS5	2100
9	Flascard	1900
10	HeadPhones	800

```
select i.buyer_name, count(b.stock_id) as items
from buyer_info i join buyer_stock b
on i.buyer_id = b.buyer_id
group by i.buyer_name order by items desc
```

100 %

Результаты | Сообщения

	buyer_name	items
1	Technics	3
2	Uni-Care	2
3	Digitals	2
4	Greentest	1
5	OptMob	1
6	Vestidots	1

```
select b.buyer_name as buyer, s.stock_name as stock, p.purchase_date as purchase
from purchase_date p join buyer_info b on p.buyer_id = b.buyer_id
join stock_decs s on p.stock_id = s.stock_id
```

100 %

Результаты | Сообщения

	buyer	stock	purchase
1	Uni-Care	Keyboard	2021-01-01
2	Uni-Care	HeadPhones	2021-01-05
3	Greentest	Flascard	2021-01-03

```
select c.costumer_name as costumer, c.costumer_address as address
from costumers c join detail_order d on c.costumer_id = d.costumer_id
join orders o on o.order_id = d.order_id
join stock_decs s on s.stock_id = o.stock_id
where s.stock_name like 'LapTop' and d.paid not like 'F'
```

100 %

Результаты | Сообщения

	costumer	address
1	Joseph Ross	Belarus, Minsk
2	Kenneth Hampton	Kazakhstan, Almaty

END TERM PROJECT (task description)

1. Создайте процедуру закупа необходимого товара на складе. Процедура проверяет наличие товара в магазине и, если его количество близко к нулю, выводится соответствующее сообщение о необходимости закупа товара с его наименованием. Заканчивающиеся товары помещаются в отдельную таблицу. Прикрепленный к таблице триггер сработает, когда весь закуп будет завершен и запасы магазина вновь будут восполнены.

2. Процедура скидочных акций на некоторые виды товаров в магазине. Один раз в месяц (а именно 3его числа каждого нового месяца) определенная группа товаров попадает под скидочную акцию, причем на каждый товар действует своя скидка: от 10 до 70%. В следующем месяце поменяется и наименование продуктов, попавших под скидки, и их количество. Триггер срабатывает автоматически при наступлении 3его числа нового месяца и обновляет цены на товар с учетом скидочных акций.

3. Создайте процедуру наличия бонусных карт у покупателей. С каждой покупки на бонусную карту начисляются бонусы в виде 1% от суммы покупки. По мере накопления не менее 5000 бонусов покупатель может либо полностью оплатить ими покупку, либо обменять их на подарок, предлагаемый администрацией магазина. Срок действия бонусной карты – не более одного года. Как только количество бонусов на карте достигает десяти тысяч (срабатывает триггер), покупатель обязан потратить их в течение трех дней, в противном случае они безвозвратно сгорят.

4. Процедура расчета прибыли магазина за месяц. Данная процедура должна проверить все возможные доходы, расходы и убытки, с учетом выплаты заработной платы сотрудникам магазина, затратами на закуп товара, скидочными тарифами, бонусными картами и тд, чтобы в конечном итоге вывести ежемесячную прибыль магазина. Затем 50% от чистой прибыли

остаются в обороте деятельности магазина, тогда как вторая половина поступает на счет директора магазина.

5. Создайте процедуру выполнения онлайн-заказов с доставкой товара на дом. Все звонки, сделанные в магазин, фиксируются в соответствующей таблице, после чего заказу присваивается идентификатор “В процессе выполнения”. Надпись на идентификаторе меняется каждый раз в зависимости от хода выполнения заказа: “Поиск товара” – “Сбор заказа” – “Отправка заказа” – “Заказ выполнен”. Прикрепленный триггер удалит из соответствующей таблицы товаров те, которые были взяты для сбора заказа.

6. Авторская процедура + триггер.

Каждая процедура также должна включать в себя хотя бы одну созданную вами функцию.

Procedure 1

1. Создайте процедуру закупа необходимого товара на складе. Процедура проверяет наличие товара в магазине и, если его количество близко к нулю, выводится соответствующее сообщение о необходимости закупа товара с его наименованием. Заканчивающиеся товары помещаются в отдельную таблицу. Прикрепленный к таблице триггер сработает, когда весь закуп будет завершен и запасы магазина вновь будут восполнены.

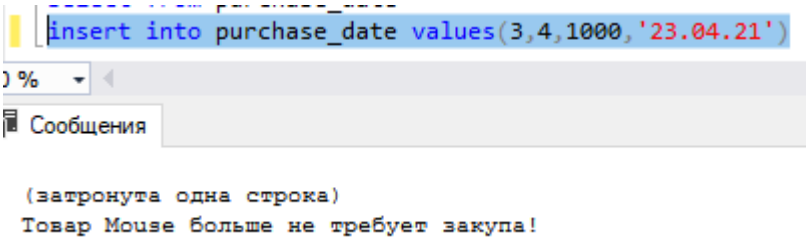
```
create table to_buy(
    stock_id integer,
    constraint pk_to_buy primary key(stock_id),
    stock_name varchar(10)
)

create index ix_stock_q on in_stock(stock_q)

create trigger purchase_done
on purchase_date
after insert, update as
begin
    update in_stock set stock_q = stock_q + (select quantity from inserted)
    where stock_id = (select stock_id from inserted)
    delete from to_buy where stock_id = (select stock_id from inserted)
end
```

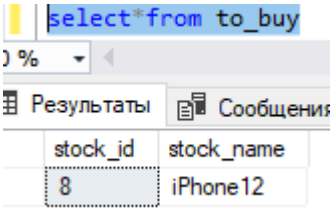
```
create trigger to_buy_insert_delete
on to_buy
after insert, delete as
begin
    declare @action varchar(20)
    set @action = (case when exists(select*from inserted)
                        then 'insert'
                        when exists(select*from deleted)
                        then 'delete'
                    end)
    declare @name varchar(10)
    if @action = 'insert'
    begin
        set @name = (select stock_name from inserted)
        print concat('Товар ',@name ,' требует закупа!')
    end
    else begin
        set @name = (select stock_name from deleted)
        print concat('Товар ',@name ,' больше не требует закупа!')
    end
end
```

```
create function stock_name
(@id integer)
returns varchar(10) as
begin
    declare @name varchar(10) =
    (select stock_name from in_stock where stock_id = @id)
    return(@name)
end

create procedure stock_to_buy as
begin
    declare @a integer = (select max(stock_id) from in_stock)
    declare @i integer = 1
    while @i <= @a
    begin
        declare @quantity integer = (select stock_q from in_stock where stock_id = @i)
        if @quantity <= 100 and @i not in (select stock_id from to_buy)
        begin
            insert into to_buy values(@i,dbo.stock_name(@i))
        end
    set @i = @i+1
    end
end
```

```
create function stock_name
(@id integer)
returns varchar(10) as
begin
    declare @name varchar(10) =
    (select stock_name from in_stock where stock_id = @id)
    return(@name)
end

create procedure stock_to_buy as
begin
    declare @a integer = (select max(stock_id) from in_stock)
    declare @i integer = 1
    while @i <= @a
    begin
        declare @quantity integer = (select stock_q from in_stock where stock_id = @i)
        if @quantity <= 100 and @i not in (select stock_id from to_buy)
        begin
            insert into to_buy values(@i,dbo.stock_name(@i))
        end
    set @i = @i+1
    end
end
```


```
create function stock_name
(@id integer)
returns varchar(10) as
begin
    declare @name varchar(10) =
    (select stock_name from in_stock where stock_id = @id)
    return(@name)
end

create procedure stock_to_buy as
begin
    declare @a integer = (select max(stock_id) from in_stock)
    declare @i integer = 1
    while @i <= @a
    begin
        declare @quantity integer = (select stock_q from in_stock where stock_id = @i)
        if @quantity <= 100 and @i not in (select stock_id from to_buy)
        begin
            insert into to_buy values(@i,dbo.stock_name(@i))
        end
    set @i = @i+1
    end
end
```





Procedure 2

1. Процедура скидочных акций на некоторые виды товаров в магазине. Один раз в месяц (а именно 3его числа каждого нового месяца) определенная группа товаров попадает под скидочную акцию, причем на каждый товар действует своя скидка: от 10 до 70%. В следующем месяце поменяется и наименование продуктов, попавших под скидки, и их количество. Триггер срабатывает автоматически при наступлении 3его числа нового месяца и обновляет цены на товар с учетом скидочных акций.

```sql
/*2*/
create table discounts(
    stock_id integer,
    constraint fk_discounts foreign key(stock_id) references in_stock(stock_id),
    discount integer
)

create function new_price(@stock_id integer, @discount integer)
returns decimal(10,2) as
begin
    declare @percent decimal(5,2) = @discount*0.01
    declare @price decimal(10,2) =
    (select (sell_price - sell_price*@percent) from in_stock where stock_id = @stock_id)
    return @price
end

create procedure set_discount as
```

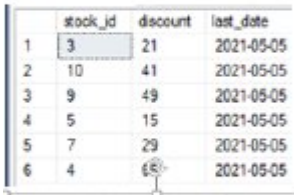
```sql
create procedure set_discount as
begin
    declare @today date = convert(date, getdate())
    declare @last_day date = (select distinct last_date from discounts)
    if day(@today) = 28 and datediff(day, @last_day, @today) >= 28
    begin
        declare @max integer = (select max(stock_id) from discounts)
        declare @i integer = 1
        while @i <= @max
        begin
            if @i in (select stock_id from discounts)
            begin
                update in_stock set sell_price =
                (select dbo.old_price(@i, discount) from discounts where stock_id = @i)
                where stock_id = @i
            end
            set @i = @i +1
        end
        delete from discounts
    end
    else select 'Discounts cannot be set today!'
end
```

```sql
create function old_price(@stock_id integer, @discount integer)
returns integer
as begin
    declare @new_discount decimal(10,2) = 1-(@discount*0.01)
    declare @old_price integer = (select sell_price/@new_discount from in_stock where stock_id = @stock_id)
    return @old_price
end
```

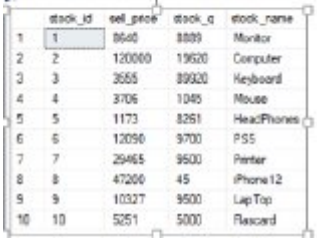
```
    end
end

create trigger new_discounts
on discounts
after delete as
begin
    declare @today date = convert(date, getdate())
    declare @max integer = (select max(stock_id) from in_stock)
    declare @amount integer = floor(RAND()*(@max-1+1))+1
    declare @i integer = 1
    while @i <= @amount
    begin
        declare @stock integer = floor(RAND()*(@max-1+1))+1
        declare @discount integer = floor(RAND()*(70-10+1))+10
        if @stock not in (select stock_id from discounts)
```

```
        declare @discount integer = floor(RAND()*(70-10+1))+10
        if @stock not in (select stock_id from discounts)
        begin
            insert into discounts values (@stock, @discount, @today)
            update in_stock set sell_price = dbo.new_price(@stock, @discount) where stock_id = @stock
        end
        set @i = @i+1
    end
end

exec set_discount
select*from discounts
select*from in_stock
update discounts set last_date = '04.04.21'

select stock_id, dbo.new_price(stock_id, 10) from in_stock
```

	stock_id	sell_price	stock_q	stock_name
1	1	8640	8889	Monitor
2	2	120000	15620	Computer
3	3	4500	89920	Keyboard
4	4	10800	1045	Mouse
5	5	1380	8261	HeadPhones
6	6	12090	9700	PS5
7	7	41500	9500	Printer
8	8	47200	45	iPhone12
9	9	20250	9500	LapTop
10	10	8900	5000	Flascard



	stock_id	discount	last_date
1	3	21	2021-05-05
2	10	41	2021-05-05
3	9	49	2021-05-05
4	5	15	2021-05-05
5	7	29	2021-05-05
6	4	65	2021-05-05



	stock_id	sell_price	stock_q	stock_name
1	1	8640	8889	Monitor
2	2	120000	19620	Computer
3	3	3655	89920	Keyboard
4	4	9706	1045	Mouse
5	5	1173	8261	HeadPhones
6	6	12090	9700	PS5
7	7	29465	9500	Printer
8	8	47200	45	iPhone12
9	9	10327	9500	LapTop
10	10	5251	5000	Flascard

Procedure 3

Создайте процедуру наличия бонусных карт у покупателей. С каждой покупки на бонусную карту начисляются бонусы в виде 1% от суммы покупки. По мере накопления не менее 5000 бонусов покупатель может либо полностью оплатить ими покупку, либо обменять их на подарок, предлагаемый администрацией магазина. Срок действия бонусной карты – не более одного года. Как только количество бонусов на карте достигает десяти тысяч (срабатывает триггер), покупатель обязан потратить их в течение трех дней, в противном случае они безвозвратно сгорят.

```
create procedure check_bonus as
begin
    declare @id integer = 1
    declare @max integer = (select max(costumer_id) from bonus_cards)
    declare @date date = convert(date, getdate())
    while @id <= @max
    begin
        if @id in (select costumer_id from bonus_cards where bonuses >= 10000 and datediff(day, last_bonus, @date) >= 3)
        begin
            update bonus_cards set bonuses = 0
            print concat('Бонусы пользователя ', dbo.cost_name(@id), ' обнулены!')
        end
        if @id in (select costumer_id from bonus_cards where datediff(year, open_date, @date) >= 1)
        begin
            delete from bonus_cards where costumer_id = @id
            print concat('Срок действия бонусной карты пользователя ', dbo.cost_name(@id), ' истек!')
        end
    set @id = @id+1
    end
end
```

```
select*from detail_order
create table bonus_cards(
    costumer_id integer,
    constraint pk_bonus primary key(costumer_id),
    constraint fk_bonus foreign key(costumer_id) references costumers(costumer_id),
    bonuses integer,
    open_date date,
    last_bonus date
)

create table presents(
    present_id integer identity(1,1),
    constraint pk_presents primary key(present_id),
    present_name varchar(50),
    present_price integer
)

insert into presents values('PowerBank', 7000)
insert into presents values('Headphones', 3000)
insert into presents values('SmartTag', 5000)
insert into presents values('Laptop Bag', 1200)
insert into presents values('Lamp', 4500)
insert into presents values('FlashCard', 2700)

create table presents_bonus(
    id integer identity(1,1),
    costumer_id integer,
    constraint pk_present_bonus primary key(id),
    constraint fk_present_bonus foreign key(costumer_id) references costumers(costumer_id),
    present_id integer,
    constraint fk_bonus_present foreign key(present_id) references presents(present_id)
```

```
create function calculate_bonuses (@id integer)
returns decimal(10,2) as
begin
    declare @bonus decimal(10,2) =
    (select sum(total_price*0.001) from detail_order
    where costumer_id = @id)
    return @bonus
end

create trigger bonus_update
on detail_order
after insert, update as
begin
    declare @id integer = (select costumer_id from inserted)
    declare @status varchar(5) = (select Status from detail_order where order_id = (select order_id from inserted))
    if @status = 'D'
    begin
        if @id in (select costumer_id from bonus_cards)
        begin
            update bonus_cards set last_bonus = (select date_ordered from inserted) where costumer_id = @id
            update bonus_cards set bonuses = dbo.calculate_bonuses(@id) where costumer_id = @id
        end
        else begin
            declare @date date = (select date_ordered from inserted)
            insert into bonus_cards values(@id,dbo.calculate_bonuses(@id), @date,@date)
        end
    end
end
```

```
create trigger presents_bonus_trade
on presents_bonus
after insert as
begin
    declare @trans integer = (select id from inserted)
    declare @id integer = (select costumer_id from inserted)
    declare @present integer = (select present_id from inserted)
    declare @bonus integer = (select bonuses from bonus_cards where costumer_id = @id)
    declare @price integer = (select present_price from presents where present_id = @present)
    if @bonus >= 5000 and @bonus >= @price
    begin
        update bonus_cards set bonuses = bonuses - @price
        where costumer_id = @id
        print ('Обмен бонусов на подарки прошел успешно!')
    end
    else begin
        delete from presents_bonus where id = @trans
        print ('Недостаточно бонусов!')
    end
end

create function cost_name (@id integer)
returns varchar(20) as
begin
    declare @name varchar(20) = (select costumer_name from costumers where costumer_id = @id)
    return @name
end
```

```
create procedure pay_by_bonuses (@order_id integer) as
begin
    declare @total integer = (select total_price from detail_order where order_id = @order_id)
    declare @costumer integer = (select costumer_id from detail_order where order_id = @order_id)
    declare @bonuses integer = (select bonuses from bonus_cards where costumer_id = @costumer)
    if @bonuses >= 5000 and @bonuses >= @total
    begin
        update bonus_cards set bonuses = bonuses - @total where costumer_id = @costumer
        update detail_order set total_price = 0 where order_id = @order_id
        update detail_order set paid = 'T' where order_id = @order_id
        print concat('Покупка номер ', @order_id, ' оплачена бонусами!')
    end
    else begin
        print 'Недостаточно бонусов!'
    end
end

create procedure presents_trade (@costumer_id integer, @present_id integer) as
begin
    insert into presents_bonus values(@costumer_id, @present_id)
end
```

```
create trigger bonus_messenges
on bonus_cards
after update as
begin
    declare @id integer = (select costumer_id from inserted)
    declare @bonus integer = (select bonuses from inserted)
    declare @day date = (select dateadd(day, 3, last_bonus) from inserted)
    if @bonus >= 5000 and @bonus < 10000
    begin
        print concat('Пользователь ', dbo.cost_name(@id), ' может начать использовать свои бонусы!')
    end
    if @bonus >= 10000
    begin
        print concat('Пользователь ', dbo.cost_name(@id), ' должен использовать бонусы до ', @day)
    end
end

create index ix_bonus on bonus_cards(bonuses)
```

```
update bonus_cards set open_date = '02.05.2020' where costumer_id = 1
exec check_bonus
```

```
%
Сообщения

(затронута одна строка)
Срок действия бонусной карты пользователя Kenneth Hampton истек!
```

	costumer_id	bonuses	open_date	last_bonus
1	2	327	2021-01-02	2021-02-12
2	3	38159	2021-01-03	2021-05-03
3	4	837	2021-01-04	2021-03-05
4	5	56	2021-01-06	2021-03-06

```
exec online_order @costumer_id = 4
```

0 %

Сообщения

```
Новый заказ с id 22
```

```
insert into orders values (22, 5, 10)
insert into orders values (22, 2, 10)
update detail_order set status = 'D' where order_id = 22
```

100 %

Результаты Сообщения

	costumer_id	bonuses	open_date	last_bonus
1	2	327	2021-01-02	2021-02-12
2	3	38159	2021-01-03	2021-05-03
3	4	2057	2021-01-04	2021-05-03
4	5	56	2021-01-06	2021-03-06

```
update bonus_cards set bonuses = 5470 where costumer_id = 4
```

%

Сообщения

```
Пользователь Joseph Ross может начать использовать свои бонусы!
```

```
update bonus_cards set bonuses = 15470 where costumer_id = 4
```

Сообщения

```
Пользователь Joseph Ross должен использовать бонусы до 2021-05-06
```

```
update bonus_cards set bonuses = 5470 where costumer_id = 4
```

Сообщения

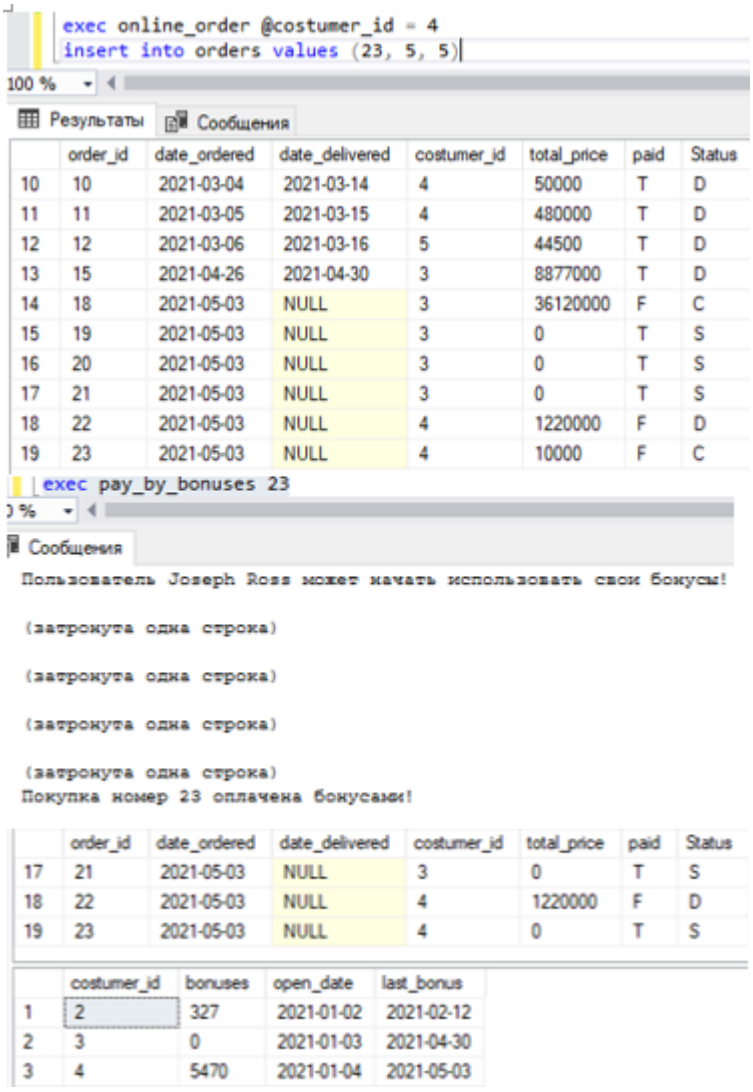
```
Пользователь Joseph Ross может начать использовать свои бонусы!
```

```
update bonus_cards set last_bonus = '30.04.2021' where costumer_id = 3
exec check_bonus
```

Сообщения

```
(затронута одна строка)
Бонусы пользователя Vivien Glenn обнулены!
```

	costumer_id	bonuses	open_date	last_bonus
1	2	327	2021-01-02	2021-02-12
2	3	0	2021-01-03	2021-04-30
3	4	15470	2021-01-04	2021-05-03
4	5	56	2021-01-06	2021-03-06



```
exec online_order @costumer_id = 4
insert into orders values (23, 5, 5)
```

100 %

Результаты | Сообщения

	order_id	date_ordered	date_delivered	costumer_id	total_price	paid	Status
10	10	2021-03-04	2021-03-14	4	50000	T	D
11	11	2021-03-05	2021-03-15	4	480000	T	D
12	12	2021-03-06	2021-03-16	5	44500	T	D
13	15	2021-04-26	2021-04-30	3	8877000	T	D
14	18	2021-05-03	NULL	3	36120000	F	C
15	19	2021-05-03	NULL	3	0	T	S
16	20	2021-05-03	NULL	3	0	T	S
17	21	2021-05-03	NULL	3	0	T	S
18	22	2021-05-03	NULL	4	1220000	F	D
19	23	2021-05-03	NULL	4	10000	F	C

```
exec pay_by_bonuses 23
```

Сообщения

```
Пользователь Joseph Ross может начать использовать свои бонусы!

(затронута одна строка)

(затронута одна строка)

(затронута одна строка)

(затронута одна строка)
Покупка номер 23 оплачена бонусами!
```

	order_id	date_ordered	date_delivered	costumer_id	total_price	paid	Status
17	21	2021-05-03	NULL	3	0	T	S
18	22	2021-05-03	NULL	4	1220000	F	D
19	23	2021-05-03	NULL	4	0	T	S

	costumer_id	bonuses	open_date	last_bonus
1	2	327	2021-01-02	2021-02-12
2	3	0	2021-01-03	2021-04-30
3	4	5470	2021-01-04	2021-05-03

Procedure 4

1. Процедура расчета прибыли магазина за месяц. Данная процедура должна проверить все возможные доходы, расходы и убытки, с учетом выплаты заработной платы сотрудникам магазина, затратами на закуп товара, скидочными тарифами, бонусными картами и тд, чтобы в конечном итоге вывести ежемесячную прибыль магазина. Затем 50% от чистой прибыли остаются в обороте деятельности магазина, тогда как вторая половина поступает на счет директора магазина.

```
674
675  create function fn_month_name_to_number (@monthname varchar(25))
676  returns int as
677  begin
678  declare @monthno as int;
679  select @monthno =
680  case @monthname
681  when 'January' then 1
682  when 'February' then 2
683  when 'March' then 3
684  when 'April' then 4
685  when 'May' then 5
686  when 'June' then 6
687  when 'July' then 7
688  when 'August' then 8
689  when 'September' then 9
690  when 'October' then 10
691  when 'November' then 11
692  when 'December' then 12
693  end
694  return @monthno
695  end
696
697  select dbo.fn_month_name_to_number('May')
```

Results Messages

	(No column name)
1	5

```
602  create table accounts(
603      acc_id int PRIMARY KEY,
604      acc_name VARCHAR(25),
605      wealth int
606  )
607  delete from accounts
608
609  insert into accounts VALUES(1,'Boss',3000,'April')
610  insert into accounts VALUES(2,'Shop',30000,'April')
611
612  alter table accounts
613  add last_upd VARCHAR(25)
614
615  select *from accounts
```

Results Messages

	acc_id	acc_name	wealth	last_upd
1	1	Boss	2100	May
2	2	Shop	29100	May

```
616
617 create table employees(
618     emp_id int IDENTITY(1,1) primary key,
619     emp_wage int
620 )
621
622 insert into employees VALUES(1000)
623 insert into employees VALUES(800)
624 insert into employees VALUES(800)
625 insert into employees VALUES(800)
626
627 select * from employees
628
```

Results Messages

	emp_id	emp_wage
1	1	1000
2	2	800
3	3	800
4	4	800

```
629  create FUNCTION profit_fun
630  (@mes varchar(25))
631  RETURNS @T table(profit_ int,mon_ VARCHAR(25))
632  as
633  BEGIN
634      declare @emp_wages int=(select sum(emp_wage) from employees);
635      with dinam as
636      (select sum(
637          case
638              when total_price=0 then -sell_price*num_ordered
639              else (sell_price-buy_price)*num_ordered
640          end
641          )-@emp_wages as profit,DATENAME(month,date_ordered) as mon
642      from detail_order do
643      join orders o on do.order_id=o.order_id
644      join buyer_stock bs on bs.stock_id=o.stock_id
645      join in_stock ins on ins.stock_id=o.stock_id
646      GROUP by DATENAME(month,date_ordered)
647      HAVING DATENAME(month,date_ordered)=@mes
648      )
649      INSERT into @T
650      select profit, mon
651      from dinam
652      RETURN
653  END
654
655  select * from dbo.profit_fun('May')
656
```

Results Messages

	profit_	mon_
1	-1800	May

```
659  create PROCEDURE month_profit
660  @mes VARCHAR(25)
661  as
662  BEGIN
663  declare @p int=(select profit_ from dbo.profit_fun(@mes))
664  update accounts
665  set wealth=wealth+0.5*@p,
666  last_upd=@mes
667  where dbo.fn_month_name_to_number(@mes)>dbo.fn_month_name_to_number(last_upd)
668  and @p is not null
669  END
670
671  exec month_profit 'May'
672
673  SELECT * from accounts
674
```

Results Messages

	acc_id	acc_name	wealth	last_upd
1	1	Boss	2100	May
2	2	Shop	29100	May

Procedure 5

1. Создайте процедуру выполнения онлайн-заказов с доставкой товара на дом. Все звонки, сделанные в магазин, фиксируются в соответствующей таблице, после чего заказу присваивается идентификатор “В процессе выполнения”. Надпись на идентификаторе меняется каждый раз в зависимости от хода выполнения заказа: “Поиск товара” – “Сбор заказа” – “Отправка заказа” – “Заказ выполнен”. Прикрепленный триггер удалит из соответствующей таблицы товаров те, которые были взяты для сбора заказа.

```
exec online_order @costumer_id = 3
```

%

Сообщения

Новый заказ с id 15

```
select*from detail_order
exec online_order @costumer_id = 3
```

Результаты | Сообщения

order_id	date_ordered	date_delivered	costumer_id	total_price	paid	Status
12	2021-03-06	2021-03-16	5	44500	T	D
15	2021-04-26	NULL	3	NULL	F	F

stock_id	sell_price	stock_q	stock_name
1	12000	9000	Monitor
2	120000	20000	Computer
3	15000	90000	Keyboard
4	10900	1045	Mouse
5	2000	8300	HeadPhones
6	12090	10000	PS5
7	50000	9500	Printer

```
insert into orders values(15,1,100)
insert into orders values(15,6,300)
```

Результаты | Сообщения

stock_id	sell_price	stock_q	stock_name
1	12000	8900	Monitor
2	120000	20000	Computer
3	15000	90000	Keyboard
4	10900	1045	Mouse
5	2000	8300	HeadPhones
6	12090	9700	PS5
7	50000	9500	Printer

order_id	date_ordered	date_delivered	costumer_id	total_price	paid	Status
12	2021-03-06	2021-03-16	5	44500	T	D
15	2021-04-26	NULL	3	4827000	F	C

```
update detail_order set paid = 'T' where order_id = 15
```

Результаты | Сообщения

order_id	date_ordered	date_delivered	costumer_id	total_price	paid	Status
12	2021-03-06	2021-03-16	5	44500	T	D
15	2021-04-26	NULL	3	8877000	T	S

```
update detail_order set date_delivered = '30.04.21' where order_id = 15
```

Результаты | Сообщения

order_id	date_ordered	date_delivered	costumer_id	total_price	paid	Status
12	2021-03-06	2021-03-16	5	44500	T	D
15	2021-04-26	2021-04-30	3	8877000	T	D

```
create table order_status(
    status varchar(5),
    constraint pk_status primary key(status),
    status_desc varchar(100)
)

insert into order_status values('F', 'Поиск заказа')
insert into order_status values('C', 'Сбор заказа')
insert into order_status values('S', 'Отправка заказа')
insert into order_status values('D', 'Заказ выполнен')

select*from orders
select*from detail_order
alter table detail_order add Status varchar(5) default 'F'
update detail_order set Status = 'D'
alter table detail_order add constraint fk_do_status foreign key(Status) references order_status(status)

create function total_price (@id integer)
returns integer as
begin
    declare @total integer = (select sum(o.num_ordered*s.sell_price)
    from orders o join in_stock s on o.stock_id = s.stock_id
    where o.order_id = @id)
    return @total
end
```

```
create procedure online_order
@costumer_id integer
as
begin
    declare @date date = convert(date, getdate())
    insert into detail_order values(@date, null, @costumer_id, null, 'F', 'F')
end

create trigger paid_status
on detail_order
after update as
begin
    declare @id integer = (select order_id from inserted)
    declare @paid varchar(1) = (select paid from detail_order where order_id = @id)
    declare @date date = (select date_delivered from detail_order where order_id = @id)
    if @paid = 'T'
        update detail_order set Status = 'S' where order_id = @id
    if @date is not null
        update detail_order set Status = 'D' where order_id = @id
end

create trigger stock_update
on orders
after insert as
begin
    declare @id integer = (select order_id from inserted)
        update detail_order set Status = 'C' where order_id = @id
        update detail_order set total_price = dbo.total_price(@id) where order_id = @id
        update in_stock set stock_q = stock_q - (select num_ordered from inserted)
        where stock_id = (select stock_id from inserted)
end
```

```
create procedure online_order
@costumer_id integer
as
begin
    declare @date date = convert(date, getdate())
    insert into detail_order values(@date, null, @costumer_id, null, 'F', 'F')
end

create trigger paid_status
on detail_order
after update as
begin
    declare @id integer = (select order_id from inserted)
    declare @paid varchar(1) = (select paid from detail_order where order_id = @id)
    declare @date date = (select date_delivered from detail_order where order_id = @id)
    if @paid = 'T'
        update detail_order set Status = 'S' where order_id = @id
    if @date is not null
        update detail_order set Status = 'D' where order_id = @id
end

create trigger stock_update
on orders
after insert as
begin
    declare @id integer = (select order_id from inserted)
        update detail_order set Status = 'C' where order_id = @id
        update detail_order set total_price = dbo.total_price(@id) where order_id = @id
        update in_stock set stock_q = stock_q - (select num_ordered from inserted)
        where stock_id = (select stock_id from inserted)
end


create trigger order_id
on detail_order
after insert as
begin
    declare @id integer = (select order_id from inserted)
    print concat('Новый заказ с id ',@id)
end
```

Procedure 6 (Авторская процедура) - Процедура доставки оплаченных товаров

```
create FUNCTION fn_check_delivery_date
(@order_id int)
RETURNS DATE
as
BEGIN
    DECLARE @date DATE
    set @date=(select date_delivered from detail_order where order_id=@order_id)
    RETURN @date
END
GO

create FUNCTION fn_check_paid
(@order_id int)
RETURNS VARCHAR(1)
as
BEGIN
    DECLARE @paid VARCHAR(1)
    set @paid=(select paid from detail_order where order_id=@order_id)
    RETURN @paid
END
GO

select dbo.fn_check_delivery_date(1)
select dbo.fn_check_paid(1)
GO

create procedure proc_deliver
(@order_id INT)
as
BEGIN
    DECLARE @date date=dbo.fn_check_delivery_date(@order_id)
    DECLARE @paid varchar(1)=dbo.fn_check_paid(@order_id)
    if @date is NULL and @paid='T'
    BEGIN
        update detail_order
        set date_delivered=GETDATE()
        where order_id=@order_id
    END
```

Procedure 7 - процедура отмены доставки

```
create function items_ordered (@order_id integer)
returns integer
as begin
    declare @items integer = (select count(*) from orders where order_id = @order_id)
    return @items
end
GO

create procedure cancel_order (@order_id integer)
as begin
    declare @total integer = (select total_price from detail_order where order_id = @order_id)
    declare @status varchar(5) = (select Status from detail_order where order_id = @order_id)
    declare @id integer = (select costumer_id from detail_order where order_id = @order_id)
    if @order_id in (select order_id from detail_order)
    begin
        if @total > 0
        begin
            declare @items integer = dbo.items_ordered(@order_id)
            while @items >0
            begin
                declare @item integer = (select max(stock_id) from orders where order_id = @order_id)
                declare @quantity integer = (select num_ordered from orders where order_id = @order_id and stock_id = @item)
                update in_stock set stock_q = stock_q + @quantity where stock_id = @item
                delete from orders where order_id = @order_id and stock_id = @item
                set @items = @items -1
            end
            if @status = 'D'
            begin
                update bonus_cards set bonuses = bonuses - @total*0.001 where costumer_id = @id
            end
        end
        else select 'Cannot cancel the order which was paid by bonuses'
    end
    else select 'Order does not exist'
end
```

Триггер:

```
create trigger delete_order
on orders
after delete
as begin
    declare @order_id integer = (select order_id from deleted)
    declare @items integer = dbo.items_ordered(@order_id)
    if @items = 0
    begin
        delete from detail_order where order_id = @order_id
    end
end
GO
```

Procedure 8 - cancel order

```
create procedure cancel_order (@order_id integer)
as begin
    declare @total integer = (select total_price from detail_order where order_id = @order_id)
    declare @status varchar(5) = (select Status from detail_order where order_id = @order_id)
    declare @id integer = (select costumer_id from detail_order where order_id = @order_id)
    if @order_id in (select order_id from detail_order)
    begin
        if @total > 0
        begin
            declare @items integer = dbo.items_ordered(@order_id)
            while @items >0
            begin
                declare @item integer = (select max(stock_id) from orders where order_id = @order_id)
                declare @quantity integer = (select num_ordered from orders where order_id = @order_id and stock_id = @item)
                update in_stock set stock_q = stock_q + @quantity where stock_id = @item
                delete from orders where order_id = @order_id and stock_id = @item
                set @items = @items -1
            end
            if @status = 'D'
            begin
                update bonus_cards set bonuses = bonuses - @total*0.001 where costumer_id = @id
            end
        end
        else select 'Cannot cancel the order which was paid by bonuses'
    end
    else select 'Order does not exist'
end
```

Trigger:

```
create trigger delete_order
on orders
after delete
as begin
    declare @order_id integer = (select order_id from deleted)
    declare @items integer = dbo.items_ordered(@order_id)
    if @items = 0
    begin
        delete from detail_order where order_id = @order_id
    end
end
GO
```

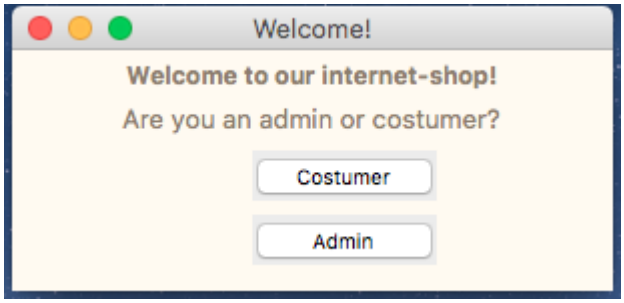
Procedure 9 - make discount 7/7

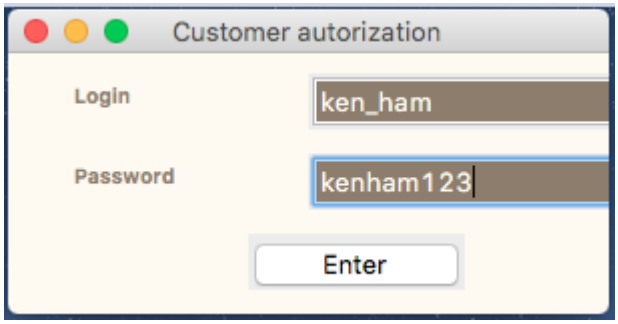
```
create PROCEDURE proc_make_discount (@order_id int, @percent float, @stock_id int)
as
BEGIN
    DECLARE @all_prod_num int;
    DECLARE @prod_num int;
    DECLARE @cheapest int;
    DECLARE @discount FLOAT;
    DECLARE @total_price INT;
    set @all_prod_num=(select prod_num from dbo.fn_check_for_discount(@order_id))
    set @prod_num=(select num_ordered from orders where order_id=@order_id and stock_id=@stock_id)
    set @total_price=(select total_price from detail_order WHERE order_id=@order_id)
    if @all_prod_num>=7 and @total_price>0
    BEGIN
        set @cheapest=(select cheapest from dbo.fn_check_for_discount(@order_id))
        set @discount=@prod_num*(@percent*@cheapest)
        UPDATE detail_order set total_price=total_price-@discount where order_id=@order_id
        select 'And, You have discount thanks to buying 7 products'
    END
    else
    BEGIN
        select 'You can have discount by ordering 7 or more items!'
    END
end
GO
```

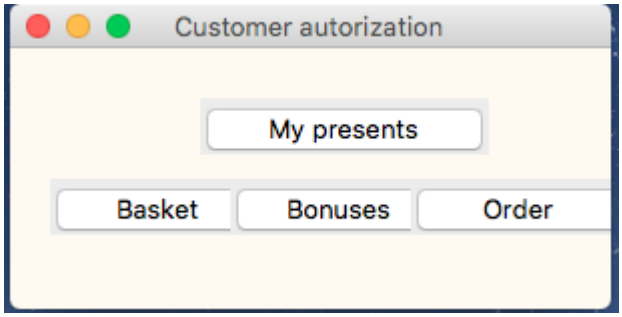
Trigger:

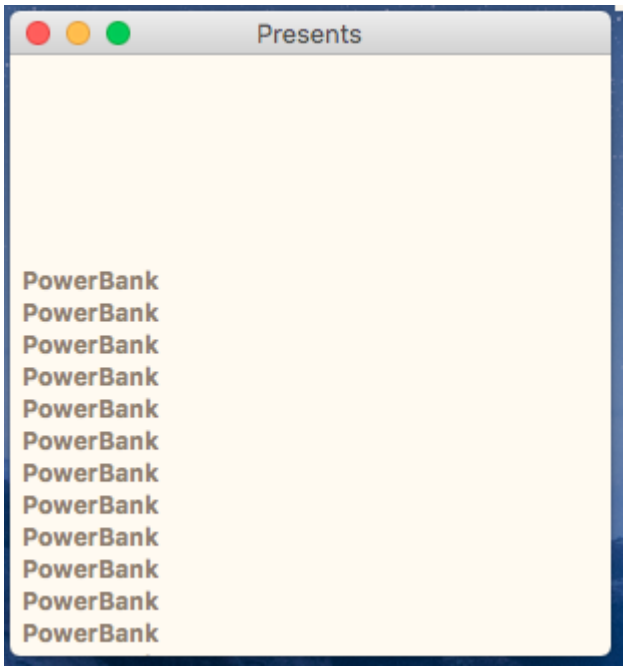
```
create TRIGGER trig_make_discount
on orders
after INSERT
as
BEGIN
    DECLARE @ord_id INT;
    DECLARE @st_id int;
    set @st_id=(select stock_id from inserted)
    set @ord_id=(select order_id from inserted)
    set NOCOUNT on
    EXEC proc_make_discount @order_id=@ord_id, @percent=0.07, @stock_id=@st_id
END
```

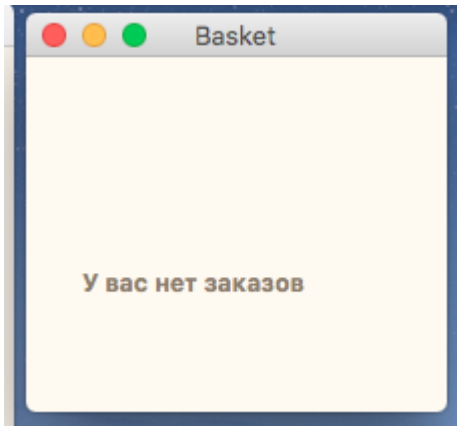
4) The interface:

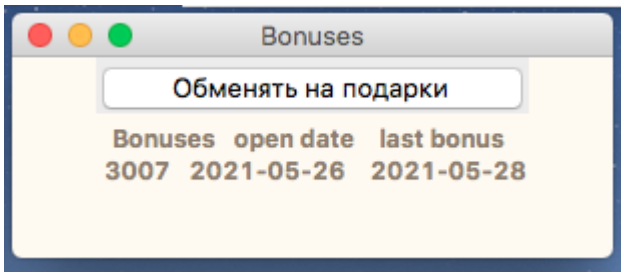


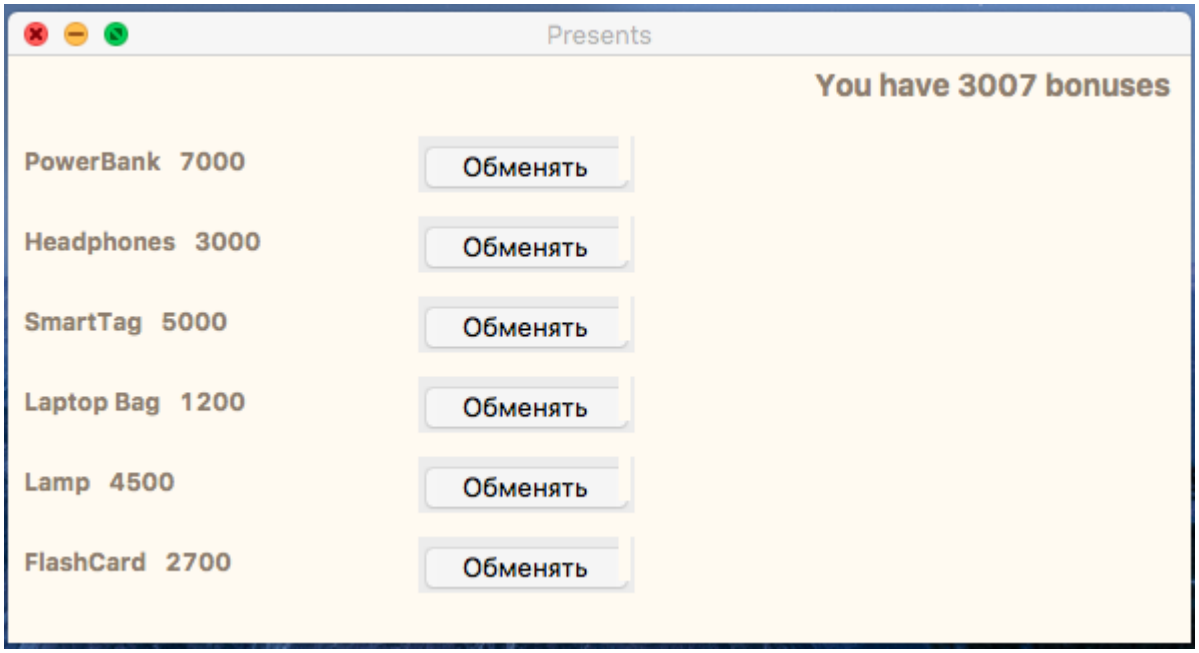


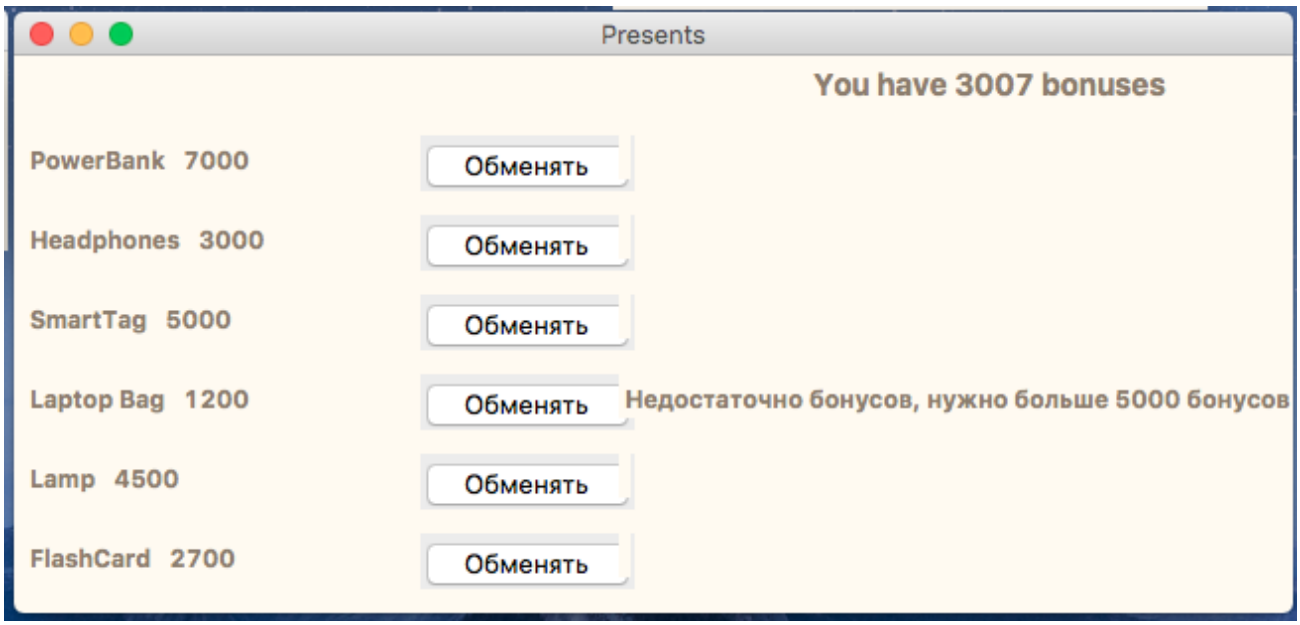


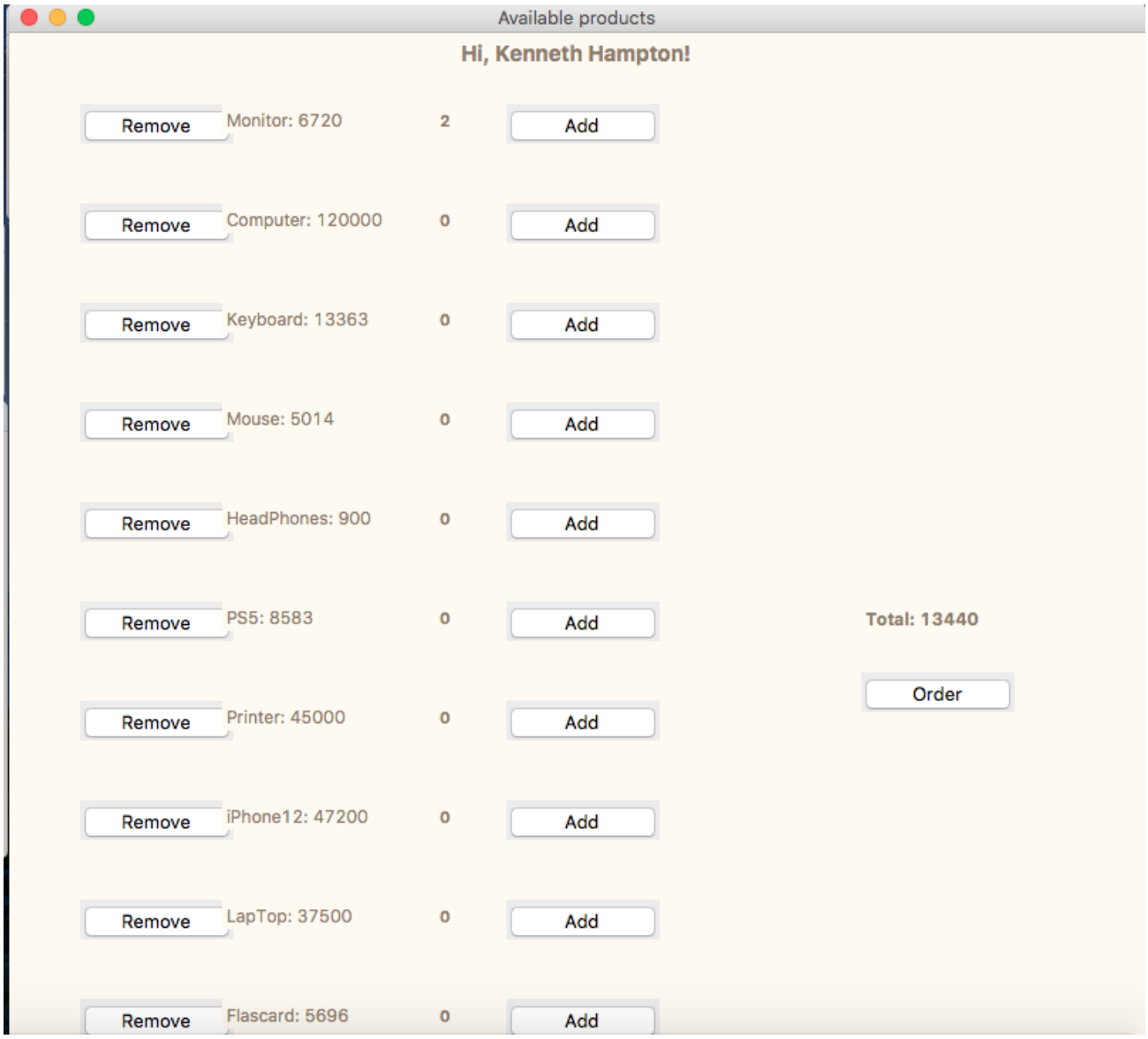

Presents
PowerBank
PowerBank
PowerBank
PowerBank
PowerBank
PowerBank
PowerBank
PowerBank
PowerBank
PowerBank
PowerBank
PowerBank

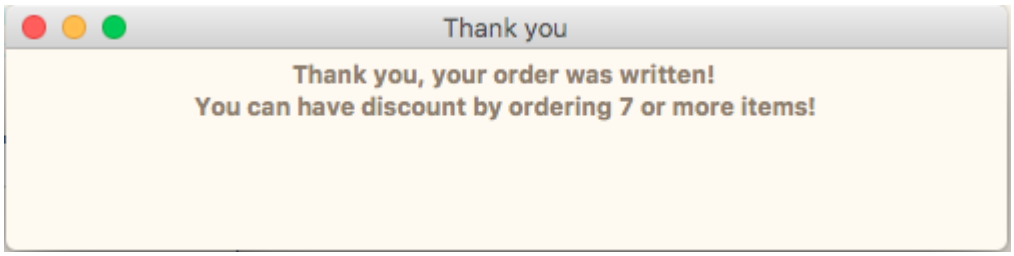

Basket
У вас нет заказов

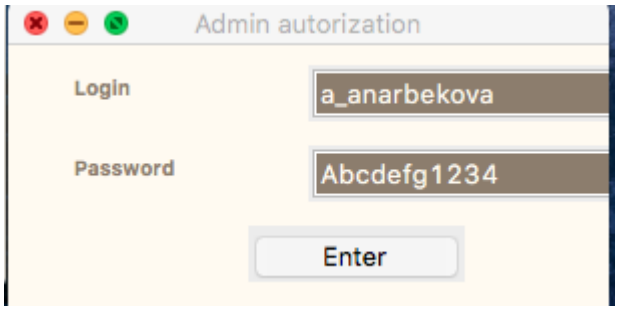

Bonuses
Обменять на подарки
Bonuses open date last bonus
3007 2021-05-26 2021-05-28

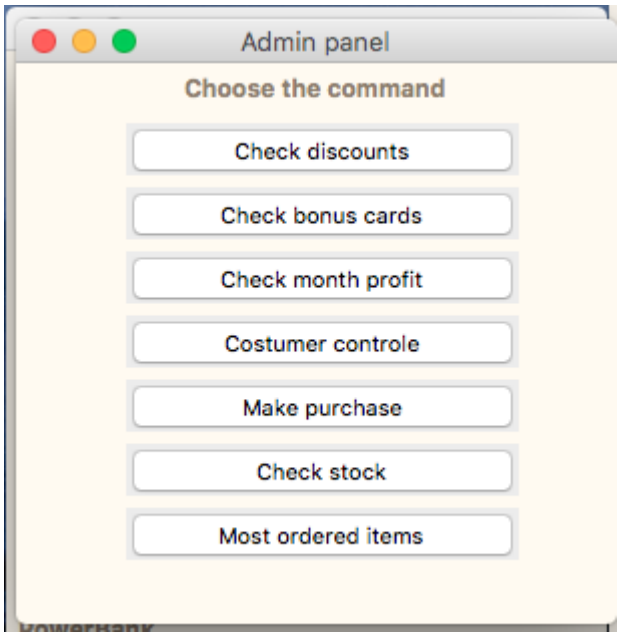

Presents
You have 3007 bonuses
PowerBank 7000
Обменять
Headphones 3000
Обменять
SmartTag 5000
Обменять
Laptop Bag 1200
Обменять
Lamp 4500
Обменять
FlashCard 2700
Обменять

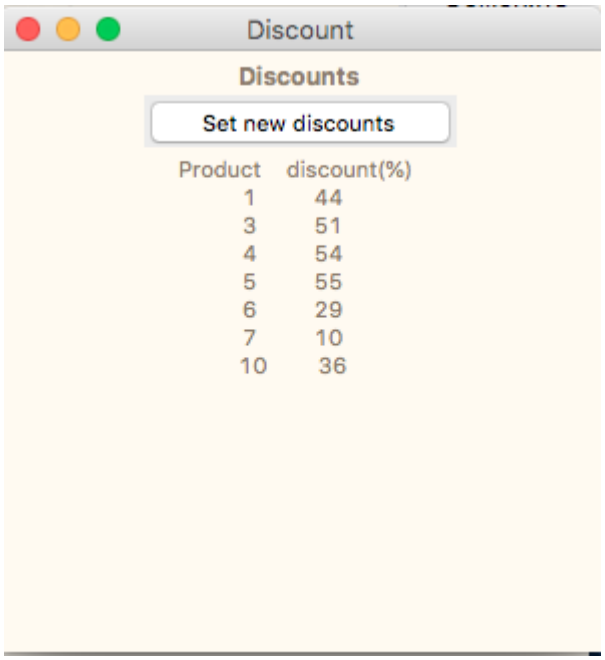

Presents
You have 3007 bonuses
PowerBank 7000
Обменять
Headphones 3000
Обменять
SmartTag 5000
Обменять
Laptop Bag 1200
Обменять
Недостаточно бонусов, нужно больше 5000 бонусов
Lamp 4500
Обменять
FlashCard 2700
Обменять

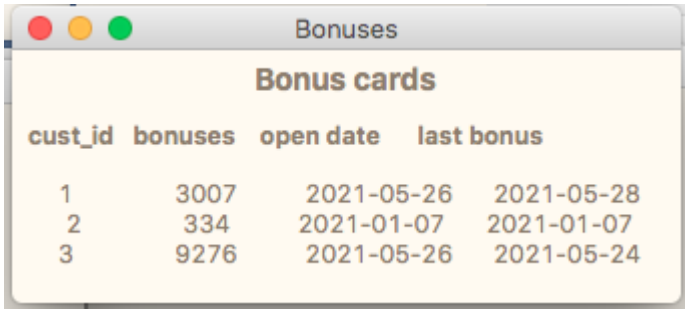

Available products
Hi, Kenneth Hampton!
Remove
Monitor: 6720
2
Add
Remove
Computer: 120000
0
Add
Remove
Keyboard: 13363
0
Add
Remove
Mouse: 5014
0
Add
Remove
HeadPhones: 900
0
Add
Remove
PS5: 8583
0
Add
Total: 13440
Order
Remove
Printer: 45000
0
Add
Remove
iPhone12: 47200
0
Add
Remove
LapTop: 37500
0
Add
Remove
Flascard: 5696
0
Add

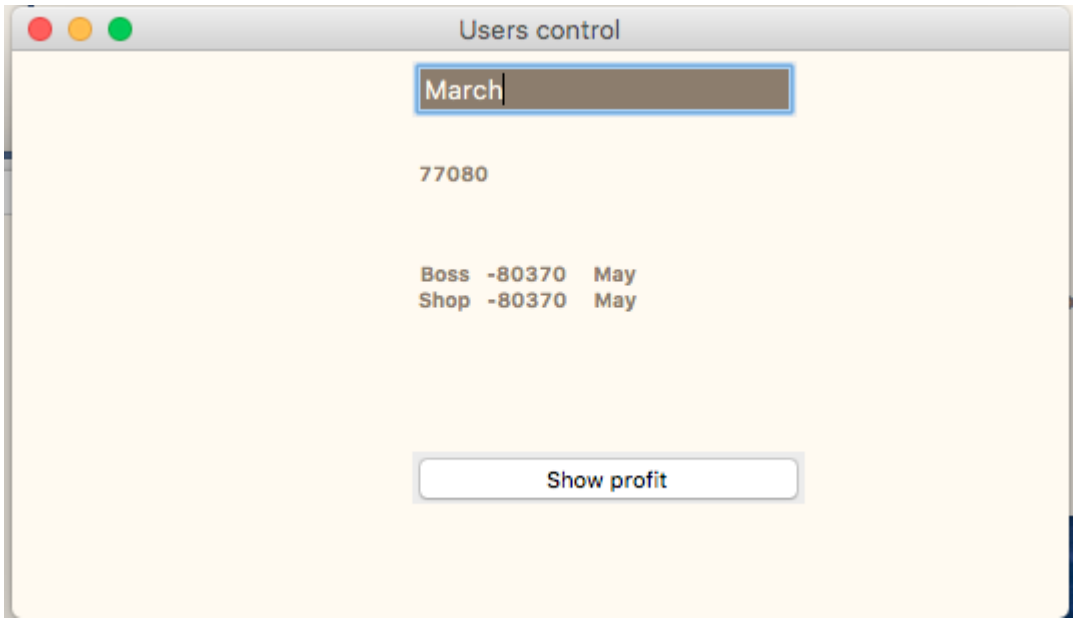

Thank you
Thank you, your order was written!
You can have discount by ordering 7 or more items!

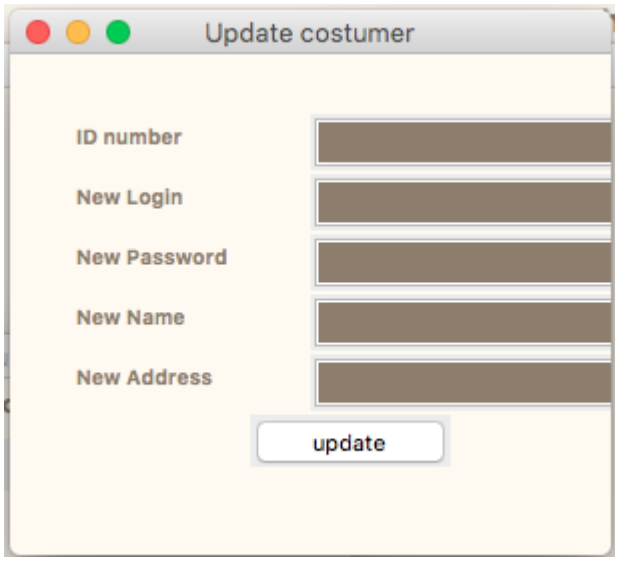

Admin autorization
Login
a_anarbekova
Password
Abcdefg1234
Enter

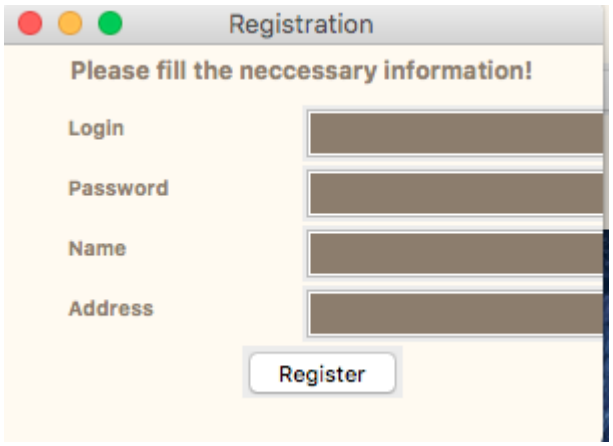

Admin panel
Choose the command
Check discounts
Check bonus cards
Check month profit
Costumer controle
Make purchase
Check stock
Most ordered items

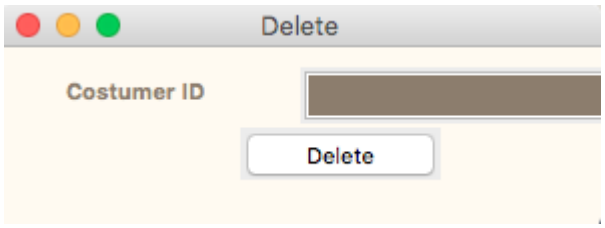

Discount
Discounts
Set new discounts
Product discount(%)
1 44
3 51
4 54
5 55
6 29
7 10
10 36

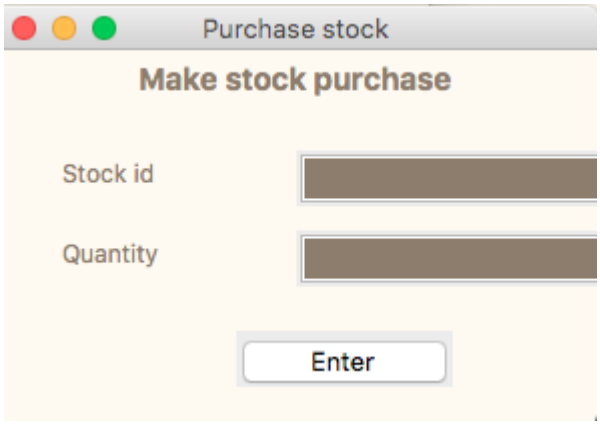

Bonuses
Bonus cards
cust_id bonuses open date last bonus
1 3007 2021-05-26 2021-05-28
2 334 2021-01-07 2021-01-07
3 9276 2021-05-26 2021-05-24

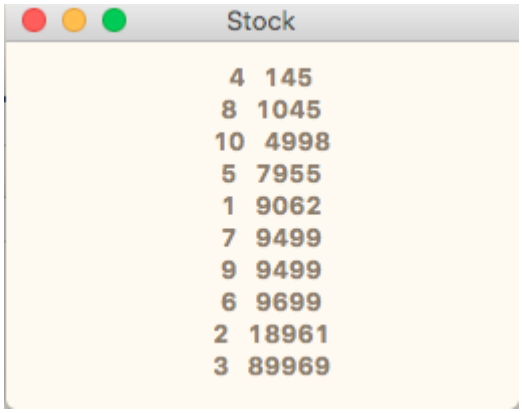

Users control
March
77080
Boss -80370 May
Shop -80370 May
Show profit

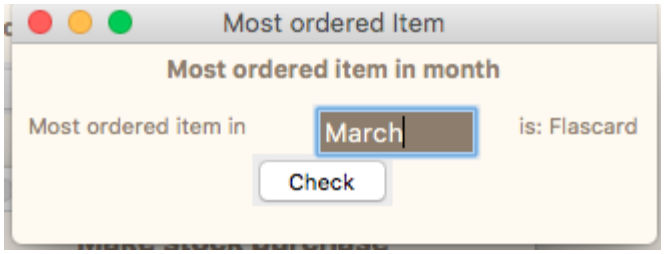

Update costumer
ID number
New Login
New Password
New Name
New Address
update


Registration
Please fill the neccessary information!
Login
Password
Name
Address
Register


Delete
Costumer ID
Delete


Purchase stock
Make stock purchase
Stock id
Quantity
Enter


Stock
4 145
8 1045
10 4998
5 7955
1 9062
7 9499
9 9499
6 9699
2 18961
3 89969


Most ordered Item
Most ordered item in month
Most ordered item in
March
is: Flascard
Check