**Задание по SQL №1:**

Выбрать все счета, которые оплачены на полную сумму. Cчета оплачиваются в порядке очередности по мере поступления средств и могут быть оплачены как частично несколькими платежами, так и одним платежом несколько счетов.

declare @clients table (id int, name varchar(20))

declare @invoices table (id int, summa numeric(18,2), clientId int)

declare @payments table (clientId int, payment numeric(18,2))

insert @clients (id, name)

values

(1, '1sr'),

(2, '2nd'),

(3, '3rd'),

(4, '4th')

insert @invoices (id, summa, clientId)

values

(1, 10, 1),

(2, 15, 1),

(3, 20, 1),

(4, 25, 1),

(5, 12, 2),

(6, 14, 2),

(7, 200, 2),

(8, 100, 3),

(9, 200, 3)

insert @payments (clientId, payment)

values

(1, 30),

(2, 500),

(3, 100),

(4, 20)

-- SOLUTION:

-- Define the resulting table

declare @paidInvoices table (id int, summa numeric(18,2), clientId int)

-- Define property for calculation of sum of client's invoices

declare @ClientIvoicesSum numeric(18,2)

declare @previousClientID as int

-- Define cursor for iterating through invoices with the client's payments

declare Invoice\_Cursor cursor for

select i.\*, p.payment

from @invoices as i

join @payments as p on i.clientId = p.clientId;

-- Define properties for our cursor

declare @InvoiceId as int

declare @InvoiceSumma as numeric(18,2)

declare @ClientID as int

declare @ClientPayment as numeric(18,2)

open Invoice\_Cursor;

-- Move to the first element of our cursor

fetch next from Invoice\_Cursor into @InvoiceId, @InvoiceSumma, @ClientID, @ClientPayment;

set @ClientIvoicesSum = 0;

set @previousClientID = 0;

while (@ClientIvoicesSum <= @ClientPayment or @previousClientID = @ClientID) and @@FETCH\_STATUS = 0

begin

set @ClientIvoicesSum = @ClientIvoicesSum + @InvoiceSumma;

if @ClientIvoicesSum > @ClientPayment

begin

set @previousClientID = @ClientID;

fetch next from Invoice\_Cursor into @InvoiceId, @InvoiceSumma, @ClientID, @ClientPayment;

-- If the new row in the cursor is for new client, then reset @ClientIvoicesSum

if @previousClientID != @ClientID

begin

set @ClientIvoicesSum = 0;

continue;

end;

-- Continue skipping the invoices for this customer when their sum is more than client's payment

fetch next from Invoice\_Cursor into @InvoiceId, @InvoiceSumma, @ClientID, @ClientPayment;

continue;

end;

-- Add fully paid invoice to the results table

insert into @paidInvoices

select \* from @invoices

where id = @InvoiceId;

set @previousClientID = @ClientID;

-- Move to the next invoice after processing the current one

fetch next from Invoice\_Cursor into @InvoiceId, @InvoiceSumma, @ClientID, @ClientPayment;

-- If the new row in the cursor is for new client, then reset @ClientIvoicesSum

if @previousClientID != @ClientID

begin

set @ClientIvoicesSum = 0

end

end;

close Invoice\_Cursor;

deallocate Invoice\_Cursor;

select \* from @paidInvoices;

**Задание по SQL №2:**

В таблице находятся числа из диапазона от 1 до 10, в произвольном порядке. Каждое число встречается один раз.   
Необходимо написать запрос, который из всего диапазона данных [min, max] вернёт все числа отсутствующие в таблице.   
Использовать спецфункции нельзя. Решение должно быть наиболее грамотным, понятным и универсальным.   
Проверить решение на таблице, в которой максимальное число в диапазоне 1 000 000.

IF Object\_id('tempdb..#test\_table') IS NOT NULL

DROP TABLE #test\_table

CREATE TABLE #test\_table

(

id INT

)

GO

INSERT INTO #test\_table

VALUES (1), (2), (8), (4), (9), (7), (3), (10) --<-- Отсутствуют числа 5 и 6

GO

SELECT \*

FROM #test\_table

GO

**Решение задания по SQL №2:**

-- SOLUTION:

-- Define block with all numbers from "#test\_table" table from MIN to MAX, including the missing ones

WITH numbers AS

(

SELECT MIN(id) AS num

FROM #test\_table

UNION ALL

SELECT num + 1

FROM numbers

WHERE num < ANY(SELECT id FROM #test\_table)

)

-- Select those numbers from "numbers" block that are not presented in "#test\_table" table

SELECT n.num

FROM #test\_table AS t

RIGHT JOIN numbers AS n ON t.id = n.num

WHERE t.id is null;