**Отчет по примерным задачам к экзамену**

**Лешук Дмитрий, ПОИБМС 7-2**

Задания выглядят следующим образом – рисунок 1.1.

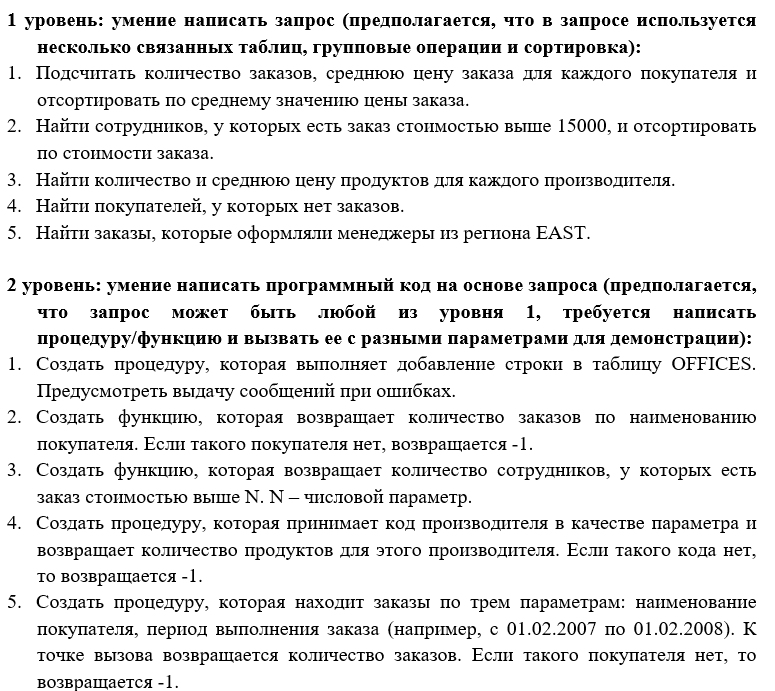


Рисунок 1.1 – Условия заданий

Начнем с заданий первого уровня. Эти пять заданий включают в себя написание select-запросов, в которых используются соединения таблиц, сортировка, группировка, агрегатные функции.

Первое задание первого уровня представлено на рисунке 1.2. Используем соединение таблиц, группировку для выполнения агрегатных функций, а также сортировку, как и сказано в условии

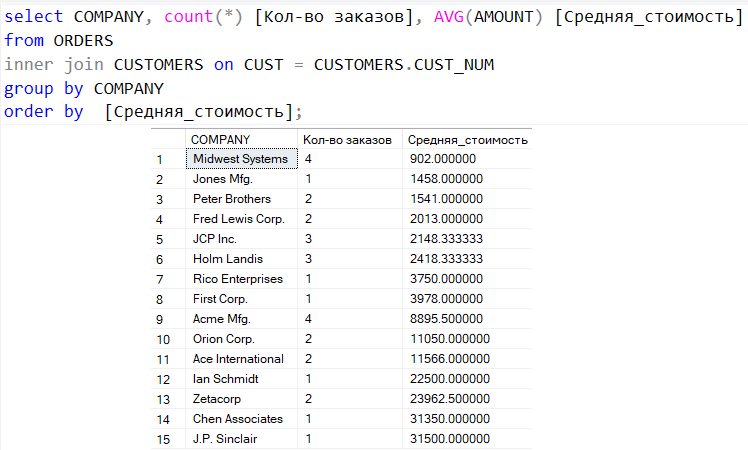


Рисунок 1.2 – Уровень 1, задание 1

Второе задание первого уровня – рисунок 1.3. Аналогично используем соединение таблиц и сортировку, а также секцию WHERE для задания условия для столбца с ценой заказа,

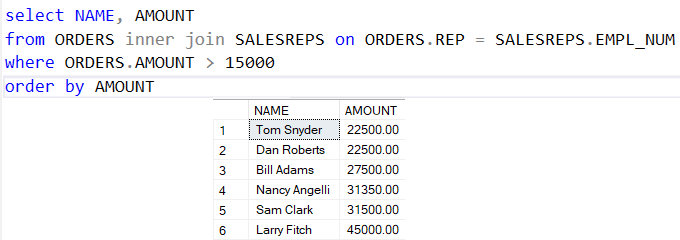


Рисунок 1.3 – Уровень 1, задание 2

Третье задание – рисунок 1.4. Запрос выполняется над одной таблицей, используем группировку и агрегатные функции для рассчёта количества и средней цены продуктов для каждого производителя.

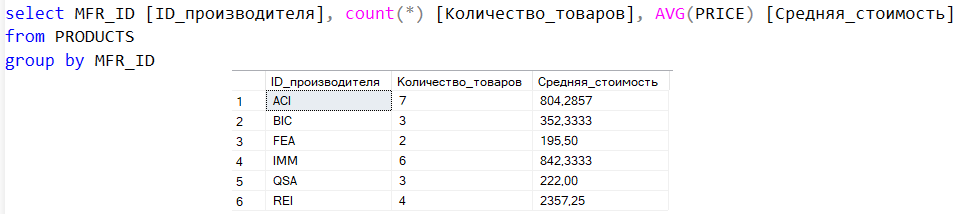


Рисунок 1.4 – Уровень 1, задание 3

Четвертое задание уже немного отличается от предыдущих, так как в нём нужно использовать коррелированный подзапрос и оператор not exists – рисунок 1.5.

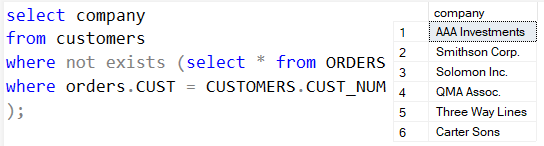


Рисунок 1.5 – Уровень 1, задание 4

Запрос пятого задания включает в себя соединение трёх таблиц и секцию WHERE – рисунок 1.6.

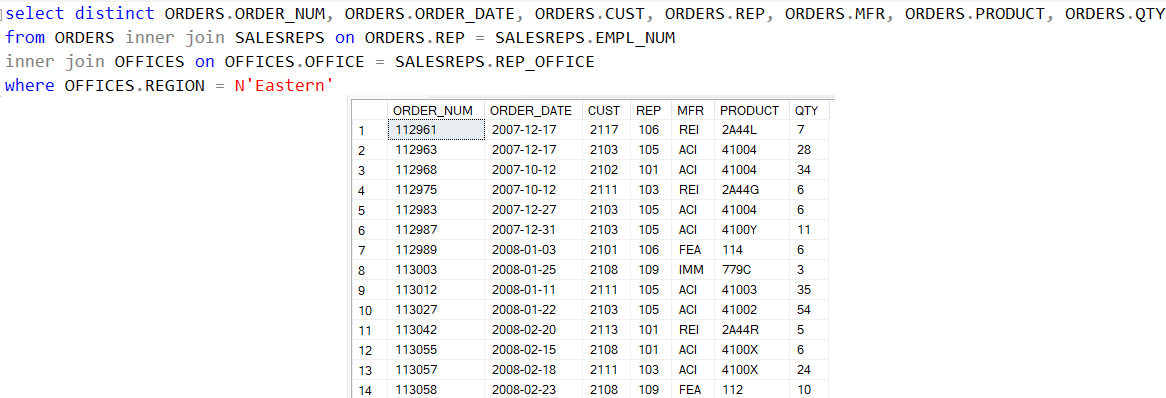


Рисунок 1.6 – Уровень 1, задание 5

Задания второго уровня включают в себя создание процедур и функций, реализующие определенные операции и возвращающие какие-либо значения.

В первом задании необходимо создать процедуру, принимающую 6 параметров, значения которых в последствии будут вставлены в таблицу OFFICES. Также добавим обработку ошибок, как и требуется в условии. Код представлен в листинге 1.1. Результат выполнения – на рисунке 1.7.

create procedure dbo.TASK1

@off int, @city varchar(15), @region varchar(10), @mgr int, @targ decimal(9, 2), @sales decimal(9, 2)

as

begin try

insert OFFICES values(@off, @city, @region, @mgr, @targ, @sales)

end try

begin catch

print N'Номер ошибки: ' + cast(error\_number() as varchar(6));

print N'Сообщение: ' + error\_message()

print N'Уровень: ' + cast(error\_severity() as varchar(6));

print N'Метка: ' + cast(error\_state() as varchar(8));

print N'Номер строки: ' + cast(error\_line() as varchar(8));

if ERROR\_PROCEDURE() is not null

print N'Имя процедуры:' + error\_procedure();

return -1;

end catch;

Листинг 1.1 – Уровень 2, задание 1

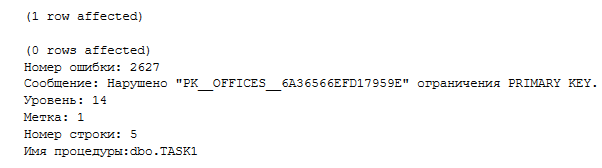


Рисунок 1.7 – Выполнение процедуры №1

Во втором задании необходимо создать функцию, возвращающее целочисленное значение с условным блоком if-else – листинг 1.2.

CREATE function TASK2TRUE(@company nvarchar(20)) returns int

as

begin

declare @rc int = 0;

IF not exists (select COMPANY from CUSTOMERS where COMPANY = @company)

set @rc = -1

else

set @rc = (select count(\*) from ORDERS inner join CUSTOMERS on ORDERS.CUST = CUSTOMERS.CUST\_NUM where COMPANY = @company)

return @rc

end;

Листинг 1.2 – Функция из второго задания, уровень 2

Проверим результат её выполнения как в select-запросе для всех компаний, так и для отдельно взятой – рисунок 1.8.

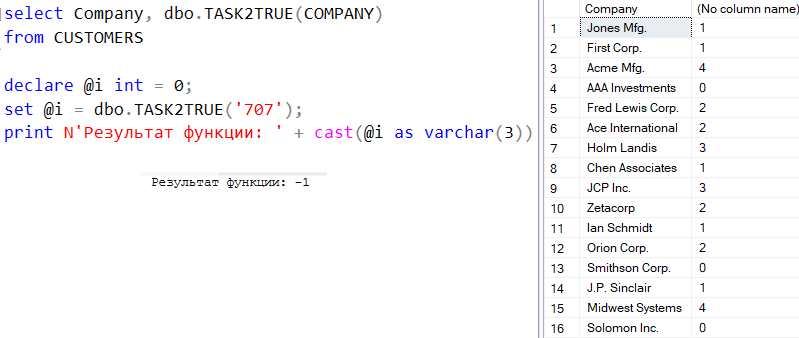


Рисунок 1.8 – Результат выполнения второго задания

Функция из третьего задания должна возвращать количество работников, удовлетворяющих условию, у которых есть заказ со стоимостью больше указанной – листинг 1.3. Результат выполнения – рисунок 1.9. Сразу после выполнения скрипта создания базы данных, в таблицe SALESREPS мы имеем всего 9 работников, которые обрабатывали заказы. Следовательно, при указании наименьшей цены мы получим все 9 работников.

alter function dbo.TASK3(@price decimal(9, 2)) returns int

as

begin

declare @rc int = 0

set @rc =

(select COUNT(employees) from (

select distinct EMPL\_NUM as [employees]

from SALESREPS inner join ORDERS

on orders.REP = SALESREPS.EMPL\_NUM

WHERE ORDERS.AMOUNT > @price

) as subquery)

return @rc

end;

Листинг 1.3 – Функция из третьего задания

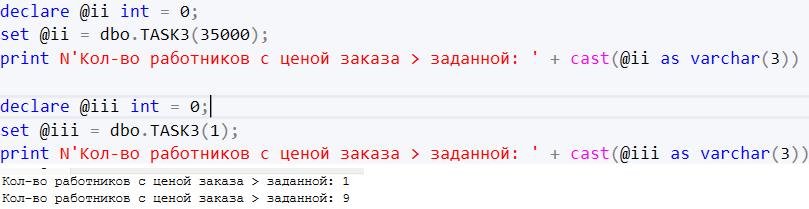


Рисунок 1.9 – Выполнение функции

В четвертом задании процедура работает только с одной таблицей – PRODUCTS и возвращает значение – листинг 1.4. Результат выполнения с определенным производителем и некорректными данными представлен на рисунке 1.10.

create procedure dbo.TASK4 @mfr varchar(20)

as

declare @rc int = 0

begin

IF not exists (select MFR\_ID from PRODUCTS where MFR\_ID = @MFR)

return -1

else

begin

set @rc = (select count(\*) from PRODUCTS where MFR\_ID = @mfr)

print N'Кол-во продуктов у производителя с кодом ' + @mfr + ': ' + cast(@rc as varchar(3))

return @rc

end

end;

Листинг 1.4 – Задание 4, уровень 2

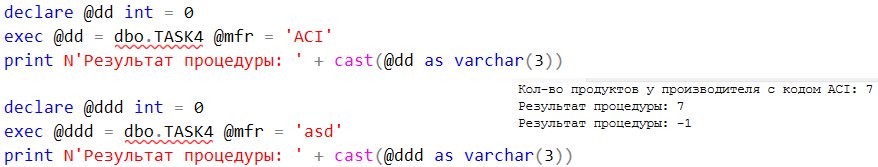


Рисунок 1.10 – Результат выполнения задания №4 второго уровня

В последнем задании процедура принимает один строковый параметр и два параметра типа date, также возвращает число, равное определенному кол-во строк, а в случае отсутствия строк, попадающих под условие, возвращается -1. Также в процедуре присутствует условный блок, как и в некоторых предыдущих заданиях такого же типа – листинг 1.5. Результат выполнения – рисунок 1.11.

create procedure dbo.TASK5 @cust varchar(20), @dstart date, @dend date

as

declare @rc int = 0

begin

IF not exists (select COMPANY from CUSTOMERS where COMPANY = @cust)

return -1

else

begin

set @rc = (select count(\*) from ORDERS inner join CUSTOMERS on ORDERS.CUST = CUSTOMERS.CUST\_NUM where COMPANY = @cust

AND ORDER\_DATE between @dstart AND @dend)

print N'Кол-во заказов: ' + cast(@rc as varchar(3))

return @rc

end

end;

Листинг 1.5 – Процедура из пятого задания

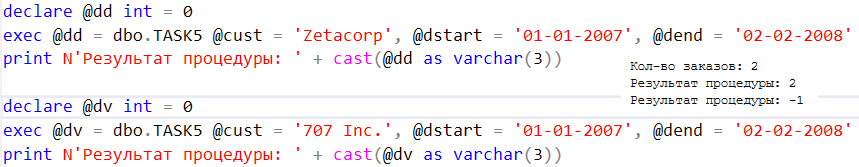


Рисунок 1.11 – Результат выполнения последней процедуры

**Вывод:**

В данном отчёте были разобраны первый и второй уровень примерных задач к экзамену по дисциплине «Базы данных», которые требуют от студента знаний в написании select-запросов и подзапросов, группировки, создании и использовании функций и процедур для решения типовых задач.