1. Таблица А имеет поля id и name и содержит следующие данные:

|  |  |
| --- | --- |
| **Table A** | |
| **id** | **name** |
| 1 | AAA |
| 2 | BBB |
| 2 | CCC |

Какой результат вернет каждый из следующих вариантов запроса?

Ответы желательно пояснить.

-- a

declare @name varchar(10); select @name = 'XXXX'

select @name = name from A where id = 0

print @name

go

-- b

declare @name varchar(10); select @name = 'XXXX'

select @name = name from A where id = 1

print @name

go

-- c

declare @name varchar(10); select @name = 'XXXX'

select @name = name from A where id = 2

print @name

go

-- d

declare @name varchar(10); select @name = 'XXXX'

set @name = (select name from A where id = 0)

print @name

go

-- e

declare @name varchar(10); select @name = 'XXXX'

set @name = (select name from A where id = 1)

print @name

go

-- f

declare @name varchar(10); select @name = 'XXXX'

set @name = (select name from A where id = 2)

print @name

go

--ответы @sqlmaster

-- a вернет XXXX так как @name = 'XXXX'

declare @name varchar(10); select @name = 'XXXX'

select @name = name from A where id = 0

print @name

go

-- b вернет AAA так как @name = name from A where id = 1

declare @name varchar(10); select @name = 'XXXX'

select @name = name from A where id = 1

print @name

go

-- c вернет CCC так как @name = name from A where id = 2

declare @name varchar(10); select @name = 'XXXX'

select @name = name from A where id = 2

print @name

go

-- d ничего, так как подзапрос (select name from A where id = 0) ничего не вернет

declare @name varchar(10); select @name = 'XXXX'

set @name = (select name from A where id = 0)

print @name

go

-- e вернет AAA так как подзапрос (select name from A where id = 1) вернет AAA

declare @name varchar(10); select @name = 'XXXX'

set @name = (select name from A where id = 1)

print @name

go

-- f ошибка так как (select name from A where id = 2) вернет 2 строки

declare @name varchar(10); select @name = 'XXXX'

set @name = (select name from A where id = 2)

print @name

go

1. Написать *один* запрос, который на основе данных таблицы A вернет следующие значения:
2. минимальное значение по столбцу amount;
3. максимальное значение по столбцу amount;
4. сумму по столбцу amount;
5. сумму всех положительных значений amount;
6. сумму всех отрицательных значений amount;
7. число уникальных значений в столбце amount.

select min(amount) min, max(amount) max, sum(amount) sum,

sum (case when amount > 0 then amount else 0 end) [>0],

sum (case when amount < 0 then amount else 0 end) [<0],

COUNT(DISTINCT amount)

from a

|  |  |  |
| --- | --- | --- |
|  | |  |
| **Table A** | |  |
| **id** | **amount** |  |
| 1 | 500 |  |
| 2 | 300 |  |
| 3 | 100 |  |
| 4 | -200 |  |
| 5 | 100 |  |
| 6 | 300 |  |
| 7 | -200 |  |
| 8 | -100 |  |

1. В таблице A (столбец id – integer identity primary key) было 2 строки. При выполнении следующего скрипта

insert A(name) values('СССC')

print @@identity

insert A(name) values('DDDD')

print @@identity

go

выводится результат:

3

2

Объясните, чем он может быть вызван, и как добиться правильного результата (4) во втором случае?

Состояние таблицы:

|  |  |  |
| --- | --- | --- |
| до вставки |  | после вставки |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table A** | |  | **Table A** | |
| **id** | **name** |  | **id** | **name** |
| 1 | AAAA |  | 1 | AAAA |
| 2 | BBBB |  | 2 | BBBB |
|  |  |  | 3 | CCCC |
|  |  |  | 4 | DDDD |

drop table a ;

CREATE TABLE A( [ID] [int] IDENTITY(1,1) NOT NULL, [name] [varchar](50));

insert into a(name) values ('AAA');

insert into a(name) values ('BBB');

insert A(name) values('СССC')

print @@identity

insert A(name) values('DDDD')

print @@identity

go

результат:

(1 row affected)

(1 row affected)

(1 row affected)

3

(1 row affected)

4

1. Таблица А содержит полностью дублирующие строки.

Написать запрос, который выберет такие записи.

Написать скрипт, который позволит избавиться от ненужных дублей (допускается модификация таблицы, использование других объектов, переименование и т.п.).

--выбираем дубли

select \* FROM

(

SELECT id,name,property,date, ROW\_NUMBER() OVER(ORDER BY id,name,property,date) AS RowNum

FROM A

)D

JOIN

(

SELECT id,name,property,date, ROW\_NUMBER() OVER(ORDER BY id,name,property,date) AS RowNum

FROM A

)E

on D.id = E.id and D.name = E.name and D.property = E.property and D.date = E.date

AND D.RowNum < E.RowNum

--удаляем дубли

delete D FROM

(

SELECT id,name,property,date, ROW\_NUMBER() OVER(ORDER BY id,name,property,date) AS RowNum

FROM A

)D

JOIN

(

SELECT id,name,property,date, ROW\_NUMBER() OVER(ORDER BY id,name,property,date) AS RowNum

FROM A

)E

on D.id = E.id and D.name = E.name and D.property = E.property and D.date = E.date

AND D.RowNum < E.RowNum

Пример таблицы

|  |  |  |  |
| --- | --- | --- | --- |
| **Table A** | | | |
| **id** | **Name** | **property** | **date** |
| 1 | AAA | 2 | 2020-10-11 |
| 2 | BBB | 0 | 2020-10-10 |
| 2 | BBB | 0 | 2020-10-10 |
| 3 | CCC | 3 | 2020-10-12 |
| 4 | DDD | 1 | 2020-10-12 |
| 4 | DDD | 1 | 2020-10-12 |
| 4 | DDD | 1 | 2020-10-12 |
| 5 | EEE | 2 | 2020-10-11 |
| 6 | FFF | 1 | 2020-10-13 |

1. Покупатель имеет бонусную карту, привязанную только к его номеру телефона. Покупатель может заменить бонусную карту по причине утери, либо какой-то другой. При этом все данные со старой карты клонируются на новую, включая номер телефона, а старая очищается. После этого старая карта считается свободной ***для следующей выдачи*** либо ***замены*** (переноса данных). Поскольку покупатель всегда имеет право поменять номер телефона и перепривязать бонусную карту к нему, идентифицируем покупателя на текущий момент как владельца той или иной бонусной карты.

Один покупатель не имеет права осуществлять более 5 переносов с карты на карту в год (@limit = 5).

Есть процедура замены, получающая на вход два номера карты – текущий (@old\_card) и новый (@new\_card).

Требуется написать скрипт для нее, определяющий, позволяется ли покупателю осуществить текущий перенос (достиг ли он лимита), если лимит достигнут, когда он сможет в следующий раз воспользоваться заменой карты.

Есть таблица переносов c карты на карту:

|  |  |  |
| --- | --- | --- |
| **Table cards\_transfer** | | |
| **old\_card** | **new\_card** | **dt** |
| 111 | 555 | 2020-01-09 |
| 222 | 223 | 2020-02-10 |
| 333 | 334 | 2020-03-11 |
| 444 | 222 | 2020-04-12 |
| 555 | 666 | 2020-05-12 |
| 666 | 777 | 2020-06-13 |
| 777 | 888 | 2020-07-14 |
| 888 | 000 | 2020-08-15 |
| 999 | 333 | 2020-09-16 |
| 223 | 111 | 2020-10-16 |

Таким образом, согласно данным в примере, владелец карты 000 не может осуществлять замену карты до 09.01.2021.

CREATE PROC change\_card @old\_card INT, @new\_card INT

AS

declare @dt date

set @dt=(select a.dt from cards\_transfer a join cards\_transfer b on a.new\_card=b.old\_card

join cards\_transfer c on b.new\_card=c.old\_card

join cards\_transfer d on c.new\_card=d.old\_card

join cards\_transfer e on d.new\_card=e.old\_card

where a.old\_card = @old\_card);

select case when @dt is null then 'можно менять карту' else (SELECT DATEADD(month, 12, @dt)) end;

--проверка и вывод даты возможной замены сделаны, далее логика добавления @new\_card в задании это не указано

exec change\_card 111,222

результат  
2021-01-09