1. მონიშნეთ არასწორი წინადადება

a. CTE - ში შეიძლება ახალი CTE იყოს განსაზღვრული

* 1. CTE - ს ტანში შეგვიძლია გამოვიყენოთ UNION,UNION ALL, EXPECT, INTERSECT
  2. CTE შეგვიძლია გამოვიყენოთ ვიუში
  3. სხვა სერვერზე არსებული ცხრილი შეგვიძლია გამოვიყენოთ CTE - ში

1. მონიშნეთ არასწორი წინადადება
2. CTE - ს შეიძლება მოყვებოდეს DELETE
3. CTE - ს შეიძლება მოყვებოდეს TRUNCATE
4. CTE - ს შეიძლება მოყვებოდეს UPDATE
5. CTE - ს შეიძლება მოყვებოდეს მხოლოდ DML ბრძანებები
6. მონიშნეთ არასწორი წინადადება პროცედურების შესახებ
7. პროცედურებში შეგვიძლია CTE - ების გამოყენება
8. პროცედურებში შეგვიძლია TRY...CATCH ის გამოყენება
9. პროცედურებში არ შეგვიძლია ლოკალური დროებითი ცხრილის გამოყენება
10. პროცედურებში შეგვიძლია While...End ის გამოყენება
11. Try - Catch ბლოკის გამოყენება შეგვიძლია
    1. პროცედურაში
    2. სკალარულ ფუნქციაში
    3. ცხრილურ ფუნქციაში
    4. ყველა პასუხი სწორია
12. როგორ შეგვიძლია დაუქომითებელი მონაცემების წაკითხვა სხვა სესიიდან (Query-დან) ?
    1. ცხრილზე NOLOCK ბრძანების მითითებით
    2. SET TRAN ISOLATION LEVEL READ UNCOMMITTED ბრძანების დაწერით სელექთამდე
    3. NOLOCK ით ან SET TRAN ISOLATION LEVEL READ UNCOMMITTED ბრძანების

დაწერით სელექთამდე

* 1. სხვა სესიიდან დაუქომითებელ მონაცემებს ვერ წავიკითხავთ

1. Case When ბრძანება შეგვიძლია გამოვიყენოთ
   1. SELECT, UPDATE, DELETE
   2. SET, IN
   3. WHERE, ORDER BY, GROUP BY
   4. ყველა ზემოთ ჩამოთვლილთან
2. ტრანზაქციის გამოყენება შეგვიძლია
   1. მხოლოდ პროცედურებში
   2. ვიუებში და პროცედურებში
   3. ცხრილურ ფუნქციებში და პროცედურებში
   4. სკალარულ ფუნქციებში და პროცედურებში
3. დროებითი ცხრილები შეგვიძლია გამოვიყენოთ
   1. მხოლოდ ცხრილურ ფუნქციებში და პროცედურებში
   2. მხოლოდ ვიუებსა და პროცედურებში
   3. მხოლოდ პროცედურებში
   4. მხოლოდ სკალარულ ფუნქციებსა და პროცედურებში
4. ცხრილში გვაქვს ID სვეტი, რომლის ზოგიერთი მნიშვნელობა არის ცარიელი (NULL), ქვემოთ მოცემული წინადადებებიდან რომელი იქნება სწორი
   1. ორივე დააბრუნებს ცხრილში სტრიქონების რაოდენობას
   2. ორივე დააბრუნებს ცხრილში არაცარიელი სტრიქონების რაოდენობას
   3. COUNT(\*) დააბრუნებს არაცარიელი სტრიქონების რაოდენობას, ხოლო COUNT(ID) მხოლოდ ყველა სტრიქონის რაოდენობას
   4. COUNT(\*) დააბრუნებს სტრიქონების რაოდენობას, მიუხედავად იმისა ცარიელი იქნება თუ შევსებული, ხოლო COUNT(ID) მხოლოდ არაცარიელი სტრიქონების რაოდენობას
5. რომელი ტიპის ფუნქციაში შეგვიძლია განვსაზღვროთ დასაბრუნებელი ცხრილის სტრუქტურა?
   1. Scalar
   2. Inline Table-valued
   3. MSTVF
   4. არც ერთში
6. ორ თარიღს შორის (20190101 , 20191013) დღეების რაოდენობას სწორად გამოიტანს შემდეგი ფორმულა
   1. SELECT DATEDIFF(dd,'20190101','20191013')
   2. SELECT DATEDIFF(dd,'20191013','20190101')
   3. SELECT DATEDIFF(ddd,'20191013','20190101')
   4. SELECT DATEDIFF(ww,'20190101','20191013')
7. COALESCE ფუნქცია გვიბრუნებს
   1. მხოლოდ რიცხვით მნიშვნელობას
   2. პირველივე რიცხვით მნიშვნელობას
   3. პირველივე არაცარიელ მნიშვნელობას
   4. არც ერთი პასუხი არ არის სწორი
8. რას დაგვიბრუნებს შემდეგი ფუნქცია



* 1. 1
  2. 0
  3. 5
  4. NULL

1. როდესაც გვაქვს 1:1 კავშირი, ასეთ შემთხვევაში მონაცემები უნდა ჩავწეროთ
   1. ორ ცხრილში, სადაც ერთში გვექნება პირველადი გადასაღები და მეორე ცხრილს დავუკავშირებთ მეორადი გასაღებით
   2. ერთ ცხრილში
   3. შემოვიღოთ კავშირის ცხრილი, რომელიც იქნება მესამე ცხრილი და დაუკავშირდება თითოეულ ცხრილს ცალ-ცალკე
   4. გააჩნია შემთხვევას
2. 101.163 რიცხვის დამრგვალებისას იგივე რიცხვის მიღება (101.163) რომელი ფუნქციით არის შესაძლებელი?
   1. Round()
   2. Floor()
   3. Ceiling()
   4. სამივე პასუხი სწორია
3. რა შედეგს დაგვიბრუნებს შემდეგი ბრძანება?



a. 4

b. 0

c. 9

d. 5

1. პირველ ცხრილში გვაქვს 3 ჩანაწერი (Row), მეორეში 11, რომელი ტიპით დაკავშირება დაგვიბრუნებს 33 ჩანაწერს
   1. FULL JOIN
   2. FULL JOIN - იმ შემთხვევაში თუ გადაბმის პირობაში არც ერთი პირობა არ ჩაჯდება, ანუ არ გადაებმევა
   3. CROSS JOIN
   4. სწორი პასუხებია - CROSS JOIN და FULL JOIN - იმ შემთხვევაში თუ გადაბმის პირობაში არც ერთი პირობა არ ჩაჯდება, ანუ არ გადაებმევა
2. სიმრავლეებზე მანიპულირებისას, იმას თუ რომელი ცხრილი ეწერება პირველი და რომელი მეორე შედეგზე გავლენა შეიძლება მოხდინოს შემდეგი ბრძანებისთვის
   1. UNION ALL
   2. EXCEPT
   3. INTERSECT
   4. UNION
3. რა შედეგს დაგვიბრუნებს შემდეგი სკრიპტი



* 1. A ს შესაბამის ASCII კოდს
  2. a ს შესაბამის ASCII კოდს
  3. [ სიმბოლოს შესაბამის ASCII კოდს
  4. NULL - ს

1. სკალარული ფუნქცია აბრუნებს
   1. მხოლოდ რიცხვით მნიშვნელობებს
   2. ცხრილს
   3. ერთ მნიშვნელობას, გარდა date, text, cursor, image, timestamp
   4. არც ერთი პასუხი არ არის სწორი
2. რომელი ტიპის ფუნქციის სინტაქსში არ გამოიყენება BEGIN ... END ბრძანებები?
   1. Scalar
   2. Inline Table-valued
   3. MSTVF
   4. ყველაში გამოიყენება
3. რომელ ტიპის ფუნქციას შეიძლება რომ არ ჰქონდეს input პარამეტრი?
   1. Inline Table-valued function
   2. Multi-statement table-valued functions
   3. სამივე პასუხი სწორია
   4. Scalar Functions
4. რა შეზღუდვებს ადებს Primary Key ველს?
   1. UNIQUE, IDENTITY, NOT NULL
   2. UNIQUE, CHECK,
   3. NOT NULL, UNIQUE
   4. IDENTITY, NOT NULL
5. რომელია ბრძანებების სწორი მიმდევრობა?
   1. SELECT, TOP, FROM, GROUP BY, HAVING, WHERE, ORDER BY
   2. SELECT, TOP, FROM, GROUP BY, WHERE, HAVING, ORDER BY
   3. SELECT, TOP, FROM, WHERE, GROUP BY, ORDER BY, HAVING
   4. SELECT, TOP, FROM, WHERE, GROUP BY, HAVING, ORDER BY
6. რომელია არასწორი ჩანაწერი?
   1. Identity(0, 1)
   2. Identity (-1, -100)
   3. Identity (1, 0.5)
   4. Identity (1, 1)

შექმენით ტრიგერი სახელად trgAfterUpdateSupplyOrder SupplyOrder ცხრილზე, რომელიც ჩაწერს ჩანაწერს OrderHistory(შექმენით ცხრილი) ცხრილში სვეტებით HistoryID, OrderID, OldTotalAmount, NewTotalAmount და ChangeDate, როდესაც განახლდება შეკვეთის TotalAmount - SupplyOrder ცხრილში.

ALTER TABLE Supplier

ADD CONSTRAINT chk\_ContactEmail\_AtSymbol

CHECK (ContactEmail LIKE '%@%');

SupplyOrder ცხრილი სვეტებით OrderID (მთელი რიცხვი, ძირითადი გასაღები), SupplierID (მთელი რიცხვი), OrderDate (თარიღი), TotalAmount (ათწილადი). დაამატეთ foreign გასაღების შეზღუდვა SupplierID-დან SupplyOrder-ში SupplierID-ს Supplier-ში და შეამოწმეთ შეზღუდვა, რათა უზრუნველყოთ TotalAmount 0-ზე მეტობა.

ALTER TABLE SupplyOrder

ADD CONSTRAINT fk\_SupplyOrder\_Supplier

FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID);

ALTER TABLE SupplyOrder

ADD CONSTRAINT chk\_TotalAmount\_Positive

CHECK (TotalAmount > 0);

ჩააინსერტეთ სამი მომწოდებელი („ABC Supplies“ ID 1-ით და ელფოსტა „abc@supplies.com“, „XYZ Distributors“ ID 2-ით და ელფოსტა „xyz@distributors.com“ და „LMN Providers“ ID 3-ით და ელფოსტა „lmn@providers.com“) მიმწოდებლის ცხრილში.

INSERT INTO Supplier (SupplierID, SupplierName, ContactEmail) VALUES

(1, 'ABC Supplies', 'abc@supplies.com'),

(2, 'XYZ Distributors', 'xyz@distributors.com'),

(3, 'LMN Providers', 'lmn@providers.com');

ჩააინსერტეთ სამი supply orders (Order ID 101 for Supplier ID 1 on '2024-01-01' with total amount 5000, Order ID 102 for Supplier ID 2 on '2024-02-01' with total amount 6000, and Order ID 103 for Supplier ID 3 on '2024-03-01' with total amount 7000) SupplyOrder თებილში.

INSERT INTO SupplyOrder (OrderID, SupplierID, OrderDate, TotalAmount) VALUES

(101, 1, '2024-01-01', 5000),

(102, 2, '2024-02-01', 6000),

(103, 3, '2024-03-01', 7000);

დააფდეითეთ შეკვეთების ჯამური რაოდენობა( TotalAmount), გაიზარდეთ 10%-ით, სადაც OrderDate არის „2024-02-15“-მდე.

UPDATE SupplyOrder

SET TotalAmount = TotalAmount \* 1.10

WHERE OrderDate < '2024-02-15';

შექმენით ფუნქცია სახელად GetSupplierOrderCount, რომელიც იღებს SupplierID-ს (მთლიანი რიცხვი) და აბრუნებს შეკვეთების რაოდენობას ამ მომწოდებლისთვის.

CREATE FUNCTION GetSupplierOrderCount (@SupplierID INT)

RETURNS INT

AS

BEGIN

DECLARE @OrderCount INT;

SELECT @OrderCount = COUNT(\*)

FROM SupplyOrder

WHERE SupplierID = @SupplierID;

RETURN @OrderCount;

END;

შექმენით ფუნქცია სახელად GetSupplierOrderCount, რომელიც იღებს SupplierID-ს (მთლიანი რიცხვი) და აბრუნებს შეკვეთების რაოდენობას ამ მომწოდებლისთვის.

CREATE FUNCTION GetSupplierOrderCount (@SupplierID INT)

RETURNS TABLE

AS

RETURN

SELECT COUNT(\*) AS OrderCount

FROM SupplyOrder

WHERE SupplierID = @SupplierID;