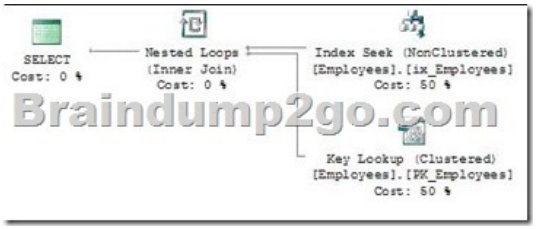
# Up-To-Date Braindump2go Microsoft 70-433 Exam Dumps PDF Files Free Download (11-20)

New Released Braindump2go Microsoft 70-433 Dumps PDF - Questions and Answers Updated with Microsoft Official Exam Center! Visit Braindump2go and download our 70-433 Exam Questions Now, Pass 70-433 100% at your first time! Exam Code: 70-433Exam Name: TS: Microsoft SQL Server 2008, Database DevelopmentCertification Provider: MicrosoftKeywords: 70-433 Exam Dumps,70-433 Practice Tests,70-433 Practice Exams,70-433 Exam Questions,70-433 PDF,70-433 VCE Free,70-433 Book,70-433 E-Book,70-433 Study Guide,70-433 Braindump,70-433 Prep Guide



QUESTION 11You need to generate the following XML document. <ProductExport> <Product Price="99">Product1</Product> <Product Price="199">Product2</Product> <Product Price="299">Product3</Product> <Product Price="399">Product4</Product> </ProductExport> Which query should you use? A. SELECT Price, ProductName FROM Products AS ProductExport FOR XML PATH('Product') B. SELECT Price, ProductName FROM Products FOR XML AUTO, ROOT('ProductExport') C. SELECT Price [@Price], ProductName AS [*] FROM Products AS ProductExport FOR XML AUTO, ELEMENTS D. SELECT Price [@Price], ProductName AS [*] FROM Products FOR XML PATH('Product'),ROOT('ProductExport') Answer: D QUESTION 12Your company's database contains Customers and Orders tables. You have been tasked to write a SELECT statement that outputs customer and order data as a valid and well-formed XML document. You are required to mix attribute and element based XML within the document. You have determined that using the FOR XML AUTO clause will not be suitable. You need to identify the correct FOR XML clause to meet the requirement. Which FOR XML statement should you use? (Each correct answer represents a complete solution. Choose two.) A. FOR BROWSE B. FOR XML RAW C. FOR XML PATH D. FOR XML EXPLICIT Answer: CD QUESTION 13Your company's database contains Customers and Orders tables. You have been tasked to write a SELECT statement that exposes the data as a valid and well-formed XML document. The XML data must be attribute-based, and the order data XML must be nested in the customer data XML. You need to write a SELECT statement to meet the requirements. Which Transact-SQL statement should you use? A. SELECT c.ContactName, o.OrderDate, o.RequiredDate FROM Customers c INNER JOIN Orders o ON c.CustomerID = o.CustomerID FOR XML RAW('Contact'), ROOT('ContactOrderDate') B. SELECT c.ContactName, o.OrderDate, o.RequiredDate FROM Customers c INNER JOIN Orders o ON c.CustomerID = o.CustomerID FOR XML PATH('ContactOrderDate') C. SELECT c.ContactName, o.OrderDate, o.RequiredDate FROM Customers c INNER JOIN Orders o ON c.CustomerID = o.CustomerID FOR XML AUTO D. SELECT c.ContactName, o.OrderDate, o.RequiredDate FROM Customers c INNER JOIN Orders o ON c.CustomerID = o.CustomerID FOR XML AUTO, ROOT('ContactOrderDate') Answer: DExplanation:SELECT c.ContactName, o.OrderDate, o.RequiredDate FROM Customers c INNER JOIN Orders oON c.CustomerID = o.CustomerIDFOR XML RAW('Contact'), ROOT('ContactOrderDate') Produce the following result:<ContactOrderDate><Contact ContactName="Paul Henriot" OrderDate="1996-07-04T00:00:00" RequiredDate="1996-08-01T00:00:00" /><Contact ContactName="Karin Josephs" OrderDate="1996-07-05T00:00:00" RequiredDate="1996-08-16T00:00:00" /><Contact ContactName="Paula Wilson" OrderDate="1998-05-06T00:00:00" RequiredDate="1998-06-03T00:00:00" /></ContactOrderDate>SELECT c.ContactName, o.OrderDate, o.RequiredDate FROM Customers c INNER JOIN Orders oON c.CustomerID = o.CustomerIDFOR XML PATH('ContactOrderDate')Produce the following result:<ContactOrderDate><ContactName>Paul Henriot</ContactName> <OrderDate>1996-07-04T00:00:00</OrderDate><RequiredDate>1996-08-01T00:00:00</RequiredDate> </ContactOrderDate> <ContactOrderDate><ContactName>Karin Josephs</ContactName><OrderDate>1996-07-05T00:00:00</OrderDate> <RequiredDate>1996-08-16T00:00:00</RequiredDate> </ContactOrderDate><ContactOrderDate><ContactName>Paula Wilson</ContactName><OrderDate>1998-05-06T00:00:00</OrderDate><RequiredDate>1998-06-03T00:00:00</RequiredDate> </ContactOrderDate>SELECT c.ContactName, o.OrderDate, o.RequiredDate FROM Customers c INNER JOIN Orders oON c.CustomerID = o.CustomerIDFOR XML AUTOProduce the following result:<c ContactName="Paul Henriot"><o OrderDate="1996-07-04T00:00:00" RequiredDate="1996-08-01T00:00:00" /> </c><c ContactName="Karin Josephs"><o

OrderDate="1996-07-05T00:00:00" RequiredDate="1996-08-16T00:00:00" /> </c><c ContactName="Paula Wilson"><o OrderDate="1998-05-06T00:00:00" RequiredDate="1998-06-03T00:00:00" /> </c>SELECT c.ContactName, o.OrderDate, o.RequiredDate FROM Customers c INNER JOIN Orders oON c.CustomerID = o.CustomerIDFOR XML AUTO, ROOT('ContactOrderDate')Produce the following result:<ContactOrderDate><c ContactName="Paul Henriot"><o OrderDate="1996-07-04T00:00:00" RequiredDate="1996-08-01T00:00:00" /> </c><c ContactName="Karin Josephs"><o OrderDate="1996-07-05T00:00:00" RequiredDate="1996-08-16T00:00:00" /> </c><c ContactName="Paula Wilson"><o OrderDate="1998-05-06T00:00:00" RequiredDate="1998-06-03T00:00:00" /> </c></ContactOrderDate> QUESTION 14You have a table named Customer that has an XML column named Locations. This column stores an XML fragment that contains details of one or more locations, as show in the following examples. <Location City="Sydney" Address="..." PhoneNumber="..." /> <Location City="Chicago" Address="..." PhoneNumber="..." /> <Location City="London" Address="..." PhoneNumber="..." /> You need to write a query that returns a row for each of the customer's locations. Each resulting row must include the customer name, city, and an XML fragment that contains the location details. Which query should you use? A. SELECT CustomerName, Locations.query('for $i in /Location return data($i/@City)'), Locations.query('for $i in /Location return $i') FROM Customer B. SELECT CustomerName, Locations.query('for $i in /Location return element Location {$i/@City, $i}') FROM Customer C. SELECT CustomerName, Locations.query('data(/Location/@City)'), Locations.query('/Location') FROM Customer D. SELECT CustomerName, Loc.value('@City','varchar(100)'), Loc.query('.') FROM Customer CROSS APPLY Customer.Locations.nodes ('/Location') Locs(Loc) Answer: D QUESTION 15Click the Exhibit button.




You have the following XML: <Site URL="http://www.contoso.com/index.htm"> <Site URL="http://www.contoso.com/finance/index.htm"> <Site URL="http://www.contoso.com/finance/reports/index.htm" /> <Site URL="http://www.contoso.com/finance/main/index.htm" /> </Site> <Site URL="http://www.contoso.com/marketing/index.htm"> <Site URL="http://www.contoso.com/marketing/reports/index.htm" /> <Site URL="http://www.contoso.com/marketing/main/index.htm" /> </Site> <Site URL="http://www.contoso.com/sales/index.htm" /> </Site> You are tasked to query the sites listed in the XML by using OPENXML. The results will have two columns, ParentSiteURL and SiteURL. The ParentSiteURL column should contain the URL attribute of the parent site. The SiteURL column should contain the URL attribute of the site itself. The output should look like that in the exhibit. You need to write the OPENXML query. Which Transact-SQL statement should you use? A. SELECT ParentSiteURL, SiteURL FROM OPENXML (@XMLDocHandle, '//@Site', 1) WITH ( ParentSiteURL nVarChar(512) '../URL', SiteURL nVarChar(512) 'URL') B. SELECT ParentSiteURL, SiteURL FROM OPENXML (@XMLDocHandle, '//URL', 1) WITH ( ParentSiteURL nVarChar(512) '../@URL', SiteURL nVarChar(512) '@URL') C. SELECT ParentSiteURL, SiteURL FROM OPENXML (@XMLDocHandle, '//Site', 1) WITH ( ParentSiteURL nVarChar(512) '../@URL', SiteURL nVarChar(512) '@URL') D. SELECT ParentSiteURL, SiteURL FROM OPENXML (@XMLDocHandle, '//@URL', 1) WITH ( ParentSiteURL nVarChar(512) '../URL', SiteURL nVarChar(512) 'URL') Answer: CExplanation:DECLARE @XMLDocHandle int,@XMLDoc varchar(1000) ='<Site URL="http://www.contoso.com/index.htm"><Site URL="http://www.contoso.com/finance/index.htm"> <Site URL="http://www.contoso.com/finance/reports/index.htm" /> <Site URL="http://www.contoso.com/finance/main/index.htm" /> </Site> <Site URL="http://www.contoso.com/marketing/index.htm"> <Site URL="http://www.contoso.com/marketing/reports/index.htm" /> <Site URL="http://www.contoso.com/marketing/main/index.htm" /></Site><Site URL="http://www.contoso.com/sales/index.htm" /> </Site>';--Create an internal representation of the XML document. EXEC sp_xml_preparedocument @XMLDocHandle OUTPUT, @XMLDoc SELECT ParentSiteURL, SiteURLFROM OPENXML (@XMLDocHandle, '//Site', 1)WITH ( ParentSiteURL nVarChar(512) '../@URL',SiteURL nVarChar(512) '@URL') QUESTION 16 Your company uses an application that passes XML to the database server by using stored procedures. The database server has a large number of XML handles that are currently active. You determine that the XML is not being flushed from SQL Server memory. You need to identify the system stored procedure to flush the XML from memory. Which Transact-SQL statement should you use? A. sp_xml_removedocument B. sp_xml_preparedocument C. sp_reserve_http_namespace D. sp_delete_http_namespace_reservation Answer: AExplanation:sp_xml_removedocument removes the internal representation of the

XML document specified by the document handle and invalidates the document handle. sp_xml_preparedocument reads the XML text provided as input, parses the text by using the MSXML parser(Msxmlsql.dll), and provides the parsed document in a state ready for consumption. This parsed document is a tree representation of the various nodes in the XML document:elements, attributes, text, comments, and so on. A parsed document is stored in the internal cache of SQL Server. The MSXML parser uses one-eighth the total memory available for SQL Server. To avoid running out of memory, run sp_xml_removedocument to free up the memory. QUESTION 17You work for a company that provides marketing data to other companies. You have the following Transact-SQL statement: DECLARE @CustomerDemographics XML SET @CustomerDemographics=N' <CustomerDemographics> <Customer CustomerID="1" Age="21" Education="High School"> <IsCoffeeDrinker>0</IsCoffeeDrinker> </Customer> <Customer CustomerID="2" Age="27" Education="College"> <IsCoffeeDrinker>1</IsCoffeeDrinker> <IsFriendly>1</IsFriendly> </Customer> <Customer CustomerID="3" Age="35" Education="Unknown"> <IsCoffeeDrinker>1</IsCoffeeDrinker> <IsFriendly>1</IsFriendly> </Customer> </CustomerDemographics>' DECLARE @OutputAgeOfCoffeeDrinkers XML SET @OutputAgeOfCoffeeDrinkers = @CustomerDemographics.query(' for $output in /child::CustomerDemographics/child::Customer[( child::IsCoffeeDrinker[1] cast as xs:boolean )] return <CoffeeDrinkingCustomer> { $output/attribute::Age } </CoffeeDrinkingCustomer>') SELECT @OutputAgeOfCoffeeDrinkers You need to determine the result of the query. What result should you expect? A. <CoffeeDrinkingCustomer Age="27" /> <CoffeeDrinkingCustomer Age="35" /> B. <CoffeeDrinkingCustomer Age="21" /> C. <CustomerDemographics> <Customer> <CoffeeDrinkingCustomer Age="21" /> </Customer> </CustomerDemographics> D. <CustomerDemographics> <Customer> <CoffeeDrinkingCustomer Age="27" /> </Customer> <Customer> <CoffeeDrinkingCustomer Age="35" /> </Customer> </CustomerDemographics> Answer: A QUESTION 18You have a table named Stores that has an XML column named OpenHours. This column contains the opening and closing times.<hours dayofWeek= "Monday" open ="8:00 AM" closed="8:00 PM"<hours dayofWeek= "Tuesday" open ="8:00 AM" closed="8:00 PM"...<hours dayofWeek= "Saturday" open ="8:00 AM" closed="8:00 PM"You need to write a query that returns a list of stores and their opening time for a specified day.Which code segment should you use? A. DECLARE @Day VARCHAR(10) = 'Tuesday' SELECT StoreName, OpenHours.value('/hours[1]/@open','time') FROM Stores WHERE OpenHours.value('/hours[1]/@dayofWeek','varchar(20)') = @Day B. DECLARE @Day VARCHAR(10) = 'Tuesday' SELECT StoreName, OpenHours.value('/hours[1]/@open','time') FROM Stores WHERE OpenHours.exist('/hours[@dayofWeek=sql:variable("@Day")]') = 1 C. DECLARE @Day VARCHAR(10) = 'Tuesday' SELECT Storename, OpenHours.query('data(/hours[@dayofWeek=sql:variable("@Day")]/@open)') FROM Stores D. DECLARE @Day VARCHAR(10) = 'Tuesday' SELECT StoreName, OpenHours.value('/hours[1][@dayofWeek=sql:variable("@Day")]/@open','time') FROM Stores Answer: CExplanation:CREATE TABLE Stores(StoreName VARCHAR(10)NOT NULL,OpenHours [xml] NULL, CONSTRAINT [PK_Stores] PRIMARY KEY CLUSTERED (StoreName)) GOINSERT INTO Stores (StoreName, OpenHours) VALUES('Store1','<hours dayofWeek= "Wednesday" open ="8:00 AM" closed="8:00 PM"/> <hours dayofWeek= "Tuesday" open ="9:00 AM" closed="8:00 PM"/> <hours dayofWeek= "Friday" open ="8:00 AM" closed="8:00 PM"/>'), ('Store2','<hours dayofWeek= "Monday" open ="8:00 AM" closed="8:00 PM"/> <hours dayofWeek= "Tuesday" open ="8:00 AM" closed="8:00 PM"/> <hours dayofWeek= "Saturday" open ="8:00 AM" closed="8:00 PM"/>') DECLARE @Day VARCHAR(10) = 'Tuesday' SELECT Storename, OpenHours.query('data(/hours[@dayofWeek=sql:variable("@Day")]/ @open)')FROM StoresGO QUESTION 19You have the following XML document that contains Product information. DECLARE @prodList xml =' <ProductList xmlns="urn:Wide_World_Importers/schemas/Products"> <Product Name="Product1" Category="Food" Price="12.3" /> <Product Name="Product2" Category="Drink" Price="1.2" /> <Product Name="Product3" Category="Food" Price="5.1" /> ... </ProductList>'; You need to return a list of products that contains the Product Name, Category, and Price of each product. Which query should you use? A. SELECT prod.value('.[1]/@Name','varchar(100)'), prod.value('.[1]/@Category','varchar(20)'), prod.value('.[1]/@Price','money') FROM @prodList.nodes('/ProductList/Product') ProdList(prod); B. SELECT prod.value('@Name','varchar(100)'), prod.value('@Category','varchar(20)'), prod.value('@Price','money') FROM @prodList.nodes('/ProductList/Product') ProdList(prod); C. WITH XMLNAMESPACES(DEFAULT 'urn;Wide_World_Importers/schemas/Products' as o) SELECT prod.value('Name[1]','varchar(100)'), prod.value('Category[1]','varchar(20)'), prod.value('Price[1]','money') FROM @prodList.nodes('/o:ProductList/o:Product') ProdList(prod); D. WITH XMLNAMESPACES(DEFAULT 'urn:Wide_World_Importers/schemas/Products') SELECT prod.value('./@Name','varchar(100)'), prod.value('./@Category','varchar(20)'), prod.value('./@Price','money') FROM @prodList.nodes('/ProductList/Product') ProdList(prod); Answer: D QUESTION 20You have a table named Products.Product. The table has columns ProductID, Name, Size, and Category. You have a variable named @XML with following XML value: <Root> <Category Name="Socks" /> <Category Name="Pants" /> <Category Name="Shirts" /> </Root>You are tasked to write a query

that lists the products in Products.Product that match the categories listed in the XML document. You need to write a query to accomplish the task. Which query should you write? A. SELECT p.ProductID, p.Name, p.Size, p.Category FROM Production.Product p CROSS APPLY @XML.nodes('//Category') as x(s) B. SELECT p.ProductID, p.Name, p.Size, p.Category FROM Production.Product p OUTER APPLY @XML.nodes('//Category') as x(s) C. WITH XMLTable AS ( SELECT s.value('@Name','varchar(20)') as Category FROM @XML.nodes('//Category') as x(s) ) SELECT p.ProductID, p.Name, p.Size, p.Category FROM Production.Product p INNER JOIN XMLTable x ON p.Category = x.Category D. WITH XMLTable AS ( SELECT s.value('@Category','varchar(20)') as Category FROM @XML.nodes('//Category') as x(s) ) SELECT p.ProductID, p.Name, p.Size, p.Category FROM Production.Product p INNER JOIN XMLTable x ON p.Category = x.Category Answer: C Braindump2go New Released 70-433 Dump PDF Free Download, 210 Questions in all, Passing Your Exam 100% Easily!



http://www.braindump2go.com/70-433.html