SQL Server 2012 – Database Development

Lab Book

Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Revision No. | Author | **Summary of Changes** |
| 25th July 2011 | 2.0 | Latha S. | Changes in Material made based on integration process |
| 3rd April, 2012 | 3.0 | Shilpa Bhosle | Changes in Lab Book are made as an upgrade to SQL Server 2008 |
| 21st Aug 2013 | 4.0 | Shashank Saudagar | Changes in Lab Book are made as an upgrade to SQL Server 2012 |
| 14th May 2015 | 5.0 | Vaishali Kasture | Changes in Lab Book as per new curriculum of SQL Server 2012 |
| 9th May 2016 | 6.0 | Shital A Patil | Changes in Material made based on integration process as per Capgemini Course Structure |

Table of Contents

[Getting Started 4](#_Toc453855038)

[Lab 1. Getting connected to the SQL Server 2012 Server 5](#_Toc453855039)

[1.1 Steps to connect to the SQL Server 2012 Server 5](#_Toc453855040)

[1.2 Getting Familiar with SQL Server 6](#_Toc453855041)

[1.3 SQL Languages – DDL- Creating Tables, Alias Data Type and Constraints 7](#_Toc453855042)

[1.4 Simple Queries & Merge Statement 12](#_Toc453855043)

[1.5 Data Retrieval - Joins, Subqueries, SET Operators and DML 20](#_Toc453855044)

[1.6 Indexes and Views 23](#_Toc453855045)

[1.7 Procedures and Exception Handling in SQL server 25](#_Toc453855046)

[Lab 2. SQL Server 2012 Stretched Assignment 28](#_Toc453855047)

[2.1 Transact-SQL Statements 28](#_Toc453855048)

[2.2 Data Retrieval - Joins, Subqueries, SET Operators and DML 29](#_Toc453855049)

[2.3 Indexes and Views 30](#_Toc453855050)

[Appendix A: Table Structure 31](#_Toc453855051)

[Appendix B: Table of Figures 34](#_Toc453855052)

Getting Started

Overview

This Lab book is a guided tour for Learning SQL server 2012. Each section contains some examples and assignments. Follow the steps provided in the solved examples and then work out the Assignments given.

Setup Checklist for SQL Server 2012

Here is what is expected on your machine in order for the lab to work.

Minimum System Requirements

Processor, HDD & RAM

* Processor - Minimum: AMD Opteron, AMD Athlon 64, Intel Xeon with Intel EM64T support, Intel Pentium IV with EM64T support
* Processor speed: Minimum: 1.4 GHz
* Recommended: 2.0 GHz or faster
* RAM - Minimum:512 MB, Recommended: 2.048 GB or more
* HDD – 150 GB

Operating System

* Windows XP Professional x64
* Windows 7 Professional 64 bit

SQL Server 2012 Developer Edition

* SQL server 2012 client and a SQL server 2012 Server instance running on the Server.

A database called Training will be available. All objects for the lab session would be stored in that database alone.

1. Getting connected to the SQL Server 2012 Server

## Steps to connect to the SQL Server 2012 Server

1. Click Start, Programs, Microsoft SQL Server 2012, SQL Server Management Studio.

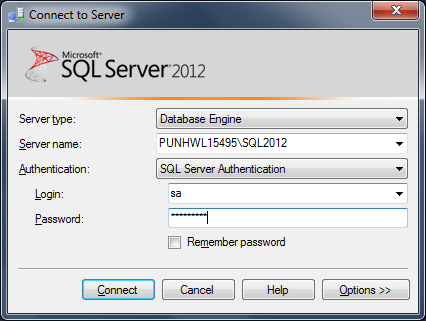


Figure 1: Connecting to SQL Server 2012



Enter the Login, Password and the Server name provided to you.

Login: <loginid> Passwd: <password>

1. Click on New Query.

## Getting Familiar with SQL Server

1. Identify all the system and user defined database in your system.
2. Make master database as your current database , by using the command

Use <databasename>

go

1. Find out if your active database is master ,by giving the command

Select DB\_NAME()

go

1. Now make Training database as your active database
2. Find out the content of the database by giving the following command. Observe the output

sp\_help

go

1. Repeat the above steps for master database and Northwind database
2. Find out the version of your SQL Server by giving the following command

Select @@version

go

1. Find out the server date by giving the following commands

Select getdate()

go

1. Make Northwind as your current database , find out information about tables using the command - Categories ,Products , Orders, Order Details , Employees

sp\_help <tablename>

go

1. Make a note of all related tables and foreign key columns
2. Repeat the above operation of Training database tables as well

## SQL Languages – DDL- Creating Tables, Alias Data Type and Constraints

The following questions will be solved using the Training database only

1. Create a Table called Customer\_<empid> with the following Columns

|  |  |  |
| --- | --- | --- |
| Customerid | Int | Unique NOT NULL |
| CustomerName | varchar(20) | Not Null |
| Address1 | varchar(30) |  |
| Address2 | varchar(30) |  |
| Contact Number | varchar(12) | Not Null |
| Postal Code | Varchar(10) |  |

1. Create a table called Employees\_<empid>

CREATE TABLE Employees

(

EmployeeId INT NOT NULL PRIMARY KEY,

Name NVARCHAR(255) NULL

);

1. Create a table called Contractors\_<empid>

CREATE TABLE Contractors

(

ContractorId INT NOT NULL PRIMARY KEY,

Name NVARCHAR(255) NULL

);

1. Create a table called TestRethrow\_<empid>

USE Training;

CREATE TABLE dbo.TestRethrow

(

ID INT PRIMARY KEY

);

In Object Explorer, go to Databases | Training| Tables and you should see Customers, Employees, Contractors and TestRethrow tables created

1. Create a user defined data type called Region, which would store a character string of size 15.

hint2**Hint:** Use Create Type Statement

1. Create a Default which would store the value ‘NA’ (North America’)

hint2**Hint:** create default

1. Bind the default to the Alias Data Type of Q1 i.e. region

**hint2Hint:** use sp\_bindefault

**Syntax -** EXEC sp\_bindefault <DefaultName>, '<AliasName>'

1. Modify the table Customers to add the a column called Customer\_Region which would be of data type:

Region

1. Add the column to the Customer Table.

Gender char (1)

1. Using alter table statement add a constraint to the Gender column such that it would not accept any other values except ‘M’,’F’ and ‘T”.
2. Create the Table Orders with the following Columns:

|  |  |  |
| --- | --- | --- |
| OrdersID | Int | NOT NULL IDENTITY with starting values 1000 |
| Customerld | Int | Not Null |
| OrdersDate | Datetime |  |
| Order\_State | char(1) | can be only ‘P’ or ‘C’ |

1. Add referential integrity constraint for Orders & Customer tables through Customerld with the name fk\_CustOrders.

Using sp\_help check if the constraints have been added properly.

1. Creating and using Sequence Numbers

Task 1 – Creating the Sequence

1. Copy and paste the following code segment in query editor.
   * + 1. SQL
       2. USE **Training**;
       3. CREATE SEQUENCE **IdSequence** AS INT
       4. START WITH **10000**
       5. INCREMENT BY **1**;
2. In Object Explorer, go to Databases | Training | Programmability | Sequences, right-click and select Refresh.

Click on the plus sign at the left of Sequences, and you should see the IdSequence sequence

Task 2 – Using the Sequence to Insert New Rows

Finally, you have both the sequence and tables to insert new rows with sequential identifiers.

1. Copy and paste the following code segment in query editor.
   * + 1. SQL
       2. USE **Training**;
       3. INSERT INTO **Employees** (**EmployeeId**, **Name**)
       4. VALUES (NEXT VALUE FOR **IdSequence**, '**Shashank**');
       5. INSERT INTO **Contractors** (**ContractorId**, **Name**)
       6. VALUES (NEXT VALUE FOR **IdSequence**, '**Aditya**');
       7. SELECT \* FROM **Employees**;
       8. SELECT \* FROM **Contractors**;
2. You should be able to see now the result of the execution below the Results tab. As you can see, the Employees table has an employee named Shashank, with EmployeeId 10000, while the Contractors table has a contractor named Aditya with EmployeeId 10001. Asking for the next value while inserting a row in both tables obtained a new value for the EmployeeId field.

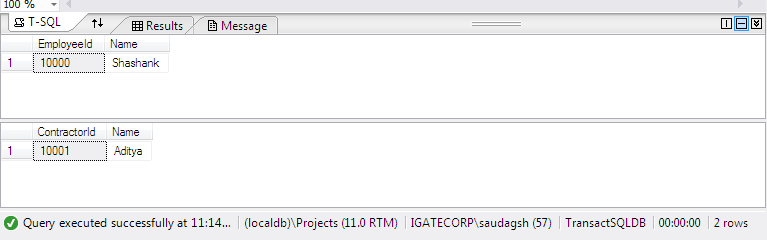


Figure 2: Result of Sequence

## Simple Queries & Merge Statement

For these questions, you will be using the University Schema; the table structure has been given in the appendix. These tables would be available in the Training database

1. List out Student\_Code, Student\_Name and Department\_Code of every Student
2. Do the same for all the staff’s
3. Retrieve the details (Name, Salary and dept code) of the employees who are working in department 20, 30 and 40.
4. Display Student\_Code, Subjects and Total\_Marks for every student. Total\_Marks will calculate as Subject1 + Subject2 + Subject3 from Student\_Marks. The records should be displayed in the descending order of Total Score
5. List out all the books which starts with ‘An’, along with price
6. List out all the department codes in which students have joined this year
7. Display name and date of birth of students where date of birth must be displayed in the format similar to “January, 12 1981” for those who were born on Saturday or Sunday.

**hint2Hint:** Use datename or datepart function

1. List out a report like this

StaffCode StaffName Dept Code Date of Joining No of years in the Company

1. List out all staffs who have joined before Jan 2000
2. Write a query which will display Student\_Name, Departmnet\_Code and DOB of all students who born between January 1, 1981 and March 31, 1983.
3. List out all student codes who did not appear in the exam subject2

Working with Merge Statement

**Case Study –** The Countryside Confectioneries is one of the well-known names in the brands of confectioneries in Switzerland. The company Database Administrator John & his team, maintains the entire business data in SQL Server. As a database administrator, John, need to perform the ETL (Extract, Transform & Load) on database quite often, wherein he needs to execute multiple INSERT, UPDATE & DELETE Operations on database target table by matching the records from the source table. For example, a products dimension table has information about the products; you need to sync-up this table with the latest information about the products from the source table.

To simplify above task John & his team uses SQL Server one of the remarkable programming enhancement called MERGE statement as MERGE SQL command to perform these operations in a single statement. He uses MERGE statement to so that he can eliminate the need of writing multiple and separate DML statements to refresh the target table with an updated product list or do lookups.

The following example demonstrates the use of MERGE statement in above given case study.

1. Create following tables in SQL Server 2012 – In this demo you will be creating Products table as Target table & UpdateProducts as Source Table. You will also populate these tables with some sample data.

CREATE TABLE Products

(

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Rate MONEY

)

--Insert records into target table

INSERT INTO Products

VALUES

(1,'Tea', 10.00),

(2, 'Coffee', 20.00),

(3, 'Muffin', 30.00),

(4, 'Biscuit', 40.00)

CREATE TABLE UpdatedProducts

(

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Rate MONEY

)

--Insert records into source table

INSERT INTO UpdatedProducts

VALUES

(1, 'Tea', 10.00),

(2, 'Coffee', 25.00),

(3, 'Muffin', 35.00),

(5, 'Pizza', 60.00)

**MERGE Statement**

--Synchronize the target table with

--refreshed data from source table

MERGE Products AS TARGET

USING UpdatedProducts AS SOURCE

ON (TARGET.ProductID = SOURCE.ProductID)

--When records are matched, update

--the records if there is any change

WHEN MATCHED AND TARGET.ProductName <> SOURCE.ProduActName

OR TARGET.Rate <> SOURCE.Rate THEN

UPDATE SET TARGET.ProductName = SOURCE.ProductName,

TARGET.Rate = SOURCE.Rate

--When no records are matched, insert

--the incoming records from source

--table to target table

WHEN NOT MATCHED BY TARGET THEN

INSERT (ProductID, ProductName, Rate)

VALUES (SOURCE.ProductID, SOURCE.ProductName, SOURCE.Rate)

--When there is a row that exists in target table and

--same record does not exist in source table

--then delete this record from target table

WHEN NOT MATCHED BY SOURCE THEN

DELETE

--$action specifies a column of type nvarchar(10)

--in the OUTPUT clause that returns one of three

--values for each row: 'INSERT', 'UPDATE', or 'DELETE',

--according to the action that was performed on that row

OUTPUT $action,

DELETED.ProductID AS TargetProductID,

DELETED.ProductName AS TargetProductName,

DELETED.Rate AS TargetRate,

INSERTED.ProductID AS SourceProductID,

INSERTED.ProductName AS SourceProductName,

INSERTED.Rate AS SourceRate;

SELECT @@ROWCOUNT;

GO

Create two tables (Target & Source) with a structure given below. Also, insert some sample data into both tables.

**TODO** – Write a MERGE Statement, that matches the EmpID attribute in the Source with the EmpID attribute of target. When a match is found in the target, the target Employee’s attributes should overwrite with the source Employee attributes. When a match is not found in the target, a new row should be inserted into the target, using the source Employee attributes. When a match is not found in the source, the target Employee row should be deleted.

Source Table

CREATE TABLE EmployeeSource

(

EmpID INT NOT NULL,

Designation VARCHAR (25) NOTNULL,

Phone VARCHAR (20) NOT NULL,

Address VARCHAR (50) NOT NULL,

CONSTRAINT PK\_EmployeeSC

PRIMARY KEY (EmpID)

)

Target Table

CREATE TABLE EmployeeTarget

(

EmpID INT NOT NULL,

Designation VARCHAR (25) NOT NULL,

Phone VARCHAR (20) NOT NULL,

Address VARCHAR (50) NOT NULL,

CONSTRAINT PK\_EmployeeTG

PRIMARY KEY (EmpID)

)

Working with Grouping Set

1. Create the following table & populate with some sample data.
2. Write following query which uses Grouping Set in the query window.

Employee Table

CREATE TABLE Employee

(

Employee\_Number INT NOT NULL PRIMARY KEY,

Employee\_Name VARCHAR(30) NULL,

Salary FLOAT NULL,

Department\_Number INT NULL,

Region VARCHAR(30) NULL

)

SELECT Region, Department\_Number, AVG (Salary) Average\_Salary

From Employee

Group BY GROUPING SETS

( (Region, Department\_Number),

(Region),

(Department\_Number)

)

1. Execute above query & observe the output.
2. The query performs following :
3. It generates result set grouped by each set mentioned in the Grouping Sets.
4. It also calculates average salary of every employee for each region and department.

One can get the same result achieved in early SQL Server versions using the following query:

**(NOTE – This part of Lab is not compulsory to perform)**

SELECT Region, Department\_Number, AVG (Salary) Average\_Salary

From Employee

Group BY (Region, Department\_Number)

UNION

SELECT Region, Department\_Number, AVG (Salary) Average\_Salary

From Employee

Group BY (Region)

UNION

SELECT Region, Department\_Number, AVG (Salary) Average\_Salary

From Employee

Group BY (Department\_Number)

## Data Retrieval - Joins, Subqueries, SET Operators and DML

1. Write a query which displays Staff Name, Department Code, Department Name, and Salary for all staff who earns more than 20000.
2. Write a query to display Staff Name, Department Code, and Department Name for all staff who do not work in Department code 10
3. Print out a report like this

Book Name No of times issued

Let us C 12

Linux Internals 9

1. List out number of students joined each department last year. The report should be displayed like this

Physics 12

Chemistry 40

1. List out a report like this

Staff Code Staff Name Manager Code Manager Name

**hint2Hint:** Use Self Join

1. Display the Staff Name, Hire date and day of the week on which staff was hired. Label the column as DAY. Order the result by the day of the week starting with Monday.
2. Display Staff Code, Staff Name, and Department Name for those who have taken more than one book.
3. List out the names of all student code whose score in subject1 is equal to the highest score
4. Modify the above query to display student names along with the codes.
5. List out the names of all the books along with the author name, book code and category which have not been issued at all. Try solving this question using EXISTS.
6. List out the code and names of all staff and students belonging to department 20.

**hint2Hint:** Use UNION

1. List out all the students who have not appeared for exams this year.
2. List out all the student codes who have never taken books
3. Add the following records to the Customers Table , created in our earlier exercises

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CustomerID** | **CustomerName** | **Address1** | **Address2** | **Contact** | **PostalCode** | **Region** | **Gender** |
| ALFKI | AlfredsFutterkiste | Obere Str. 57 | Berlin,Germany | 030-0074321 | 12209 | NULL | NULL |
| ANATR | Ana Trujillo Emparedados y helados | Avda. de la Constitución 2222 | México D.F.,Mexico | (5) 555-4729 | 5021 | NA | NULL |
| ANTON | Antonio Moreno Taquería | Mataderos 2312 | México D.F.,Mexico | (5) 555-3932 | 5023 | NULL | NULL |
| AROUT | Around the Horn | 120 Hanover Sq. | London,UK | (171) 555-7788 | WA1 1DP | NULL | NULL |
| BERGS | Berglundssnabbköp | Berguvsvägen 8 | Luleå,Sweden | 0921-12 34 65 | S-958 22 | NULL | NULL |
| BLAUS | Blauer See Delikatessen | Forsterstr. 57 | Mannheim,Germany | 0621-08460 | 68306 | NA | NULL |
| BLONP | Blondesddslpère et fils | 24, place Kléber | Strasbourg,France | 88.60.15.31 | 67000 | NULL | NULL |
| BOLID | BólidoComidaspreparadas | C/ Araquil, 67 | Madrid,Spain | (91) 555 22 82 | 28023 | EU | NULL |
| BONAP | Bon app' | 12, rue des Bouchers | Marseille,France | 91.24.45.40 | 13008 | NULL | NULL |
| BOTTM | Bottom-Dollar Markets | 23 Tsawassen Blvd. | Tsawassen,Canada | (604) 555-4729 | T2F 8M4 | BC |  |

1. Replace the contact number of Customer id ANATR to (604) 3332345.
2. Update the Address and Region of Customer BOTTM to the following

19/2 12th Block, Spring Fields.

Ireland - UK

Region - EU

1. Insert the following records in the Orders table. The Order id should be automatically generated

Save the commands in a script file (Script file has a .sql extension)

|  |  |  |
| --- | --- | --- |
| **Customer ID** | **OrderDate** | **Order State** |
| AROUT | 4-Jul-96 | P |
| ALFKI | 5-Jul-96 | C |
| BLONP | 8-Jul-96 | P |
| ANTON | 8-Jul-96 | P |
| ANTON | 9-Jul-96 | P |
| BOTTM | 10-Jul-96 | C |
| BONAP | 11-Jul-96 | P |
| ANATR | 12-Jul-96 | P |
| BLAUS | 15-Jul-96 | C |
| HILAA | 16-Jul-96 | P |

1. Delete all the Customers whose Orders have been cleared.
2. Remove all the records from the table using the truncate command. Rerun the script to populate the records once again
3. Change the order status to C, for all orders before `15th July.

## Indexes and Views

1. Create a Unique index on Department Name for Department master Table.
2. Try inserting the following values and observe the output

|  |  |
| --- | --- |
| **Dept Code** | **Dept Name** |
| 100 | Home Science |
| 200 | Home Science |
| 300 | NULL |
| 400 | NULL |

1. Create a non-clustered index for Book\_Trans table on the following columns

Boo\_code, Staff\_name, student name, date of issue. Try adding some values. Do you experience any difficulties?

1. List the indexes created in the previous questions, from the sysindexes table.
2. Create a View with the name StaffDetails\_view with the following column name

Staff Code, Staff Name, Department Name, Desig Name salary

1. Try inserting some records in the view; Are you able to add records? Why not? Write your answers here.
2. Working with Filtered Index – The following Filtered Index created on

Production.BillOfMaterials table, cover queries that return the columns defined in

The index and that select only rows with a non-NULL value for EndDate.

USE Adventure Works;

GO

CREATE NONCLUSTERED INDEX FIBillOfMaterialsWithEndDate

ON Production.BillOfMaterials (ComponentID, StartDate)

WHERE EndDate IS NOT NULL;

GO

1. View the definition of the view using the following syntax.

Sp\_helptext <viewname>

1. Using the view , List out all the staffs who have joined in the month of June
2. Create a non-clustered column store index on EmployeeID of Employees table

## Procedures and Exception Handling in SQL server

1. Write a procedure that accept Staff\_Code and updates the salary and store the old salary details in Staff\_Master\_Back (Staff\_Master\_Back has the same structure without any constraint) table. The procedure should return the updated salary as the return value

Exp< 2 then no Update

Exp>= 2 and <= 5 then 20% of salary

Exp> 5 then 25% of salary

1. Write a procedure to insert details into Book\_Transaction table. Procedure should accept the book code and staff/student code. Date of issue is current date and the expected return date should be 10 days from the current date. If the expected return date falls on Saturday or Sunday, then it should be the next working day. Suitable exceptions should be handled.
2. Modify question 1 and display the results by specifying With result sets
3. Create a procedure that accepts the book code as parameter from the user. Display the details of the students/staff that have borrowed that book and has not returned the same. The following details should be displayed

Student/StaffCode Student/StaffName IssueDate Designation ExpectedRet\_Date

1. Write a procedure to update the marks details in the Student\_marks table. The following is the logic.

* The procedure should accept student code , and marks as input parameter
* Year should be the current year.
* Student code cannot be null, but marks can be null.
* Student code should exist in the student master.
* The entering record should be unique ,i.e. no previous record should exist
* Suitable exceptions should be raised and procedure should return -1.
* IF the data is correct, it should be added in the Student marks table and a success value of 0 should be returned.

Working with THROW Statement

Task 1 – Raising and Catching an Exception

Now, we can use the **TestRethrow** table to force an exception. As you will see, the query runs successfully, but catches the error when attempting to insert the same primary key twice in the table, and shows an error message.

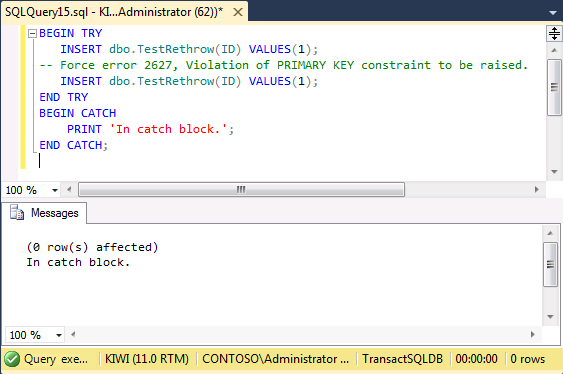
1. Copy and paste the following code segment in query editor.
   * 1. SQL
     2. USE **Training;**
     3. BEGIN TRY
     4. INSERT **dbo.TestRethrow**(**ID**) VALUES(**1**);
     5. -- **Force error 2627, Violation of PRIMARY KEY constraint to be raised**.
     6. INSERT **dbo.TestRethrow**(**ID**) VALUES(**1**);
     7. END TRY
     8. BEGIN CATCH
     9. PRINT **'In catch block.**';
     10. END CATCH;
     11. 

Figure 3: Attempting to insert the same row twice raises an exception

Task 3 – Using Throw to Raise an Exception Again in a Catch Block

Finally, we can add a **Throw** statement in the **Catch** block. This can be useful when there is a chain of procedures executed, so exceptions are bubbled up.

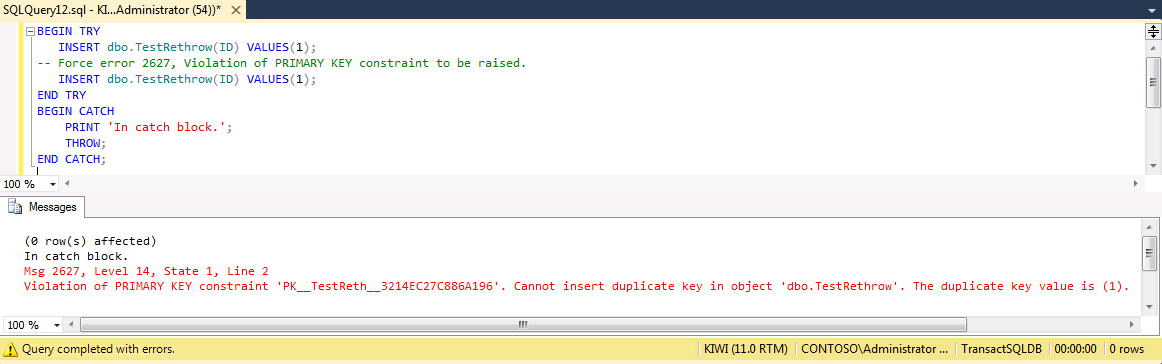
1. Copy and paste the following code segment in query editor.
   * 1. SQL
     2. USE **Training**;
     3. BEGIN TRY
     4. INSERT **dbo.TestRethrow**(**ID**) VALUES(**1**);
     5. -- **Force error 2627, Violation of PRIMARY KEY constraint to be raised**.
     6. INSERT **dbo.TestRethrow**(**ID**) VALUES(**1**);
     7. END TRY
     8. BEGIN CATCH
     9. PRINT **'In catch block.**';
     10. THROW;
     11. END CATCH;
     12. 

Figure 4:Re-throwing the exception shows actual error, & the query completes with errors

1. SQL Server 2012 Stretched Assignment
2. Transact-SQL Statements
3. List the empno, name and Department No of the employees who have got experience of more than 18 years.
4. Display the name and salary of the staff. Salary should be represented as X. Each X represents a 1000 in salary. It is assumed that a staff’s salary to be multiples of 1000 , for example a salary of 5000 is represented as XXXXX

Sample Output

JOHN 10000 XXXXXXXXXX

ALLEN 12000 XXXXXXXXXXXX

1. List out all the book code and library member codes whose return is still pending
2. List all the staff’s whose birthday falls on the current month
3. How many books are stocked in the library?
4. How many books are there for topics Physics and Chemistry?
5. How many members are expected to return their books today?
6. Display the Highest, Lowest, Total & Average salary of all staff. Label the columns Maximum, Minimum, Total and Average respectively. Round the result to nearest whole number
7. How many staffs are managers”?
8. List out year wise total students passed. The report should be as given below. A student is considered to be passed only when he scores 60 and above in all 3 subjects individually

Year No of students passed

1. List out all the departments which is having a headcount of more than 10
2. List the total cost of library inventory ( sum of prices of all books )
3. List out category wise count of books costing more than Rs 1000 /-
4. How many students have joined in Physics dept (dept code is 10) last year?
5. Data Retrieval - Joins, Subqueries, SET Operators and DML
6. Write a query that displays Staff Name, Salary, and Grade of all staff. Grade depends on the following table.

|  |  |
| --- | --- |
| Salary | Grade |
| Salary >=50000 | A |
| Salary >= 25000 < 50000 | B |
| Salary>=10000 < 25000 | C |

1. Generate a report which contains the following information.

Staff Code, Staff Name, Designation, Department, Book Code, Book Name,

Author, Fine

For the staff who have not return the book. Fine will be calculated as Rs. 5 per day.

Fine = 5 \* (No. of days = Current Date – Expected return date), for others it should be displayed as –

1. List out all the staffs who are reporting to the same manager to whom staff 100060 reports to.
2. List out all the students along with the department who reads the same books which the professors read
3. List out all the authors who have written books on same category as written by Author David Gladstone.
4. Display the Student report Card for this year. The report Card should contain the following information.

Student Code Student Name Department Name Total Marks Grade

Grade is calculated as follows. If a student has scored < 60 or has not attempted an exam he is considered to an F

>80 - E

70-80 - A

60- 69 - B

<60 – F

1. Indexes and Views
2. Create a Filtered Index HumanResources.Employee table present in the AdventureWorks database for the column EmployeeID. The index should cover all the queries that uses EmployeeID for its search & that select only rows with “Marketing Manager” for Title column.

Appendices

## Appendix A: Table Structure

1. Desig\_Master

|  |  |  |
| --- | --- | --- |
| **Name** | **Null?** | **Type** |
| Design\_code | Not Null | int |
| Design\_name |  | Varchar(50) |

1. Department\_Master

|  |  |  |
| --- | --- | --- |
| **Name** | **Null?** | **Type** |
| Dept\_Code | Not Null | int |
| Dept\_name |  | Varchar(50) |

1. Student\_Master Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Null?** | **Type** |  |
| Student\_Code | Not Null | int |  |
| Student\_name | Not Null | Varchar2(50) |  |
| Dept\_Code |  | int | FK ->Dept\_Master |
| Student\_dob |  | Datetime |  |
| Student\_Address |  | Varchar(240) |  |

1. Student\_Marks

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Null?** | **Type** |  |
| Student\_Code |  | int | FK->Student\_master |
| Student\_Year | Not Null | int |  |
| Subject1 |  | int |  |
| Subject2 |  | int |  |
| Subject3 |  | int |  |

1. Staff\_Master

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Null?** | **Type** |  |
| Staff\_code | Not Null | int |  |
| Staff\_Name | Not Null | Varchar(50) |  |
| Design\_code |  | int | FK->Design\_master |
| Dept\_code |  | int | FK->Dept\_Master |
| HireDate |  | Datetime |  |
| Staff\_dob |  | Datetime |  |
| Staff\_address |  | Varchar(240) |  |
| Mgr\_code |  | int |  |
| Staff\_sal |  | decimal (10,2) |  |

1. Book\_Master

|  |  |  |
| --- | --- | --- |
| **Name** | **Null?** | **Type** |
| Book\_Code | Not Null | int |
| Book\_Name | Not Null | Varchar(50) |
| Book\_pub\_year |  | int |
| Book\_pub\_author | Not Null | Varchar(50) |
| Book\_category | Not null | Varchar(10) |

1. Book\_Transaction

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Null?** | **Type** |  |
| Book\_Code |  | int | Fk ->Book\_master |
| Student\_code | Null | int | FK->Student\_master |
| Staff\_code | Null | int | FK->Staff\_master |
| Book\_Issue\_date | Not Null | Datetime |  |
| Book\_expected\_return\_date | Not Null | Datetime |  |
| Book\_actual\_return\_date | Null | Datetime |  |

## Appendix B: Table of Figures

[Figure 1: Connecting to SQL Server 2012 5](#_Toc453854739)

[Figure 2: Result of Sequence 11](#_Toc453854740)

[Figure 3: Attempting to insert the same row twice raises an exception 26](#_Toc453854741)

[Figure 4:Re-throwing the exception shows actual error, & the query completes with errors 27](#_Toc453854742)