



### Topik

1. Execution plan
2. Pemrograman T-SQL
3. Error Handling

### Tujuan

1. Memahami komponen dari query dengan performa yang baik
2. Memahami indexes dan statistics pada SQL Server
3. Mahasiswa memahami cara menggunakan elemen bahasa T-SQL dalam pemrograman dasar.
4. Mahasiswa memahami tentang batches dan bagaimana penanganannya dalam SQL Server.
5. Mahasiswa memahami cara mendeklarasikan & menugaskan variabel dan sinonim.
6. Mahasiswa memahami cara menggunakan blok IF dan WHILE dalam flow program.
7. Mahasiswa memahami bagaimana SQL Server menangani error yang muncul di kode T-SQL.
8. Mahasiswa memahami cara mengimplementasikan penanganan exception yang terstruktur dalam T-SQL.
9. Mahasiswa memahami cara mendapatkan informasi tentang error dari system objects.

### Petunjuk Umum

1. Ikuti langkah-langkah pada bagian-bagian praktikum sesuai dengan urutan yang diberikan.
2. Anda dapat menggunakan SQL Server 2012 Standard Edition untuk mencoba praktikum pada jobsheet ini. Sesuaikan dengan kondisi komputer Anda.
3. Jawablah semua pertanyaan bertanda **[Soal-X]** yang terdapat pada langkah-langkah tertentu di setiap bagian praktikum.
4. Dalam setiap langkah pada praktikum terdapat penjelasan yang akan membantu Anda dalam menjawab pertanyaan-pertanyaan pada petunjuk nomor 3, maka baca dan kerjakanlah semua bagian praktikum dalam jobsheet ini.
5. Tulis jawaban dari soal-soal pada petunjuk nomor 3 pada sebuah laporan yang dikerjakan menggunakan aplikasi word processing (Word, OpenOffice, atau yang lain yang sejenis). Eksport sebagai file **PDF** dengan format nama sebagai berikut:
  - **BDL\_Tugas13\_Kelas\_2DigitNomorAbsen\_NamaLengkapAnda.pdf**
  - Contoh:
    - o **BDL\_Tugas13\_TI2Q\_99\_Suneo.pdf**
  - Perhatikan baik-baik format penamaannya.
  - Kumpulkan file PDF tersebut sebagai laporan praktikum kepada dosen pengampu.
  - Selain pada nama file, cantumkan juga identitas Anda pada halaman pertama laporan tersebut.

## TOPIK 13.1 – QUERY PERFORMANCE

### Praktikum – Bagian 1: Percobaan Viewing Query Execution Plans

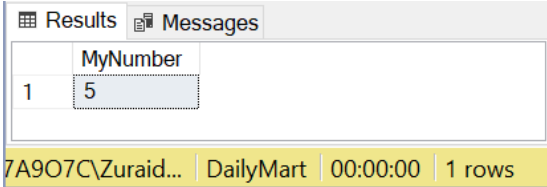
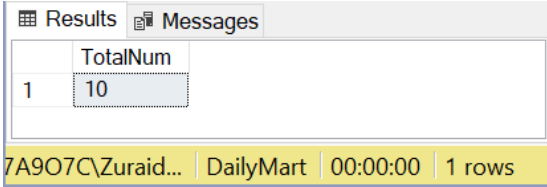
Langkah	Keterangan
1	Pada bagian ini Anda akan melakukan percobaan untuk membuat dan menambahkan isi tabel Sales.TempOrders
2	<p>Departemen IT akan membuat dan mengisi tabel sample bernama Sales.TempOrders dengan perintah berikut ini :</p> <pre>IF OBJECT_ID('Sales.TempOrders') IS NOT NULL DROP TABLE Sales.TempOrders; SELECT     OrderID, CustomerID, EmployeeID, OrderDate, RequiredDate, ShippedDate,     ShipperID, Freight, ShipName, ShipAddress, ShipCity, ShipRegion,     ShipPostalCode, ShipCountry INTO Sales.TempOrders FROM Sales.Orders AS o CROSS JOIN dbo.Nums AS n WHERE n.n &lt;= 120;</pre>
3	[Soal-1] Jalankan dan catat outputnya
4	[Soal-2] Buat perintah TSQL untuk mengembalikan kolom OrderID, CustomerID, dan OrderDate dari table Sales.TempOrders
5	Highlight perintah soal-2 dan tunjukkan estimated execution plan. Amati elemen-elemen yang ditampilkan.
6	Arahkan kursor ke Table Scan pada Execution Plans dan lihat properti yang ditampilkan di Yellow tooltip box. Perhatikan kata “estimated” di berbagai properti.
7	[Soal-3] Posisikan kursor mouse diatas panah antara operator SELECT dan operator Table Scan dalam execution plans. Informasi apa saja yang ditampilkan?
8	[Soal-4] Tampilkan semua properti operator SELECT dalam execution plans dengan mengklik kanan operator dan pilih “Properties”
9	[Soal-5] Klik tombol Include Actual Execution Plan pada SQL Editor toolbar (atau tekan ctrl+M pada keyboard) dan execute dengan query SELECT
10	Analisa actual execution plan dan jelaskan!
11	[Soal-6] Salin perintah SELECT sebelumnya dan lakukan modifikasi untuk mengambil satu baris dengan menggunakan klausa TOP. Tunjukkan estimated execution plan.
12	[Soal-7] Bandingkan execution plan dengan pada soal-2 dan soal-6. Komponen/operator apa yang berbeda?

## Praktikum – Bagian 2: Percobaan Viewing Index Usage dan Penggunaan perintah SET STATISTICS

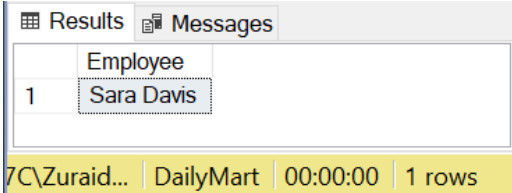
Langkah	Keterangan
1	Sekarang anda akan mempelajari tentang bagaimana cara mengaktifkan I/O statistic, menggunakan indexing dan mengimplementasikan perintah SET STATISTIC.
2	Eksekusi perintah dibawah ini CREATE CLUSTERED INDEX CX_Sales_TempOrders_orderdate ON Sales.TempOrders (OrderDate ASC);
3	[Soal-8] Tuliskan perintah SELECT untuk mengambil kolom OrderID, CustomerID, dan OrderDate dari tabel Sales.TempOrders. Tampilkan baris dengan tahun pemesanan 2017 dan bulan pemesanan sama dengan 6.
4	Aktifkan I/O Statistic dengan mengeksekusi perintah SET STATISTICS IO ON
5	[Soal-9] Salin query pada [soal 8] kemudian eksekusi. Perhatikan jumlah logical reads yang ditampilkan pada message tab. Angka tersebut berdasarkan I/O Statistic. Apa yang dimaksud dengan logical reads?
6	[Soal-10] Salin perintah query pada [soal 8] dan modifikasi dengan mengganti klausa WHERE dengan menggunakan range (tanggal awal dan akhir) berdasarkan kolom OrderDate. Jalankan kemudian catat hasilnya.
7	Perhatikan angka pada logical reads pada message tab.
8	Tampilkan query execution plan. Perhatikan operator baru bernama Clustered Index Seek.
10	[Soal-11] Salin query pada pada soal-8 dan soal-10. Block kedua query tersebut kemudian jalankan
11	[Soal-12] Perhatikan perbedaan logical reads. Meskipun hasilnya sama, query pada soal 10 memindai data lebih sedikit dibandingkan query soal 8. Analisa mengapa hal tersebut bisa terjadi?

## TOPIK 13.2 – T-SQL PROGRAMMING

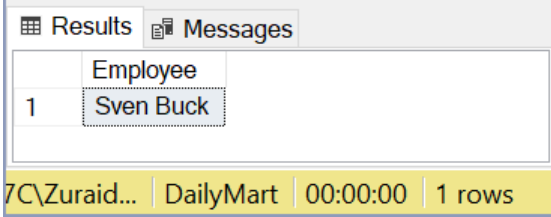
### Praktikum Bagian 3 – DEKLARASI VARIABEL & BATCH: Mendeklarasikan variabel dan mendapatkan nilai variabel

Langkah	Keterangan
1	<p><b>[Soal-13]</b> Deklarasikan sebuah variable bernama <code>@num</code> bertipe integer lalu inisialisasi dengan nilai 5. Tampilkan nilai variabel tersebut dengan menggunakan alias <code>MyNumber</code> lalu eksekusi.</p> <p>Hasil yang benar ditunjukkan pada gambar berikut:</p>  <p>The screenshot shows a SQL Server Results window with two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with one column named 'MyNumber' and one row containing the value '5'. Below the table, the status bar indicates '7A9O7C\Zuraid...   DailyMart   00:00:00   1 rows'.</p>
2	<p><b>[Soal-14]</b> Pada akhir script di atas (langkah 1) tambahkan batch delimiter (GO). Selanjutnya, deklarasikan 2 variabel bernama <code>@num1</code> dan <code>@num2</code> bertipe integer. Set nilainya masing-masing 4 dan 6. Buat sebuah query dengan SELECT yang menampilkan jumlah kedua variable tersebut sebagai <code>TotalNum</code>.</p> <p>Hasil yang benar ditunjukkan pada gambar berikut:</p>  <p>The screenshot shows a SQL Server Results window with two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with one column named 'TotalNum' and one row containing the value '10'. Below the table, the status bar indicates '7A9O7C\Zuraid...   DailyMart   00:00:00   1 rows'.</p>

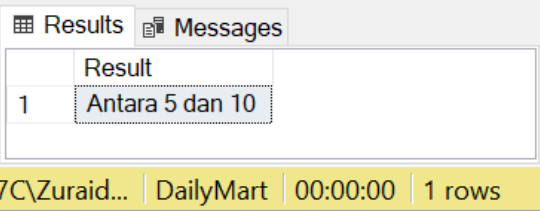
## Praktikum Bagian 4 – DEKLARASI VARIABEL & BATCH: Memberi nilai terhadap variabel menggunakan query SELECT

Langkah	Keterangan
1	<p><b>[Soal-15]</b> Deklarasikan variabel <code>@employeeName</code> bertipe <code>nvarchar(30)</code>. Selanjutnya, set nilai variabel tersebut sebagai hasil query SELECT terhadap tabel <b>HR.Employees</b> dengan <code>EmployeeID</code> bernilai 1. Gabungkan kolom <code>FirstName</code> dan <code>LastName</code> dengan dipisahkan spasi.</p> <p>Terakhir, tampilkan nilai variabel <code>@employeeName</code> dengan menggunakan query SELECT dan beri nama alias sebagai <code>Employee</code>.</p> <p>Hasil yang benar ditunjukkan pada gambar berikut:</p>  <p>The screenshot shows a SQL Server query results window. The 'Results' tab is active, displaying a table with one row. The first column is labeled 'Employee' and the value is 'Sara Davis'. The status bar at the bottom indicates '1 rows'.</p>
2	<p><b>[Soal-16]</b> Apakah yang terjadi apabila query SELECT terhadap tabel <b>HR.Employees</b> mengembalikan lebih dari 1 baris? Lakukan uji coba misalnya dengan menghilangkan filter "WHERE <code>EmployeeID = 1</code>".</p>

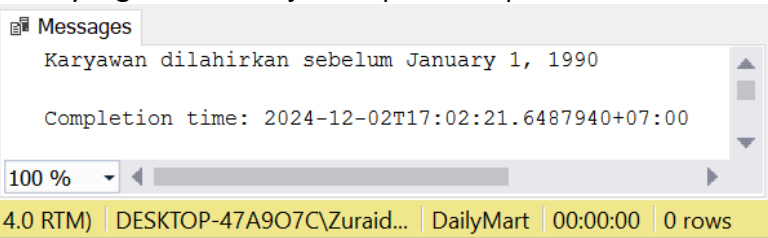
## Praktikum Bagian 5 – DEKLARASI VARIABEL & BATCH: Menggunakan sebuah variabel dalam klausa WHERE

Langkah	Keterangan
1	<p><b>[Soal-17]</b> Salinlah script T-SQL dari <b>[Soal- 15]</b>. Lakukan modifikasi dengan mendeklarasikan sebuah variabel baru bernama <code>@employeeID</code> bertipe integer lalu insialisasi dengan nilai 5. Gunakan variabel <code>@employeeID</code> sebagai filter untuk kolom <code>EmployeeID</code> dalam klausa WHERE</p> <p>Hasil yang benar ditunjukkan pada tampilan berikut:</p>  <p>The screenshot shows a SQL Server query results window. The 'Results' tab is active, displaying a table with one row. The first column is labeled 'Employee' and the value is 'Sven Buck'. The status bar at the bottom indicates '1 rows'.</p>
2	<p><b>[Soal-18]</b> Salinlah script T-SQL dari <b>[Soal-17]</b> di atas. Tambahkan <i>batch delimiter</i> GO beserta query SELECT seperti di bawah ini:</p> <pre>... GO SELECT @employeeName AS Employee;</pre> <p>Setelah mengeksekusi script tersebut, apakah yang terjadi? Mengapa demikian?</p>

## Praktikum Bagian 6 – CONTROL OF FLOW: Membuat conditional logic sederhana


Langkah	Keterangan
1	<p><b>[Soal-19]</b> Buatlah sebuah script T-SQL dengan mendeklarasikan variabel <code>@result</code> bertipe <code>nvarchar(20)</code> dan variabel <code>@i</code> bertipe integer dengan nilai 8.</p> <p>Tambahkan statement IF yang memenuhi <i>logic</i> di bawah ini:</p> <ul style="list-style-type: none"> <li>• Jika variabel <code>@i</code> bernilai kurang dari 5, set nilai variabel <code>@result</code> menjadi “Kurang dari 5”</li> <li>• Jika variabel <code>@i</code> bernilai antara 5 dan 10, set nilai variabel <code>@result</code> menjadi “Antara 5 dan 10”</li> <li>• Jika variabel <code>@i</code> bernilai lebih dari 10, set nilai variabel <code>@result</code> menjadi “Lebih dari 10”</li> <li>• Selain dari itu, set nilai variabel <code>@result</code> menjadi “Unknown”</li> </ul> <p>Di bagian akhir, tambahkan sebuah query SELECT untuk menampilkan nilai variabel <code>@result</code> dengan memberi alias <code>Result</code>.</p> <p>Eksekusi script yang sudah dibuat dan bandingkan dengan hasil berikut ini:</p>  <p>The screenshot shows a SQL Server Results window with two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with one column named 'Result' and one row containing the value 'Antara 5 dan 10'. Below the table, a status bar indicates '7C\Zuraid...   DailyMart   00:00:00   1 rows'.</p>
2	<p><b>[Soal-20]</b> Modifikasilah script dari [Soal-19] di atas dengan mengganti statement IF menjadi ekspresi CASE dan pastikan hasilnya sama.</p>

## Praktikum Bagian 7 – CONTROL OF FLOW: Membuat dan mengeksekusi stored procedure

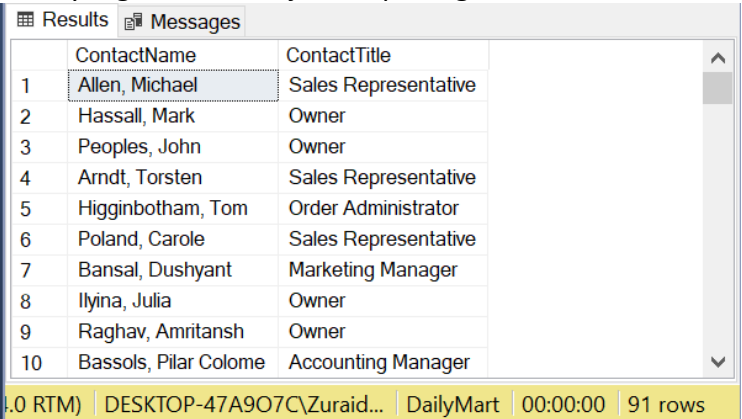
Langkah	Keterangan
1	<p>Salin &amp; jalankan script T-SQL berikut ini:</p> <pre> CREATE PROCEDURE Sales.CheckPersonBirthDate     @employeeID INT,     @otherDate date AS DECLARE @birthDate date;  SET @birthDate = (SELECT BirthDate FROM HR.Employees WHERE EmployeeID = @employeeID)  IF @birthDate &lt; @otherDate     PRINT 'Karyawan dilahirkan sebelum ' + FORMAT(@otherDate, 'MMMM d, yyyy', 'en-US'); ELSE     PRINT 'Karyawan dilahirkan pada atau setelah ' + FORMAT(@otherDate, 'MMMM d, yyyy', 'en-US');</pre>
2	<p><b>[Soal-21]</b> Stored procedure bernama <b>Sales.CheckPersonBirthDate</b> pada di atas mempunyai 2 parameter, yakni @employeeID (untuk memfilter data employee) dan @otherDate (untuk tanggal yang akan dibandingkan).</p> <p>Lakukan perintah EXECUTE pada stored procedure tersebut dengan nilai parameter @employeeID = 3 dan @otherDate yang diset tanggal January 1, 1990.</p> <p>Hasil yang benar ditunjukkan pada tampilan berikut:</p> 

## Praktikum Bagian 8 – CONTROL OF FLOW: Melakukan loop/pengulangan menggunakan pernyataan WHILE

Langkah	Keterangan
1	<p><b>[Soal-22]</b> Buatlah sebuah script T-SQL yang berisi looping/pengulangan dengan mengikuti langkah berikut:</p> <ul style="list-style-type: none"> <li>• Pertama, deklarasikan sebuah variabel @i yang bertipe integer bernilai 1</li> <li>• Lalu buatlah sebuah pengulangan dengan menggunakan WHILE. Selama nilai variabel @i kurang dari 10, cetak nilai variabel @i dan tambahkan nilai @i secara incremental dengan menambah 1 (@i+1).</li> </ul> <p>Hasil yang benar ditunjukkan pada tampilan berikut:</p>

	
--	--

## Praktikum Bagian 9 – SYNONYMS: Membuat dan menggunakan synonym

Langkah	Keterangan
1	<p><b>[Soal-23]</b> Tulislah sebuah script T-SQL yang membuat SYNONYM bernama <b>dbo.Pelanggan</b> untuk tabel <b>Sales.Customers</b>. Eksekusi script tersebut.</p> <p>Selanjutnya, buatlah query SELECT terhadap synonym <b>dbo.Pelanggan</b> yang menampilkan kolom <i>ContactTitle</i> dan <i>ContactName</i>. Eksekusi script tersebut.</p> <p>Hasil yang benar ditunjukkan pada gambar berikut:</p> 
2	<p>Untuk menghapus synonym yang dibuat sebelumnya, jalankan script berikut:</p> <p><b>DROP SYNONYM dbo.Pelanggan;</b></p>



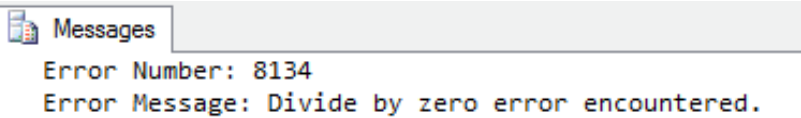
### TOPIK 13.3 – ERROR HANDLING

#### Praktikum Bagian 10 – TRY / CATCH: Membuat blok TRY / CATCH sederhana

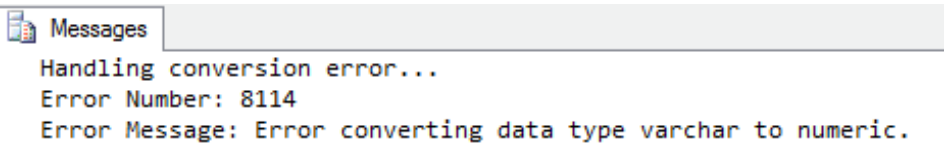
Langkah	Keterangan
1	<p>Salin dan eksekusi pernyataan SELECT berikut ini:</p> <pre>SELECT CAST(N'Ini teks loh' AS int);</pre> <p>Perhatikan error yang terjadi saat dieksekusi.</p>
2	<p><b>[Soal-24]</b> Buatlah konstruksi TRY / CATCH dengan menempatkan query pada Langkah 1 tersebut dalam blok TRY. Sedangkan dalam blok CATCH, isikan perintah untuk menampilkan pesan “Error”. Jalankan script T-SQL tersebut, maka tab Messages, akan menampilkan pesan:</p> <pre>(0 row(s) affected) Error</pre>

#### Praktikum Bagian 11 – TRY / CATCH: Menampilkan kode & pesan error

Langkah	Keterangan
1	<p>Salin dan eksekusi script T-SQL berikut ini dan perhatikan hasilnya:</p> <pre>DECLARE @num varchar(20) = '0';  BEGIN TRY     PRINT 5.0/ CAST(@num AS numeric(10,4)) END TRY BEGIN CATCH END CATCH</pre>
2	<p><b>[Soal-25]</b> Pada langkah 1 di atas, jika memperhatikan nilai variabel @num, semestinya dihasilkan error “division by zero”, tetapi nyatanya tidak. Mengapa demikian?</p>
3	<p><b>[Soal-26]</b> Modifikasilah script T-SQL pada Langkah 1 di atas dengan menambahkan 2 statement PRINT pada blok CATCH. Statement yang pertama untuk menampilkan nomer error dengan menggunakan fungsi <code>ERROR_NUMBER()</code> dan statemtn kedua untuk menampilkan pesan error dengan memakai fungsi <code>ERROR_MESSAGE()</code>.</p> <p>Untuk memperjelas, tambahkan string label “Error Number:” pada pesan pertama dan string label “Error Message:” pada pesan kedua.</p> <p>Eksekusi dan bandingkan hasilnya dengan tampilan berikut:</p>

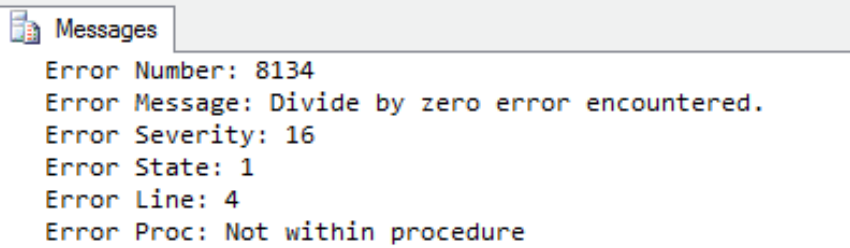
	
4	Sebagai uji coba, masih dengan script T-SQL yang sama, ubah nilai variabel @num dari 0 menjadi 'A'. Bagaimana outputnya?
5	Sebagai uji coba, masih dengan script T-SQL yang sama, ubah nilai variabel @num dari 'A' menjadi 1000000000. Bagaimana outputnya?

### Praktikum Bagian 12 – TRY / CATCH: Menambahkan conditional logic pada blok CATCH

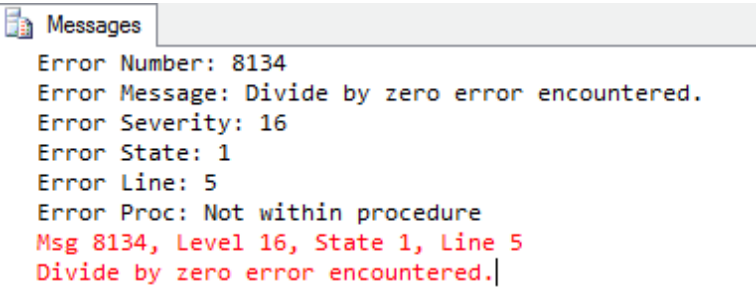
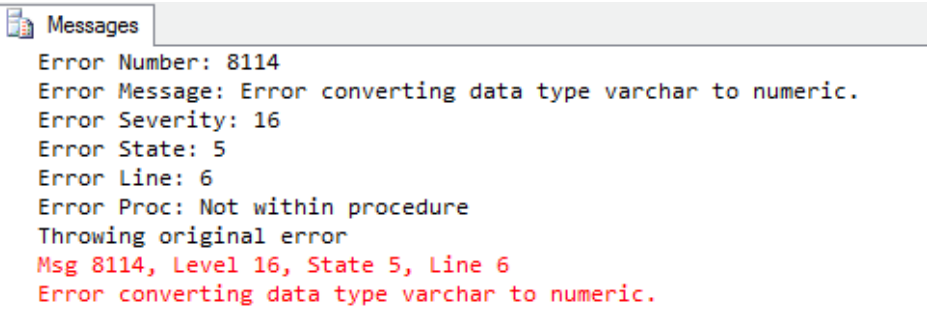
Langkah	Keterangan
1	<p>[Soal-27] Salin script T-SQL pada soal sebelumnya. Set kembali nilai variabel @num dengan 'A'. Lakukan modifikasi dengan menambahkan pernyataan IF pada bagian blok CATCH sebelum pernyataan PRINT.</p> <ul style="list-style-type: none"> <li>Jika error number = 245 atau 8114, tampilkan pesan "Handling conversion error..."</li> <li>Jika tidak sama dengan 245 atau 8114, tampilkan pesan "Handling NON conversion error..."</li> </ul> <p>Bandingkan hasilnya dengan tampilan berikut:</p> 
2	Lakukan uji cobadengan mengubah nilai variabel @num dari 'A' menjadi '0'. Bagaimana outputnya?

### Praktikum Bagian 13 – TRY / CATCH: Mengeksekusi stored procedure pada blok CATCH

1	<p>Salin dan eksekusi script T-SQL yang membuat sebuah stored procedure dbo.GetErrorInfo di bawah ini.</p> <pre>CREATE PROCEDURE dbo.GetErrorInfo AS PRINT 'Error Number: ' + CAST(ERROR_NUMBER() AS varchar(10)); PRINT 'Error Message: ' + ERROR_MESSAGE(); PRINT 'Error Severity: ' + CAST(ERROR_SEVERITY() AS varchar(10)); PRINT 'Error State: ' + CAST(ERROR_STATE() AS varchar(10)); PRINT 'Error Line: ' + CAST(ERROR_LINE() AS varchar(10)); PRINT 'Error Proc: ' + COALESCE(ERROR_PROCEDURE(), 'Not within procedure');</pre>
2	[Soal-28] Buatlah sebuah konstruksi TRY / CATCH. Pada block TRY lakukan pembagian dengan

	<p>0 dan pada blok CATCH, lakukan eksekusi stored procedure yang telah dibuat pada Langkah 1 di atas, lalu jalankan.</p> 
--	---

#### Praktikum Bagian 14 – THROW: Menggunakan THROW untuk mengirimkan kembali pesan error

1	<p><b>[Soal-29]</b> Modifikasi script T-SQL dari soal sebelumnya dengan menambahkan perintah THROW yang ditempatkan setelah pernyataan eksekusi stored procedure. Bandingkan hasilnya dengan contoh berikut.</p> 
2	<p><b>[Soal-30]</b> Modifikasi script T-SQL dari soal sebelumnya dengan mengganti perintah THROW dengan pernyataan IF.</p> <ul style="list-style-type: none"> <li>• IF error number = 8114, tampilkan pesan “Handling division by zero...”</li> <li>• Jika tidak, tampilkan pesan “Throwing original error”</li> </ul> <p>Set nilai variabel @num sebagai ‘A’, lalu eksekusi script T-SQL tersebut. Bandingkan hasilnya dengan tampilan berikut:</p> 
3	<p>Untuk menghapus stored procedure yang dibuat sebelumnya, jalankan script berikut:</p> <pre>DROP PROCEDURE Sales.CheckPersonBirthDate; DROP PROCEDURE dbo.GetErrorInfo;</pre>

--- Selamat Mengerjakan ---