70-461 Exam Notes

Ron Wu

Last update: 5/26/15

Create database objects (24%)

1. Create and alter tables using T-SQL syntax (simple statements)
   - https://msdn.microsoft.com/en-us/library/ms188264.aspx
   - Create a base table
   - Table name-- identifiers: regular and delimited[],””. Letters in the Unicode, Decimal numbers, the first character must be a letter or an underscore, cannot be a digit. Variables begin with an at sign (@), Temporary tables or procedures must begin with a number sign (#); Subsequent identifier characters can include: Letters Numerals at sign (@), the dollar sign ($), the number sign (#), and the underscore; A regular identifier cannot be a T-SQL reserved word and cannot include embedded spaces; “” requires SET QUOTED_IDENTIFIER ON, which is ON by default. maximum length of 128 characters.
   - table variables,

```
CREATE TABLE Production.Categories(
       categoryid INT IDENTITY(1,1) NOT NULL,
       categoryname NVARCHAR(15) NOT NULL,
       description NVARCHAR(200) NOT NULL DEFAULT ('')
       longtext AS categoryname + description -- concatenate string
       CONSTRAINT PK_CategoriesTest PRIMARY KEY(categoryid)
       ) ON [PRIMARY] -- primary filegroup
       WITH (DATA_COMPRESSION = ROW); --row level compression
GO
```

   - Schema, dbo is the default. Schemas are not nested

```
CREATE SCHEMA Production
-- moves the table to the Sales database schema
ALTER SCHEMA Sales TRANSFER Production.Categories;
--alter its compression
ALTER TABLE Production.Categories
REBUILD WITH (DATA_COMPRESSION = PAGE);
```

   - Alter; cannot change column name, add identity property, remove identity property

```
--Adding columns
ALTER TABLE Production.Categories
       ADD categoryname NVARCHAR(15) NOT NULL;
GO
--alter Column size
ALTER TABLE Production.Categories
       ALTER COLUMN desc NVARCHAR(500) NOT NULL;
GO
--Make the column allow NULL
```

```
ALTER TABLE Production.Categories
       ALTER COLUMN description NVARCHAR(500) NULL ;
GO
--change constraint
```

- Identity, only one column can have IDENTITY.

```
-- identity-related functions
SELECT
  SCOPE_IDENTITY() --returns the last identity value generated in
                   -- the session in the current scope
  @@IDENTITY --returns the last identity value generated in the
                   --session regardless of scope;
  IDENT_CURRENT('Sales.MyOrders') --the last identity used in the table.

-- create sequence
IF OBJECT_ID(N'Sales.SeqOrderIDs', N'SO') IS NOT NULL DROP SEQUENCE Sales.SeqOrderIDs;

CREATE SEQUENCE Sales.SeqOrderIDs AS INT
  MINVALUE 1
  CYCLE;

SELECT TYPE_NAME(system_type_id) AS type,
  start_value, minimum_value, current_value, increment, is_cycling
FROM sys.sequences
WHERE object_id = OBJECT_ID(N'Sales.SeqOrderIDs', N'SO');

-- request a new value; run 3 times
SELECT NEXT VALUE FOR Sales.SeqOrderIDs;

-- alter sequence properties (all besides type, cannot change sequence type)
ALTER SEQUENCE Sales.SeqOrderIDs
  RESTART WITH 1;

IF OBJECT_ID(N'Sales.MyOrders', N'U') IS NOT NULL DROP TABLE Sales.MyOrders;
GO

CREATE TABLE Sales.MyOrders
(
  orderid INT NOT NULL
    CONSTRAINT PK_MyOrders_orderid PRIMARY KEY,
  custid  INT NOT NULL
    CONSTRAINT CHK_MyOrders_custid CHECK(custid > 0),
  empid   INT NOT NULL
    CONSTRAINT CHK_MyOrders_empid CHECK(empid > 0),
  orderdate DATE NOT NULL
);

INSERT INTO Sales.MyOrders(orderid, custid, empid, orderdate)
  SELECT
    NEXT VALUE FOR Sales.SeqOrderIDs OVER(ORDER BY orderid),
    custid, empid, orderdate
  FROM Sales.Orders
  WHERE custid = 1;

-- add as default constraint
ALTER TABLE Sales.MyOrders
  ADD CONSTRAINT DFT_MyOrders_orderid
```

```
    DEFAULT(NEXT VALUE FOR Sales.SeqOrderIDs) FOR orderid;
```

- Drop

```
--drop the table
IF OBJECT_ID('Production.CategoriesTest','U') IS NOT NULL
    DROP TABLE Production.CategoriesTest;
GO
```

- Table expression: derived table, common table expression, views, inline table-valued functions. ORDED BY is not allowed with some exception like OFFSET-FETCH.
- Derived table,

```
SELECT categoryid, productid, productname, unitprice
FROM (SELECT
        ROW_NUMBER() OVER(PARTITION BY categoryid
                          ORDER BY unitprice, productid) AS rownum,
        categoryid, productid, productname, unitprice
      FROM Production.Products) AS D
WHERE rownum <= 2;
```

- temporary table(local#, global##), table variable(@)

```
CREATE TABLE #T1
(
  col1 INT NOT NULL
);

INSERT INTO #T1(col1) VALUES(10);

EXEC('SELECT col1 FROM #T1;'); -- table variable not visible inner levels, error if
@T1

-- table variable
DECLARE @T1 AS TABLE
(
  col1 INT NOT NULL
);

INSERT INTO @T1(col1) VALUES(10);

SELECT name FROM tempdb.sys.objects WHERE name LIKE '#%';
```

- CTE

```
WITH C AS
(
  SELECT
    ROW_NUMBER() OVER(PARTITION BY categoryid
                      ORDER BY unitprice, productid) AS rownum,
    categoryid, productid, productname, unitprice
  FROM Production.Products
)
SELECT categoryid, productid, productname, unitprice
```

```
FROM C
WHERE rownum <= 2;

-- Recursive CTE
WITH EmpsCTE AS
(
  SELECT empid, mgrid, firstname, lastname, 0 AS distance
  FROM HR.Employees
  WHERE empid = 9

  UNION ALL

  SELECT M.empid, M.mgrid, M.firstname, M.lastname, S.distance + 1 AS distance
  FROM EmpsCTE AS S
    JOIN HR.Employees AS M
      ON S.mgrid = M.empid
)
SELECT empid, mgrid, firstname, lastname, distance
FROM EmpsCTE;
GO
```

2. Create and alter views (simple statements)
   - https://msdn.microsoft.com/en-us/library/ms187956%28v=sql.110%29.aspx

```
CREATE VIEW Sales.OrderTotalsByYear
  WITH SCHEMABINDING –the TALBE cannot be altered without dropping the view
  WITH ENCRYPTION
  WITH VIEW_METADATA
AS
SELECT
  YEAR(O.orderdate) AS orderyear,
  SUM(OD.qty) AS qty
FROM Sales.Orders AS O
  JOIN Sales.OrderDetails AS OD
    ON OD.orderid = O.orderid
GROUP BY YEAR(orderdate);
WITH CHECK OPTION --alter rows that will not fall outside WHERE
GO
--call
SELECT orderyear, qty
FROM Sales.OrderTotalsByYear;
```

- CREATE VIEW statement must be the first statement in a batch. View name is unquie in a batabase. Only one SELECT or use UNION or UNION ALL. Cannot have ORDER BY, cannot pass parameters, cannot create a table. Cannot refer to a temporary table. INSERT, DELECT, UPDATE used in one table. Directly to the column not to aggregate function, …
- ALTER

```
ALTER VIEW Sales.OrderTotalsByYear
  WITH SCHEMABINDING
AS
SELECT
  O.shipregion,
  YEAR(O.orderdate) AS orderyear,
```

```
  SUM(OD.qty) AS qty
FROM Sales.Orders AS O
  JOIN Sales.OrderDetails AS OD
    ON OD.orderid = O.orderid
GROUP BY YEAR(orderdate), O.shipregion;
GO
```

- DROP

```
IF OBJECT_ID('Sales.OrderTotalsByYear', 'V') IS NOT NULL
       DROP VIEW Sales.OrderTotalsByYear;
GO
```

- Check sys views

```
SELECT name, object_id, principal_id, schema_id, type
FROM sys.views;  --combined with sys.sql_modules

-- also query the INFORMATION_SCHEMA.TABLES system view
SELECT SCHEMA_NAME, TABLE_NAME, TABLE_TYPE
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_TYPE = 'VIEW';
```

- Index view,

```
-- Create a view from the query
IF OBJECT_ID(N'Sales.QuantityByCountry', N'V') IS NOT NULL
   DROP VIEW Sales.QuantityByCountry;
GO
CREATE VIEW Sales.QuantityByCountry
WITH SCHEMABINDING
AS
SELECT O.shipcountry, SUM(OD.qty) AS total_ordered,
 COUNT_BIG(*) AS number_of_rows
FROM Sales.OrderDetails AS OD
 INNER JOIN Sales.Orders AS O
  ON OD.orderid = O.orderid
GROUP BY O.shipcountry;
GO
-- Index the view
CREATE UNIQUE CLUSTERED INDEX idx_cl_shipcountry
ON Sales.QuantityByCountry(shipcountry);
GO


-- both implementations get same better efficiency
SELECT O.shipcountry, SUM(OD.qty) AS totalordered
FROM Sales.OrderDetails AS OD
 INNER JOIN Sales.Orders AS O
  ON OD.orderid = O.orderid
GROUP BY O.shipcountry;
-- or
SELECT shipcountry, total_ordered
FROM Sales.QuantityByCountry


DROP VIEW Sales.QuantityByCountry;
```

- partitioned views https://msdn.microsoft.com/en-us/library/ms190019.aspx

- Inline functions, allow parameterization, must have (), cannot change state of a database, cannot create or access temporary tables, cannot call stored procedure, cannot execute dynamic sql

```
IF OBJECT_ID('Sales.fn_extension', 'FN') IS NOT NULL
       DROP FUNCTION Sales.fn_extension
GO
CREATE FUNCTION Sales.fn_extension
(
  @unitprice AS MONEY,
  @qty AS INT
)
RETURNS MONEY
AS
BEGIN
    RETURN @unitprice * @qty
END;
GO

SELECT Orderid, unitprice, qty, Sales.fn_extension(unitprice, qty) AS extension
FROM Sales.OrderDetails;

SELECT Orderid, unitprice, qty, Sales.fn_extension(unitprice, qty) AS extension
FROM Sales.OrderDetails
WHERE Sales.fn_extension(unitprice, qty) > 1000;


IF OBJECT_ID (N'Sales.fn_OrderTotalsByYearCustShip', N'IF') IS NOT NULL
    DROP FUNCTION Sales.fn_OrderTotalsByYearCustShip;
GO
CREATE FUNCTION [Sales].[fn_OrderTotalsByYearCustShip] (@lowqty int, @highqty int)
RETURNS TABLE -- table-valued function
AS
RETURN
       (
       SELECT
         C.companyname AS customercompany,
         S.companyname AS shippercompany,
         YEAR(O.orderdate) AS orderyear,
         SUM(OD.qty) AS qty,
         CAST(SUM(OD.qty * OD.unitprice * (1 - OD.discount))
               AS NUMERIC(12, 2)) AS val
       FROM Sales.Orders AS O
         JOIN Sales.OrderDetails AS OD
              ON OD.orderid = O.orderid
         JOIN Sales.Customers AS C
              ON O.custid = C.custid
         JOIN Sales.Shippers AS S
              ON O.shipperid = S.shipperid
       GROUP BY YEAR(O.orderdate), C.companyname, S.companyname
       HAVING SUM(OD.qty) >= @lowqty AND SUM(OD.qty) <= @highqty
       );
GO

SELECT customercompany, shippercompany, orderyear, qty, val
FROM [Sales].[fn_OrderTotalsByYearCustShip] (100, 200)
ORDER BY customercompany, shippercompany, orderyear;
```

```
-- Multistatement table-valued UDF
IF OBJECT_ID('Sales.fn_FilteredExtension2', 'TF') IS NOT NULL
       DROP FUNCTION Sales.fn_FilteredExtension2;
GO
CREATE FUNCTION Sales.fn_FilteredExtension2
(
  @lowqty AS SMALLINT,
  @highqty AS SMALLINT
 )
RETURNS @returntable TABLE
(
       orderid  INT,
       unitprice  MONEY,
       qty  SMALLINT
)
AS
BEGIN
  INSERT @returntable
       SELECT orderid, unitprice, qty
       FROM Sales.OrderDetails
       WHERE qty BETWEEN @lowqty AND @highqty
  RETURN
END;
GO
```

- Drop

```
IF OBJECT_ID (N'Sales.fn_OrderTotalsByYearCustShip', N'IF') IS NOT NULL
    DROP FUNCTION Sales.fn_OrderTotalsByYearCustShip;
GO
```

3. Design views
   - Ensure code non regression by keeping consistent signature for procedure, views and function (interfaces); security implications, sql injection https://msdn.microsoft.com/en-us/library/ms161953%28SQL.105%29.aspx

4. Create and modify constraints (simple statements)
   - https://msdn.microsoft.com/library/ms188066%28v=sql.110%29.aspx
   - Unique name in the database, create surrogate key as support to natural key.
   - Create primary key, not null, must be unique, only one primary key(clustered index); if use column combination, no more than 16 columns and no more than 900 bytes of data per row.

```
ALTER TABLE Production.Categories
       ADD CONSTRAINT PK_Categories PRIMARY KEY(categoryid);
GO

-- To view key
SELECT *
FROM sys.key_constraints
WHERE type = 'PK';
-- Also the clustered index is created automatically
SELECT *
FROM sys.indexes
```

```
WHERE object_id = OBJECT_ID(N'Production.Categories', 'U') AND name = 'PK_Categories';
```

- unique constraint , allow NULL but only one row NULL

```
USE TSQL2012;
ALTER TABLE Production.Categories WITH CHECK  -- check any prior violations
       ADD CONSTRAINT UC_Categories UNIQUE (categoryname);
GO
-- to view
SELECT *
FROM sys.key_constraints
WHERE type = 'UQ';

ALTER TABLE Production.Products
    DROP  CONSTRAINT U_Productname;
```

- foreign key constraints, must declare with ALTER, unique name across database, the reference key must have unique index, both have to be some data type.

```
ALTER TABLE Production.[Products]  WITH CHECK  -- check any prior violations e.g.  --
some prior categoryid in products table does not link any categoryid in categories
       ADD  CONSTRAINT [FK_Products_Categories] FOREIGN KEY(categoryid)
       REFERENCES Production.Categories (categoryid) -- primary key in categories
GO

-- to view
SELECT *
FROM sys.foreign_keys
WHERE name = 'FK_Products_Categories';

-- Drop the foreign key constraint
ALTER TABLE Production.Products
    DROP CONSTRAINT FK_Products_Categories;
```

- check constraints, cannot reference to deleted value, must use trigger, no customize error message

```
ALTER TABLE Production.Products  WITH CHECK
       ADD  CONSTRAINT CHK_Products_unitprice
       CHECK  (unitprice>=0); -- NULL passes
GO

SELECT *
FROM sys.check_constraints
WHERE parent_object_id = OBJECT_ID(N'Production.Products', N'U');
```

- default constraints, allow INSERT to skip values

```
 CREATE TABLE Production.Products
(
…
  unitprice    MONEY        NOT NULL
    CONSTRAINT DFT_Products_unitprice DEFAULT(0),
  discontinued BIT          NOT NULL
    CONSTRAINT DFT_Products_discontinued DEFAULT(0)
);
```

```
SELECT *
FROM sys.default_constraints
WHERE parent_object_id = OBJECT_ID(N'Production.Products', 'U');
```

5. Create and alter DML triggers.
   - Inserted and deleted tables; nested triggers, maximum depth of nested trigger executions is 32; disable it

```
EXEC sp_configure 'nested triggers', 0;
RECONFIGURE
```

   - types of triggers; update functions; handle multiple rows in a session; performance implications of triggers
   - After triggers, defined only on permanent table

```
IF OBJECT_ID('Production.tr_ProductionCategories_categoryname', 'TR') IS NOT NULL
       DROP TRIGGER Production.tr_ProductionCategories_categoryname;
GO
CREATE TRIGGER Production.tr_ProductionCategories_categoryname
ON Production.Categories
AFTER INSERT, UPDATE – or FOR INSERT, UPDATE, because after trigger is default
AS
BEGIN
  IF @@ROWCOUNT = 0 RETURN; -- Must be 1st statement
  SET NOCOUNT ON;
  IF EXISTS (SELECT COUNT(*)
        FROM Inserted AS I
        JOIN Production.Categories AS C
          ON I.categoryname = C.categoryname
              GROUP BY I.categoryname
              HAVING COUNT(*) > 1 )
    BEGIN
      THROW 50000, 'Duplicate category names not allowed', 0; -- rollback
    END;
END;
GO
```

   - Instead of triggers, defined on permanent table and view

```
IF OBJECT_ID('Production.tr_ProductionCategories_categoryname', 'TR') IS NOT NULL
DROP TRIGGER Production.tr_ProductionCategories_categoryname;
GO
CREATE TRIGGER Production.tr_ProductionCategories_categoryname
ON Production.Categories
INSTEAD OF INSERT
AS
BEGIN
  SET NOCOUNT ON;
  IF EXISTS (SELECT COUNT(*)
        FROM Inserted AS I
        JOIN Production.Categories AS C
          ON I.categoryname = C.categoryname
              GROUP BY I.categoryname
              HAVING COUNT(*) > 0 )
    BEGIN
```

```
      THROW 50000, 'Duplicate category names not allowed', 0;
    END;
  ELSE
    INSERT Production.Categories (categoryname, description)
      SELECT categoryname, description FROM Inserted;
END;
GO
-- Cleanup
IF OBJECT_ID('Production.tr_ProductionCategories_categoryname', 'TR') IS NOT NULL
  DROP TRIGGER Production.tr_ProductionCategories_categoryname;
```

Work with data (27%)

6. Query data by using SELECT statements
   - https://msdn.microsoft.com/library/ms189499.aspx
   - SELECT>FROM>WHERE>GROUP BY>HAVING>ORDER BY
   - Logical process: FROM>WHERE>GROUP BY>HAVING>SELECT>ORDER BY

```
SELECT DISTINCT country, YEAR(hiredate) AS yearhired, COUNT(*) AS numemployees
FROM HR.Employees
WHERE hiredate >= '20030101'
GROUP BY country, YEAR(hiredate)
HAVING COUNT(*) > 1
ORDER BY country, yearhired DESC; --Default ascendsing -- DISTINCT must return one
row per country
```

   - Data type and building function
   - https://msdn.microsoft.com/en-us/library/ms187752%28v=SQL.110%29.aspx
     https://msdn.microsoft.com/en-us/library/ms174318%28v=SQL.110%29.aspx
   - exact numeric (INT, NUMERIC, BIGINT, SMALLINT, MONEY),
   - character strings (CHAR, VARCHAR),
   - Unicode character strings (NCHAR, NVARCHAR),
   - approximate numeric (FLOAT(8 bytes), REAL(4bytes)),
   - binary strings (BINARY, VARBINARY),
   - date and time (DATE(3 bytes), TIME, DATETIME2(6-8 bytes), SMALLDATETIME(4 bytes), DATETIME(8 bytes), DATETIMEOFFSET),
   - Other type (rowversion)
   - CAST, CONVERT, PARSE, DATEPART

```
DECLARE @f AS FLOAT = '29545428.022495';
SELECT CAST(@f AS NUMERIC(28, 14)) AS value;
GO
SELECT TRY_CAST('abc' AS INT);

SELECT
  GETDATE()           AS [GETDATE],
  CURRENT_TIMESTAMP   AS [CURRENT_TIMESTAMP],
  GETUTCDATE()        AS [GETUTCDATE],
  SYSDATETIME()       AS [SYSDATETIME],
```

```
  SYSUTCDATETIME()    AS [SYSUTCDATETIME],
  SYSDATETIMEOFFSET() AS [SYSDATETIMEOFFSET];

SELECT
  CAST(SYSDATETIME() AS DATE) AS [current_date],
  CAST(SYSDATETIME() AS TIME) AS [current_time];

SELECT DATEPART(month, '20120212');
SELECT DAY('20120212') AS theday,
SELECT DATENAME(month, '20090212');
-- fromparts
SELECT
  DATEFROMPARTS(2012, 02, 12),
  DATETIME2FROMPARTS(2012, 02, 12, 13, 30, 5, 1, 7),
  DATETIMEFROMPARTS(2012, 02, 12, 13, 30, 5, 997),
  DATETIMEOFFSETFROMPARTS(2012, 02, 12, 13, 30, 5, 1, -8, 0, 7),
  SMALLDATETIMEFROMPARTS(2012, 02, 12, 13, 30),
  TIMEFROMPARTS(13, 30, 5, 1, 7);

SELECT EOMONTH(SYSDATETIME());--2015-5-31—end of current month
SELECT DATEADD(year, 1, '20120212');
SELECT DATEDIFF(day, '20110212', '20120212');

SELECT SUBSTRING('abcde', 1, 3); -- 'abc'
SELECT LEFT('abcde', 3); -- 'abc'
SELECT RIGHT('abcde', 3); -- 'cde'
SELECT CHARINDEX(' ','abcde fghilmn'); -- 6
SELECT PATINDEX('%[0-9]%', 'abcd123efgh'); -- 5 -- [0-9]% string where first char is
a digit; %[0-9] string where last char is a digit; [^0-9]% string where first char is
not a digit;  [ABC]% string where first char is a letter A or B or C;  _D% string
where second char is a D;
SELECT LEN(N'xyz'); -- 3
SELECT DATALENGTH(N'xyz'); -- 6
SELECT REPLACE('.1.2.3.', '.', '/'); -- '/1/2/3/'
SELECT REPLICATE('0', 10); -- '0000000000'
SELECT STUFF(',x,y,z', 1, 1, ''); -- 'x,y,z'
SELECT UPPER('aBcD'); -- 'ABCD'
SELECT LOWER('aBcD'); -- 'abcd'
SELECT RTRIM(LTRIM('   xyz   ')); -- 'xyz'
SELECT FORMAT(1759, '000000000'); -- '0000001759'

SELECT empid, firstname, lastname
FROM HR.Employees
WHERE lastname LIKE N'D%';


SELECT orderid, orderdate, empid, custid
FROM Sales.Orders
WHERE orderdate >= '20070201' AND orderdate < '20070301';
```

- COALESCE(NULL, OUTPUT) = OUTPUT, NULL + ‘STRING’ = NULL, CASE, ISNULL

```
COALESCE(<exp1>,<exp2>,...)

is equivalent

CASE
```

```
WHEN <exp1> IS NOT NULL THEN <exp1>
WHEN <exp2> IS NOT NULL THEN <exp2>
...
END
```

- Filer with NULL

```
SELECT empid, firstname, lastname, country, region, city
FROM HR.Employees
WHERE region <> N'WA'
   OR region IS NULL;
```

- TOP (doesn't require ORDER BY) and OFFSET-FETCH (require ORDER BY)

```
SELECT TOP (3) orderid, orderdate, custid, empid
FROM Sales.Orders
ORDER BY (SELECT NULL); --optional

SELECT TOP (1) PERCENT orderid, orderdate, custid, empid
FROM Sales.Orders
ORDER BY orderdate DESC;

SELECT TOP (3) WITH TIES orderid, orderdate, custid, empid
FROM Sales.Orders
ORDER BY orderdate DESC;

SELECT orderid, orderdate, custid, empid
FROM Sales.Orders
ORDER BY (SELECT NULL)
OFFSET 0 ROWS FETCH FIRST 3 ROWS ONLY; -- must have OFFSET

OFFSET 5 ROWS FETCH NEXT 5 ROWS ONLY;

OFFSET 5 ROWS
```

- Inner join (default), left/right outer join, cross join

```
SELECT C.custid, C.companyname, O.orderid, O.orderdate
FROM Sales.Customers AS C
  LEFT OUTER JOIN Sales.Orders AS O
    ON C.custid = O.custid;
```

- except, intersect, INTERSECT precedes UNION and EXCEPT, and UNION and EXCEPT are considered equal from left to right.

```
SELECT country, region, city
FROM HR.Employees

INTERSECT

SELECT country, region, city
FROM Sales.Customers;
```

- synonyms refer to database object, chain is not allowed, no alter synonym, to change, must drop. Can refer to non existing object

```
CREATE SYNONYM dbo.Categories FOR Production.Categories;
GO
```

```
--Then the end user can select from Categories without needing to specify a schema
SELECT categoryid, categoryname, description
FROM Categories;

DROP SYNONYM dbo.Categories
```

- write and perform queries efficiently using the new (SQL 2005/8->) code items such as, and joins (except, intersect); implement logic which uses dynamic SQL and system metadata; write efficient,; determine what code may or may not execute based on the tables provided; given a table with constraints, determine which statement set would load a table;
- creating full text catalogs and index. Search for simple term, prefix term, generation term, proximity term, thesaurus term, weighted terms, statistical semantic, similar documents. Install Full-Text Search

```
CREATE TABLE dbo.Documents
(
  id INT IDENTITY(1,1) NOT NULL,
  title NVARCHAR(100) NOT NULL,
  doctype NCHAR(4) NOT NULL,
  docexcerpt NVARCHAR(1000) NOT NULL,
  doccontent VARBINARY(MAX) NOT NULL,
  CONSTRAINT PK_Documents
   PRIMARY KEY CLUSTERED(id)
);
GO

INSERT INTO dbo.Documents
(title, doctype, docexcerpt, doccontent)
SELECT N'Columnstore Indices and Batch Processing',
 N'docx',  N'You should use a columnstore index… ',
 bulkcolumn
FROM OPENROWSET(BULK 'C:\Users\Ron\Desktop
\ColumnstoreIndicesAndBatchProcessing.docx',
                SINGLE_BLOB) AS doc;

CREATE SEARCH PROPERTY LIST WordSearchPropertyList;
GO
ALTER SEARCH PROPERTY LIST WordSearchPropertyList
 ADD 'Authors'
 WITH (PROPERTY_SET_GUID = 'F29F85E0-4FF9-1068-AB91-08002B27B3D9',
       PROPERTY_INT_ID = 4,
       PROPERTY_DESCRIPTION = 'System.Authors - authors of a given item.');
GO

CREATE FULLTEXT INDEX ON dbo.Documents
(
  docexcerpt Language 1033,
  doccontent TYPE COLUMN doctype
  Language 1033
  STATISTICAL_SEMANTICS
)
```

```
KEY INDEX PK_Documents
ON DocumentsFtCatalog
WITH STOPLIST = SQLStopList,
     SEARCH PROPERTY LIST = WordSearchPropertyList,
        CHANGE_TRACKING AUTO;
GO
```

- using CONTAINS and FREETEXT(inflectional terms)

```
SELECT id, title, docexcerpt
FROM dbo.Documents
WHERE CONTAINS(docexcerpt, N'data');

SELECT id, title, docexcerpt
FROM dbo.Documents
WHERE CONTAINS(docexcerpt, N'FORMSOF(THESAURUS, need)');

SELECT id, title, docexcerpt
FROM dbo.Documents
WHERE CONTAINS(PROPERTY(doccontent,'Authors'), 'Dejan');

-- Proximity with max distance
SELECT id, title, docexcerpt
FROM dbo.Documents
WHERE CONTAINS(docexcerpt, N'NEAR((problem, data),5)');
```

- CONTAINSTABLE, FREETEXTABLE, SEMANTICKEYPHRASETABLE, SEMANTICSIMILARITYDETAILSTABLE, SEMANTICSIMILARITYTABLE,

```
SELECT D.id, D.title, CT.[RANK], D.docexcerpt
FROM CONTAINSTABLE(dbo.Documents, docexcerpt,
      N'data OR level') AS CT
 INNER JOIN dbo.Documents AS D
  ON CT.[KEY] = D.id
ORDER BY CT.[RANK] DESC;

SELECT D.id, D.title, FT.[RANK], D.docexcerpt
FROM FREETEXTTABLE (dbo.Documents, docexcerpt,
      N'data level') AS FT
 INNER JOIN dbo.Documents AS D
  ON FT.[KEY] = D.id
ORDER BY FT.[RANK] DESC;

-- Top 20 semantic key phrases
SELECT TOP (20)
 D.id, D.title, SKT.keyphrase, SKT.score
FROM SEMANTICKEYPHRASETABLE
      (dbo.Documents, doccontent) AS SKT
 INNER JOIN dbo.Documents AS D
  ON SKT.document_key = D.id
ORDER BY SKT.score DESC;

-- Documents that are similar to document 1
SELECT SST.matched_document_key,
```

```
 D.title, SST.score
FROM SEMANTICSIMILARITYTABLE
     (dbo.Documents, doccontent, 1) AS SST
 INNER JOIN dbo.Documents AS D
  ON SST.matched_document_key = D.id
ORDER BY SST.score DESC;

--  Semantic search key phrases that are common to two documents
SELECT SSDT.keyphrase, SSDT.score
FROM SEMANTICSIMILARITYDETAILSTABLE
      (dbo.Documents, doccontent, 1,
       doccontent, 4) AS SSDT
ORDER BY SSDT.score DESC;
GO
```

- use and understand different data access technologies;

7. Implement sub-queries
   - sub-queries: self contained, correlated

```
SELECT productid, productname, unitprice
FROM Production.Products
WHERE supplierid IN  -- IN predicate
  (SELECT supplierid
   FROM Production.Suppliers
   WHERE country = N'Japan');

SELECT categoryid, productid, productname, unitprice
FROM Production.Products AS P1
WHERE unitprice =
  (SELECT MIN(unitprice)
   FROM Production.Products AS P2
   WHERE P2.categoryid = P1.categoryid);
```

   - cross apply->inner join, outer apply->left outer join

```
SELECT S.supplierid, S.companyname AS supplier, A.*
FROM Production.Suppliers AS S
  OUTER APPLY (SELECT productid, productname, unitprice
               FROM Production.Products AS P
               WHERE P.supplierid = S.supplierid
               ORDER BY unitprice, productid
               OFFSET 0 ROWS FETCH FIRST 2 ROWS ONLY) AS A
WHERE S.country = N'Japan';
```

   - pivot, cannot directly be result of expression, only one aggregate function like COUNT(<column name>)

```
WITH PivotData AS
(
  SELECT
    custid   , -- grouping column
    shipperid, -- spreading column
    freight    -- aggregation column
  FROM Sales.Orders
)
SELECT custid, [1], [2], [3]
FROM PivotData
  PIVOT(SUM(freight) FOR shipperid IN ([1],[2],[3]) ) AS P;
```

- unpivot, rotate the input data from columns to rows

```
SELECT custid, shipperid, freight
FROM Sales.FreightTotals
  UNPIVOT( freight FOR shipperid IN([1],[2],[3]) ) AS U;
```

8. Implement data types
   - Use appropriate data; understand the uses and limitations of each data type; impact of GUID (newid, newsequentialid) on database performance; when to use what data type for columns
9. Implement aggregate queries
   - Group, set functions: COUNT, SUM, AVG, MIN, MAX

```
SELECT shipperid,
  COUNT(*) AS numorders,
  COUNT(shippeddate) AS shippedorders, --ignore NULLs
  COUNT(DISTINCT shippeddate) AS DistinctShippedorders,
  MIN(shippeddate) AS firstshipdate,
  MAX(shippeddate) AS lastshipdate,
  SUM(val) AS totalvalue
FROM Sales.OrderValues
GROUP BY shipperid;

SELECT
  shipcountry, GROUPING(shipcountry) AS grpcountry, --Grouping 1 for placeholder
  shipregion , GROUPING(shipregion) AS grpregion,
  shipcity   , GROUPING(shipcity) AS grpcity,
  COUNT(*) AS numorders
FROM Sales.Orders
GROUP BY ROLLUP( shipcountry, shipregion, shipcity );
```

   - Window aggregate functions, only in SELECT AND ORDER BY

```
WITH RunningTotals AS
(
  SELECT custid, orderid, orderdate, val,
    SUM(val) OVER(PARTITION BY custid
                  ORDER BY orderdate, orderid
                  ROWS BETWEEN UNBOUNDED PRECEDING
                           AND CURRENT ROW) AS runningtotal
  FROM Sales.OrderValues
)
SELECT *
FROM RunningTotals
WHERE runningtotal < 1000.00;
```

   - Window ranking function: ROW_NUMBER, RANK(jump tie), DENSE_RANK(continue number), NTILE

```
SELECT custid, orderid, val,
  ROW_NUMBER() OVER(ORDER BY val) AS rownum,
  RANK()       OVER(ORDER BY val) AS rnk,
```

```
  DENSE_RANK() OVER(ORDER BY val) AS densernk,
  NTILE(100)   OVER(ORDER BY val) AS ntile100
FROM Sales.OrderValues;
```

Window offset functions: LAG, LEAD, FIRST_VALUE, LAST_VALUE

```
SELECT custid, orderid, orderdate, val,
  LAG(val)  OVER(PARTITION BY custid
                 ORDER BY orderdate, orderid) AS prev_val,
  LEAD(val) OVER(PARTITION BY custid
                 ORDER BY orderdate, orderid) AS next_val
FROM Sales.OrderValues;

SELECT custid, orderid, orderdate, val,
  FIRST_VALUE(val)  OVER(PARTITION BY custid
                         ORDER BY orderdate, orderid
                         ROWS BETWEEN UNBOUNDED PRECEDING   -- Beginning partition
                                  AND CURRENT ROW) AS first_val,
  LAST_VALUE(val) OVER(PARTITION BY custid
                       ORDER BY orderdate, orderid
                       ROWS BETWEEN CURRENT ROW
                                AND UNBOUNDED FOLLOWING) - end partition
                                      AS last_val
FROM Sales.OrderValues;
```

10. Query and manage XML data

- XML is case sensitive, attribute centric

```
SELECT Customer.custid, Customer.companyname,
 [Order].orderid, [Order].orderdate
FROM Sales.Customers AS Customer
 INNER JOIN Sales.Orders AS [Order]
  ON Customer.custid = [Order].custid
WHERE Customer.custid <= 2
  AND [Order].orderid %2 = 0
ORDER BY Customer.custid, [Order].orderid
FOR XML RAW

<row custid="1" companyname="Customer NRZBB" orderid="10692" orderdate="2007-10-
03T00:00:00" />
<row custid="1" companyname="Customer NRZBB" orderid="10702" orderdate="2007-10-
13T00:00:00" />
<row custid="1" companyname="Customer NRZBB" orderid="10952" orderdate="2008-03-
16T00:00:00" />
<row custid="2" companyname="Customer MLTDN" orderid="10308" orderdate="2006-09-
18T00:00:00" />
<row custid="2" companyname="Customer MLTDN" orderid="10926" orderdate="2008-03-
04T00:00:00" />


-- Create XML example with FOR XML AUTO option, attribute-centric
SELECT Customer.custid, Customer.companyname,
 [Order].orderid, [Order].orderdate
FROM Sales.Customers AS Customer
 INNER JOIN Sales.Orders AS [Order]
  ON Customer.custid = [Order].custid
WHERE Customer.custid <= 2
  AND [Order].orderid %2 = 0
```

```
ORDER BY Customer.custid, [Order].orderid
FOR XML AUTO, ROOT('CustomersOrders');

 <CustomersOrders>
  <Customer custid="1" companyname="Customer NRZBB">
    <Order orderid="10692" orderdate="2007-10-03T00:00:00" />
    <Order orderid="10702" orderdate="2007-10-13T00:00:00" />
    <Order orderid="10952" orderdate="2008-03-16T00:00:00" />
  </Customer>
  <Customer custid="2" companyname="Customer MLTDN">
    <Order orderid="10308" orderdate="2006-09-18T00:00:00" />
    <Order orderid="10926" orderdate="2008-03-04T00:00:00" />
  </Customer>
</CustomersOrders>
```

- Element centric

```
-- XML with AUTO option, element-centric, with namespace
WITH XMLNAMESPACES('TK461-CustomersOrders' AS co)
SELECT [co:Customer].custid AS [co:custid],
 [co:Customer].companyname AS [co:companyname],
 [co:Order].orderid AS [co:orderid],
 [co:Order].orderdate AS [co:orderdate]
FROM Sales.Customers AS [co:Customer]
 INNER JOIN Sales.Orders AS [co:Order]
  ON [co:Customer].custid = [co:Order].custid
WHERE [co:Customer].custid <= 2
  AND [co:Order].orderid %2 = 0
ORDER BY [co:Customer].custid, [co:Order].orderid
FOR XML AUTO, ELEMENTS, ROOT('CustomersOrders');
GO

 <CustomersOrders xmlns:co="TK461-CustomersOrders">
  <co:Customer>
    <co:custid>1</co:custid>
    <co:companyname>Customer NRZBB</co:companyname>
    <co:Order>
      <co:orderid>10692</co:orderid>
      <co:orderdate>2007-10-03T00:00:00</co:orderdate>
    </co:Order>
    <co:Order>
      <co:orderid>10702</co:orderid>
      <co:orderdate>2007-10-13T00:00:00</co:orderdate>
    </co:Order>
    <co:Order>
      <co:orderid>10952</co:orderid>
      <co:orderdate>2008-03-16T00:00:00</co:orderdate>
    </co:Order>
  </co:Customer>
  <co:Customer>
    <co:custid>2</co:custid>
    <co:companyname>Customer MLTDN</co:companyname>
    <co:Order>
      <co:orderid>10308</co:orderid>
      <co:orderdate>2006-09-18T00:00:00</co:orderdate>
    </co:Order>
    <co:Order>
      <co:orderid>10926</co:orderid>
```

```
      <co:orderdate>2008-03-04T00:00:00</co:orderdate>
    </co:Order>
  </co:Customer>
</CustomersOrders>
```

- XPath

```
SELECT Customer.custid AS [@custid],  -- attribute centric
 Customer.companyname AS [companyname] -- element centric
FROM Sales.Customers AS Customer
WHERE Customer.custid <= 2
ORDER BY Customer.custid
FOR XML PATH ('Customer'), ROOT('Customers');
GO
```

- XQuery FLWOR expression: for, let, where, order by, return

```
DECLARE @x AS XML;
SET @x = N'
<CustomersOrders>
  <Customer custid="1">
    <!-- Comment 111 -->
    <companyname>Customer NRZBB</companyname>
    <Order orderid="10692">
      <orderdate>2007-10-03T00:00:00</orderdate>
    </Order>
    <Order orderid="10702">
      <orderdate>2007-10-13T00:00:00</orderdate>
    </Order>
    <Order orderid="10952">
      <orderdate>2008-03-16T00:00:00</orderdate>
    </Order>
  </Customer>
  <Customer custid="2">
    <!-- Comment 222 -->
    <companyname>Customer MLTDN</companyname>
    <Order orderid="10308">
      <orderdate>2006-09-18T00:00:00</orderdate>
    </Order>
    <Order orderid="10952">
      <orderdate>2008-03-04T00:00:00</orderdate>
    </Order>
  </Customer>
</CustomersOrders>';
SELECT @x.query('for $i in CustomersOrders/Customer/Order
                 let $j := $i/orderdate
                 where $i/@orderid < 10900
                 order by ($j)[1]
                 return
                 <Order-orderid-element>
                  <orderid>{data($i/@orderid)}</orderid>
                  {$j}
                 </Order-orderid-element>')
GO
```

- Using XML data type for dynamic schema

```
-- Using the XML Data Type for Dynamic Schema
ALTER TABLE Production.Products
 ADD additionalattributes XML NULL;
GO

-- Auxiliary tables
CREATE TABLE dbo.Beverages
(
  percentvitaminsRDA INT
);
CREATE TABLE dbo.Condiments
(
  shortdescription NVARCHAR(50)
);
GO
-- Store the Schemas in a Variable and Create the Collection
DECLARE @mySchema NVARCHAR(MAX);
SET @mySchema = N'';
SET @mySchema = @mySchema +
  (SELECT *
   FROM Beverages
   FOR XML AUTO, ELEMENTS, XMLSCHEMA('Beverages'));
SET @mySchema = @mySchema +
  (SELECT *
   FROM Condiments
   FOR XML AUTO, ELEMENTS, XMLSCHEMA('Condiments'));
SELECT CAST(@mySchema AS XML);
CREATE XML SCHEMA COLLECTION dbo.ProductsAdditionalAttributes AS @mySchema;
GO
-- Drop Auxiliary Tables
DROP TABLE dbo.Beverages, dbo.Condiments;
GO
ALTER TABLE Production.Products
  ALTER COLUMN additionalattributes
   XML(dbo.ProductsAdditionalAttributes);
GO
-- Function to Retrieve the Namespace
CREATE FUNCTION dbo.GetNamespace(@chkcol XML)
 RETURNS NVARCHAR(15)
AS
BEGIN
 RETURN @chkcol.value('namespace-uri((/*)[1])','NVARCHAR(15)')
END;
GO
-- Function to Retrieve the Category Name
CREATE FUNCTION dbo.GetCategoryName(@catid INT)
 RETURNS NVARCHAR(15)
AS
BEGIN
 RETURN
  (SELECT categoryname
   FROM Production.Categories
   WHERE categoryid = @catid)
END;
GO
-- Add the Constraint
ALTER TABLE Production.Products ADD CONSTRAINT ck_Namespace
 CHECK (dbo.GetNamespace(additionalattributes) =
```

```
        dbo.GetCategoryName(categoryid));
GO

-- Valid Data
-- Beverage
UPDATE Production.Products
   SET additionalattributes = N'
<Beverages xmlns="Beverages">
  <percentvitaminsRDA>27</percentvitaminsRDA>
</Beverages>'
WHERE productid = 1;
-- Condiment
UPDATE Production.Products
   SET additionalattributes = N'
<Condiments xmlns="Condiments">
  <shortdescription>very sweet</shortdescription>
</Condiments>'
WHERE productid = 3;
GO
```

- XML data type: value(), exist(), query(), nodes(), modify()

```
-- shred order info for the first customer
SELECT  T.c.value('./@orderid[1]', 'INT') AS [Order Id],
 T.c.value('./orderdate[1]', 'DATETIME') AS [Order Date]
FROM @x.nodes('//Customer[@custid=1]/Order')
      AS T(c);

-- Update the name of the first customer
SET @x.modify('replace value of
    /CustomersOrders[1]/Customer[1]/companyname[1]/text()[1]
       with "New Company Name"');
SELECT @x.value('(/CustomersOrders/Customer/companyname)[1]',
       'NVARCHAR(20)')
       AS [First Customer New Name];
GO

-- Check whether the companyname and address nodes exist
SELECT @x.exist('(/CustomersOrders/Customer/companyname)')
       AS [Company Name Exists],
          @x.exist('(/CustomersOrders/Customer/address)')
       AS [Address Exists];
-- Return orders for the first customer
SELECT @x.query('//Customer[@custid=1]/Order')
       AS [Customer 1 orders];
```

- Understand XML datatypes and their schemas and interop w/, limitations and restrictions; implement XML schemas and handling of XML data; XML data: how to handle it in SQL Server and when and when not to use it, including XML namespaces; import and export XML; XML indexing

Modify data (24%)

11. Create and alter stored procedures (simple statements)
    - Insert procedure

```
IF OBJECT_ID(N'Sales.OrdersForCountry', N'P') IS NOT NULL
  DROP PROC Sales.OrdersForCountry;
GO

CREATE PROC Sales.OrdersForCountry
  @country AS NVARCHAR(15)
AS
SELECT orderid, custid, empid, orderdate, shipcountry, freight
FROM Sales.Orders
WHERE shipcountry = @country;
GO

INSERT INTO Sales.MyOrders(orderid, custid, empid, orderdate, shipcountry, freight)
  EXEC Sales.OrdersForCountry
    @country = N'Portugal';

-- insert stored procedure
IF OBJECT_ID('Production.InsertProducts', 'P') IS NOT NULL
  DROP PROCEDURE Production.InsertProducts
GO
CREATE PROCEDURE Production.InsertProducts
  @productname AS NVARCHAR(40)
  , @supplierid AS INT
  , @categoryid AS INT
  , @unitprice AS MONEY = 0
  , @discontinued AS BIT = 0
AS
BEGIN
  DECLARE @ClientMessage NVARCHAR(100);
  BEGIN TRY
    -- Test parameters
    IF NOT EXISTS(SELECT 1 FROM Production.Suppliers
      WHERE supplierid = @supplierid)
      BEGIN
        SET @ClientMessage = 'Supplier id '
            + CAST(@supplierid AS VARCHAR) + ' is invalid';
        THROW 50000, @ClientMessage, 0;
      END
    IF NOT EXISTS(SELECT 1 FROM Production.Categories
        WHERE categoryid = @categoryid)
      BEGIN
        SET @ClientMessage = 'Category id '
           + CAST(@categoryid AS VARCHAR) + ' is invalid';
        THROW 50000, @ClientMessage, 0;
      END;
    IF NOT(@unitprice >= 0)
      BEGIN
        SET @ClientMessage = 'Unitprice '
          + CAST(@unitprice AS VARCHAR) + ' is invalid. Must be >= 0.';
        THROW 50000, @ClientMessage, 0;
      END;
    -- Perform the insert
    INSERT Production.Products (productname, supplierid, categoryid,
      unitprice, discontinued)
```

```
    VALUES (@productname, @supplierid, @categoryid, @unitprice, @discontinued);
  END TRY
  BEGIN CATCH
    THROW;
  END CATCH;
END;
GO


EXEC Production.InsertProducts
  @productname = 'Test Product'
  , @supplierid = 10
  , @categoryid = 1
  , @unitprice  = 100
  , @discontinued = 0;
GO
```

- Procedure cannot USE <database> command; cannot use a CREATE with AGGRAGATE, RULE, DEFAULT, CREATE, FUNCTION, TRIGGER, PROCEDURE, VIEW.

```
IF OBJECT_ID('Sales.GetCustomerOrders', 'P') IS NOT NULL
       DROP PROC Sales.GetCustomerOrders;
GO
CREATE PROC Sales.GetCustomerOrders
       @custid   AS INT,
       @orderdatefrom AS DATETIME = '19000101',
       @orderdateto   AS DATETIME = '99991231',
       @numrows  AS INT = 0 OUTPUT -- OUTPUT is both input and output
AS
BEGIN
       SET NOCOUNT ON;
       SELECT orderid, custid, shipperid, orderdate, requireddate, shippeddate
       FROM Sales.Orders
       WHERE custid = @custid
              AND orderdate >= @orderdatefrom
              AND orderdate < @orderdateto;
       SET @numrows = @@ROWCOUNT;
       RETURN;
END

DECLARE @rowsreturned AS INT;
EXEC Sales.GetCustomerOrders
  @custid   = 37,
  @orderdatefrom = '20070401',
  @orderdateto  = '20070701',
  @numrows  = @rowsreturned OUTPUT;
SELECT @rowsreturned AS "Rows Returned";

--BACKUP DATABASE command
IF OBJECT_ID('dbo.BackupDatabases', 'P') IS NOT NULL
       DROP PROCEDURE dbo.BackupDatabases
GO
CREATE PROCEDURE dbo.BackupDatabases
AS
```

```
BEGIN
  DECLARE @databasename AS NVARCHAR(128)
    , @timecomponent AS NVARCHAR(50)
    , @sqlcommand AS NVARCHAR(1000);
  SET @databasename = (SELECT MIN(name) FROM sys.databases
    WHERE name NOT IN ('master', 'model', 'msdb', 'tempdb'));
  WHILE @databasename IS NOT NULL
    BEGIN
      SET @timecomponent = REPLACE(REPLACE(REPLACE(
          CONVERT(NVARCHAR, GETDATE(), 120), ' ', '_'), ':', ''), '-', '');
      SET @sqlcommand = 'BACKUP DATABASE ' + @databasename + ' TO DISK =
          ''C:\Backups\' + @databasename + '_' + @timecomponent + '.bak''';
      PRINT @sqlcommand;
      EXEC(@sqlcommand);
      SET @databasename = (SELECT MIN(name) FROM sys.databases WHERE name
          NOT IN ('master', 'model', 'msdb', 'tempdb') AND name > @databasename);
  END;
  RETURN;
END;
GO
```

- Branching logic: if/else , while

```
DECLARE @categoryname AS NVARCHAR(15);
SET @categoryname = (SELECT MIN(categoryname) FROM Production.Categories);
WHILE @categoryname IS NOT NULL
BEGIN
  PRINT @categoryname;
  SET @categoryname = (SELECT MIN(categoryname) FROM Production.Categories
    WHERE categoryname > @categoryname);
END;
GO
```

- Write a stored procedure to meet a given set of requirements; branching logic; create stored procedures and other programmatic objects; techniques for developing stored procedures; different types of storeproc result; create stored procedure for data access layer; program stored procedures, triggers, functions with T-SQL

12. Modify data by using INSERT, UPDATE, and DELETE statements
    - Insert data

```
IF OBJECT_ID(N'Sales.MyOrders', N'U') IS NOT NULL
     DROP TABLE Sales.MyOrders;

CREATE TABLE Sales.MyOrders
(
  orderid INT NOT NULL IDENTITY(1, 1)
    CONSTRAINT PK_MyOrders_orderid PRIMARY KEY,
  custid  INT NOT NULL,
  empid   INT NOT NULL,
  orderdate DATE NOT NULL
    CONSTRAINT DFT_MyOrders_orderdate DEFAULT (CAST(SYSDATETIME() AS DATE)),
  shipcountry NVARCHAR(15) NOT NULL,
```

```
  freight MONEY NOT NULL
);

INSERT INTO Sales.MyOrders(custid, empid, orderdate, shipcountry, freight) VALUES
  (2, 11, '20120620', N'USA', 50.00),
  (5, 13, '20120620', N'USA', 40.00),
  (7, 17, '20120620', N'USA', 45.00);

INSERT INTO Sales.MyOrders(orderid, custid, empid, orderdate, shipcountry, freight)
  SELECT orderid, custid, empid, orderdate, shipcountry, freight
  FROM Sales.Orders
  WHERE shipcountry = N'Norway';


--select into doesn't carry indexes, constraints, trigger, permission...
SELECT orderid, custid, orderdate, shipcountry, freight
INTO Sales.MyOrders
FROM Sales.Orders
WHERE shipcountry = N'Norway';


-- examine the structure of the Sales.Customers table
EXEC sp_describe_first_result_set N'SELECT * FROM Sales.Customers;';
```

- Update

```
UPDATE Sales.MyOrderDetails
  SET discount += 0.05
WHERE orderid = 10251;

-- using a CTE
WITH C AS
(
  SELECT TGT.custid,
    TGT.country AS tgt_country, SRC.country AS src_country,
    TGT.postalcode AS tgt_postalcode, SRC.postalcode AS src_postalcode
  FROM Sales.MyCustomers AS TGT
    INNER JOIN Sales.Customers AS SRC
      ON TGT.custid = SRC.custid
)
UPDATE C
  SET tgt_country = src_country,
      tgt_postalcode = src_postalcode;

-- using derived table
UPDATE D
  SET tgt_country = src_country,
      tgt_postalcode = src_postalcode
FROM (
        SELECT TGT.custid,
          TGT.country AS tgt_country, SRC.country AS src_country,
          TGT.postalcode AS tgt_postalcode, SRC.postalcode AS src_postalcode
        FROM Sales.MyCustomers AS TGT
          INNER JOIN Sales.Customers AS SRC
            ON TGT.custid = SRC.custid
     ) AS D;
```

```
-- UPDATE based on join
UPDATE TGT
  SET TGT.country = SRC.country,
      TGT.postalcode = SRC.postalcode
FROM Sales.MyCustomers AS TGT
  INNER JOIN Sales.Customers AS SRC
    ON TGT.custid = SRC.custid;

-- using just the FROM
UPDATE Sales.MyCustomers
  SET MyCustomers.country = SRC.country,
      MyCustomers.postalcode = SRC.postalcode
FROM Sales.Customers AS SRC
WHERE MyCustomers.custid = SRC.custid;

-- equivalent using cross join
UPDATE TGT
  SET TGT.country = SRC.country,
      TGT.postalcode = SRC.postalcode
FROM Sales.MyCustomers AS TGT
  CROSS JOIN Sales.Customers AS SRC
WHERE TGT.custid = SRC.custid;
```

- Delete

```
DELETE FROM Sales.MyOrderDetails
WHERE productid = 11;

-- TRUNCATE no filter; delete doesn’t reset identity, truncate does; truncate is not
allow if a foreign key
TRUNCATE TABLE Sales.MyOrderDetails;

-- reseed
SELECT IDENT_CURRENT('Sales.MyOrdersDetails') AS [IDENT_CURRENT];
DBCC CHECKIDENT('Sales.MyOrderDetails’, RESEED, 4);

-- DELETE based on a join, not standard
DELETE FROM O
FROM Sales.MyOrders AS O
  INNER JOIN Sales.MyCustomers AS C
    ON O.custid = C.custid
WHERE C.country = N'USA';

-- alternative using a subquery, standard sql
DELETE FROM Sales.MyOrders
WHERE EXISTS
  (SELECT *
   FROM Sales.MyCustomers
   WHERE MyCustomers.custid = MyOrders.custid
     AND MyCustomers.country = N'USA');

-- DELETE using table expressions
WITH OldestOrders AS
(
  SELECT TOP (100) *
  FROM Sales.MyOrders
```

```
  ORDER BY orderdate, orderid
)
DELETE FROM OldestOrders;
```

- use output statement

```
-- INSERT with OUTPUT
INSERT INTO Sales.MyOrders(custid, empid, orderdate)
  OUTPUT
    inserted.orderid, inserted.custid, inserted.empid, inserted.orderdate
    INTO SomeTable(orderid, custid, empid, orderdate)
  SELECT custid, empid, orderdate
  FROM Sales.Orders
  WHERE shipcountry = N'Norway';

-- archive deleted orders
INSERT INTO Sales.MyOrdersArchive(orderid, custid, empid, orderdate)
  SELECT orderid, custid, empid, orderdate
  FROM (DELETE FROM Sales.MyOrders
          OUTPUT deleted.*
        WHERE orderdate < '20070101') AS D
  WHERE custid IN (17, 19);

-- compsable DML
DECLARE @InsertedOrders AS TABLE
(
  orderid   INT  NOT NULL PRIMARY KEY,
  custid    INT  NOT NULL,
  empid     INT  NOT NULL,
  orderdate DATE NOT NULL
);

INSERT INTO @InsertedOrders(orderid, custid, empid, orderdate)
  SELECT orderid, custid, empid, orderdate
  FROM (MERGE INTO Sales.MyOrders AS TGT
        USING (VALUES(1,        70,      1,      '20061218'),
                     (2,        70,      7,      '20070429'),
                     (3,        70,      7,      '20070820'),
                     (4,        70,      3,      '20080114'),
                     (5,        70,      1,      '20080226'),
                     (6,        70,      2,      '20080410'))
               AS SRC(orderid,  custid,  empid,  orderdate )
          ON SRC.orderid = TGT.orderid
        WHEN MATCHED AND (   TGT.custid    <> SRC.custid
                          OR TGT.empid     <> SRC.empid
                          OR TGT.orderdate <> SRC.orderdate) THEN UPDATE
          SET TGT.custid    = SRC.custid,
              TGT.empid     = SRC.empid,
              TGT.orderdate = SRC.orderdate
        WHEN NOT MATCHED THEN INSERT
          VALUES(SRC.orderid, SRC.custid, SRC.empid, SRC.orderdate)
        WHEN NOT MATCHED BY SOURCE THEN
          DELETE
        OUTPUT
          $action AS the_action, inserted.*) AS D
  WHERE the_action = 'INSERT';
```

- deal with identity insert

```
--insert with IDENTITY_INSERT
SET IDENTITY_INSERT Production.CategoriesDuplicate ON;
INSERT Production.CategoriesTest (categoryid, categoryname, description)
       SELECT categoryid, categoryname, description
       FROM Production.Categories;
GO
SET IDENTITY_INSERT Production.CategoriesDuplicate OFF;
GO
```

13. Combine datasets
    - UNION (return distinct) and UNION all (with duplicates)

```
SELECT country, region, city
FROM HR.Employees

UNION ALL   or   UNION

SELECT country, region, city
FROM Sales.Customers;
```

- Difference between UNION and UNION all; case versus isnull versus coalesce;
- modify data by using MERGE statements

```
-- update where exists, insert where not exists
DECLARE
  @orderid   AS INT  = 1,
  @custid    AS INT  = 1,
  @empid     AS INT  = 2,
  @orderdate AS DATE = '20120620';

MERGE INTO Sales.MyOrders WITH (HOLDLOCK) AS TGT
USING (VALUES(@orderid, @custid, @empid, @orderdate))
       AS SRC( orderid,  custid,  empid,  orderdate)
  ON SRC.orderid = TGT.orderid
WHEN MATCHED THEN UPDATE
  SET TGT.custid    = SRC.custid,
      TGT.empid     = SRC.empid,
      TGT.orderdate = SRC.orderdate
WHEN NOT MATCHED THEN INSERT
  VALUES(SRC.orderid, SRC.custid, SRC.empid, SRC.orderdate);
GO

-- table as source
DECLARE @Orders AS TABLE
(
  orderid   INT  NOT NULL PRIMARY KEY,
  custid    INT  NOT NULL,
  empid     INT  NOT NULL,
  orderdate DATE NOT NULL
);

INSERT INTO @Orders(orderid, custid, empid, orderdate) VALUES
```

```
  (2, 1, 3, '20120612'),
  (3, 2, 2, '20120612'),
  (4, 3, 5, '20120612');

-- update where exists (only if different), insert where not exists,
-- delete when exists in target but not in source
MERGE INTO Sales.MyOrders AS TGT
USING @Orders AS SRC
  ON SRC.orderid = TGT.orderid
WHEN MATCHED AND (   TGT.custid    <> SRC.custid
                  OR TGT.empid     <> SRC.empid
                  OR TGT.orderdate <> SRC.orderdate) THEN UPDATE -- or DELETE
  SET TGT.custid    = SRC.custid,
      TGT.empid     = SRC.empid,
      TGT.orderdate = SRC.orderdate
WHEN NOT MATCHED THEN INSERT
  VALUES(SRC.orderid, SRC.custid, SRC.empid, SRC.orderdate)
WHEN NOT MATCHED BY SOURCE THEN
  DELETE; -- or UPDATE

-- With rows in a table expression
WITH SRC AS
(
  SELECT *
  FROM Sales.Orders
  WHERE shipcountry = N'Norway'
)
MERGE INTO Sales.MyOrders AS TGT
USING SRC
  ON  SRC.orderid = TGT.orderid
WHEN MATCHED AND (   TGT.custid    <> SRC.custid
                  OR TGT.empid     <> SRC.empid
                  OR TGT.orderdate <> SRC.orderdate) THEN UPDATE
  SET TGT.custid    = SRC.custid,
      TGT.empid     = SRC.empid,
      TGT.orderdate = SRC.orderdate
WHEN NOT MATCHED THEN INSERT
  VALUES(SRC.orderid, SRC.custid, SRC.empid, SRC.orderdate);

-- alternative with derived table
MERGE INTO Sales.MyOrders AS TGT
USING ( SELECT *
        FROM Sales.Orders
        WHERE shipcountry = N'Norway' )
 AS SRC
  ON  SRC.orderid = TGT.orderid
WHEN MATCHED AND (   TGT.custid    <> SRC.custid
                  OR TGT.empid     <> SRC.empid
                  OR TGT.orderdate <> SRC.orderdate) THEN UPDATE
  SET TGT.custid    = SRC.custid,
      TGT.empid     = SRC.empid,
      TGT.orderdate = SRC.orderdate
WHEN NOT MATCHED THEN INSERT
  VALUES(SRC.orderid, SRC.custid, SRC.empid, SRC.orderdate);
```

14. Work with functions
    - Understand deterministic, non-deterministic functions; scalar and table values; apply built-in scalar functions; create and alter user-defined functions (UDFs)

Troubleshoot and optimize (25%)

15. Optimize queries
    - Join algorithm: nested loops, merge, hash, bitmap filtering optimized hash.
    - Tsql(statmenat to excute)->parsing(check syntax, parse logical tree)->binding(name resolution, algebrize tree)->optimization(find plan, execute plan)->execution(query execute, plan cache)
    - Use extended events sessions
    - SET STATISTICS IO / TIME ON, Turn Actual Execution Plan on, table scan, clustered index scan, clustered index seek, index scan, index seek, RID lookup, key lookup, hash match join, merge join, nested loop, stream aggregate(ordered), hash match aggregate, filter, sort

```
DBCC DROPCLEANBUFFERS;
SET STATISTICS IO ON;
SET STATISTICS TIME ON;
--Turn on the actual execution plan
SELECT * FROM Sales.Customers;
SELECT * FROM Sales.Orders;
GO
```

- Use dynamic management objects:

```
--sys.dm_os_sys_info
SELECT cpu_count AS logical_cpu_count,
 cpu_count / hyperthread_ratio AS physical_cpu_count,
 CAST(physical_memory_kb / 1024. AS int) AS physical_memory__mb,
 sqlserver_start_time
FROM sys.dm_os_sys_info;

-- sys.dm_os_waiting_tasks, sys.dm_exec_sessions
SELECT S.login_name, S.host_name, S.program_name,
 WT.session_id, WT.wait_duration_ms, WT.wait_type,
 WT.blocking_session_id, WT.resource_description
FROM sys.dm_os_waiting_tasks AS WT
 INNER JOIN sys.dm_exec_sessions AS S
  ON WT.session_id = S.session_id
WHERE s.is_user_process = 1;

--sys.dm_exec_requests, sys.dm_exec_sessions, sys.dm_exec_sql_text
SELECT S.login_name, S.host_name, S.program_name,
 R.command, T.text,
 R.wait_type, R.wait_time, R.blocking_session_id
FROM sys.dm_exec_requests AS R
 INNER JOIN sys.dm_exec_sessions AS S
  ON R.session_id = S.session_id
 OUTER APPLY sys.dm_exec_sql_text(R.sql_handle) AS T
WHERE S.is_user_process = 1;

--sys.dm_exec_sql_text, sys.dm_exec_query_stats
SELECT TOP (5)
```

```
 (total_logical_reads + total_logical_writes) AS total_logical_IO,
 execution_count,
 (total_logical_reads/execution_count) AS avg_logical_reads,
 (total_logical_writes/execution_count) AS avg_logical_writes,
 (SELECT SUBSTRING(text, statement_start_offset/2 + 1,
    (CASE WHEN statement_end_offset = -1
          THEN LEN(CONVERT(nvarchar(MAX),text)) * 2
          ELSE statement_end_offset
     END - statement_start_offset)/2)
   FROM sys.dm_exec_sql_text(sql_handle)) AS query_text
FROM sys.dm_exec_query_stats
ORDER BY (total_logical_reads + total_logical_writes) DESC;

--sys.dm_db_index_usage_stats
SELECT OBJECT_NAME(I.object_id) AS objectname,
 I.name AS indexname,
 I.index_id AS indexid
FROM sys.indexes AS I
 INNER JOIN sys.objects AS O
  ON O.object_id = I.object_id
WHERE I.object_id > 100
  AND I.type_desc = 'NONCLUSTERED'
  AND I.index_id NOT IN
       (SELECT S.index_id
        FROM sys.dm_db_index_usage_stats AS S
        WHERE S.object_id=I.object_id
          AND I.index_id=S.index_id
          AND database_id = DB_ID('TSQL2012'))
ORDER BY objectname, indexname;

--sys.dm_db_missing_index
SELECT MID.statement AS [Database.Schema.Table],
 MIC.column_id AS ColumnId,
 MIC.column_name AS ColumnName,
 MIC.column_usage AS ColumnUsage,
 MIGS.user_seeks AS UserSeeks,
 MIGS.user_scans AS UserScans,
 MIGS.last_user_seek AS LastUserSeek,
 MIGS.avg_total_user_cost AS AvgQueryCostReduction,
 MIGS.avg_user_impact AS AvgPctBenefit
FROM sys.dm_db_missing_index_details AS MID
 CROSS APPLY sys.dm_db_missing_index_columns (MID.index_handle) AS MIC
 INNER JOIN sys.dm_db_missing_index_groups AS MIG
       ON MIG.index_handle = MIG.index_handle
 INNER JOIN sys.dm_db_missing_index_group_stats AS MIGS
       ON MIG.index_group_handle=MIGS.group_handle
ORDER BY MIGS.avg_user_impact DESC;
GO
```

- Understand statistics;

```
DBCC SHOW_STATISTICS(N'Sales.Orders',N'idx_nc_empid') WITH HISTOGRAM;
DBCC SHOW_STATISTICS(N'Sales.Orders',N'idx_nc_empid') WITH STAT_HEADER;

SELECT OBJECT_NAME(s.object_id) AS table_name,
 S.name AS statistics_name, C.name AS column_name
```

```
FROM sys.stats AS S
 INNER JOIN sys.stats_columns AS SC
  ON S.stats_id = SC.stats_id
 INNER JOIN sys.columns AS C
  ON S.object_id= C.object_id AND SC.column_id = C.column_id
WHERE S.object_id = OBJECT_ID(N'Sales.Orders', N'U')
  AND auto_created = 1;

SET AUTO_CREATE_STATISTICS ON WITH NO_WAIT;
EXEC sys.sp_updatestats;
```

- read query plans; plan guides; DMVs;; statistics IO;; describe the different join types (HASH, MERGE, LOOP) and describe the scenarios they would be used in
- parameterized queries,

```
-- Clearing the cache
DBCC FREEPROCCACHE;

--sys.dm_exec_query_stats, sys.dm_exec_sql_text
SELECT qs.execution_count AS cnt,
 qt.text
FROM sys.dm_exec_query_stats AS qs
 CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) AS qt
WHERE qt.text LIKE N'%Orders%'
  AND qt.text NOT LIKE N'%qs.execution_count%'
ORDER BY qs.execution_count;
GO

CREATE PROCEDURE Sales.GetOrder
(@orderid INT)
AS
SELECT orderid, custid, empid, orderdate
FROM Sales.Orders
WHERE orderid = @orderid;
GO

ALTER PROCEDURE Sales.GetCustomerOrders
(@custid INT)
WITH RECOMPILE
AS
SELECT orderid, custid, empid, orderdate
FROM Sales.Orders
WHERE custid = @custid;
GO


-- columnstore index
CREATE COLUMNSTORE INDEX idx_cs_fact
  ON dbo.Fact(key1, key2, key3, measure1, measure2, measure3);
```

- Hint(table, query, join), plan guide

```
SELECT qty, COUNT(*) AS num
FROM Sales.OrderDetails
GROUP BY qty
OPTION (ORDER GROUP); --query hint
GO

SELECT orderid, productid, qty
FROM Sales.OrderDetails WITH (INDEX(idx_nc_productid)) --table hint
WHERE productid BETWEEN 10 AND 30
ORDER BY productid;

SELECT O.custid, O.orderdate, OD.orderid, OD.productid,OD.qty
FROM Sales.Orders AS O
 INNER MERGE JOIN Sales.OrderDetails AS OD  --join merge hint
  ON O.orderid = OD.orderid
WHERE O.orderid < 10250;

CREATE PROCEDURE Sales.GetCustomerOrders
(@custid INT)
AS
SELECT orderid, custid, empid, orderdate
FROM Sales.Orders
WHERE custid = @custid;
GO

-- Creating a plan guide
-- Optimizing for custid = 71
EXEC sys.sp_create_plan_guide
 @name = N'Cust71',
 @stmt = N'
  SELECT orderid, custid, empid, orderdate
  FROM Sales.Orders
  WHERE custid = @custid;',
 @type = N'OBJECT',
 @module_or_batch = N'Sales.GetCustomerOrders',
 @params = NULL,
 @hints = N'OPTION (OPTIMIZE FOR (@custid = 71))';
GO
```

- dynamic vs. parameterized queries, Table name cannot be variables

```
DECLARE @SQLString AS NVARCHAR(4000)
       , @tablename AS NVARCHAR(261) = '[Production].[Products]';
SET @SQLString = N'SELECT COUNT(*) AS TableRowCount FROM ' + @tablename;
EXEC(@SQLString);

-- Using sp_executesql avaid sql injection
DECLARE @SQLString AS NVARCHAR(4000), @address AS NVARCHAR(60);
SET @SQLString = N'
SELECT custid, companyname, contactname, contacttitle, address
FROM [Sales].[Customers]
WHERE address = @address';
SET @address = N'5678 rue de l''Abbaye';
EXEC sp_executesql
       @statement = @SQLString
       , @params = N'@address NVARCHAR(60)'
       , @address = @address;
```

```
--output parameters with sp_executesql
DECLARE @SQLString AS NVARCHAR(4000)
       , @outercount AS int;
SET @SQLString = N'SET @innercount = (SELECT COUNT(*) FROM Production.Products)';
EXEC sp_executesql
       @statment = @SQLString
       , @params = N'@innercount AS int OUTPUT'
       , @innercount = @outercount OUTPUT;
SELECT @outercount AS  'RowCount';
```

- indexes and statistics, a page is 8 kb unit, first page index allocation map, a balance tree nonclustered index on a heap->RID, nonclustered index on a clustered table->cluster key,

```
SELECT index_type_desc, page_count,
 record_count, avg_page_space_used_in_percent
FROM sys.dm_db_index_physical_stats
    (DB_ID(N'tempdb'), OBJECT_ID(N'dbo.TestStructure',N'U'), NULL, NULL ,
'DETAILED');
EXEC dbo.sp_spaceused @objname = N'dbo.TestStructure', @updateusage = true;
GO

CREATE CLUSTERED INDEX idx_cl_id ON dbo.TestStructure(id);

ALTER INDEX idx_cl_filler1 ON dbo.TestStructure REBUILD;
GO

SELECT OBJECT_NAME(S.object_id) AS table_name,
 I.name AS index_name,
 S.user_seeks, S.user_scans, s.user_lookups
FROM sys.dm_db_index_usage_stats AS S
 INNER JOIN sys.indexes AS i
  ON S.object_id = I.object_id
   AND S.index_id = I.index_id
WHERE S.object_id = OBJECT_ID(N'Sales.Orders', N'U');
GO

CREATE NONCLUSTERED INDEX idx_nc_shipregion ON Sales.Orders(shipregion);
CREATE NONCLUSTERED INDEX idx_nc_shipcity_i_custid ON Sales.Orders(shipcity)
INCLUDE (custid);
```

16. Manage transactions
    - Implicit vs explicit

```
--  Implicit Transactions
USE TSQL2012;
SET IMPLICIT_TRANSACTIONS ON;
SELECT @@TRANCOUNT; -- 0
INSERT INTO Production.Products(productid, productname, supplierid, categoryid,
       unitprice, discontinued)
       VALUES(101, N'Test2: Bad categoryid', 1, 1, 18.00, 0);
SELECT @@TRANCOUNT; -- 1
COMMIT TRAN;
SELECT @@TRANCOUNT; -- 0
SET IMPLICIT_TRANSACTIONS OFF;

-- Explicit Transactions
```

```
USE TSQL2012;
SELECT @@TRANCOUNT; -- 0
BEGIN TRAN;
	SELECT @@TRANCOUNT; -- 1
	INSERT INTO Production.Products(productid, productname, supplierid,
categoryid,   unitprice, discontinued)
		VALUES(101, N'Test2: Bad categoryid', 1, 1, 18.00, 0);
	SELECT @@TRANCOUNT; -- 1
COMMIT TRAN;
SELECT @@TRANCOUNT; -- 0

-- Nested Transactions with COMMIT TRAN
SELECT @@TRANCOUNT; -- = 0
BEGIN TRAN;
	SELECT @@TRANCOUNT; -- = 1
	BEGIN TRAN;
		SELECT @@TRANCOUNT; -- = 2
		-- Issue data modification or DDL commands here
	COMMIT
	SELECT @@TRANCOUNT; -- = 1
COMMIT TRAN;
SELECT @@TRANCOUNT; -- = 0
```

- isolation levels; https://msdn.microsoft.com/en-us/library/ms173763%28v=sql.110%29.aspx
- read committed—default, cannot read under writing data; read uncommitted—dirty read, reader is not blocked by writers; read committed snapshot—read before under writing data. Stronger—repeatable read—update, delete are prevented, phantom read; weaker—snapshot—read before under writing data; serializable, holdlock—strongest—all reads are repeatable, and no new rows will be inserted if they will appear in the select.

```
-- Working with READ UNCOMMITTED
-- Session 1
BEGIN TRAN;

UPDATE HR.Employees
SET region = N'1004'
WHERE empid = 1;

ROLLBACK TRAN;

-- Session 2
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

SELECT lastname, firstname, region
FROM HR.Employees

<Results returned: region = 1004 for empid = 1>

SELECT lastname, firstname, region
FROM HR.Employees;

<Results returned: region = original value for empid = 1>

-- Using READ COMMITTED SNAPSHOT
```

```
-- Session 1
ALTER DATABASE Master SET READ_COMMITTED_SNAPSHOT ON;
BEGIN TRAN;

UPDATE HR.Employees
SET postalcode = N'10007'
WHERE empid = 1;

ROLLBACK TRAN;

<region for empid = 1 rolled back to original value>

-- Session 2
USE TSQL2012;

SELECT lastname, firstname, region
FROM HR.Employees;

<Results returned show region in original state for empid = 1>
```

17. Evaluate the use of row-based operations vs. set-based operations

- Use Cursors to drop auto-created statistics

```
DECLARE @statistics_name AS NVARCHAR(128), @ds AS NVARCHAR(1000);
DECLARE acs_cursor CURSOR FOR
SELECT name AS statistics_name
FROM sys.stats
WHERE object_id = OBJECT_ID(N'Sales.Orders', N'U')
  AND auto_created = 1;
OPEN acs_cursor;
FETCH NEXT FROM acs_cursor INTO @statistics_name;
WHILE @@FETCH_STATUS = 0
BEGIN
 SET @ds = N'DROP STATISTICS Sales.Orders.' + @statistics_name +';';
 EXEC(@ds);
 FETCH NEXT FROM acs_cursor INTO @statistics_name;
END;
CLOSE acs_cursor;
DEALLOCATE acs_cursor;
GO
```

- use cursors;

```
DECLARE @curcustid AS INT;

DECLARE cust_cursor CURSOR FAST_FORWARD FOR
  SELECT custid
  FROM Sales.Customers;

OPEN cust_cursor;

FETCH NEXT FROM cust_cursor INTO @curcustid;

WHILE @@FETCH_STATUS = 0
BEGIN
  EXEC Sales.ProcessCustomer @custid = @curcustid;
```

```
  FETCH NEXT FROM cust_cursor INTO @curcustid;
END;

CLOSE cust_cursor;

DEALLOCATE cust_cursor;
GO
```

- impact of scalar UDFs; combine multiple DML operations

18. Implement error handling

- Error, message: error number, severity level, state, error message

```
RAISERROR ('Error in % stored procedure', 16, 0, N'usp_InsertCategories');

DECLARE @message AS NVARCHAR(1000) = N'Error in % stored procedure';
SELECT @message = FORMATMESSAGE (@message, N'usp_InsertCategories');
RAISERROR (@message, 16, 0);

THROW 50000, 'Error in usp_InsertCategories stored procedure', 0;

DECLARE @errnum AS int;
BEGIN TRAN;
SET IDENTITY_INSERT Production.Products ON;
INSERT INTO Production.Products(productid, productname, supplierid, categoryid,
unitprice, discontinued)
       VALUES(1, N'Test1: Ok categoryid', 1, 1, 18.00, 0);
SET @errnum = @@ERROR;
IF @errnum <> 0 -- Handle the error
       BEGIN
              PRINT 'Insert into Production.Products failed with error ' +
CAST(@errnum AS VARCHAR);
       END
```

- XACT_ABORT with THROW, XACT_ABORT skips the batches when error > 10 or see a THROW

```
SET XACT_ABORT ON;
PRINT 'Before error';
SET IDENTITY_INSERT Production.Products ON;
INSERT INTO Production.Products(productid, productname, supplierid, categoryid,
unitprice, discontinued)
       VALUES(1, N'Test1: Ok categoryid', 1, 1, 18.00, 0);
SET IDENTITY_INSERT Production.Products OFF;
PRINT 'After error';
GO
PRINT 'New batch';
SET XACT_ABORT OFF;


SET XACT_ABORT ON;
PRINT 'Before error';
THROW 50000, 'Error in usp_InsertCategories stored procedure', 0;
```

```
PRINT 'After error';
GO
PRINT 'New batch';
SET XACT_ABORT OFF;
```

- XACT_ABORT in TRAN, skip the batch and rollback

```
DECLARE @errnum AS int;
SET XACT_ABORT ON;
BEGIN TRAN;
	SET IDENTITY_INSERT Production.Products ON;
	-- Insert #1 will fail because of duplicate primary key
	INSERT INTO Production.Products(productid, productname, supplierid,
categoryid,   unitprice, discontinued)
		VALUES(1, N'Test1: Ok categoryid', 1, 1, 18.00, 0);
	SET @errnum = @@ERROR;
	IF @errnum <> 0
		BEGIN
			IF @@TRANCOUNT > 0 ROLLBACK TRAN;
			PRINT 'Error in first INSERT';
		END;
	-- Insert #2 no longer succeeds
	INSERT INTO Production.Products(productid, productname, supplierid,
categoryid,   unitprice, discontinued)
		VALUES(101, N'Test2: Bad categoryid', 1, 1, 18.00, 0);
	SET @errnum = @@ERROR;
	IF @errnum <> 0
		BEGIN
			-- Take actions based on the error
			IF @@TRANCOUNT > 0 ROLLBACK TRAN;
			PRINT 'Error in second INSERT';
		END;
	SET IDENTITY_INSERT Production.Products OFF;
	IF @@TRANCOUNT > 0 COMMIT TRAN;
GO

SELect * FROM Production.Products WHERE productid = 101;
PRINT 'Deleted ' + CAST(@@ROWCOUNT AS VARCHAR) + ' rows';
SET XACT_ABORT OFF;
GO
SELECT XACT_STATE(), @@TRANCOUNT;
```

- TRY/CATCH

```
BEGIN TRY
BEGIN TRAN;
	SET IDENTITY_INSERT Production.Products ON;
	INSERT INTO Production.Products(productid, productname, supplierid,
categoryid,          unitprice, discontinued)
		VALUES(1, N'Test1: Ok categoryid', 1, 1, 18.00, 0);
	INSERT INTO Production.Products(productid, productname, supplierid,
categoryid,          unitprice, discontinued)
```

```
            VALUES(101, N'Test2: Bad categoryid', 1, 10, 18.00, 0);
      SET IDENTITY_INSERT Production.Products OFF;
COMMIT TRAN;
END TRY
BEGIN CATCH
      SELECT XACT_STATE() as 'XACT_STATE', @@TRANCOUNT as '@@TRANCOUNT';
      IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION;
      THROW;
END CATCH;
GO
SELECT XACT_STATE() as 'XACT_STATE', @@TRANCOUNT as '@@TRANCOUNT';
```

- Implement try/catch/throw; use set based rather than row based logic; transaction management

70-462 Exam Notes

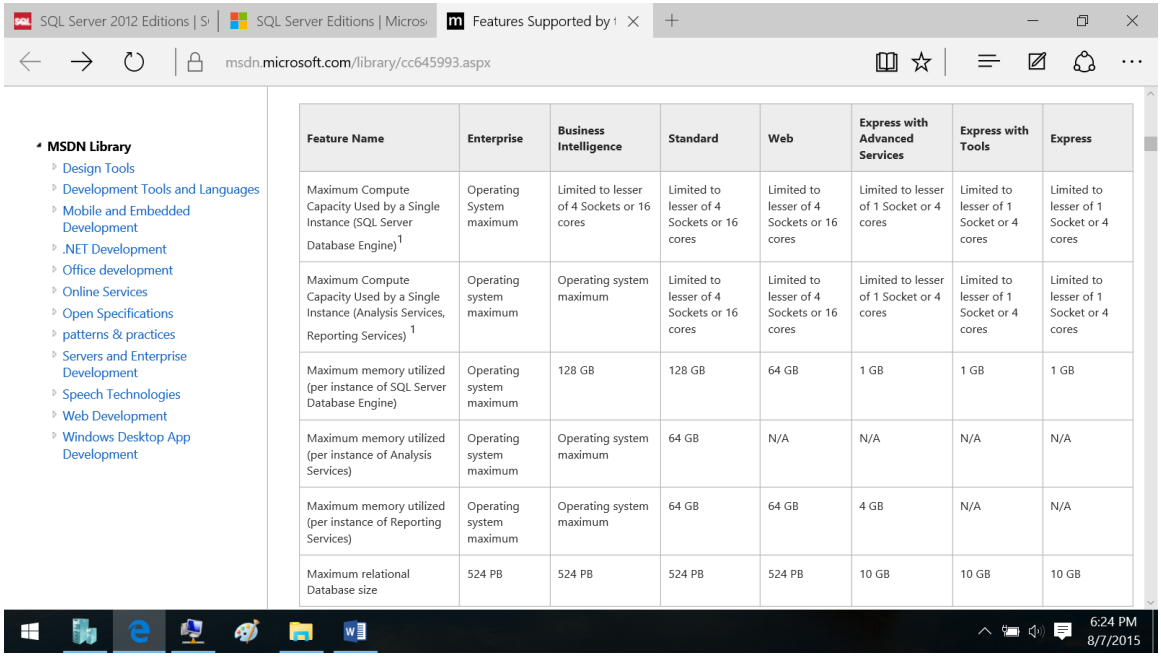
Ron Wu

Last update: 8/9/15

Install and configure (19%)

1. Plan installation

    - Evaluate installation requirements;



Feature Name	Enterprise	Business Intelligence	Standard	Web	Express with Advanced Services	Express with Tools	Express
Maximum Compute Capacity Used by a Single Instance (SQL Server Database Engine)[1]	Operating System maximum	Limited to lesser of 4 Sockets or 16 cores	Limited to lesser of 4 Sockets or 16 cores	Limited to lesser of 4 Sockets or 16 cores	Limited to lesser of 1 Socket or 4 cores	Limited to lesser of 1 Socket or 4 cores	Limited to lesser of 1 Socket or 4 cores
Maximum Compute Capacity Used by a Single Instance (Analysis Services, Reporting Services) [1]	Operating system maximum	Operating system maximum	Limited to lesser of 4 Sockets or 16 cores	Limited to lesser of 4 Sockets or 16 cores	Limited to lesser of 1 Socket or 4 cores	Limited to lesser of 1 Socket or 4 cores	Limited to lesser of 1 Socket or 4 cores
Maximum memory utilized (per instance of SQL Server Database Engine)	Operating system maximum	128 GB	128 GB	64 GB	1 GB	1 GB	1 GB
Maximum memory utilized (per instance of Analysis Services)	Operating system maximum	Operating system maximum	64 GB	N/A	N/A	N/A	N/A
Maximum memory utilized (per instance of Reporting Services)	Operating system maximum	Operating system maximum	64 GB	64 GB	4 GB	N/A	N/A
Maximum relational Database size	524 PB	524 PB	524 PB	524 PB	10 GB	10 GB	10 GB

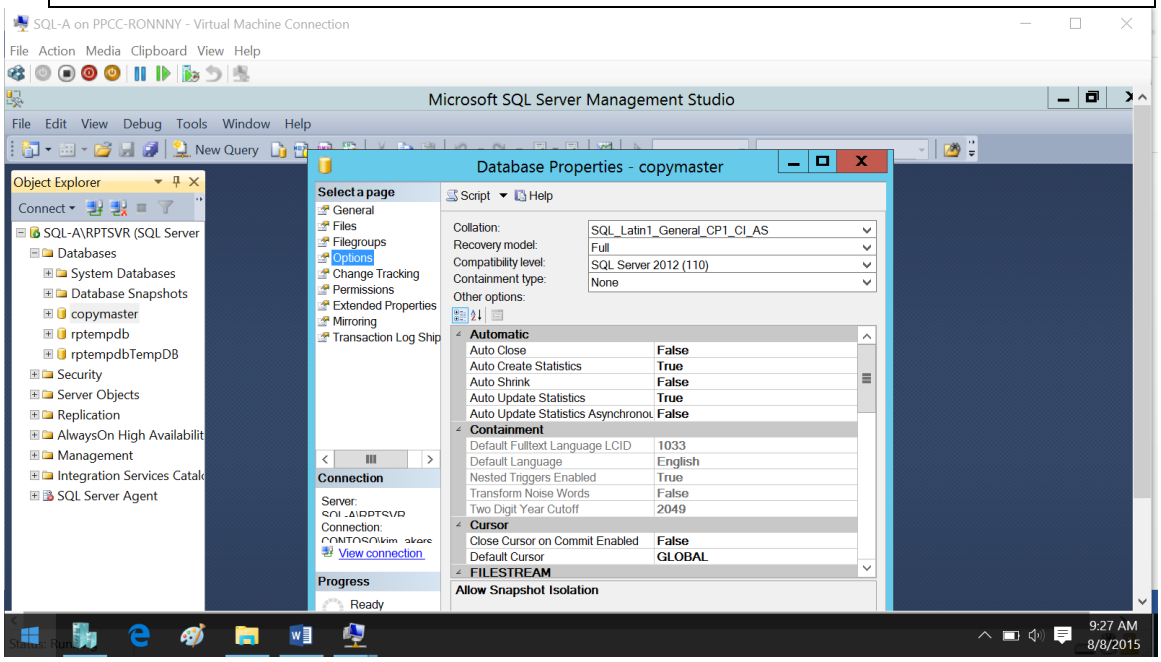
    - design the installation of SQL Server and its components (drives, service accounts, etc.);

    Cross-Box Scale Limits, High Availability, Scalability and Performance; Security ,Replication, Management Tools , RDBMS Manageability , Development Tools , Programmability , Integration Services , Integration Services-Advanced Adapters , Integration Services-Advanced Transforms , Master Data Services , Data Warehouse , Analysis Services , BI Semantic Model (Multidimensional) , BI Semantic Model (Tabular) , PowerPivot for SharePoint , Data Mining , Reporting Services , Business Intelligence Clients , Spatial and Location Services , Additional Database Services

    - plan scale-up vs. scale-out basics; plan for capacity, including if/when to shrink, grow, autogrow, and monitor growth;

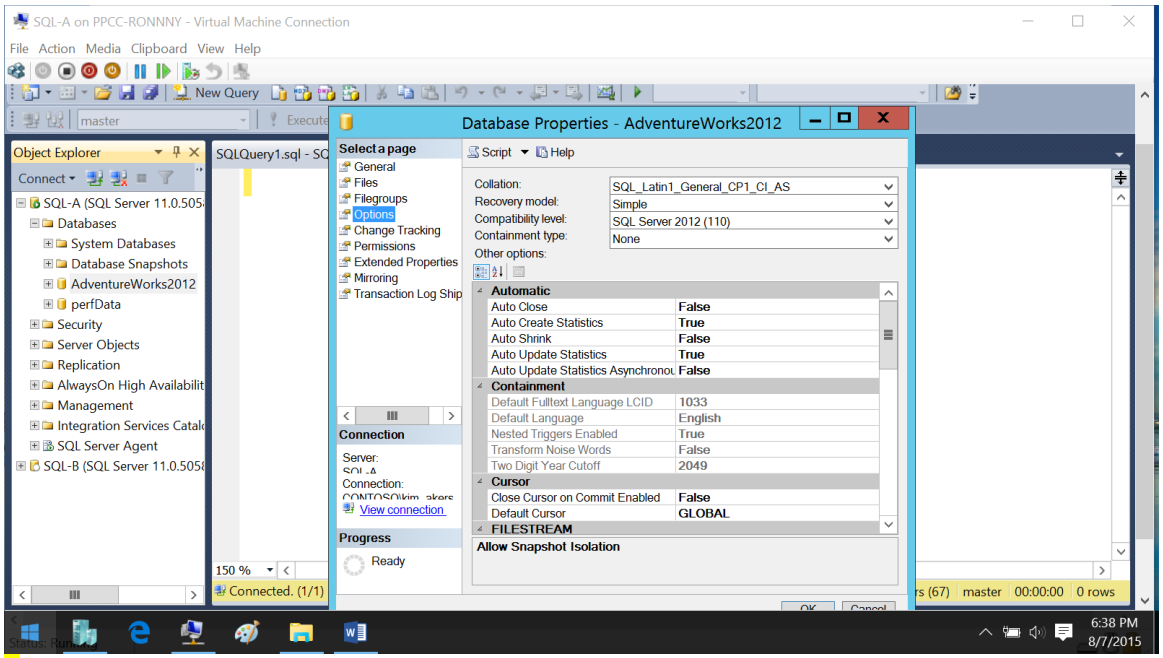
```
USE [master]
GO
ALTER DATABASE [AdventureWorks2012] MODIFY FILE
```

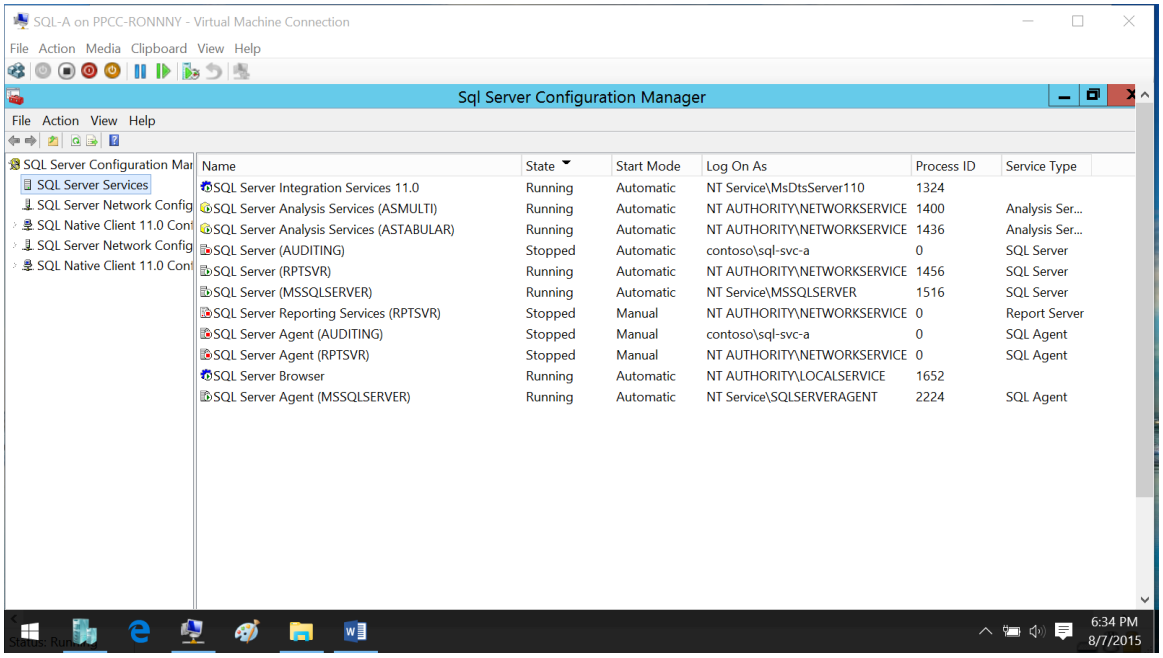
```
( NAME= N'AdventureWorks2012_Data', SIZE = 256000KB )
GO

Use [AdventureWorks2012]
GO
DBCC SHRINKDATABASE(N'AdventureWorks2012')
GO
```



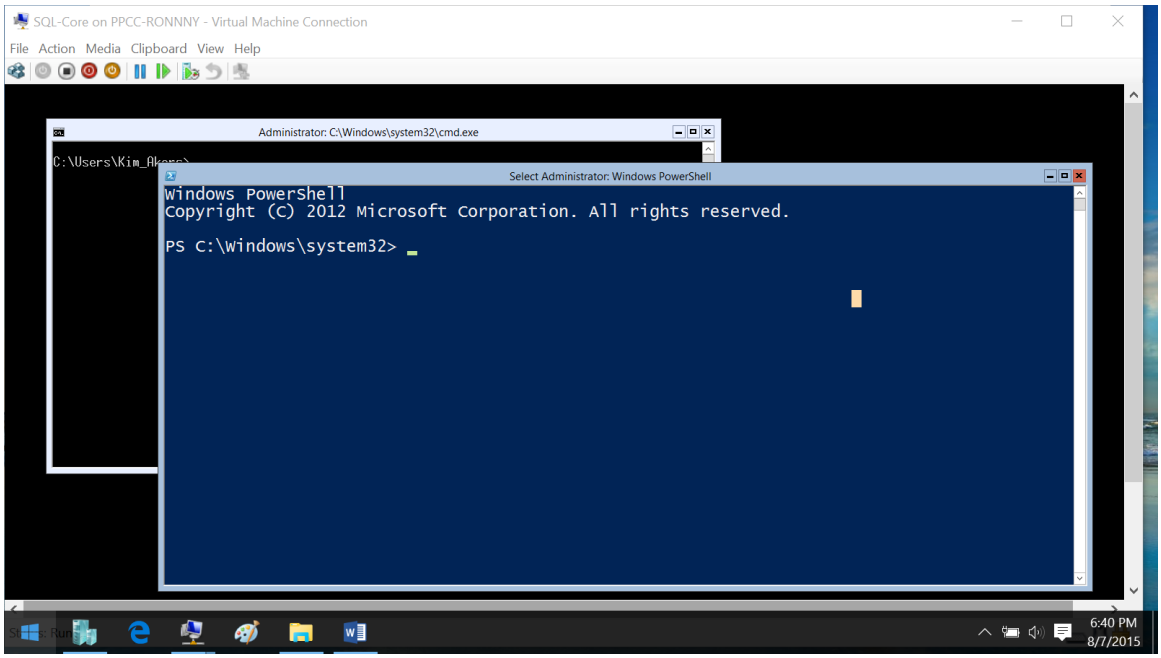
- manage the technologies that influence SQL architecture (for example, service broker, full text, scale out, etc.); design the storage for new databases (drives, filegroups, partitioning); design database infrastructure; configure a SQL Server standby database for reporting purposes; Windows-level security and service level security;

```
ALTER DATABASE [AdventureWorks2012] ADD FILE ( NAME = N'ExtraFile',
FILENAME = N'C:\AdventureWorks2012\ExtraFile.ndf' , SIZE = 4096KB,
FILEGROWTH = 1024KB ) TO FILEGROUP [PRIMARY]

ALTER DATABASE [AdventureWorks2012] ADD FILEGROUP [Additional
```

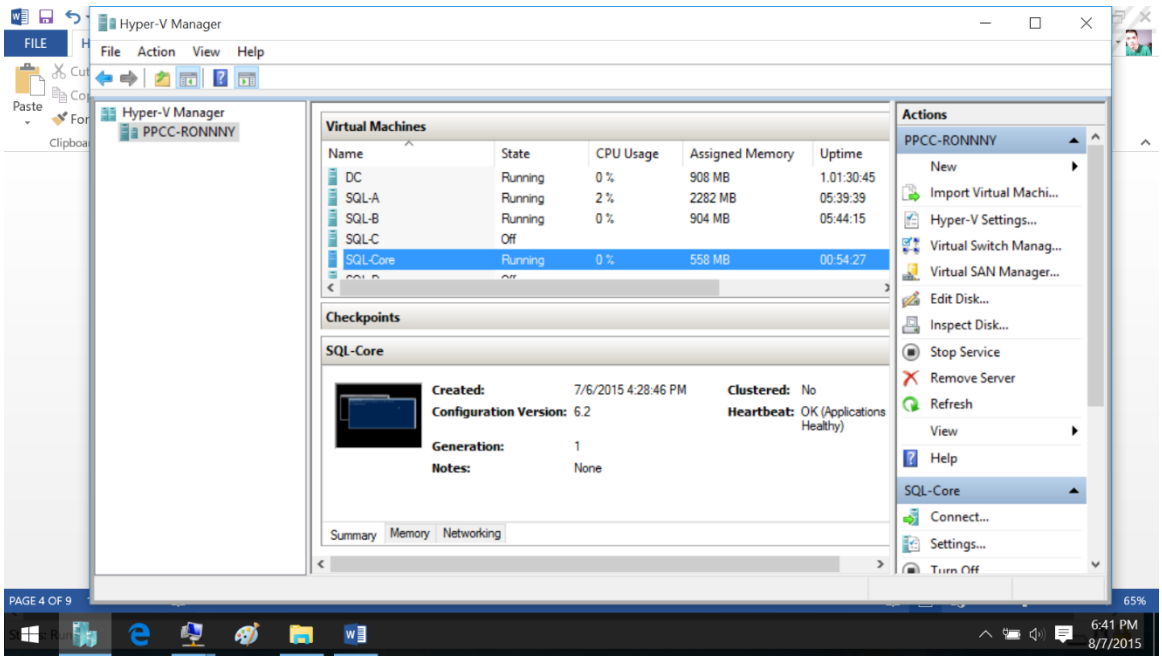




- Core mode installation; benchmark a server before using it in a production environment (SQLIO, Tests on SQL Instance); choose the right hardware

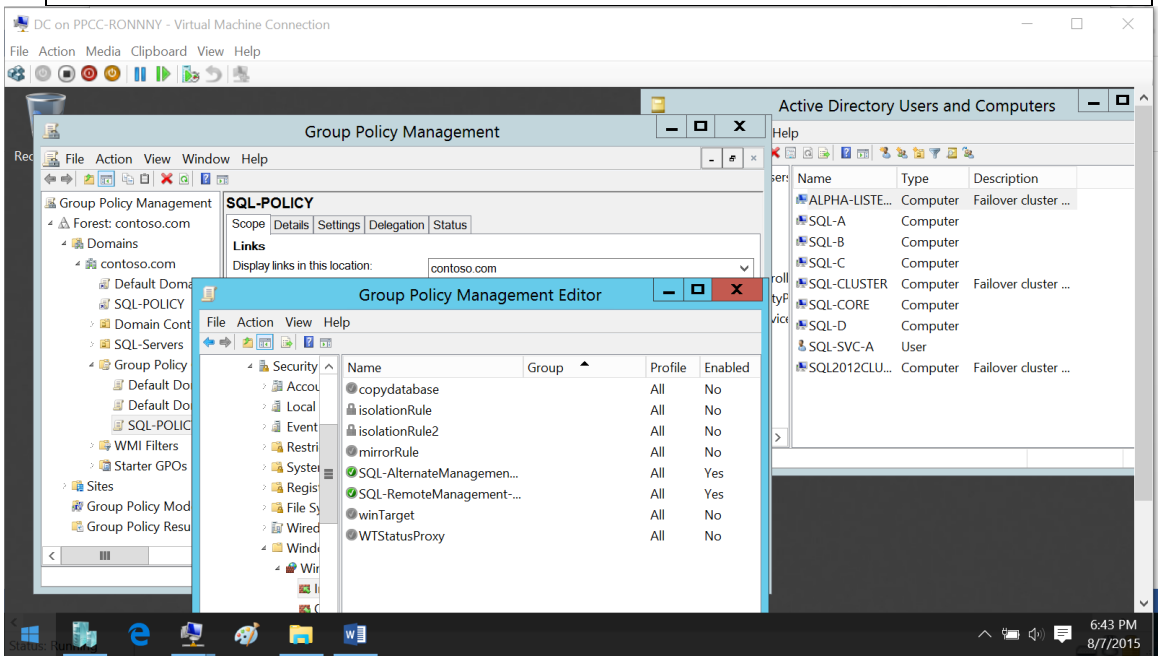


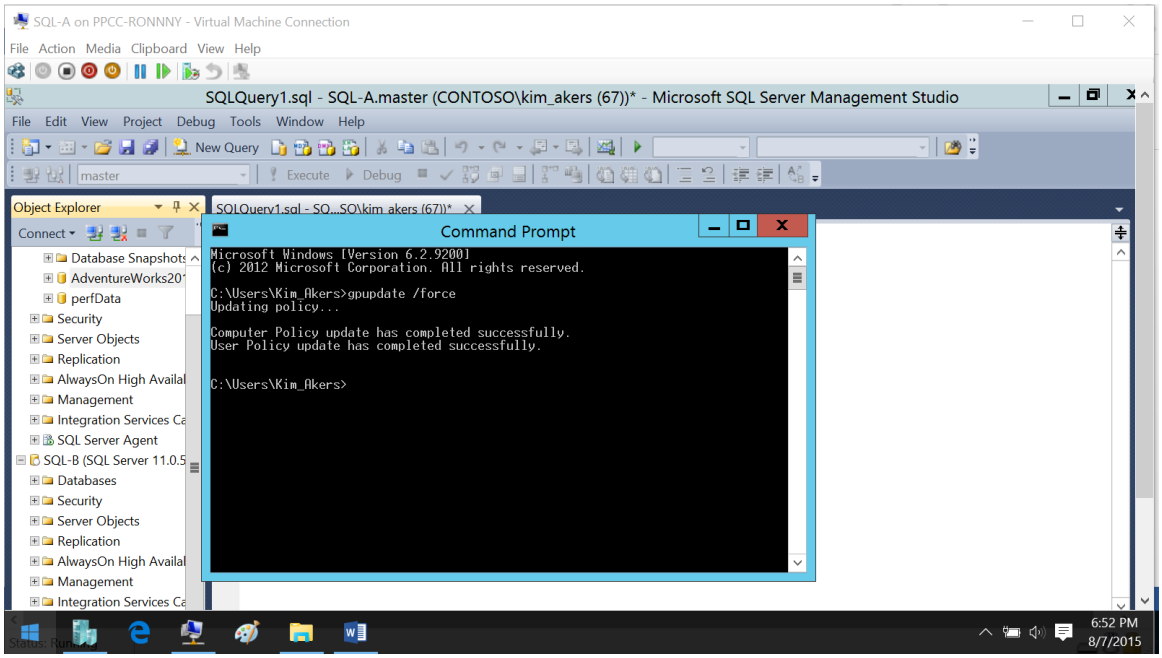
- Install SQL Server and related services
  - Test connectivity; enable and disable features; install SQL Server database engine and SSIS (not SSRS and SSAS); configure an OS disk




```
Add-WindowsFeature NET-Framework-Core -Source:D:\sources\sxs\

Setup.exe /qs /ACTION=Install /FEATURES=SQL /INSTANCENAME=MSSQLSERVER
/updateEnabled=fales /SQLSVCACCOUNT="contoso\kim_akers" /SQLSVCPASSWORD="Pa$$w0rd"
```

```
/SQLSYSADMINACCOUNTS="contoso\kim_akers" /AGTSVCACCOUNT="NT AUTHORITY\Network Service" /IACCEPTSQLSERVERLICENSETERMS
```



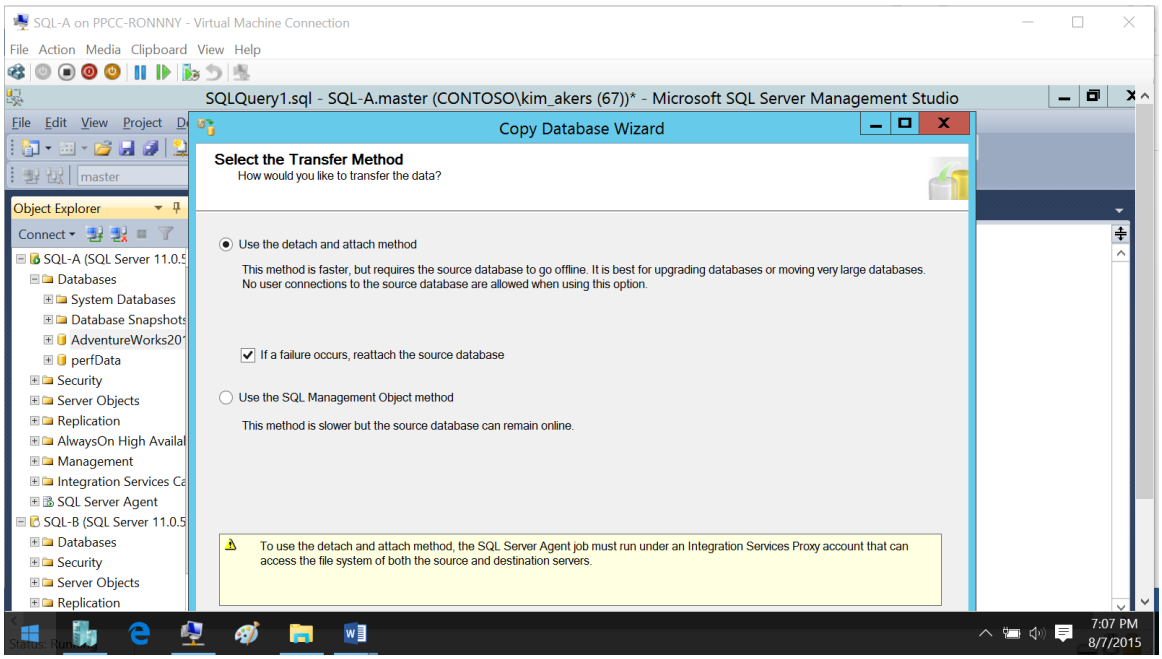


- Implement a migration strategy
  - Restore vs detach/attach; migrate security; migrate from a previous version; migrate to new hardware; migrate systems and data from other sources

```
USE master;
```

```
GO
EXEC sp_detach_db @dbname = AdventureWorks2012;
GO

USE master;
GO
CREATE DATABASE AdventureWorks2012 ON (Filename =
'C:\AdventureWorks2012\AdventureWorks2012.mdf'),
(FILENAME = 'C:\AdventureWorks2012\AdventureWorks2012_log.ldf') FOR ATTACH;
GO
```

Turn on sql server agent->Copy wizard->integration services->integration services proxy



migration of logins: create script - KB918992 article

```
USE master
GO
IF OBJECT_ID ('sp_hexadecimal') IS NOT NULL
  DROP PROCEDURE sp_hexadecimal
GO
CREATE PROCEDURE sp_hexadecimal
    @binvalue varbinary(256),
    @hexvalue varchar (514) OUTPUT
AS
DECLARE @charvalue varchar (514)
DECLARE @i int
DECLARE @length int
DECLARE @hexstring char(16)
SELECT @charvalue = '0x'
SELECT @i = 1
SELECT @length = DATALENGTH (@binvalue)
SELECT @hexstring = '0123456789ABCDEF'
WHILE (@i <= @length)
BEGIN
  DECLARE @tempint int
  DECLARE @firstint int
  DECLARE @secondint int
```

```
  SELECT @tempint = CONVERT(int, SUBSTRING(@binvalue,@i,1))
  SELECT @firstint = FLOOR(@tempint/16)
  SELECT @secondint = @tempint - (@firstint*16)
  SELECT @charvalue = @charvalue +
    SUBSTRING(@hexstring, @firstint+1, 1) +
    SUBSTRING(@hexstring, @secondint+1, 1)
  SELECT @i = @i + 1
END

SELECT @hexvalue = @charvalue
GO

IF OBJECT_ID ('sp_help_revlogin') IS NOT NULL
  DROP PROCEDURE sp_help_revlogin
GO
CREATE PROCEDURE sp_help_revlogin @login_name sysname = NULL AS
DECLARE @name sysname
DECLARE @type varchar (1)
DECLARE @hasaccess int
DECLARE @denylogin int
DECLARE @is_disabled int
DECLARE @PWD_varbinary  varbinary (256)
DECLARE @PWD_string  varchar (514)
DECLARE @SID_varbinary varbinary (85)
DECLARE @SID_string varchar (514)
DECLARE @tmpstr  varchar (1024)
DECLARE @is_policy_checked varchar (3)
DECLARE @is_expiration_checked varchar (3)

DECLARE @defaultdb sysname

IF (@login_name IS NULL)
  DECLARE login_curs CURSOR FOR

      SELECT p.sid, p.name, p.type, p.is_disabled, p.default_database_name,
l.hasaccess, l.denylogin FROM
sys.server_principals p LEFT JOIN sys.syslogins l
      ON ( l.name = p.name ) WHERE p.type IN ( 'S', 'G', 'U' ) AND p.name <> 'sa'
ELSE
  DECLARE login_curs CURSOR FOR


      SELECT p.sid, p.name, p.type, p.is_disabled, p.default_database_name,
l.hasaccess, l.denylogin FROM
sys.server_principals p LEFT JOIN sys.syslogins l
      ON ( l.name = p.name ) WHERE p.type IN ( 'S', 'G', 'U' ) AND p.name =
@login_name
OPEN login_curs

FETCH NEXT FROM login_curs INTO @SID_varbinary, @name, @type, @is_disabled,
@defaultdb, @hasaccess, @denylogin
IF (@@fetch_status = -1)
BEGIN
  PRINT 'No login(s) found.'
  CLOSE login_curs
  DEALLOCATE login_curs
  RETURN -1
END
```

```
SET @tmpstr = '/* sp_help_revlogin script '
PRINT @tmpstr
SET @tmpstr = '** Generated ' + CONVERT (varchar, GETDATE()) + ' on ' +
@@SERVERNAME + ' */'
PRINT @tmpstr
PRINT ''
WHILE (@@fetch_status <> -1)
BEGIN
  IF (@@fetch_status <> -2)
  BEGIN
    PRINT ''
    SET @tmpstr = '-- Login: ' + @name
    PRINT @tmpstr
    IF (@type IN ( 'G', 'U'))
    BEGIN -- NT authenticated account/group

      SET @tmpstr = 'CREATE LOGIN ' + QUOTENAME( @name ) + ' FROM WINDOWS WITH
DEFAULT_DATABASE = [' + @defaultdb + ']'
    END
    ELSE BEGIN -- SQL Server authentication
        -- obtain password and sid
            SET @PWD_varbinary = CAST( LOGINPROPERTY( @name, 'PasswordHash' ) AS
varbinary (256) )
        EXEC sp_hexadecimal @PWD_varbinary, @PWD_string OUT
        EXEC sp_hexadecimal @SID_varbinary,@SID_string OUT

        -- obtain password policy state
        SELECT @is_policy_checked = CASE is_policy_checked WHEN 1 THEN 'ON' WHEN 0
THEN 'OFF' ELSE NULL END FROM sys.sql_logins WHERE name = @name
        SELECT @is_expiration_checked = CASE is_expiration_checked WHEN 1 THEN
'ON' WHEN 0 THEN 'OFF' ELSE NULL END FROM sys.sql_logins WHERE name = @name

            SET @tmpstr = 'CREATE LOGIN ' + QUOTENAME( @name ) + ' WITH PASSWORD =
' + @PWD_string + ' HASHED, SID = ' + @SID_string + ', DEFAULT_DATABASE = [' +
@defaultdb + ']'

        IF ( @is_policy_checked IS NOT NULL )
        BEGIN
          SET @tmpstr = @tmpstr + ', CHECK_POLICY = ' + @is_policy_checked
        END
        IF ( @is_expiration_checked IS NOT NULL )
        BEGIN
          SET @tmpstr = @tmpstr + ', CHECK_EXPIRATION = ' + @is_expiration_checked
        END
    END
    IF (@denylogin = 1)
    BEGIN -- login is denied access
      SET @tmpstr = @tmpstr + '; DENY CONNECT SQL TO ' + QUOTENAME( @name )
    END
    ELSE IF (@hasaccess = 0)
    BEGIN -- login exists but does not have access
      SET @tmpstr = @tmpstr + '; REVOKE CONNECT SQL TO ' + QUOTENAME( @name )
    END
    IF (@is_disabled = 1)
    BEGIN -- login is disabled
      SET @tmpstr = @tmpstr + '; ALTER LOGIN ' + QUOTENAME( @name ) + ' DISABLE'
    END
    PRINT @tmpstr
```

```
  END

  FETCH NEXT FROM login_curs INTO @SID_varbinary, @name, @type, @is_disabled,
@defaultdb, @hasaccess, @denylogin
   END
CLOSE login_curs
DEALLOCATE login_curs
RETURN 0
GO
```

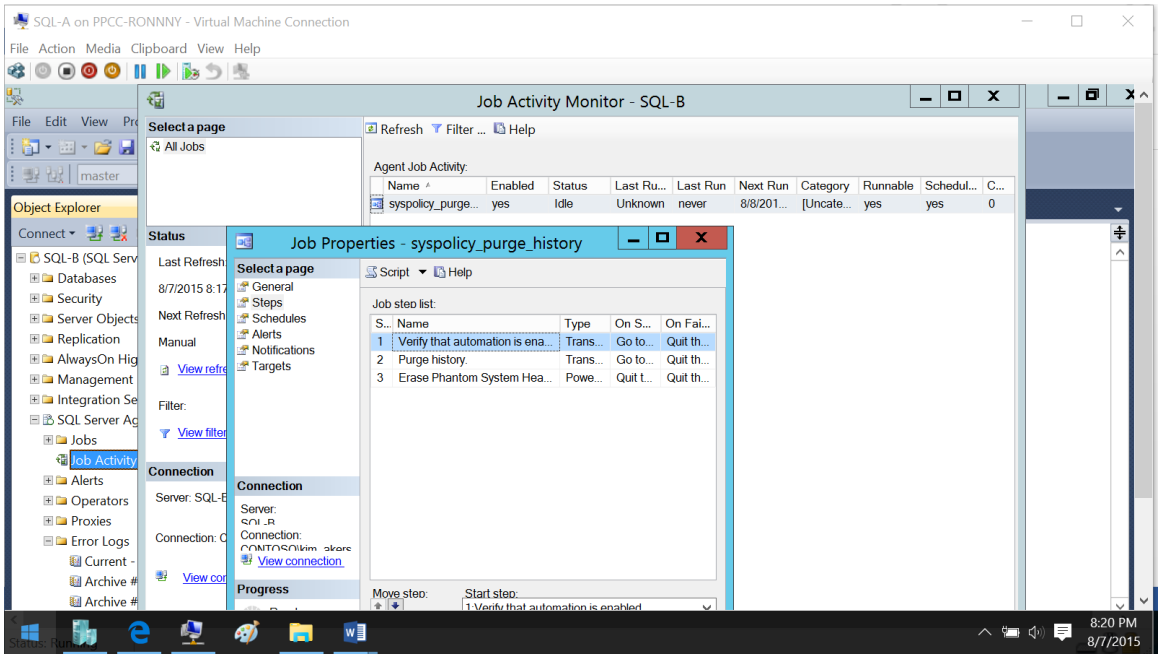
- Configure additional SQL Server components
  - Set up and configure all SQL Server components (Engine, AS, RS and SharePoint integration) in a complex and highly secure environment; configure full-text indexing; SSIS security; filestream; filetable

```
USE [master]
GO
CREATE DATABASE [Litware2012]
GO
ALTER DATABASE [Litware2012] ADD FILEGROUP [Tertiary]
GO
ALTER DATABASE [Litware2012] ADD FILEGROUP [Quaternary]
GO
EXEC sp_configure filestream_access_level, 2
GO
RECONFIGURE
GO
ALTER DATABASE Litware2012 ADD FILEGROUP FileStreamFileGroup CONTAINS FILESTREAM;
GO
ALTER DATABASE Litware2012 ADD FILE (
NAME = FileStrmFile,
FILENAME = 'C:\FSTRM')
TO FILEGROUP FileStreamFileGroup;
ALTER DATABASE Litware2012
SET FILESTREAM (NON_TRANSACTED_ACCESS = FULL, DIRECTORY_NAME = 'FTBLE');
USE [LitWare2012]
CREATE TABLE DocStore as FileTable;
GO
sp_configure 'contained database authentication', 1;
GO
RECONFIGURE;
GO
USE [master]
GO
ALTER DATABASE [Litware2012] SET CONTAINMENT = PARTIAL
GO
CREATE USER contained user WITH PASSWORD = 'Pa$$w0rd';
USE master;
GO
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'P@ssw0rd';
GO
CREATE CERTIFICATE ServerCertificate WITH SUBJECT = 'Server Certificate';
GO
USE [LitWare2012];
GO
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_128
```

```
ENCRYPTION BY SERVER CERTIFICATE ServerCertificate;
GO
ALTER DATABASE LitWare2012
SET ENCRYPTION ON;
GO
```

- Manage SQL Server Agent
  - Create, maintain, and monitor jobs;

    Mail profile, error log
  - administer jobs and alerts;

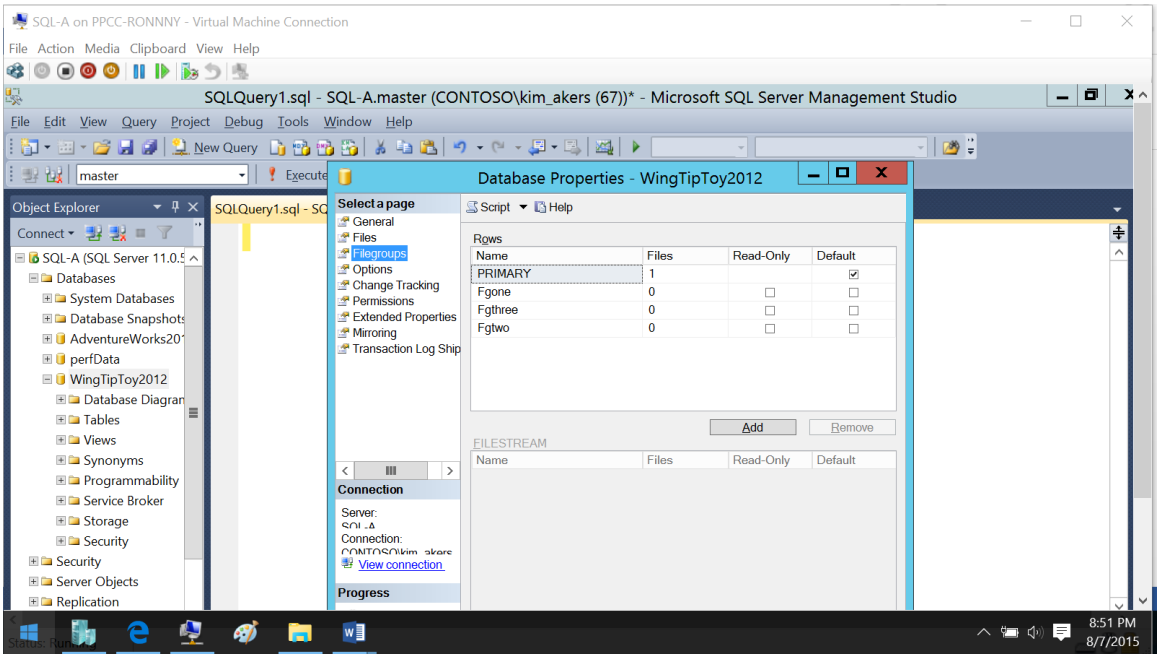


  - automate (setup, maintenance, monitoring) across multiple databases and multiple instances; send to "Manage SQL Server Agent jobs"

Maintain instances and databases (17%)

- Manage and configure databases

- Design multiple file groups; database configuration and standardization: autoclose, autoshrink, recovery models; manage file space, including adding new filegroups and moving objects from one filegroup to another;



- implement and configure contained databases;

```
sp_configure 'contained database authentication', 1;
GO
RECONFIGURE;
GO
```

- data compression;

```
USE [AdventureWorks2012]
ALTER TABLE [Sales].[Customer] REBUILD PARTITION = ALL
WITH (DATA_COMPRESSION = ROW)

ALTER INDEX indexName ON tableName REBUILD PARTITION ALL WITH
(DATA_COMPRESSION=ROW)

--OR

ALTER TABLE [Sales].[Customer] REBUILD PARTITION = ALL
WITH
(DATA_COMPRESSION = PAGE)
```

- configure TDE;

```
USE master---must be in master
GO
CREATE MASTER KEY ENCRYPTION BY PASSWORD = '<MasterKeyPasswordHere>';
GO
---must be in master
CREATE CERTIFICATE ServerCertificate WITH SUBJECT = 'Server Certificate';
GO
```

```
USE AdventureWorks2012;
GO
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_128
ENCRYPTION BY SERVER CERTIFICATE ServerCertificate;
GO
ALTER DATABASE AdventureWorks2012
SET ENCRYPTION ON;
GO
```

- partitioning;

```
CREATE PARTITION FUNCTION PartFunction (int)
as RANGE LEFT FOR VALUES (50);
GO

CREATE PARTITION SCHEME PartScheme
AS PARTITION PartFunction
   TO (FgOne, FgTwo);
GO
```

- manage log file growth; DBCC:

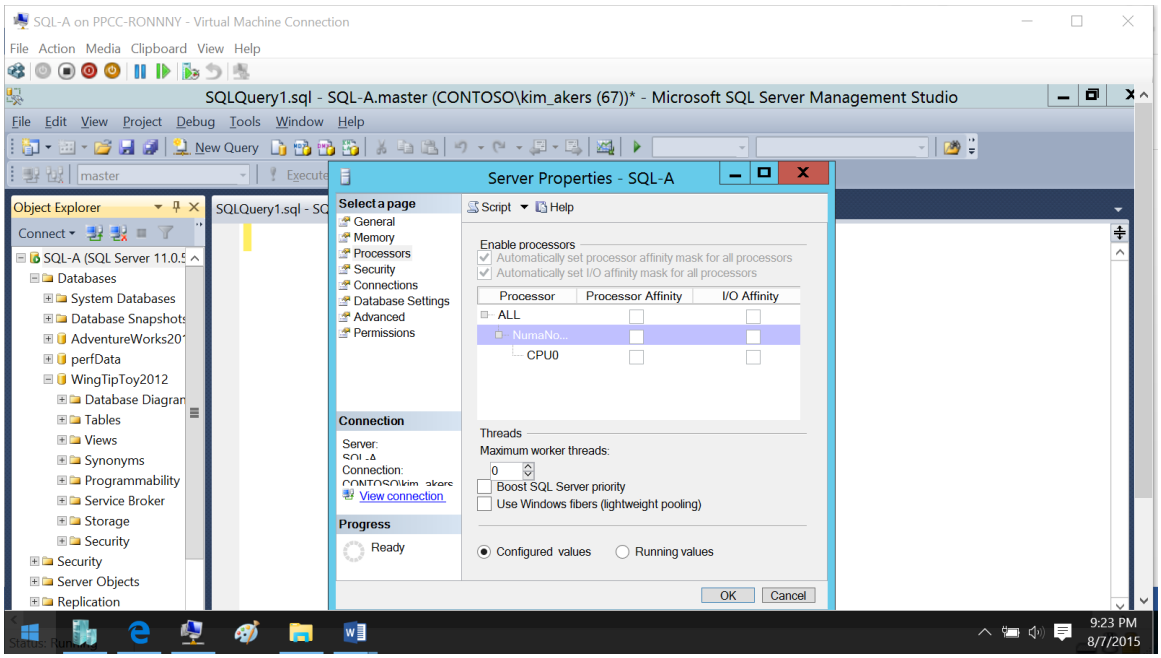
  DBCC SQLPERF(LOGSPACE), DBCC CHECKDB

- Configure SQL Server instances
  - Configure and standardize a database: autoclose, autoshrink, recovery models;

```
USE [master]
GO
ALTER DATABASE [model] SET RECOVERY FULL WITH NO_WAIT
GO
ALTER DATABASE [model] SET AUTO_CLOSE ON WITH NO_WAIT
GO
ALTER DATABASE [model] SET AUTO_SHRINK ON WITH NO_WAIT
```

- install default and named instances; configure SQL to use only certain CPUs (affinity masks, etc.);

```
ALTER SERVER CONFIGURATION SET PROCESS AFFINITY CUP = AUTO
```



- configure server level settings; configure many databases/instance, many instances/server, virtualization; configure clustered instances including MSDTC(distributed transaction coordinator); memory allocation;

```
EXEC sys.sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
EXEC sys.sp_configure 'min server memory', 1024;
GO
EXEC sys.sp_configure 'max server memory', 4096;
GO
RECONFIGURE;
GO
```

- database mail;

```
sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
sp_configure 'Database Mail XPs', 1;
GO
RECONFIGURE;
GO
EXECUTE msdb.dbo.sysmail_add_account_sp
@account_name = 'Example',
@email_address = 'example@contoso.com',
@mailserver_name = 'smtp.contoso.com';

EXEC sys.sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
```

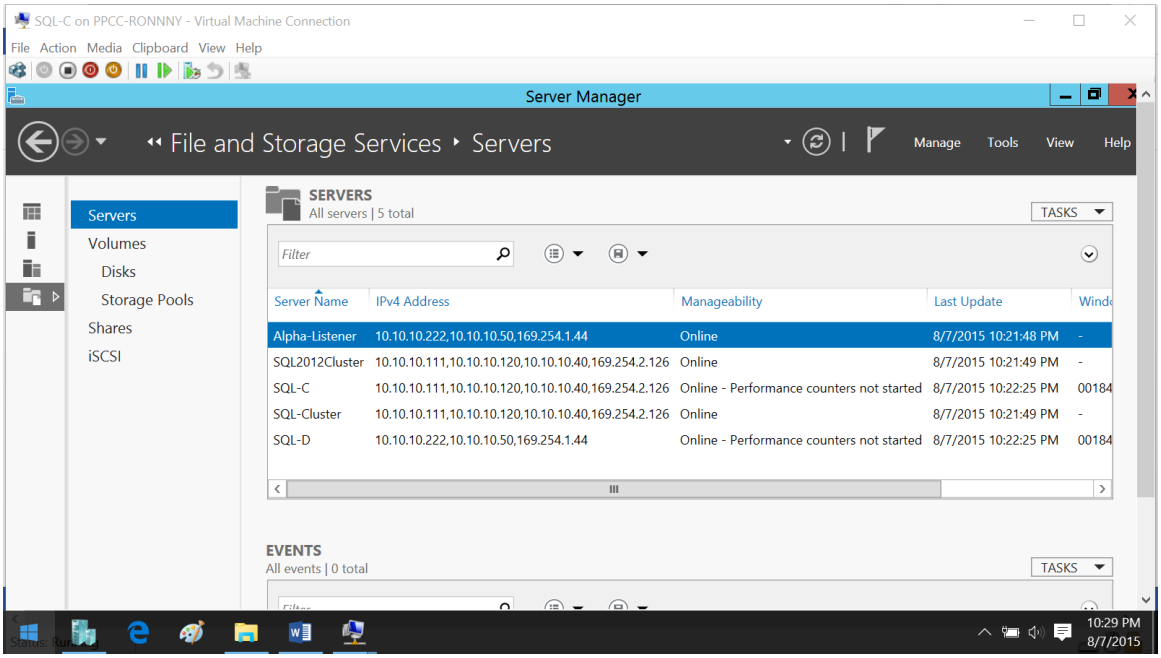
```
EXEC sys.sp_configure 'min server memory', 1024;
GO
EXEC sys.sp_configure 'max server memory', 4096;
GO
sp_configure 'fill factor', 90;
GO
RECONFIGURE;
GO
USE [master]
GO
ALTER DATABASE [model] SET RECOVERY FULL WITH NO_WAIT
GO
sp_configure 'Database Mail XPs', 1;
GO
RECONFIGURE;
GO
EXECUTE msdb.dbo.sysmail_add_account_sp
@account_name = 'Litware2012 Administrator',
@email_address = 'litware2012@contoso.com',
@mailserver_name = 'smtp.contoso.com' ;
EXECUTE msdb.dbo.sysmail_add_profile_sp
@profile_name = 'Litware2012 Mail Profile',
@description = 'Profile used for administrative mail.' ;
EXECUTE msdb.dbo.sysmail_add_profileaccount_sp
@profile_name = 'LitWare2012 Mail Profile',
@account_name = 'Litware2012 Administrator',
@sequence_number = 1;
EXECUTE msdb.dbo.sysmail_add_principalprofile_sp
@profile_name = 'LitWare2012 Mail Profile',
@principal_id = 0,
@is_default = 1;
EXECUTE msdb.dbo.sp_set_sqlagent_properties @email_save_in_sent_folder=1,
@databasemail_profile='LitWare2012 Mail Profile',
@use_databasemail=1;
```

- configure SQL Server engine: memory, filffactor, sp_configure, default options

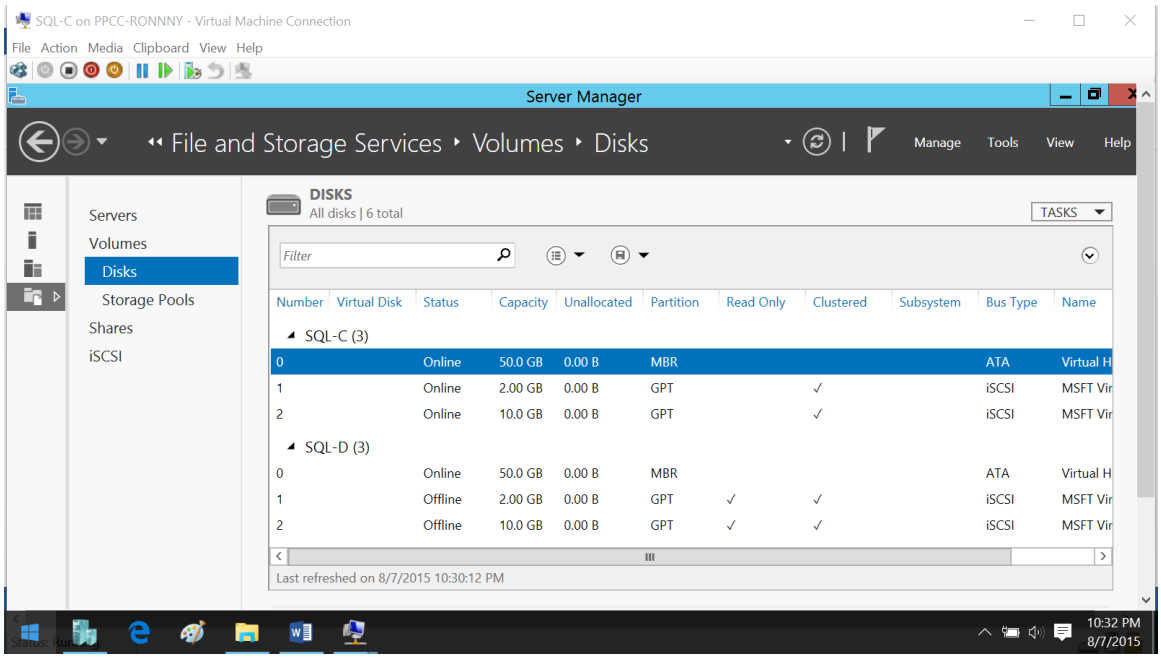
```
sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
sp_configure 'fill factor', 90;
GO
RECONFIGURE;
GO
```

- implement a SQL Server clustered instance
  - Install a cluster;

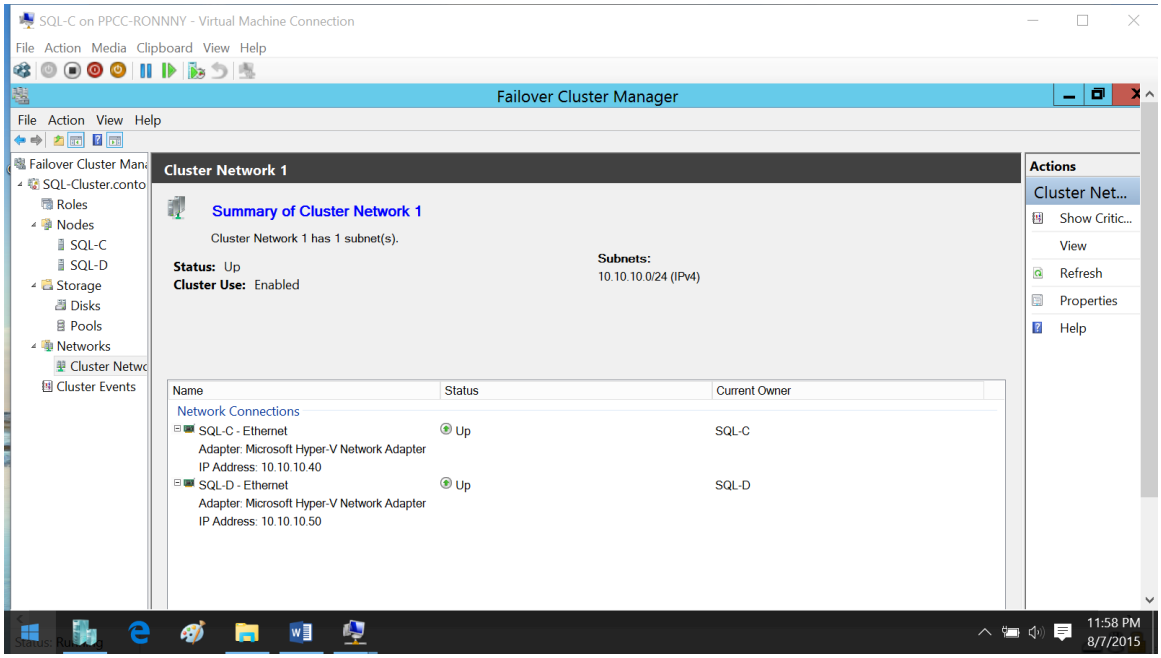
Windows server failover cluster, storage area network,



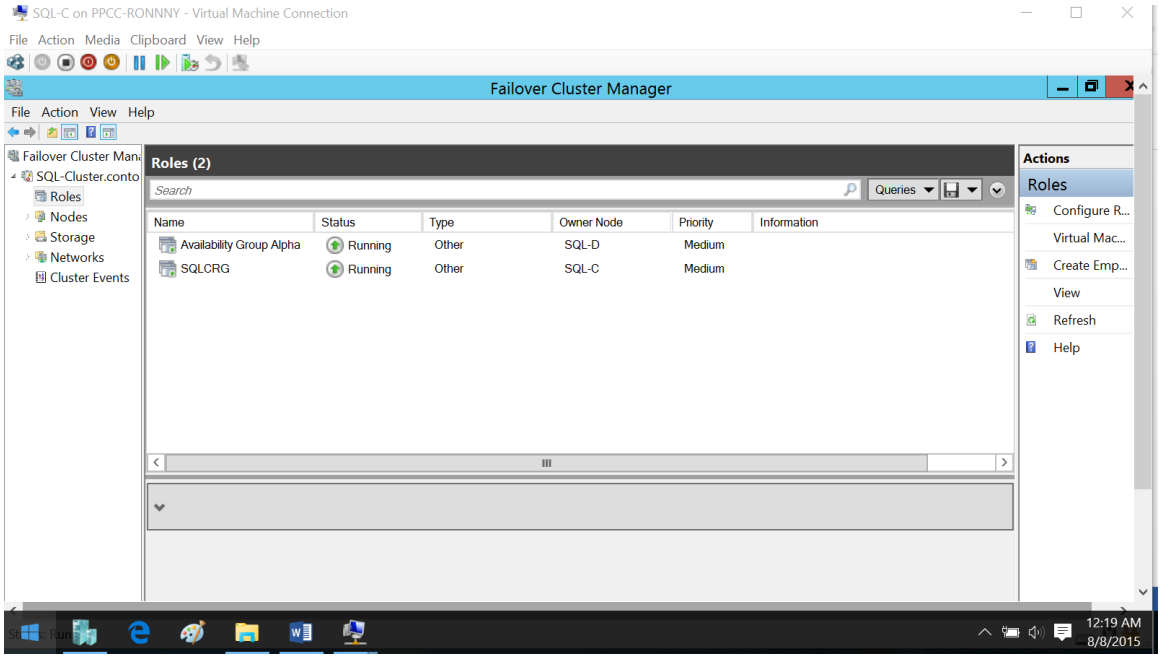
S


SQL-C on PPCC-RONNNY - Virtual Machine Connection
File Action Media Clipboard View Help
Server Manager
File and Storage Services ▸ Volumes ▸ Disks
Manage Tools View Help
Servers
Volumes
Disks
Storage Pools
Shares
iSCSI
DISKS
All disks | 6 total
TASKS
Filter
Number Virtual Disk Status Capacity Unallocated Partition Read Only Clustered Subsystem Bus Type Name
SQL-C (3)
0 Online 50.0 GB 0.00 B MBR ATA Virtual H
1 Online 2.00 GB 0.00 B GPT ✓ iSCSI MSFT Vir
2 Online 10.0 GB 0.00 B GPT ✓ iSCSI MSFT Vir
SQL-D (3)
0 Online 50.0 GB 0.00 B MBR ATA Virtual H
1 Offline 2.00 GB 0.00 B GPT ✓ ✓ iSCSI MSFT Vir
2 Offline 10.0 GB 0.00 B GPT ✓ ✓ iSCSI MSFT Vir
Last refreshed on 8/7/2015 10:30:12 PM
10:32 PM
8/7/2015

S


SQL-C on PPCC-RONNNY - Virtual Machine Connection
File Action Media Clipboard View Help
Failover Cluster Manager
File Action View Help
Failover Cluster Mana
SQL-Cluster.conto
Roles
Nodes
SQL-C
SQL-D
Storage
Disks
Pools
Networks
Cluster Netwc
Cluster Events
Cluster Network 1
Summary of Cluster Network 1
Cluster Network 1 has 1 subnet(s).
Status: Up
Cluster Use: Enabled
Subnets:
10.10.10.0/24 (IPv4)
Name Status Current Owner
Network Connections
SQL-C - Ethernet Up SQL-C
Adapter: Microsoft Hyper-V Network Adapter
IP Address: 10.10.10.40
SQL-D - Ethernet Up SQL-D
Adapter: Microsoft Hyper-V Network Adapter
IP Address: 10.10.10.50
Actions
Cluster Net...
Show Critic...
View
Refresh
Properties
Help
11:58 PM
8/7/2015

S



- manage multiple instances on a cluster; set up subnet clustering; recover from a failed cluster node

- Manage SQL Server instances
  - Install an instance; manage interaction of instances; SQL patch management; install additional instances; manage resource utilization by using Resource Governor;

```
CREATE RESOURCE POOL poolAlpha
WITH (MIN_CPU_PERCENT = 20);
GO
ALTER RESOURCE GOVERNOR RECONFIGURE;

CREATE WORKLOAD GROUP groupBeta
USING poolAlpha;
GO
ALTER RESOURCE GOVERNOR RECONFIGURE;
GO
ALTER RESOURCE GOVERNOR with (CLASSIFIER_FUNCTION = <function name>)
ALTER RESOURCE GOVERNOR RECONFIGURE
GO
```

  - cycle error logs

```
EXEC sp_cycle_errorlog;
GO
USE msdb;
GO
EXEC dbo.sp_cycle_agent_errorlog;
GO
```

Optimize and troubleshoot (14%)

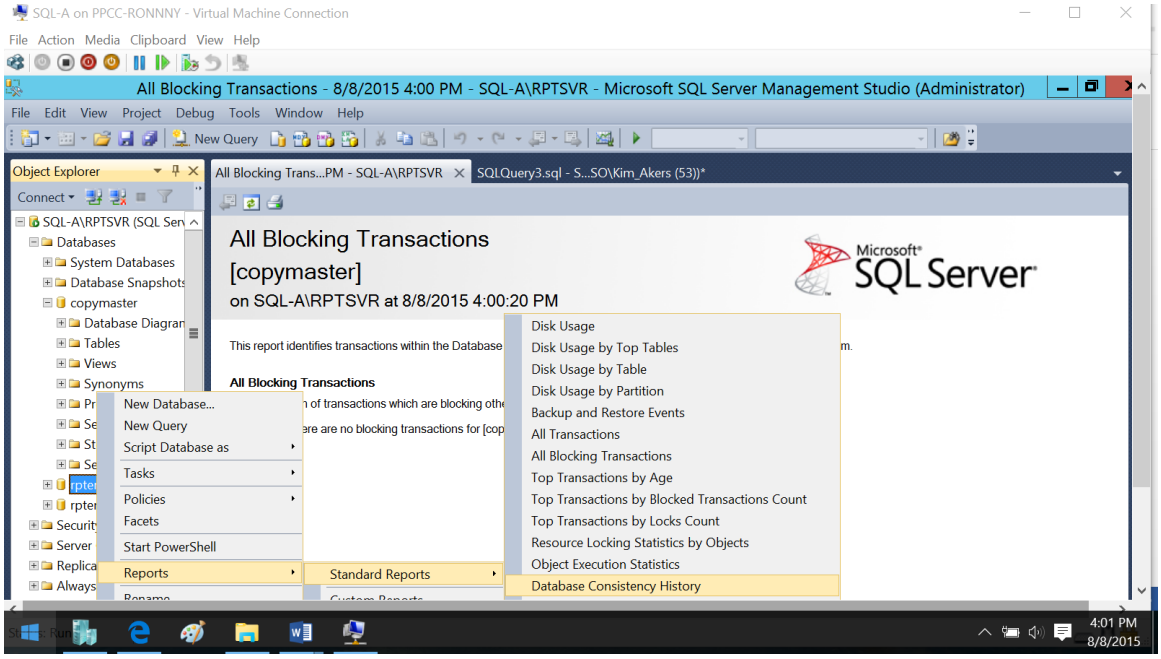
- Identify and resolve concurrency problems
    - Check isolation level, monitoring locks

```
USE [AdventureWorks2012];
GO
DBCC USEROPTIONS;
GO
```

- Examine deadlocking issues using the SQL server logs using trace flags;

  Flag 1204, 1025, 1222

- design reporting database infrastructure (replicated databases); monitor via DMV or other MS product;

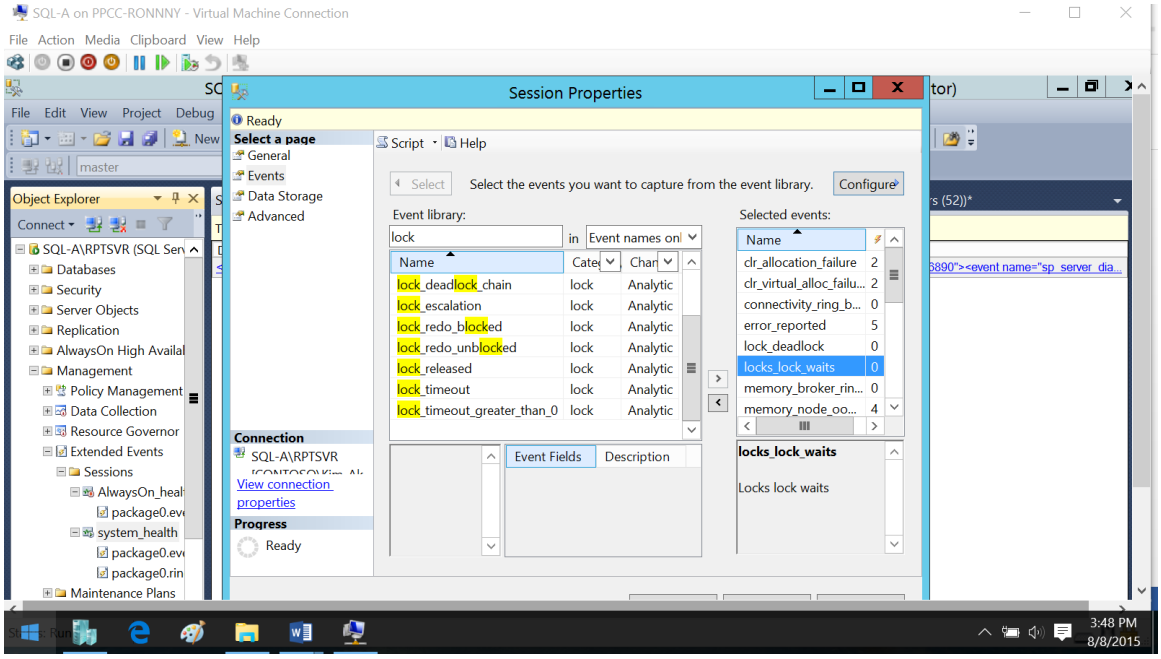



```
USE [AdventureWorks2012]
GO
ALTER DATABASE AdventureWorks2012 SET ALLOW_SNAPSHOT_ISOLATION ON
GO
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
GO
BEGIN TRANSACTION;
GO
--...
GO
COMMIT TRANSACTION;
GO
```

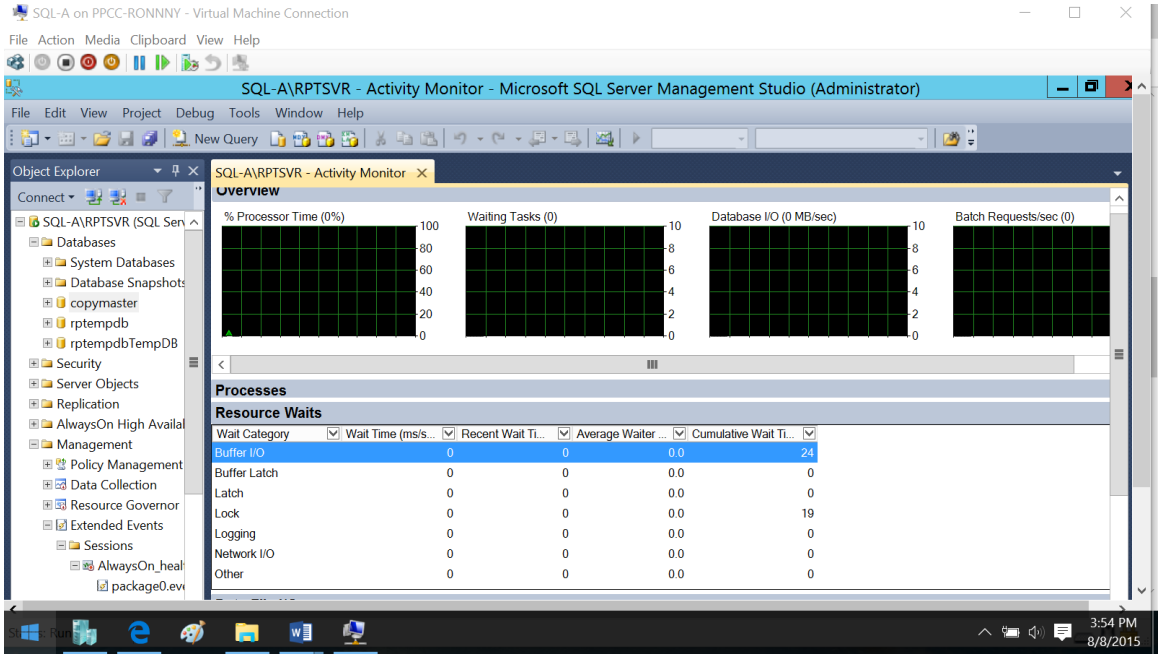
- diagnose blocking, live locking and deadlocking;

```
SELECT wt.session_id,wt.wait_duration_ms,wait_type, blocking_session_id, status
FROM sys.dm_os_waiting_tasks AS wt
```

```
JOIN sys.dm_exec_sessions AS s ON wt.session_id = s.session_id
GO
```



- diagnose waits; performance detection with built in DMVs; know what affects performance;
- active monitor

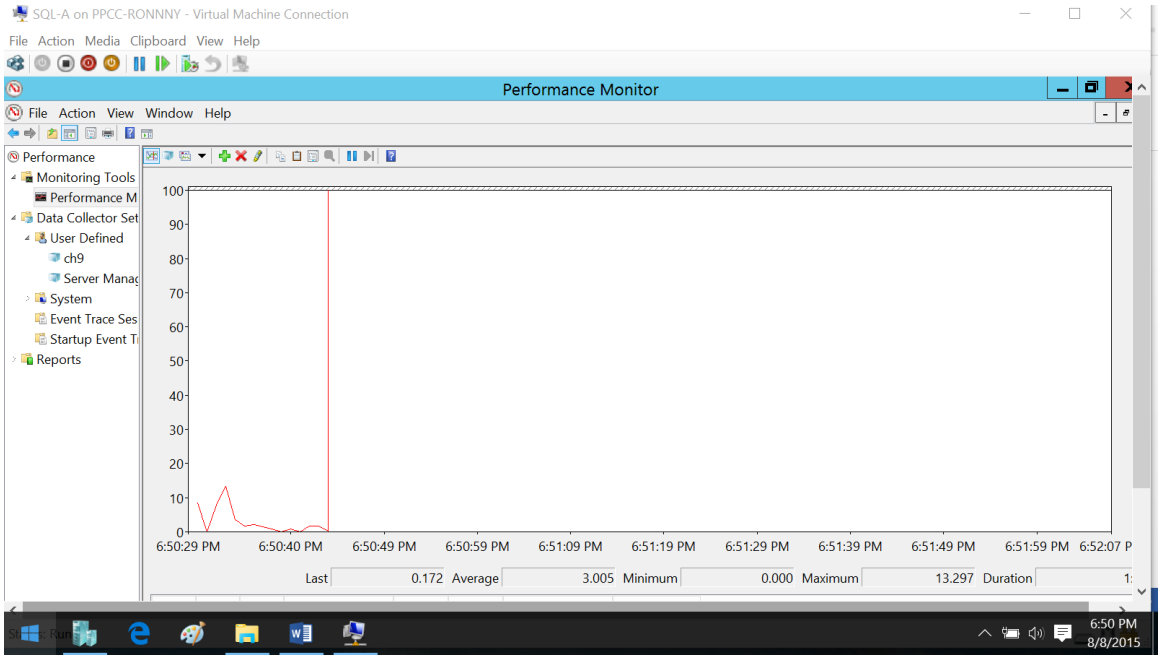


- locate and if necessary kill processes that are blocking or claiming all resources

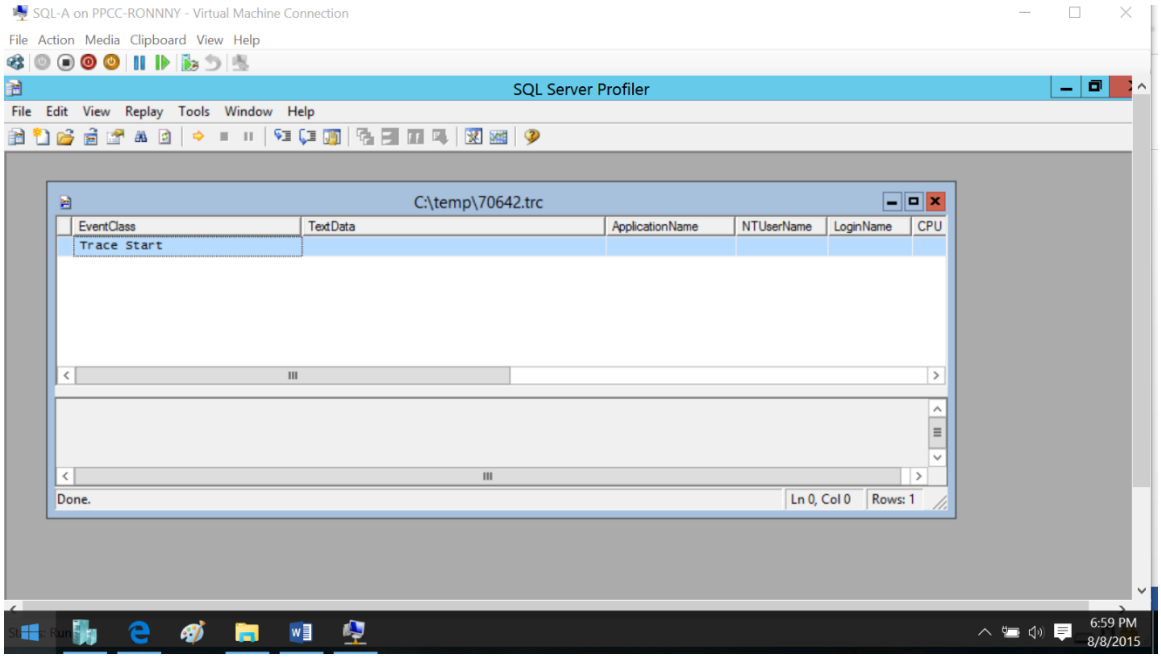
```
KILL 57 WITH STATUSONLY;
```

- Collect and analyze troubleshooting data
    - Monitor using Profiler; collect performance data by using System Monitor;

      physicalDisk, SQL server buffer/memory manager, processor: % privileged time/used time, queue length



- collect trace data by using SQL Server Profiler; tracing output



- identify transactional replication problems; identify and troubleshoot data access problems; missing index, gather performance metrics; identify potential problems before they cause service interruptions; identify performance problems;, use XEvents and DMVs;

```
SELECT *
FROM
(SELECT user_seeks * avg_total_user_cost * (avg_user_impact * 0.01) AS
index_advantage, migs.*
FROM sys.dm_db_missing_index_group_stats migs) AS migs_adv
JOIN sys.dm_db_missing_index_groups AS mig
ON migs_adv.group_handle = mig.index_group_handle
JOIN sys.dm_db_missing_index_details AS mid
ON mig.index_handle = mid.index_handle
ORDER BY migs_adv.index_advantage
```

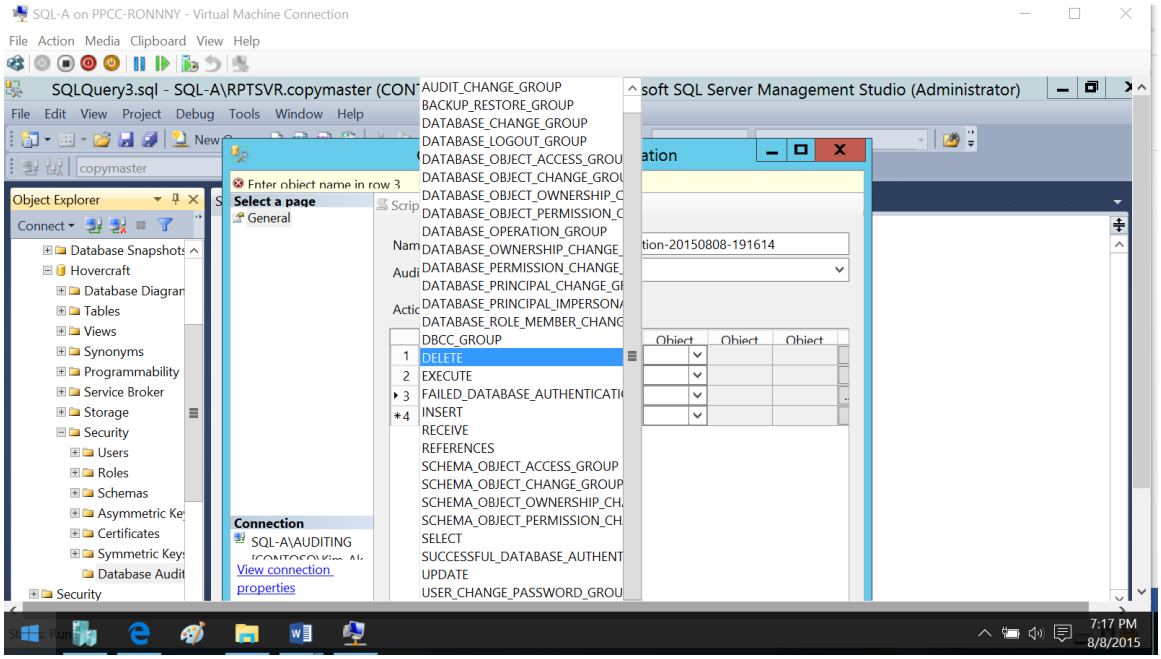
- create alerts on critical server condition; monitor data and server access by creating audit and other controls; identify IO vs. memory vs. CPU bottlenecks; use the Data Collector tool

- Audit SQL Server instances

- Implement a security strategy for auditing and controlling the instance; configure an audit; configure server audits; track who modified an object;

```
CREATE SERVER AUDIT [BETA-AUDIT]
TO APPLICATION_LOG
WITH
( QUEUE_DELAY = 1000
```

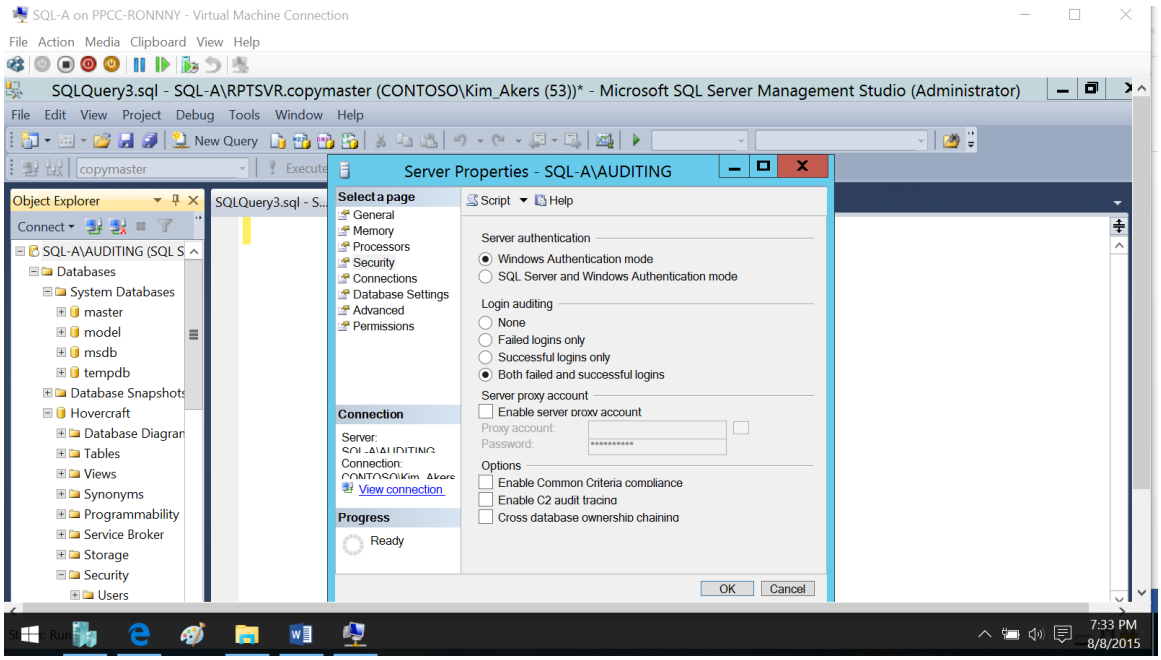
```
,ON_FAILURE = SHUTDOWN
)

CREATE SERVER AUDIT SPECIFICATION [BETA-SPECIFICATION]
FOR SERVER AUDIT [BETA-AUDIT]
ADD (DATABASE_CHANGE_GROUP)

ALTER SERVER AUDIT SPECIFICATION [BETA-SPECIFICATION]
ADD (DATABASE_LOGOUT_GROUP)

CREATE DATABASE AUDIT SPECIFICATION [Audit_Spec_1]
FOR SERVER AUDIT [Srv_Audit_1]
ADD (INSERT ON OBJECT::[dbo].[Paradigm] BY [Exemplar])
```



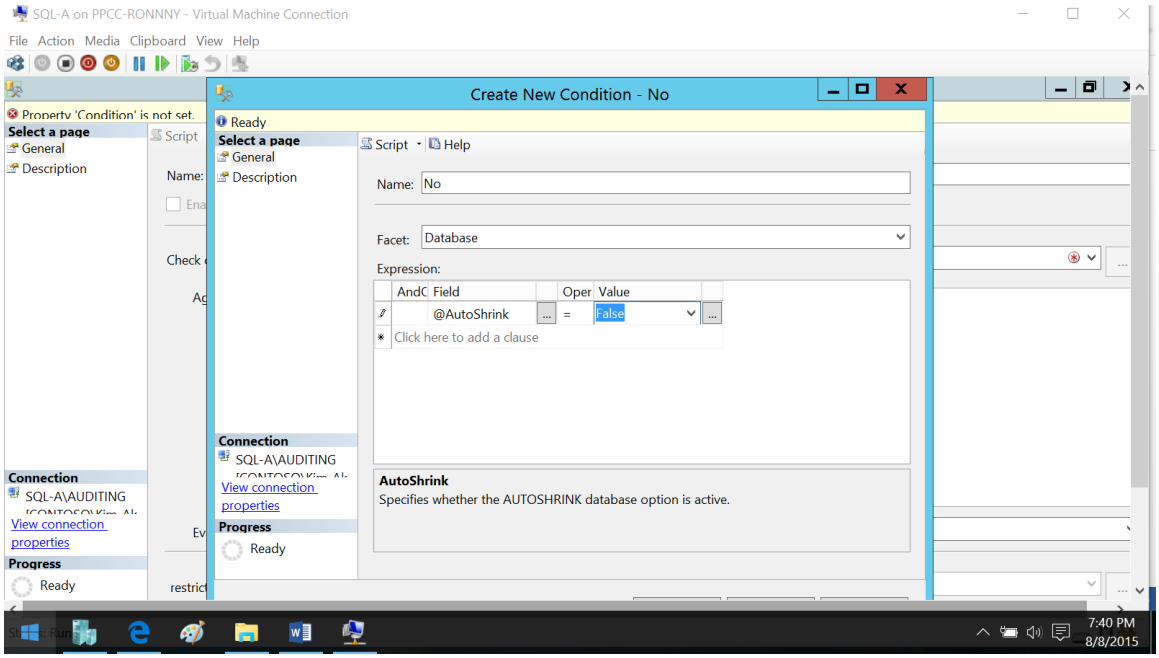
- monitor elevated privileges as well as unsolicited attempts to connect; login auditing

```
CREATE SERVER AUDIT [INSTANCE_AUDIT]
TO APPLICATION_LOG
WITH
( QUEUE_DELAY = 1000
 ,ON_FAILURE = SHUTDOWN
)
CREATE SERVER AUDIT SPECIFICATION [INSTANCE_SPEC]
FOR SERVER AUDIT [INSTANCE_AUDIT]
ADD (DATABASE_CHANGE_GROUP)
ALTER SERVER AUDIT SPECIFICATION [INSTANCE_SPEC]
ADD (DATABASE_LOGOUT_GROUP)
CREATE DATABASE AUDIT SPECIFICATION [DATABASE_SPEC]
FOR SERVER AUDIT [INSTANCE_AUDIT]
ADD (USER_CHANGE_PASSWORD_GROUP)
ALTER DATABASE AUDIT SPECIFICATION [DATABASE_SPEC]
```

```
ADD (SUCCESSFUL_DATABASE_AUTHENTICATION_GROUP)
```



- policy-based management, manage audit across multiple servers.



Manage data (19%)

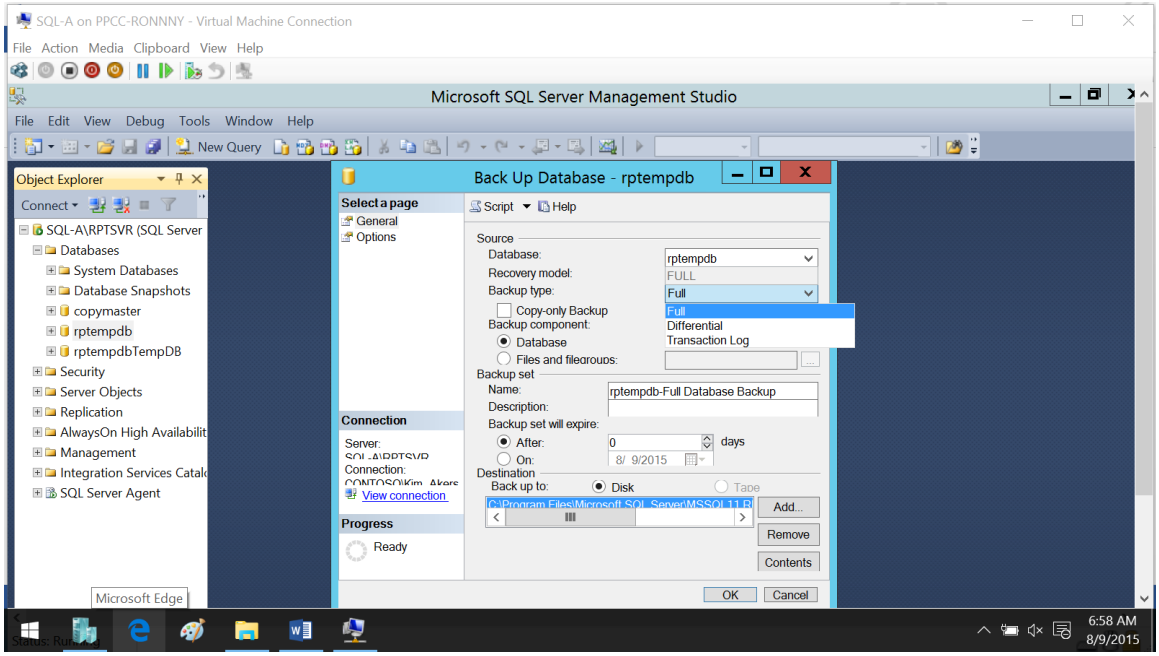
- Configure and maintain a back-up strategy

- Manage different backup models, including point-in-time recovery;

  Backup: full, differential, transaction log; recovery: simple, full, bulk-logged.

```
ALTER DATABASE [AdventureWorks2012] SET RECOVERY FULL;
```

```
EXEC sys.sp_configure [backup compression default], 1
GO
RECONFIGURE WITH OVERRIDE
GO
```

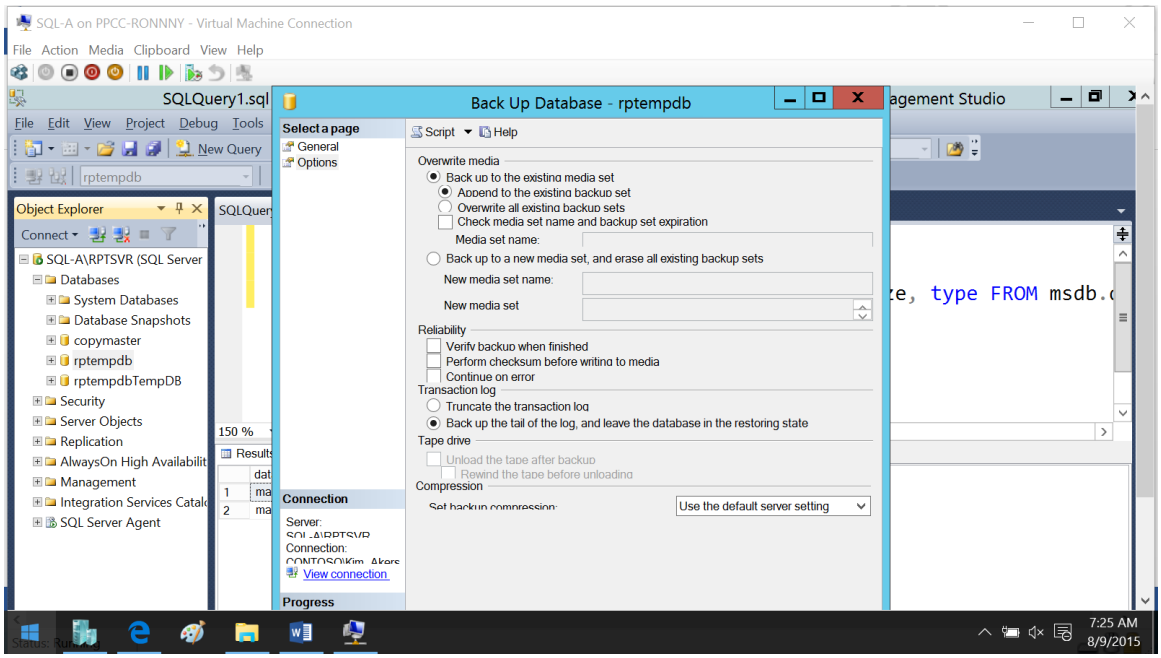


- protect customer data even if backup media is lost; perform backup/restore based on proper strategies including backup redundancy; recover from a corrupted drive;

```
EXEC sp_addumpdevice 'disk', 'Alpha-Backup', 'c:\backup\alpha-backup.bak'

BACKUP Database AdventureWorks2012 to [Alpha-Backup] WITH DIFFERENTIAL;
BACKUP LOG [AdventureWorks2012] to [Alpha-Backup]
```

- manage a multi-TB database; implement and test a database implementation and a backup strategy (multiple files for user database and tempdb, spreading database files, backup/restore); back up a SQL Server environment;

Tail-log

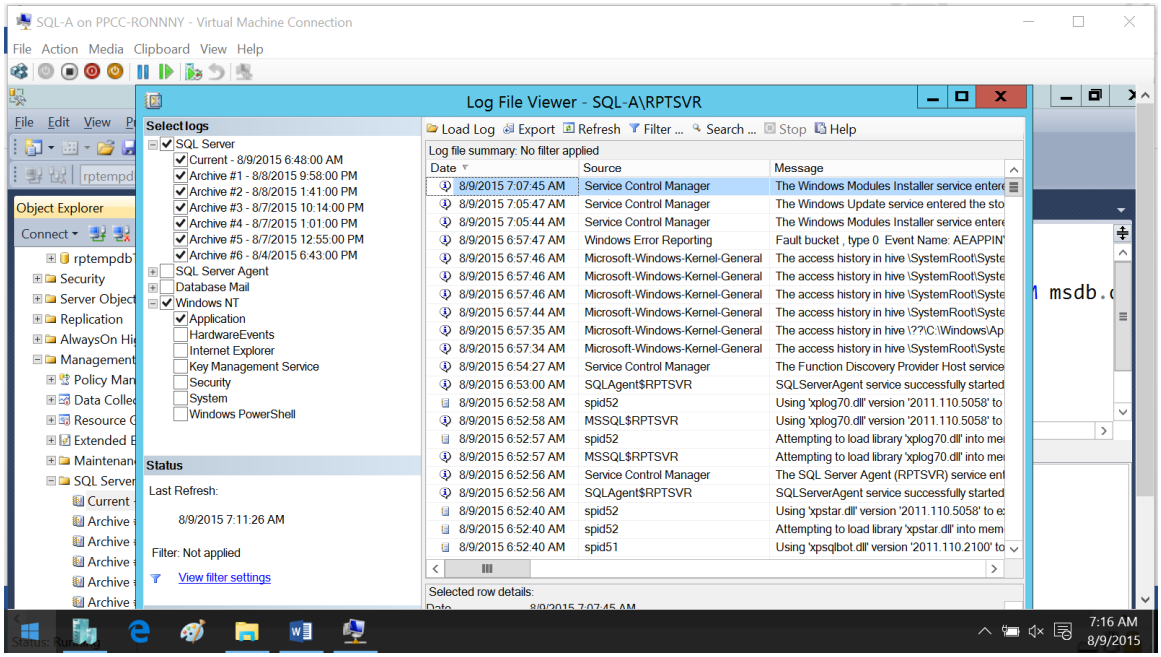


AlwaysOn replicas

```
ALTER AVAILABILITY GROUP [AG_ALPHA] SET ( AUTOMATED_BACKUP_PREFERENCE =
SECONDARY_ONLY
);
```

View backup history

```
SELECT database_name, backup_finish_date, backup_size, type
FROM msdb.dbo.backupset;
```

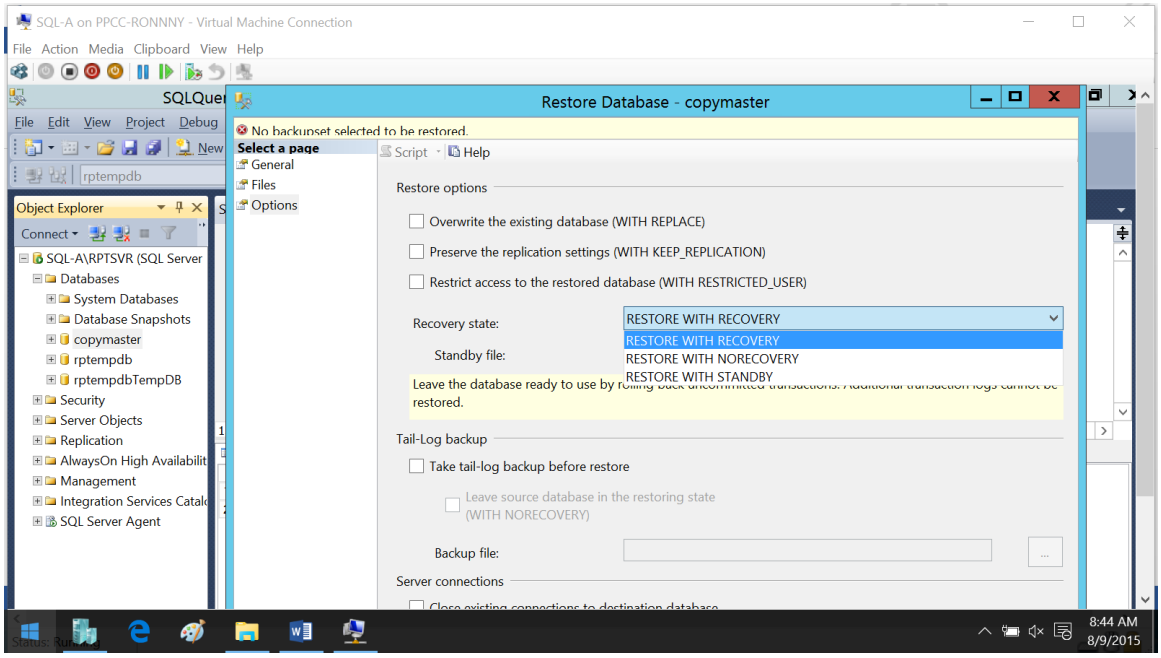


- back up system databases: master, msdb, model, configure distribution

- Restore databases

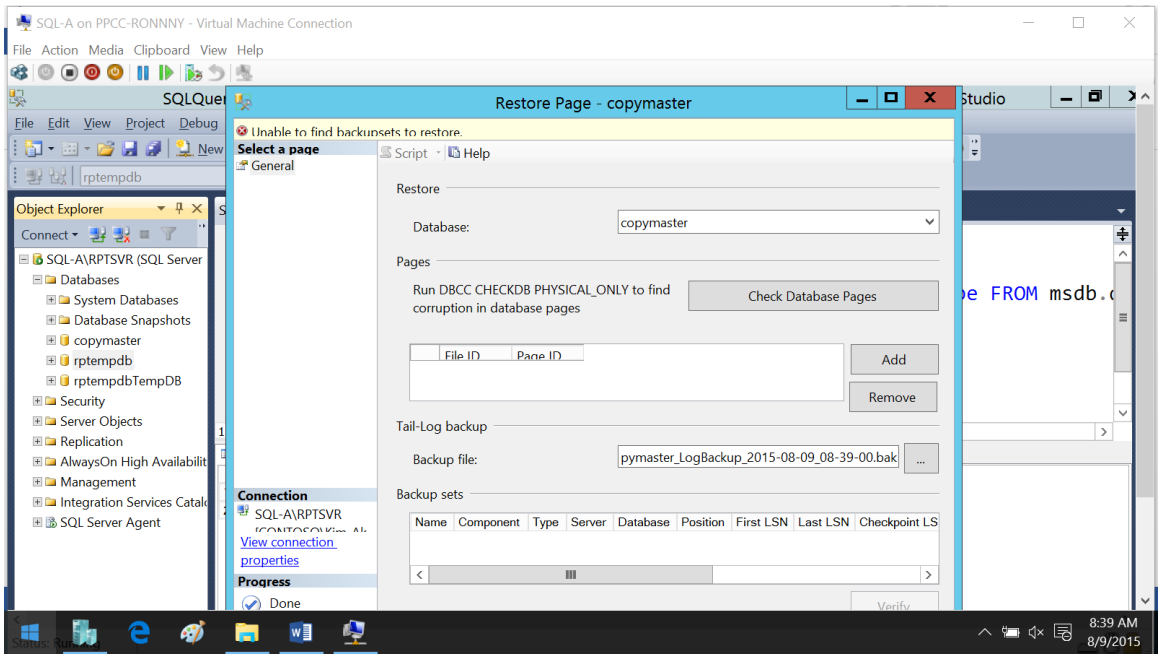
  - Restore a database s ecured with transparent Data Encryption; recover data from a damaged DB (several errors in DBCC checkdb); restore to a point in time;

    Restore with recovery, with norecovery, with standby



  - file group restore; page level restore

```
RESTORE DATABASE HOVERCRAFT PAGE='2:42, 2:81, 2:1023' FROM beta_file_backup WITH
NORECOVERY;
```



Restore system

```
RESTORE DATABASE master from <backup_device> WITH REPLACE

Setup.exe /qs /ACTION=REBUILDDATABASE /INSTANCENAME=MSSQLSERVER
```

- Implement and maintain indexes
    - Inspect physical characteristics of indexes and perform index maintenance; identify fragmented indexes;

```
USE [AdventureWorks2012]
GO
SELECT SCHEMA_NAME(so.schema_id) AS [SchemaName],
OBJECT_NAME(idx.OBJECT_ID) AS [TableName],
idx.name AS [IndexName],
idxstats.index_type_desc AS [Index_Type_Desc],
CAST(idxstats.avg_fragmentation_in_percent
AS decimal(5,2)) AS [Frag_Pct],
idxstats.fragment_count,
idxstats.page_count,
idx.fill_factor
FROM sys.dm_db_index_physical_stats
(DB_ID(), NULL, NULL, NULL, 'DETAILED') idxstats
INNER JOIN sys.indexes idx
ON idx.OBJECT_ID = idxstats.OBJECT_ID
AND idx.index_id = idxstats.index_id
INNER JOIN sys.objects so
ON so.object_id = idx.object_id
WHERE idxstats.avg_fragmentation_in_percent > 20
ORDER BY idxstats.avg_fragmentation_in_percent DESC
```

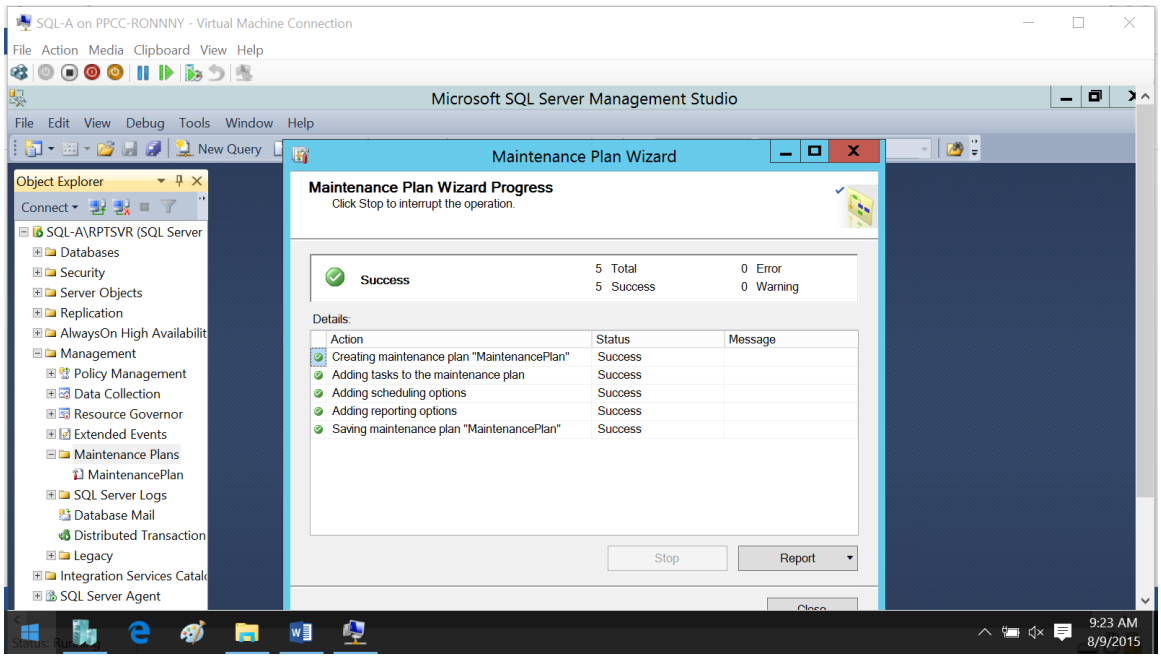
- identify unused indexes; missing indexes

```
SELECT user_seeks
* avg_total_user_cost
* (avg_user_impact * 0.01) AS [Index_Useful]
,igs.last_user_seek
,id.statement AS [Statement]
,id.equality_columns
,id.inequality_columns
,id.included_columns
,igs.unique_compiles
,igs.user_seeks
,igs.avg_total_user_cost
,igs.avg_user_impact
FROM sys.dm_db_missing_index_group_stats AS igs
INNER JOIN sys.dm_db_missing_index_groups AS ig
ON igs.group_handle = ig.index_group_handle
INNER JOIN sys.dm_db_missing_index_details AS id
ON ig.index_handle = id.index_handle
ORDER BY [Index_Useful] DESC;
```

Unused index

```
SELECT
OBJECT_SCHEMA_NAME(i.OBJECT_ID) AS [SchemaName],
OBJECT_NAME(i.OBJECT_ID) AS [ObjectName],
i.name AS [IndexName],
i.type_desc AS [IndexType],
ius.user_updates AS [UserUpdates],
ius.last_user_update AS [LastUserUpdate]
FROM sys.indexes i
INNER JOIN sys.dm_db_index_usage_stats ius
ON ius.OBJECT_ID = i.OBJECT_ID AND ius.index_id = i.index_id
WHERE OBJECTPROPERTY(i.OBJECT_ID, 'IsUserTable') = 1 -- User Indexes
AND NOT(user_seeks > 0 OR user_scans > 0 or user_lookups > 0)
AND i.is_primary_key = 0
AND i.is_unique = 0
ORDER BY ius.user_updates DESC, SchemaName, ObjectName, IndexName
```

- implement indexes; defrag/rebuild indexes; set up a maintenance strategy for indexes and statistics;




```
UPDATE STATISTICS schemaname.tablename WITH FULLSCAN
UPDATE STATISTICS schemaname.tablename WITH FULLSCAN, INDEX
UPDATE STATISTICS schemaname.tablename WITH FULLSCAN, COLUMNS
UPDATE STATISTICS schemaname.tablename WITH SAMPLE 10000 ROWS
UPDATE STATISTICS schemaname.tablename WITH SAMPLE 80 PERCENT
```

- optimize indexes (full, filter index); statistics (full, filter) force or fix queue;

```
SELECT OBJECT_NAME(object_id),name, auto_created
FROM sys.stats
WHERE auto_created = 1
```

- when to rebuild vs. reorg and index; full text indexes;

```
USE [AdventureWorks2012]
GO
ALTER INDEX ALL ON [Production].[Product] REBUILD
WITH (FILLFACTOR = 80);
GO
```

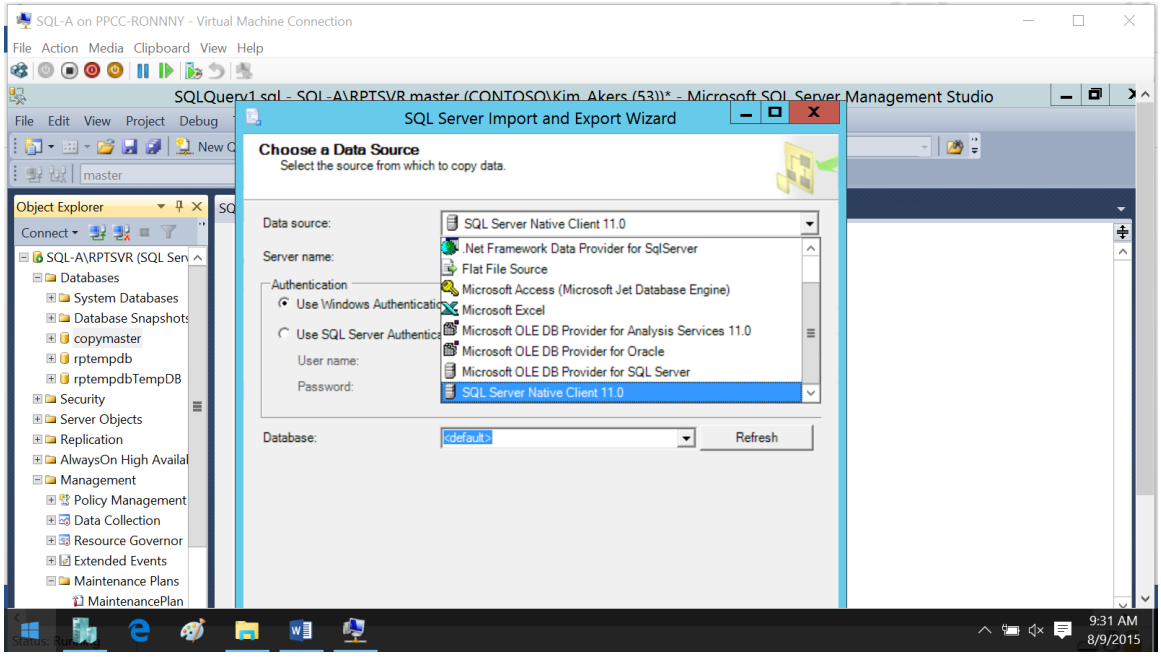
```
DECLARE @db_id SMALLINT, @object_id INT;
SET @db_id = DB_ID(N'AdventureWorks2012');
SET @object_id = OBJECT_ID(N'AdventureWorks2012.Person.Person');
SELECT
index_id, index_type_desc, alloc_unit_type_desc,
index_level, index_depth,
record_count, forwarded_record_count,
page_count, compressed_page_count,
avg_page_space_used_in_percent, avg_fragmentation_in_percent
FROM sys.dm_db_index_physical_stats
(@db_id, @object_id, NULL, NULL, 'Detailed');
```

```
USE [AdventureWorks2012]
GO
SELECT SCHEMA_NAME(so.schema_id) AS [SchemaName],
OBJECT_NAME(idx.OBJECT_ID) AS [TableName],
idx.name AS [IndexName],
idxstats.index_type_desc AS [Index_Type_Desc],
CAST(idxstats.avg_fragmentation_in_percent
AS decimal(5,2)) AS [Frag_Pct],
idxstats.fragment_count,
idxstats.page_count,
idx.fill_factor
FROM sys.dm_db_index_physical_stats
(DB_ID(), NULL, NULL, NULL, 'DETAILED') idxstats
INNER JOIN sys.indexes idx
ON idx.OBJECT_ID = idxstats.OBJECT_ID
AND idx.index_id = idxstats.index_id
INNER JOIN sys.objects so
ON so.object_id = idx.object_id
WHERE idxstats.avg_fragmentation_in_percent > 20
ORDER BY idxstats.avg_fragmentation_in_percent DESC
```

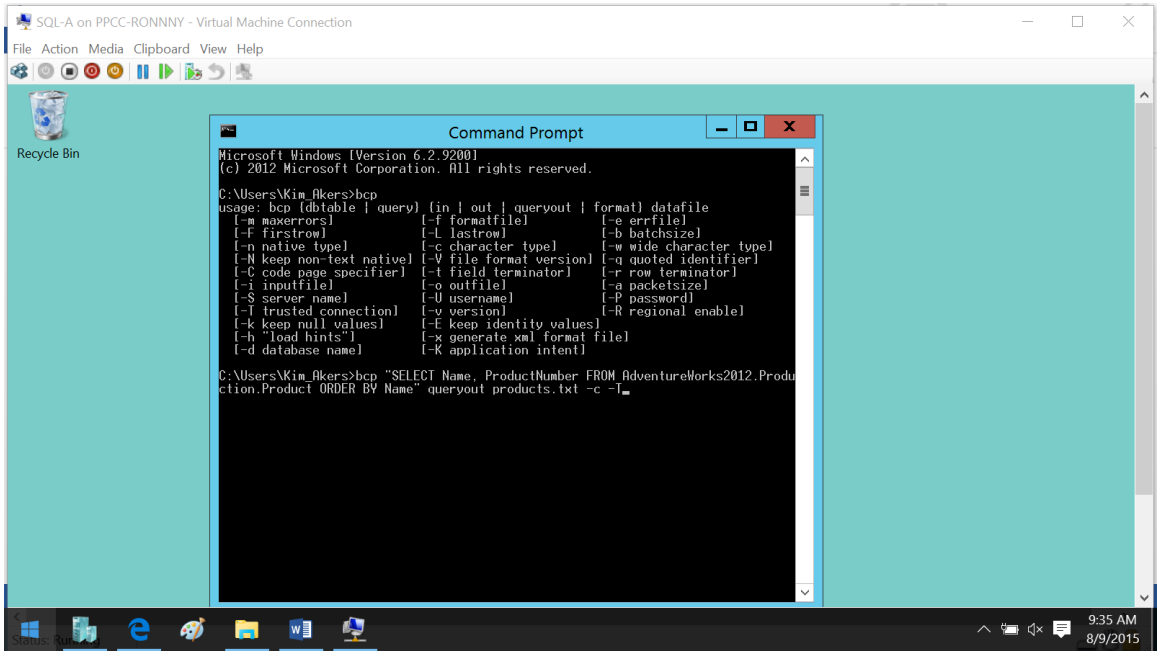
- column store indexes

```
USE [AdventureWorksDW2012]
GO
CREATE NONCLUSTERED COLUMNSTORE INDEX [IX_CS_FactProductInventory]
ON dbo.FactProductInventory
(ProductKey, DateKey, UnitCost, UnitsIn, UnitsOut, UnitsBalance)
GO
```

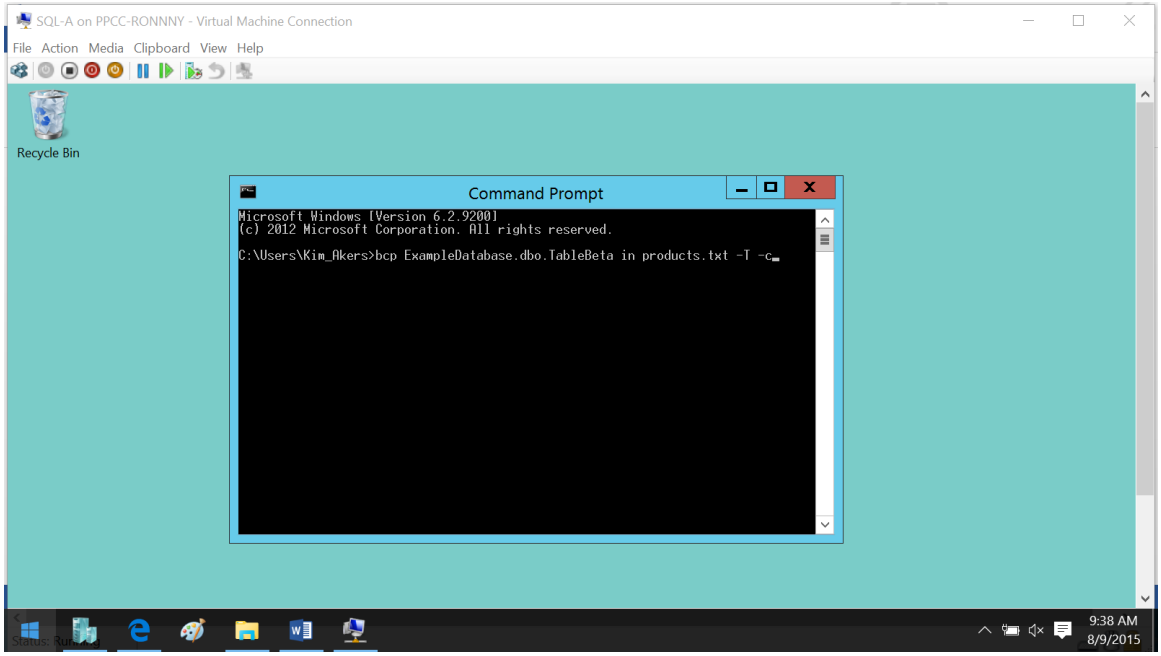
- Import and export data
  - Transfer data;



command line



insert



- bulk copy; bulk insert

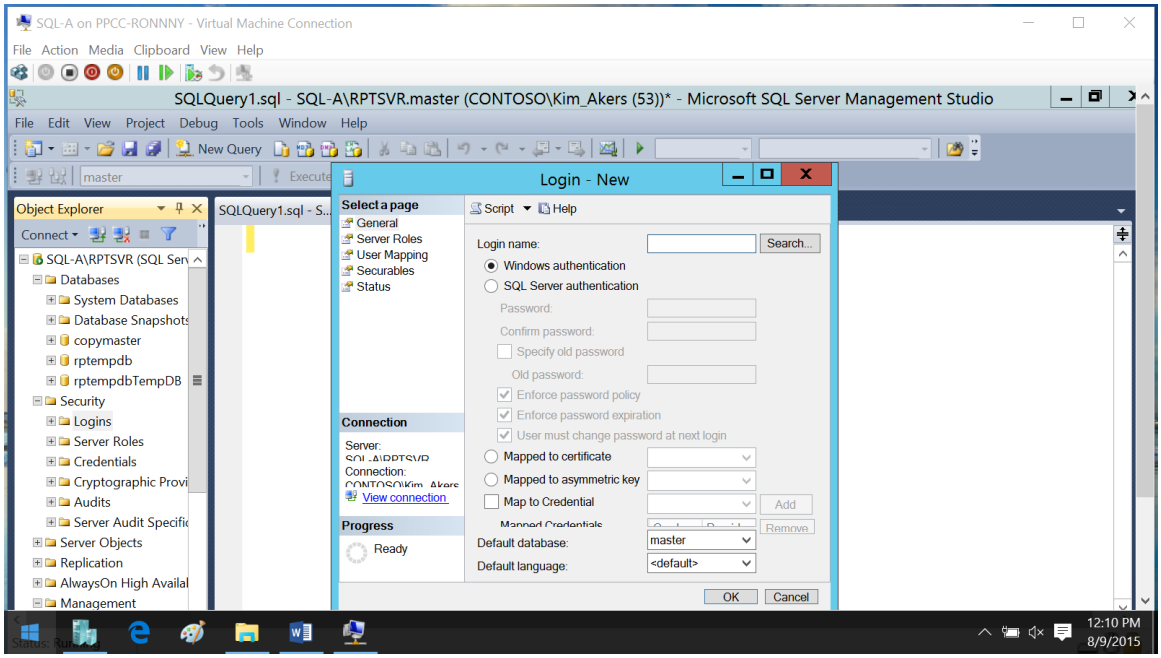
```
BULK INSERT ExampleDatabase.dbo.TableAlpha FROM '\\SQL-A\DATA\products.txt';
GO

SELECT * INTO ExampleDatabase.dbo.HumanResourcesDepartmentCopy FROM
AdventureWorks2012.
HumanResources.Department;
```

Implement security (18%)

- Manage logins and server roles
    - Configure server security; secure the SQL Server using Windows Account / SQL Server accounts,

Windows

```
CREATE LOGIN "SQL-A\Local_One" FROM WINDOWS; --must use double quote
CREATE LOGIN "SQL-A\Group_One" FROM WINDOWS;
```

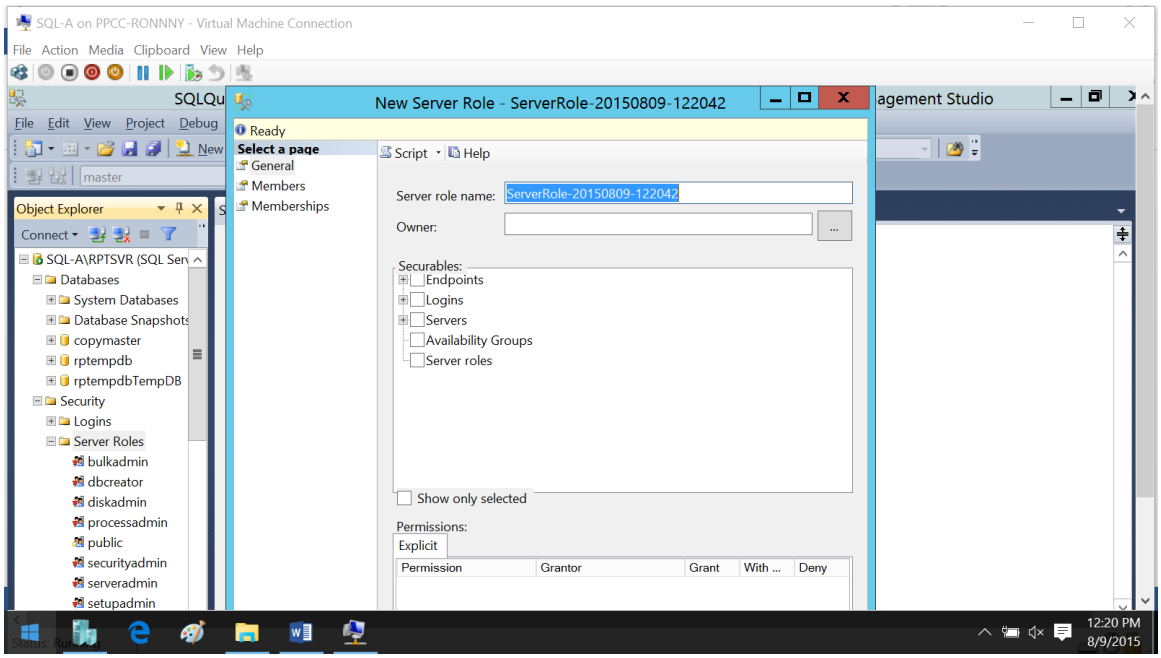


sql account

```
CREATE LOGIN sql_user_a WITH PASSWORD = 'Pa$$w0rd';


ALTER LOGIN sql_user_a DISABLE;
```

- server roles; create log in accounts; manage access to the server, SQL Server instance, and databases; create and maintain user-defined server roles;

```
ALTER SERVER ROLE serveradmin ADD MEMBER "contoso\domain_group_b";

CREATE SERVER ROLE Modify_Databases;
```



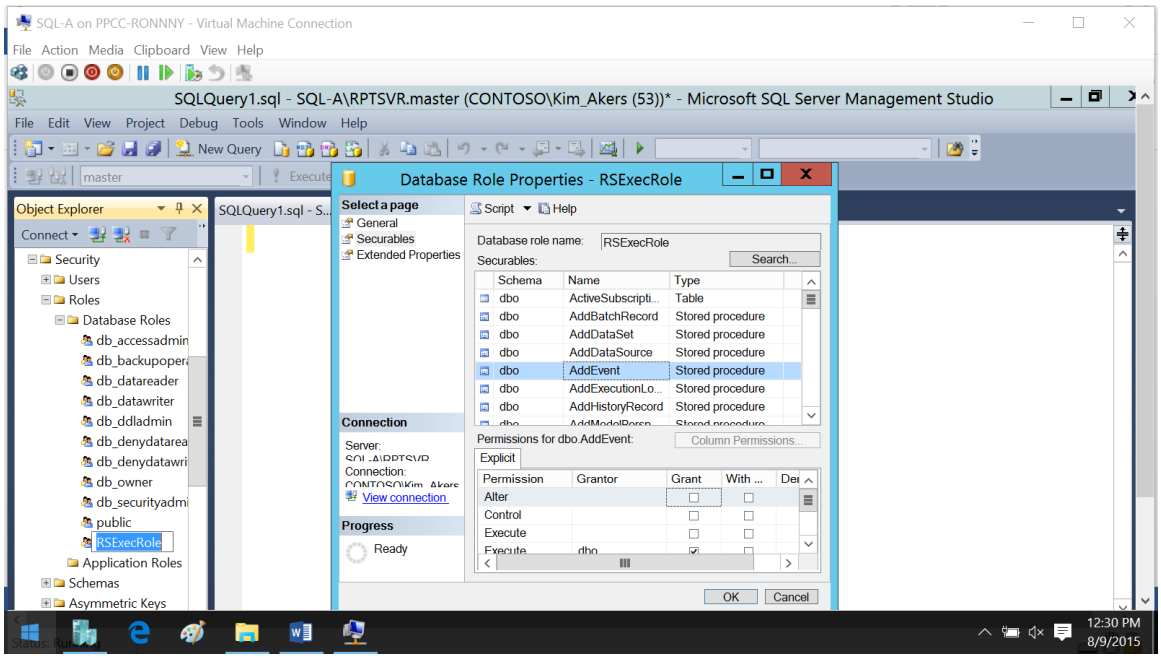

```
GRANT ALTER ANY DATABASE TO Modify_Databases;
```

- manage certificate logins

```
CREATE CERTIFICATE Dan_Bacon
WITH SUBJECT = 'Dan Bacon certificate in master database',
EXPIRY_DATE = '01/01/2018';
GO
CREATE LOGIN Dan_Bacon FROM CERTIFICATE Dan_Bacon;
```

- Manage database permissions
  - Configure database security; database level, permissions; protect objects from being modified

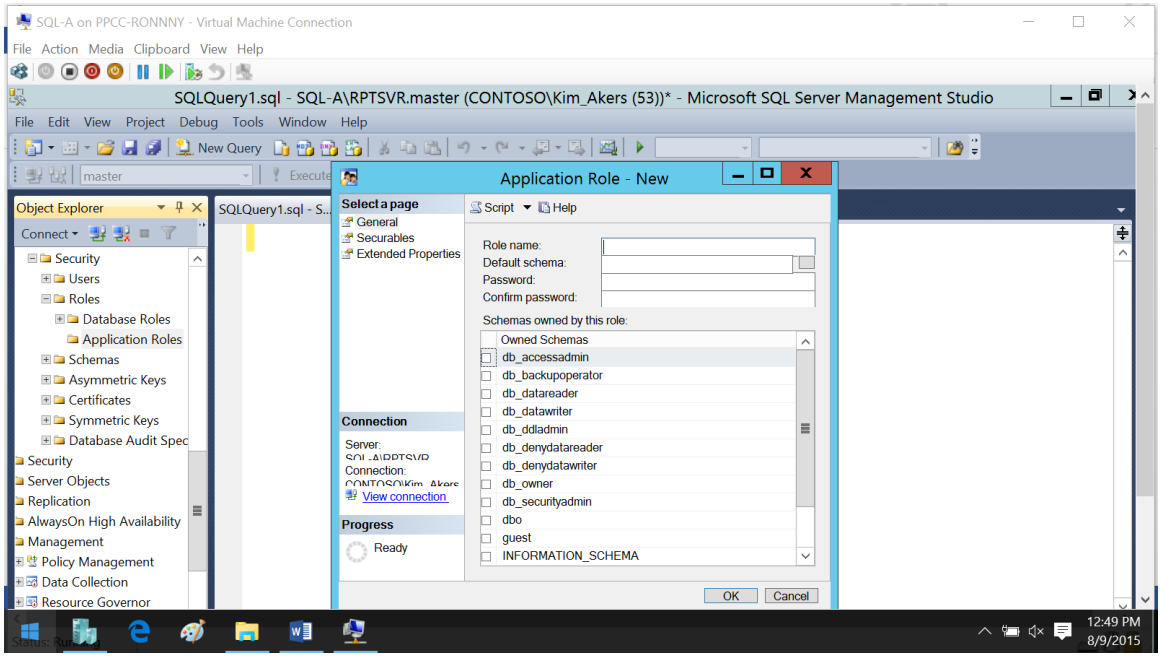
```
USE [AdventureWorks2012]
CREATE ROLE [Alpha-Role]
GO
GRANT INSERT ON [Person].[Address] TO [Alpha-Role]
GO
DENY INSERT ON [Person].[Address] TO [Alpha-Role]
DENY ALTER ON [Person].[Address] TO [Alpha-Role]
DENY UPDATE ON [Person].[Address] TO [Alpha-Role]
GO
REVOKE INSERT ON [Person].[Address] TO [Alpha-Role]
GO

REVOKE INSERT ON SCHEMA::Orbits FROM Moon_table_editors
```




```
REVOKE INSERT ON SCHEMA::Orbits FROM Moon_table_editors
```

- Manage users and database roles
    - Create access to server / database with least privilege; Application role

```
USE [AdventureWorks2012]
GO
CREATE APPLICATION ROLE app_role_alpha WITH PASSWORD = 'Pa$$w0rd';
GO
```



- manage security roles for users and administrators; create database user accounts; contained logins

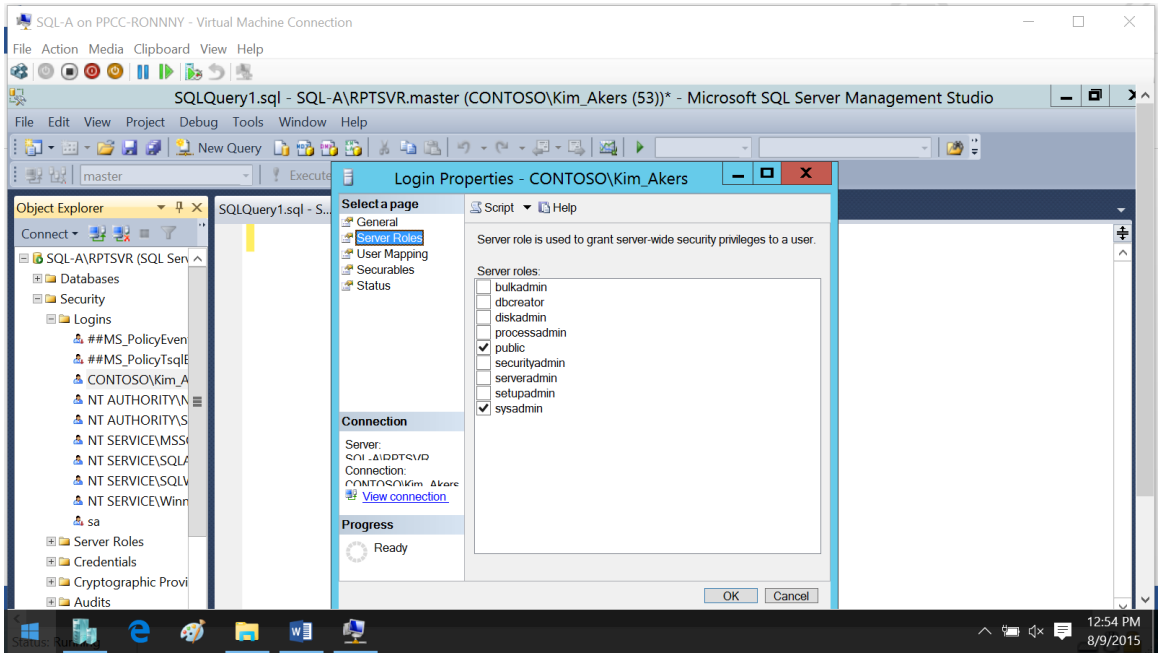
```
USE [master]
GO
CREATE LOGIN "CONTOSO\Account_Two" FROM WINDOWS;
CREATE LOGIN "CONTOSO\Group_Two" FROM WINDOWS;
CREATE LOGIN sql_user_a WITH PASSWORD = 'Pa$$w0rd';
CREATE CERTIFICATE Dan_Bacon
WITH SUBJECT = 'Dan Bacon certificate in master database',
EXPIRY_DATE = '01/01/2018';
CREATE LOGIN Dan_Bacon FROM CERTIFICATE Dan_Bacon;
CREATE ASYMMETRIC KEY sql_user_e WITH ALGORITHM = RSA_2048;
CREATE LOGIN sql_user_e FROM ASYMMETRIC KEY sql_user_e;
ALTER LOGIN sql_user_a DISABLE;
DENY CONNECT SQL TO [contoso\Account_Two];
ALTER SERVER ROLE serveradmin ADD MEMBER "contoso\Group_Two";
CREATE SERVER ROLE Modify_Databases;
GRANT ALTER ANY DATABASE TO Modify_Databases;
CREATE CREDENTIAL RemoteFTP with IDENTITY = 'FTP_Login', SECRET = 'Pa$$w0rd';
USE [AdventureWorks2012]
GO
CREATE USER "contoso\Group_Two" FOR LOGIN "contoso\Group_Two";
CREATE ROLE TableCreator AUTHORIZATION "contoso\administrator";
CREATE ROLE [Alpha-Role] AUTHORIZATION "contoso\administrator";
CREATE ROLE [Beta-Role] AUTHORIZATION "contoso\administrator";
GRANT CREATE TABLE TO TableCreator;
EXEC sp_addrolemember 'TableCreator', "contoso\Account_Two";
EXEC sp_addrolemember 'Alpha-Role', [Dan_Bacon]
GRANT INSERT ON [Person].[Address] TO [Alpha-Role]
DENY INSERT ON [Person].[Address] TO [Beta-Role]


USE [AdventureWorks2012]
GO
CREATE USER "contoso\domain_group_a" FOR LOGIN "contoso\domain_group_a";
GO
EXEC sp_addrolemember 'db_datawriter', "contoso\domain_user_b";
GO

CREATE ROLE TableCreator AUTHORIZATION "contoso\kim_akers";
GO
GRANT CREATE TABLE TO TableCreator;
GO
EXEC sp_addrolemember 'TableCreator', "contoso\domain_user_b";
GO
```

- contained logins

```
sp_configure 'show advanced', 1;
RECONFIGURE WITH OVERRIDE;
GO
sp_configure 'contained database authentication', 1;
RECONFIGURE WITH OVERRIDE;
GO
CREATE DATABASE partially_contained_db
CONTAINMENT = PARTIAL
GO
```

```
CREATE USER contained_user WITH PASSWORD = 'Pa$$w0rd';
```

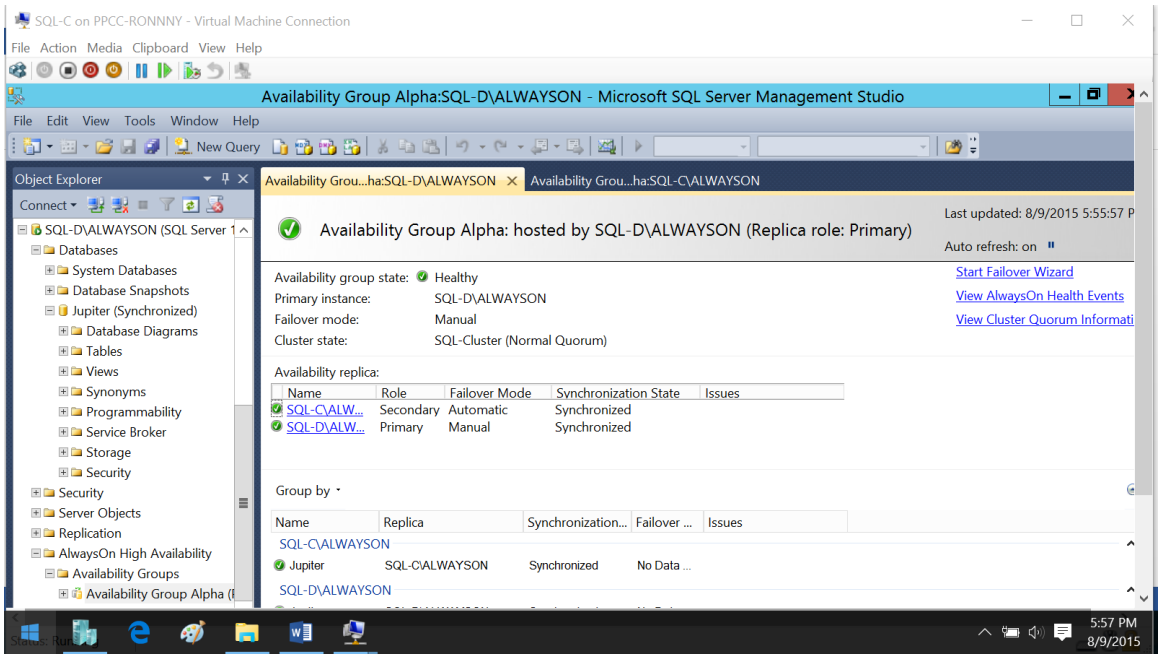
- Troubleshoot security
    - Manage certificates and keys; endpoints



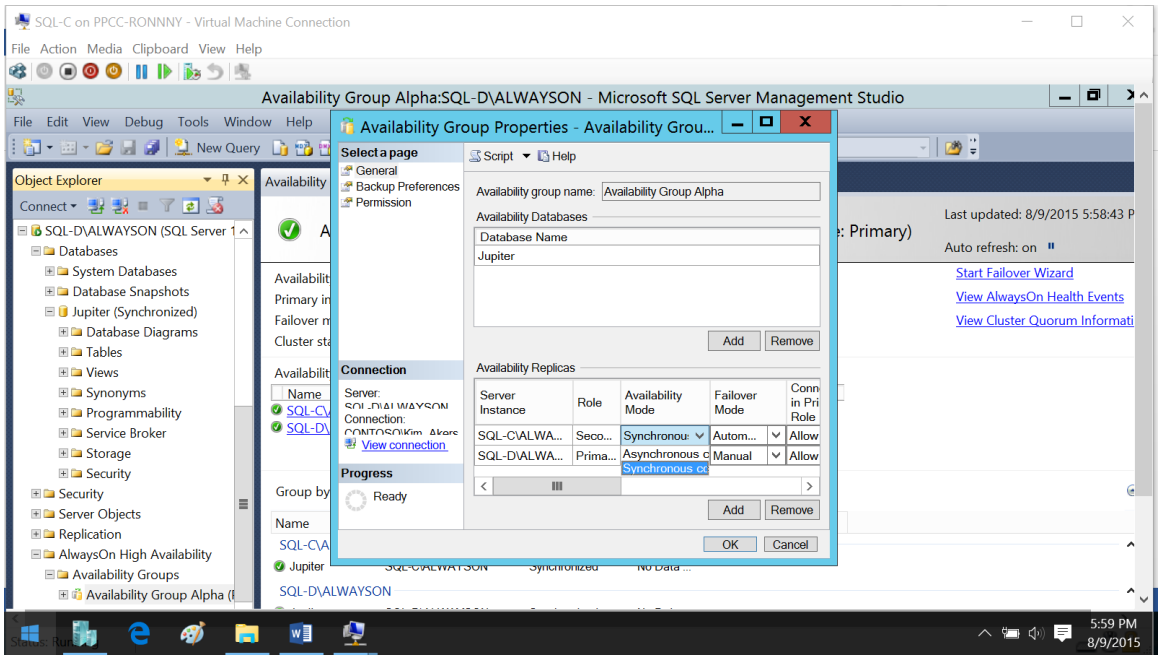

```
sys.certificates
sys.endpoints

sys.database_permissions
sys.database_principals
```

Implement high availability (12%)

- Implement AlwaysOn
    - Implement a mirroring solution using AlwaysOn; failover



X




```
ALTER AVAILABILITY GROUP AG-ALPHA MODIFY REPLICA ON 'SQL-C\AlwaysOn' WITH
( AVAILABILITY_MODE = SYNCHRONOUS_COMMIT);

ALTER AVAILABILITY GROUP AG-Alpha FAILOVER;

ALTER AVAILABILITY GROUP AG-Alpha FORCE_FAILOVER_ALLOW_DATA_LOSS;

Configure Endpoint->

$endpoint = New-SqlHadrEndpoint AlwaysOnEndpoint -Port 7028 -Path
```

```
SQLSERVER:\SQL\SQL-A\
ALTERNATE

Set-SqlHadrEndpoint -InputObject $endpoint -State "Started"

Enable-SqlAlwaysOn -Path SQLSERVER:\SQL\SQL-B\ALTERNATE

CREATE AVAILABILITY GROUP AG-BETA
FOR
DATABASE Saturn
REPLICA ON
'SQL1\newinstance' WITH
(
ENDPOINT_URL = 'TCP://sql1.contoso.com:7030',
AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
FAILOVER_MODE = MANUAL
),
'SQL2\newinstance' WITH
(
ENDPOINT_URL = 'TCP://sql2.contoso.com:7030',
AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
FAILOVER_MODE = MANUAL
);
GO
Create listener->

ALTER AVAILABILITY GROUP [AG-Alpha]
ADD LISTENER 'Beta-Listener' (with IP (('10.0.0.222','255.0.0.0')), PORT=7028);
GO

New-SqlAvailabilityGroupListener -Name Gamma-Listener -StaticIP
'10.0.0.224/255.0.0.0'
-Port 7030 -Path SQLSERVER:\SQL\SQL-C\ALWAYSON\AvailabilityGroups\AG-Gamma
```

- Implement database mirroring
    - Set up mirroring; monitor the performance of database mirroring

Full recovery->

```
USE MASTER;
ALTER DATABASE [AdventureMirror] SET RECOVERY FULL;
```

Restore to the replica server by using no recovery->

```
BACKUP DATABASE [AdventureMirror] TO DISK = N'C:\backup\adventuremirror.bak' WITH
FORMAT;

RESTORE DATABASE AdventureMirror
FROM DISK = 'C:\backup\adventuremirror.bak'
WITH NORECOVERY
GO

BACKUP LOG AdventureMirror
TO DISK = 'C:\backup\adventuremirror.trn'
GO
```

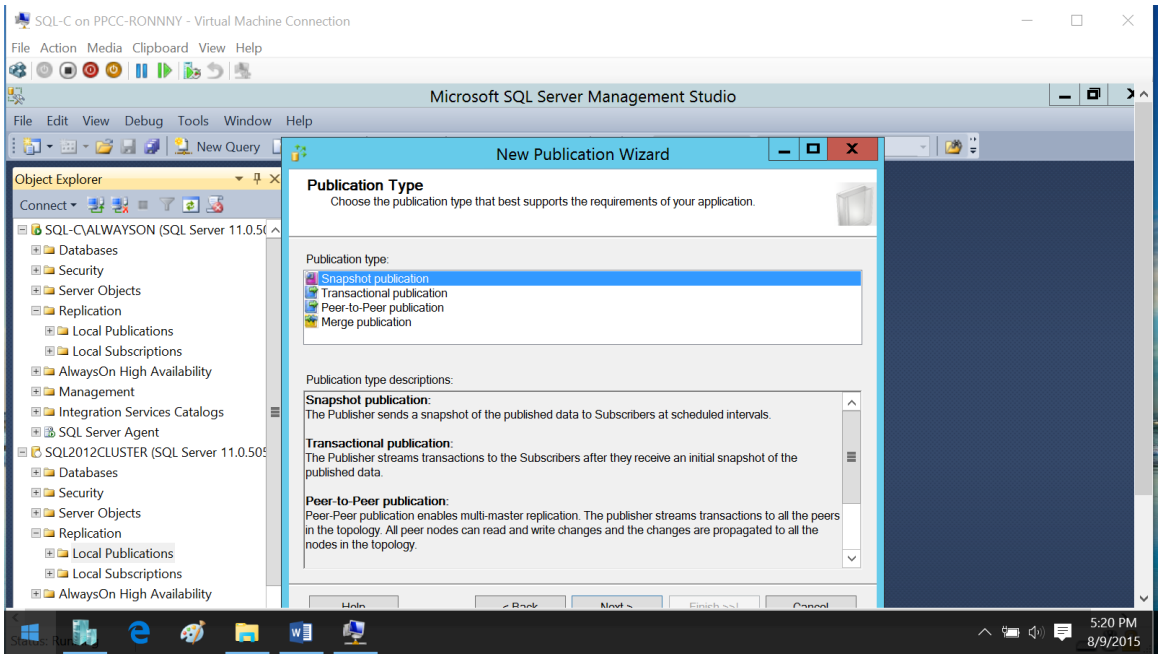
```
RESTORE LOG AdventureMirror
FROM DISK = 'C:\backup\adventuremirror.trn'
WITH FILE=1, NORECOVERY
GO
```

Configure endpoint with certificate authentication ->

```
USE master;
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'Pa$$w0rd';
GO

CREATE CERTIFICATE SQL_A_cert
WITH SUBJECT = 'SQL_A certificate';
GO

CREATE ENDPOINT Endpoint_Mirroring
       STATE = STARTED
       AS TCP (
       LISTENER_PORT=7024
       , LISTENER_IP = ALL
       )
       FOR DATABASE_MIRRORING (
       AUTHENTICATION = CERTIFICATE SQL_A_cert
       , ENCRYPTION = REQUIRED ALGORITHM AES
       , ROLE = ALL
);
GO

BACKUP CERTIFICATE SQL_A_cert TO FILE = 'C:\BACKUP\SQL_A_cert.cer';
GO

USE master;
CREATE LOGIN SQL_B_login WITH PASSWORD = 'Pa$$w0rd';
GO
CREATE USER SQL_B_user FOR LOGIN SQL_B_login;
GO
CREATE CERTIFICATE SQL_B_cert
AUTHORIZATION SQL_B_user
FROM FILE = 'C:\backup\SQL_B_cert.cer'
GO

GRANT CONNECT ON ENDPOINT::Endpoint_Mirroring TO [SQL_B_login];
GO

ALTER DATABASE AdventureMirror
       SET PARTNER = 'TCP://sql-a.contoso.com:7024';
GO

ALTER DATABASE AdventureMirror
       SET PARTNER = 'TCP://sql-b.contoso.com:7024';
GO
```

- Troubleshoot replication problems; identify appropriate replication strategy

Snapshot replication, transactional replication, peer-to-peer replication, merge replication


SQL-C on PPCC-RONNNY - Virtual Machine Connection
File Action Media Clipboard View Help
Microsoft SQL Server Management Studio
File Edit View Debug Tools Window Help
New Query
Object Explorer
Connect
SQL-C\ALWAYSON (SQL Server 11.0.5(
Databases
Security
Server Objects
Replication
Local Publications
Local Subscriptions
AlwaysOn High Availability
Management
Integration Services Catalogs
SQL Server Agent
SQL2012CLUSTER (SQL Server 11.0.505
Databases
Security
Server Objects
Replication
Local Publications
Local Subscriptions
AlwaysOn High Availability
New Publication Wizard
Publication Type
Choose the publication type that best supports the requirements of your application.
Publication type:
Snapshot publication
Transactional publication
Peer-to-Peer publication
Merge publication
Publication type descriptions:
Snapshot publication:
The Publisher sends a snapshot of the published data to Subscribers at scheduled intervals.
Transactional publication:
The Publisher streams transactions to the Subscribers after they receive an initial snapshot of the published data.
Peer-to-Peer publication:
Peer-Peer publication enables multi-master replication. The publisher streams transactions to all the peers in the topology. All peer nodes can read and write changes and the changes are propagated to all the nodes in the topology.
Help
< Back
Next >
Finish >>|
Cancel
5:20 PM
8/9/2015