imaginet

Using SQL Database Projects

With Visual Studio

SQL Server Data Tools

https://github.com/xhead/SqlSaturdayATL-2024

Mike Diehl, Miked@imaginet.Com
Practice Lead
Data Engineering and Business Intelligence

@xhead

/mikediehlsqlbi

THANK YOU SPONSORS

improving
It's what we do.™

Microsoft

ProcureSQL
DATA ARCHITECT AS A SERVICE

info river
INTELLIGENT ANALYTICS

Truviz

WhereScape®
Data Automation

TIMEXTENDER

COZYROC

KEY 2 Consulting

POWER BI SENTINEL
Governance, Disaster Recovery and Auditing for Power BI & Fabric

concretely.AI

matillion

DesignMind

SQL GENE

Please visit our sponsors

imaginet

Professional services company providing custom software solutions to organizations across North America.

Founded in 1997
Located in Dallas, TX and Winnipeg, MB with 100 full-time employees across US & Canada (developers, QA, UX, Cloud Architecture, Microsoft 365, project managers).

20-Time Microsoft Gold Partner
Core specialties: Data and Analytics, Power Platform, SharePoint, Teams
10+ years experience with Microsoft 365.

25+
Years in Business

20x
Microsoft Gold Partner

1,400+
Satisfied Customers

3,200+
Successful Engagements

imaginet

Can you answer these questions?

- What did this database look like six months ago?
- Can I change the name of a column or table and not break something?
- Are there any broken references in this database?
- Can I deploy this database to a new database?
- Can I reliably upgrade the objects in an existing database?

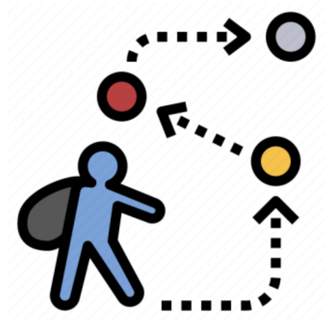
Database Projects

- Visual Studio (Data storage and processing)
- SQL files:
  - CREATE statements
    - Most files are a single CREATE; table scripts can include CREATE INDEX statements and other; separated by GO
  - Other scripts
    - One Pre-Deploy script
    - One Post-Deploy script
    - Call other scripts from these two scripts

Desired State Configuration (DSC)

The source code defines the desired end-state.

At deployment, the actions to move from current state to desired state are determined and executed



Migration-based configuration

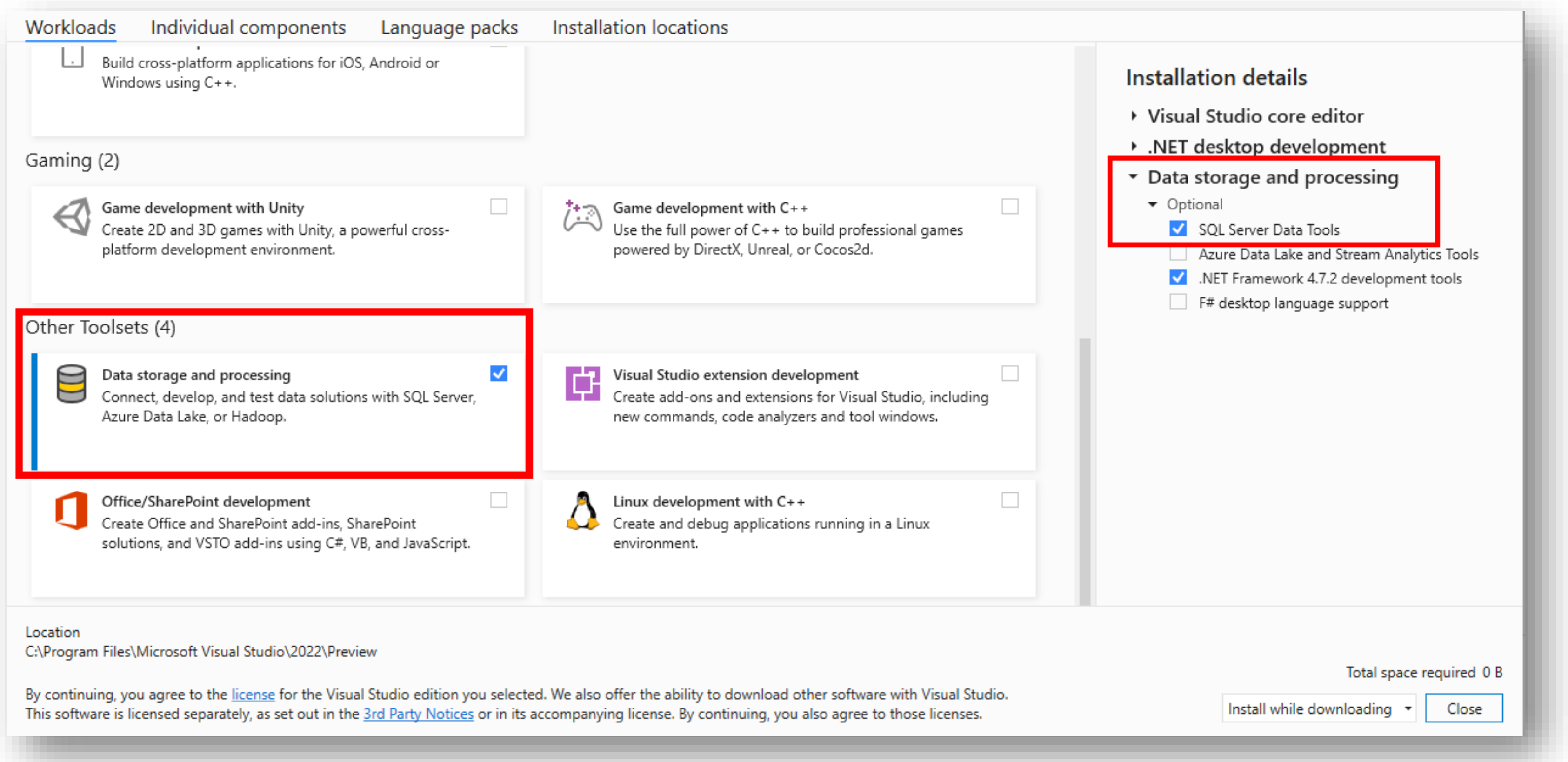
Each migration specifies actions to move from state to state

From A to B

From B to C

Software

- Visual Studio (Community Edition)
  - Other toolsets
  - Data storage and processing



Workloads | Individual components | Language packs | Installation locations

Build cross-platform applications for iOS, Android or Windows using C++.

Gaming (2)

☐ Game development with Unity
Create 2D and 3D games with Unity, a powerful cross-platform development environment.

☐ Game development with C++
Use the full power of C++ to build professional games powered by DirectX, Unreal, or Cocos2d.

Other Toolsets (4)

✓ Data storage and processing
Connect, develop, and test data solutions with SQL Server, Azure Data Lake, or Hadoop.

☐ Visual Studio extension development
Create add-ons and extensions for Visual Studio, including new commands, code analyzers and tool windows.

☐ Office/SharePoint development
Create Office and SharePoint add-ins, SharePoint solutions, and VSTO add-ins using C#, VB, and JavaScript.

☐ Linux development with C++
Create and debug applications running in a Linux environment.

Installation details
- Visual Studio core editor
- .NET desktop development
- Data storage and processing
  - Optional
    - ✓ SQL Server Data Tools
    - ☐ Azure Data Lake and Stream Analytics Tools
    - ✓ .NET Framework 4.7.2 development tools
    - ☐ F# desktop language support

Location
C:\Program Files\Microsoft Visual Studio\2022\Preview

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

Total space required 0 B

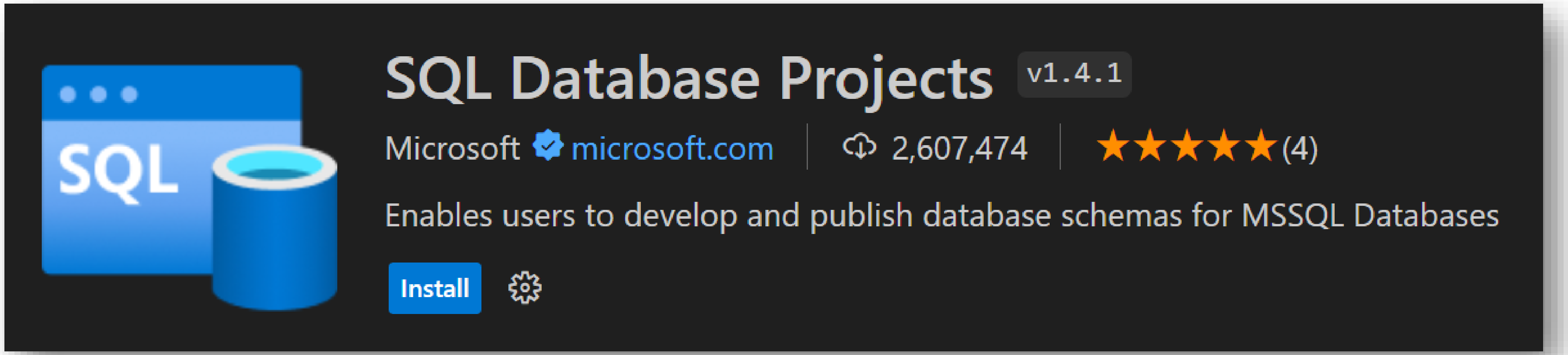
Install while downloading | Close

imaginet

Software

- Visual Studio Code: extension for SQL Database Projects

Visual Studio Code



SQL Database Projects v1.4.1

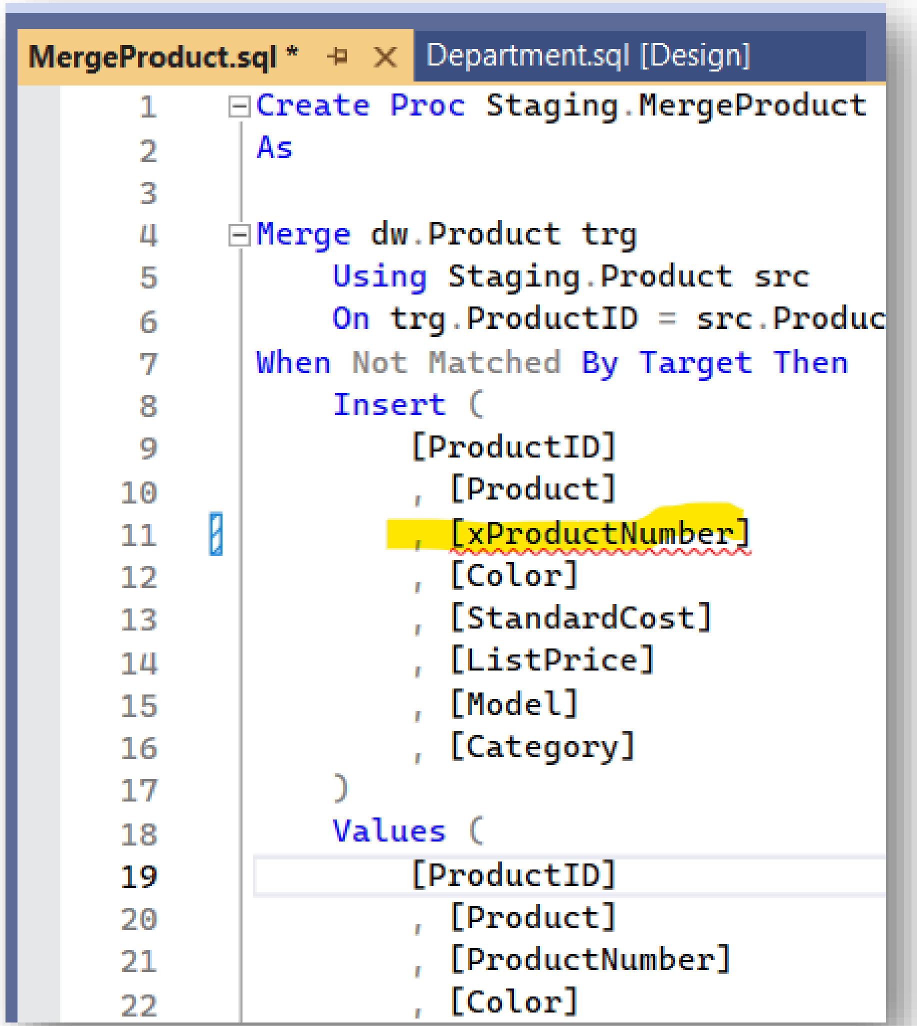
Microsoft microsoft.com | 2,607,474 | ★★★★★ (4)

Enables users to develop and publish database schemas for MSSQL Databases

imaginet

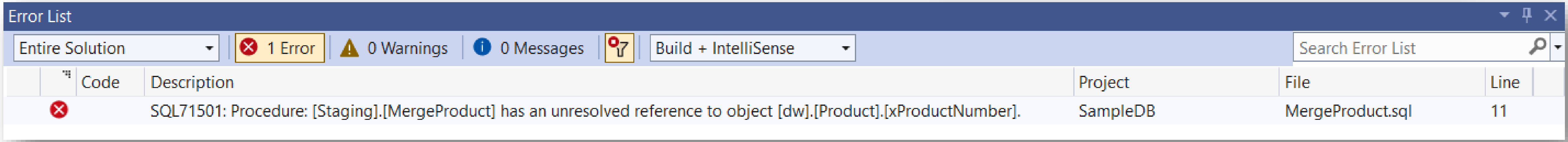
Visual Studio bonus value

- Build project
  - Detects syntax errors and invalid references
  - Sanity check before attempting to deploy
  - DACPAC file generated



MergeProduct.sql * | Department.sql [Design]

```
Create Proc Staging.MergeProduct
As

Merge dw.Product trg
    Using Staging.Product src
    On trg.ProductID = src.Produc
When Not Matched By Target Then
    Insert (
        [ProductID]
        , [Product]
        , [xProductNumber]
        , [Color]
        , [StandardCost]
        , [ListPrice]
        , [Model]
        , [Category]
    )
    Values (
        [ProductID]
        , [Product]
        , [ProductNumber]
        , [Color]
```



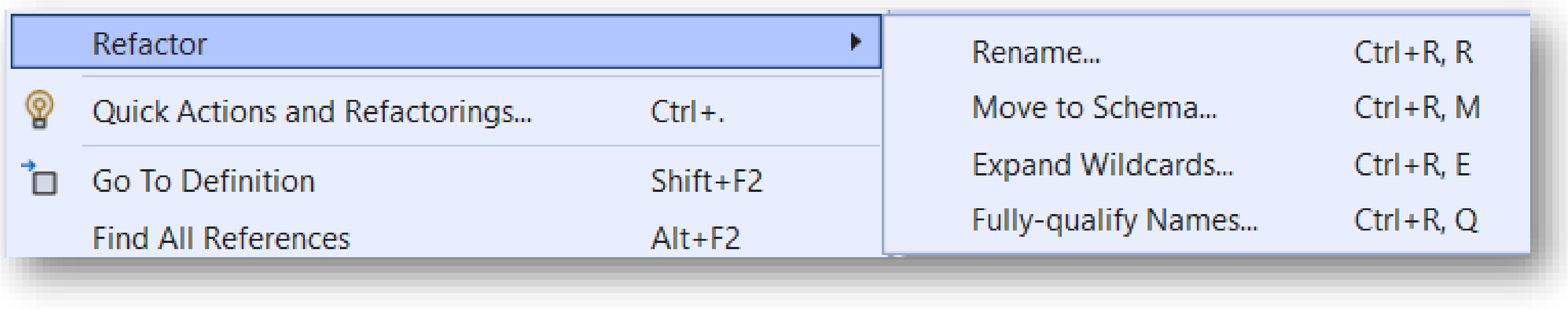
Error List

Entire Solution | 1 Error | 0 Warnings | 0 Messages | Build + IntelliSense | Search Error List

	Code	Description	Project	File	Line
✖		SQL71501: Procedure: [Staging].[MergeProduct] has an unresolved reference to object [dw].[Product].[xProductNumber].	SampleDB	MergeProduct.sql	11

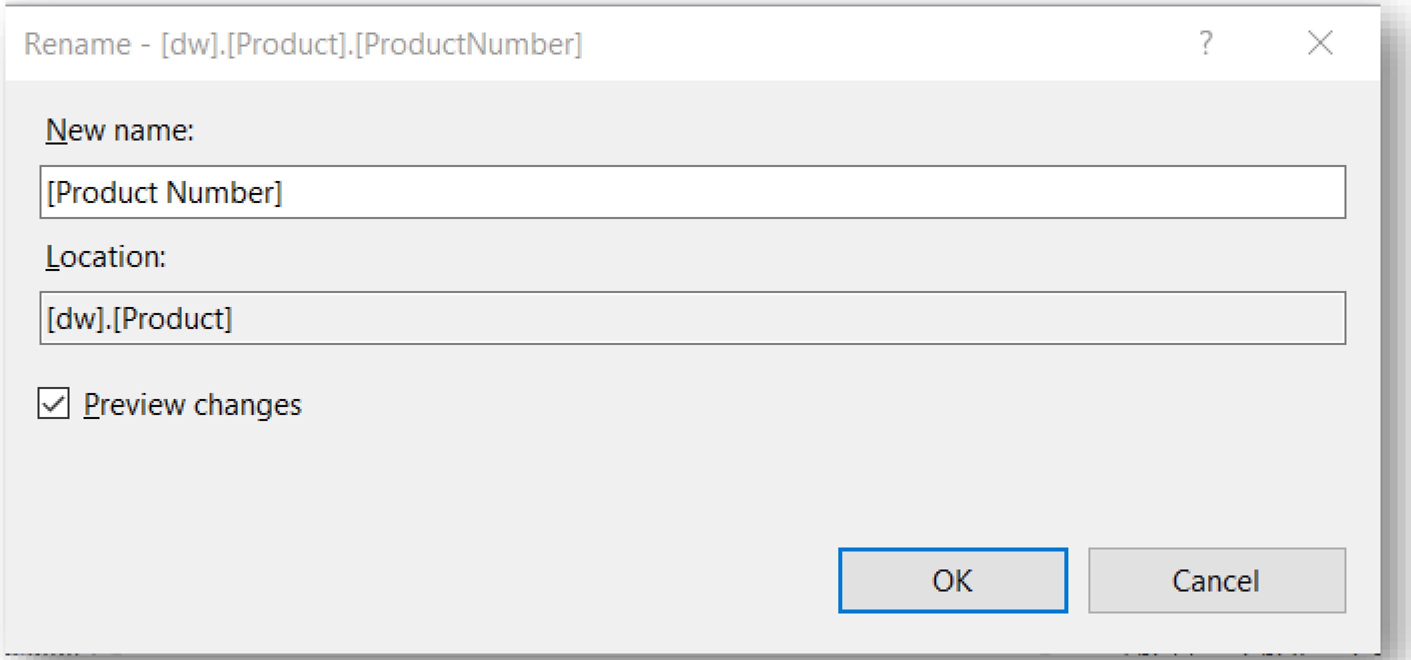
Visual Studio bonus value

- Context menu
  - Refactor
  - Go To Definition
  - Find All References


Refactor
Quick Actions and Refactorings... Ctrl+.
Go To Definition Shift+F2
Find All References Alt+F2
Rename... Ctrl+R, R
Move to Schema... Ctrl+R, M
Expand Wildcards... Ctrl+R, E
Fully-qualify Names... Ctrl+R, Q

Visual Studio bonus value

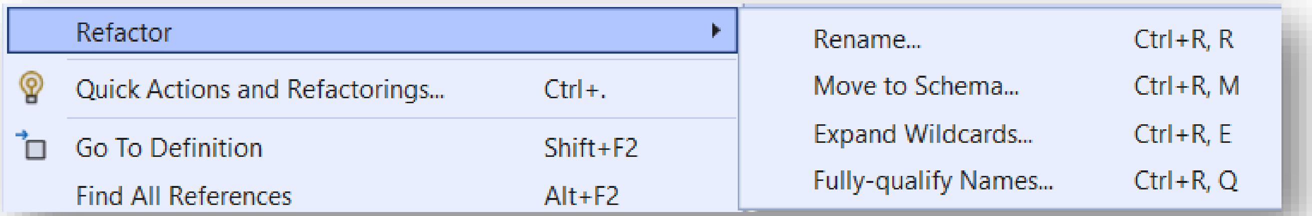
- Context menu
  - Refactor
  - Critical to avoiding data loss on deployment


Rename - [dw].[Product].[ProductNumber]

New name:
[Product Number]

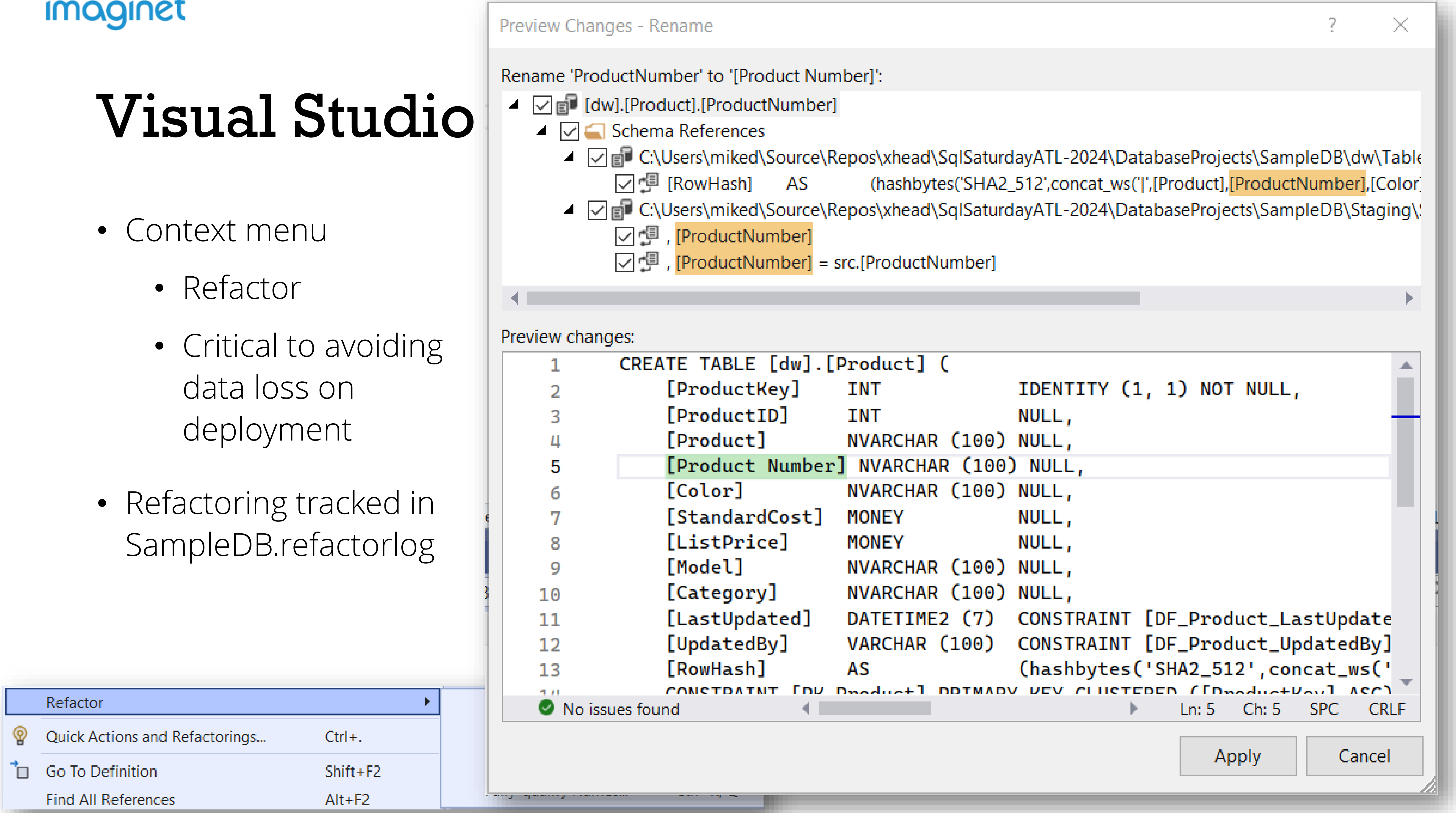
Location:
[dw].[Product]

☑ Preview changes


Refactor
Quick Actions and Refactorings... Ctrl+.
Go To Definition Shift+F2
Find All References Alt+F2
Rename... Ctrl+R, R
Move to Schema... Ctrl+R, M
Expand Wildcards... Ctrl+R, E
Fully-qualify Names... Ctrl+R, Q

Visual Studio

- Context menu
  - Refactor
  - Critical to avoiding data loss on deployment
- Refactoring tracked in SampleDB.refactorlog



Preview Changes - Rename

Rename 'ProductNumber' to '[Product Number]':

```
[dw].[Product].[ProductNumber]
  Schema References
    C:\Users\miked\Source\Repos\xhead\SqlSaturdayATL-2024\DatabaseProjects\SampleDB\dw\Table
      [RowHash]      AS          (hashbytes('SHA2_512',concat_ws('|',[Product],[ProductNumber],[Color]
    C:\Users\miked\Source\Repos\xhead\SqlSaturdayATL-2024\DatabaseProjects\SampleDB\Staging\
      , [ProductNumber]
      , [ProductNumber] = src.[ProductNumber]
```

Preview changes:

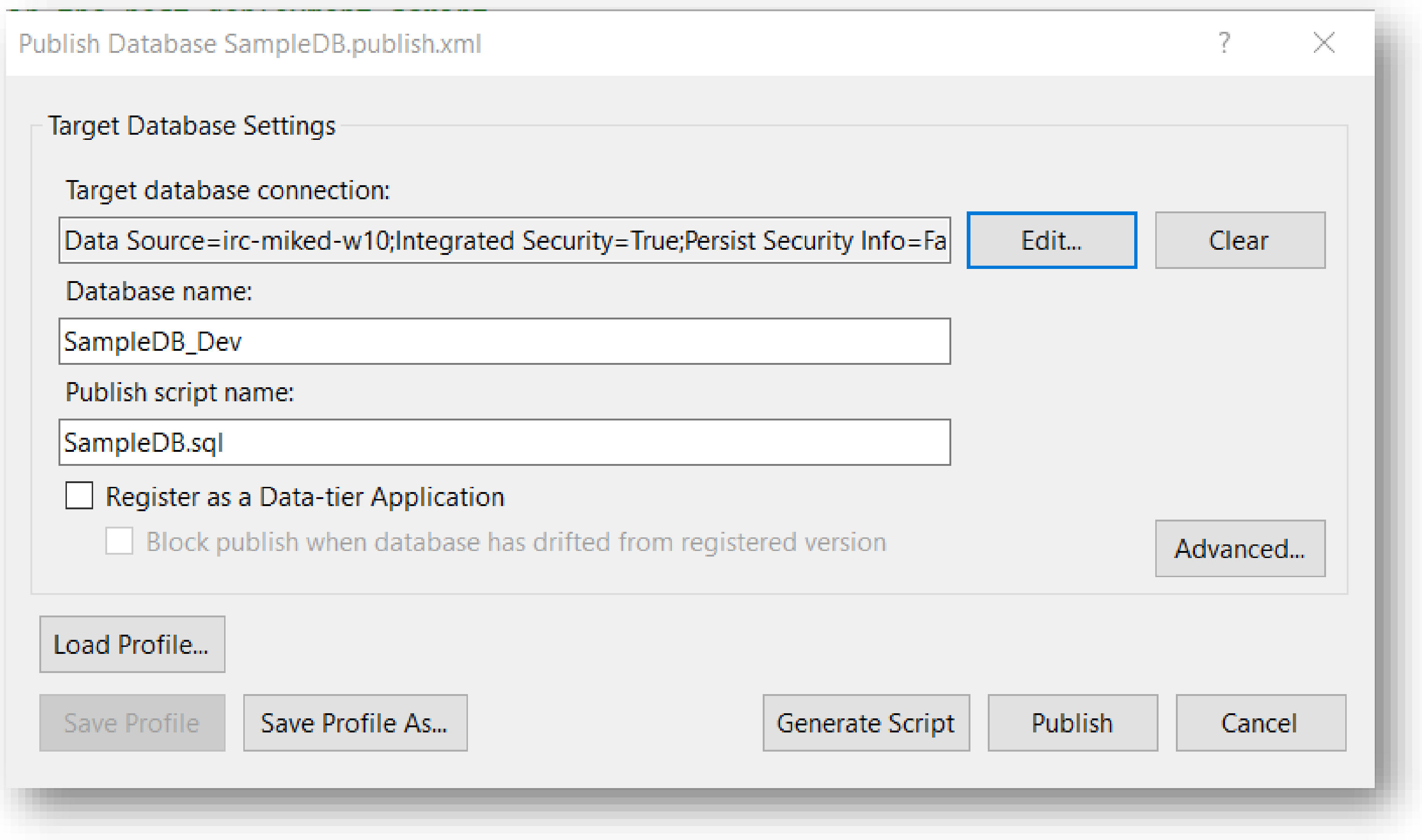
```sql
CREATE TABLE [dw].[Product] (
    [ProductKey]     INT             IDENTITY (1, 1) NOT NULL,
    [ProductID]      INT             NULL,
    [Product]        NVARCHAR (100)  NULL,
    [Product Number] NVARCHAR (100) NULL,
    [Color]          NVARCHAR (100)  NULL,
    [StandardCost]   MONEY           NULL,
    [ListPrice]      MONEY           NULL,
    [Model]          NVARCHAR (100)  NULL,
    [Category]       NVARCHAR (100)  NULL,
    [LastUpdated]    DATETIME2 (7)   CONSTRAINT [DF_Product_LastUpdate
    [UpdatedBy]      VARCHAR (100)   CONSTRAINT [DF_Product_UpdatedBy]
    [RowHash]        AS              (hashbytes('SHA2_512',concat_ws('
    CONSTRAINT [PK_Product] PRIMARY KEY CLUSTERED ([ProductKey] ASC)
```

No issues found   Ln: 5   Ch: 5   SPC   CRLF

Apply
Cancel
Refactor
Quick Actions and Refactorings... Ctrl+.
Go To Definition Shift+F2
Find All References Alt+F2

Publish from Visual Studio

- Rt-click, Deploy
    - Create script
    - Publish directly
- Publish profiles (*.publish.xml)
- Critical settings in publish profile
    - Use Smart Defaults
    - Ignore column order


Publish Database SampleDB.publish.xml

Target Database Settings

Target database connection: Data Source=irc-miked-w10;Integrated Security=True;Persist Security Info=Fa

Database name: SampleDB_Dev

Publish script name: SampleDB.sql

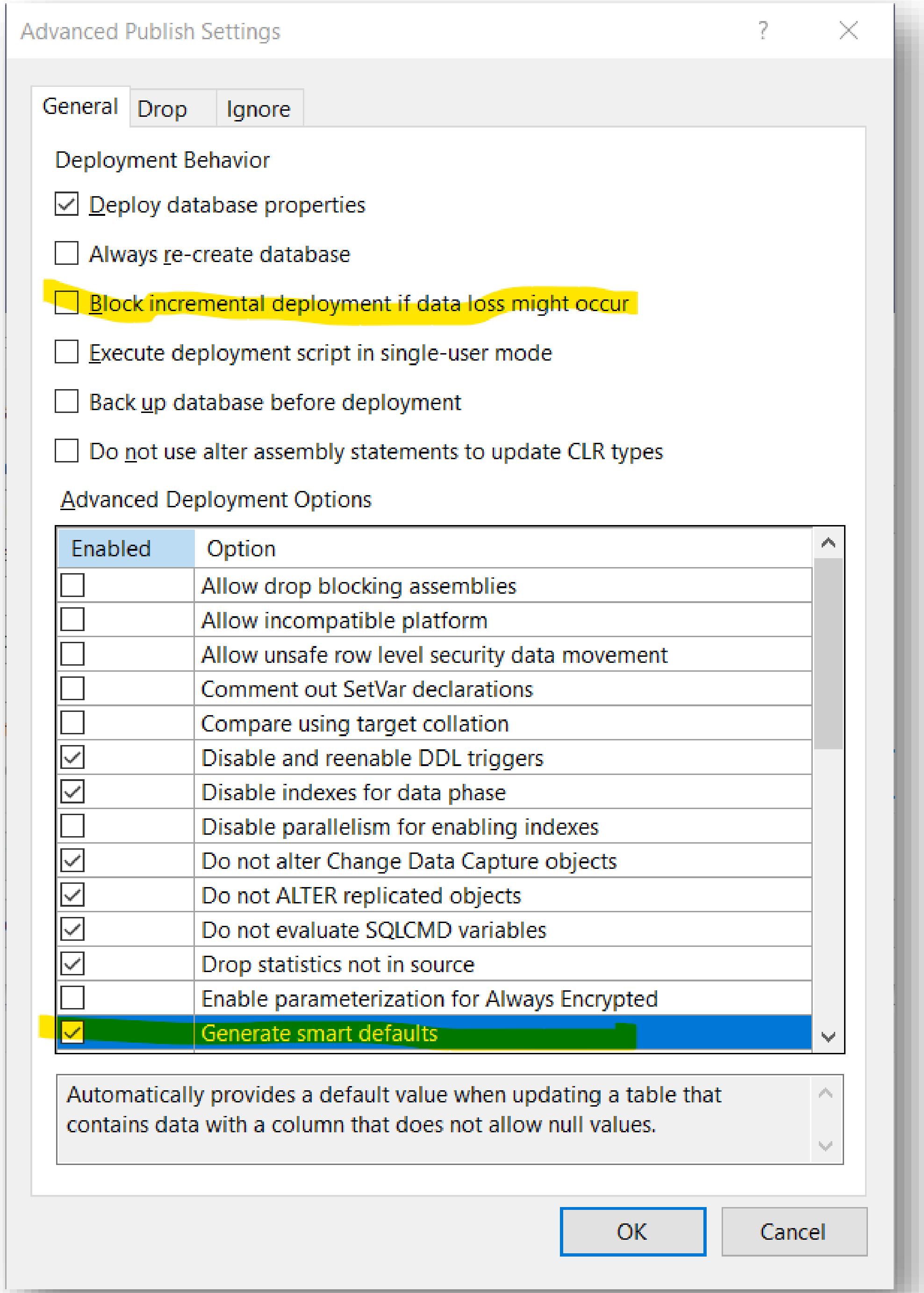
☐ Register as a Data-tier Application

☐ Block publish when database has drifted from registered version

imaginet

Publish from Visual Studio

- Rt-click, Deploy
  - Create script
  - Publish directly
- Publish profiles (*.publish.xml)
- Critical settings in publish profile
  - Use Smart Defaults
  - Ignore column order



Advanced Publish Settings

Deployment Behavior

- [✓] Deploy database properties
- [ ] Always re-create database
- [ ] Block incremental deployment if data loss might occur
- [ ] Execute deployment script in single-user mode
- [ ] Back up database before deployment
- [ ] Do not use alter assembly statements to update CLR types

Advanced Deployment Options

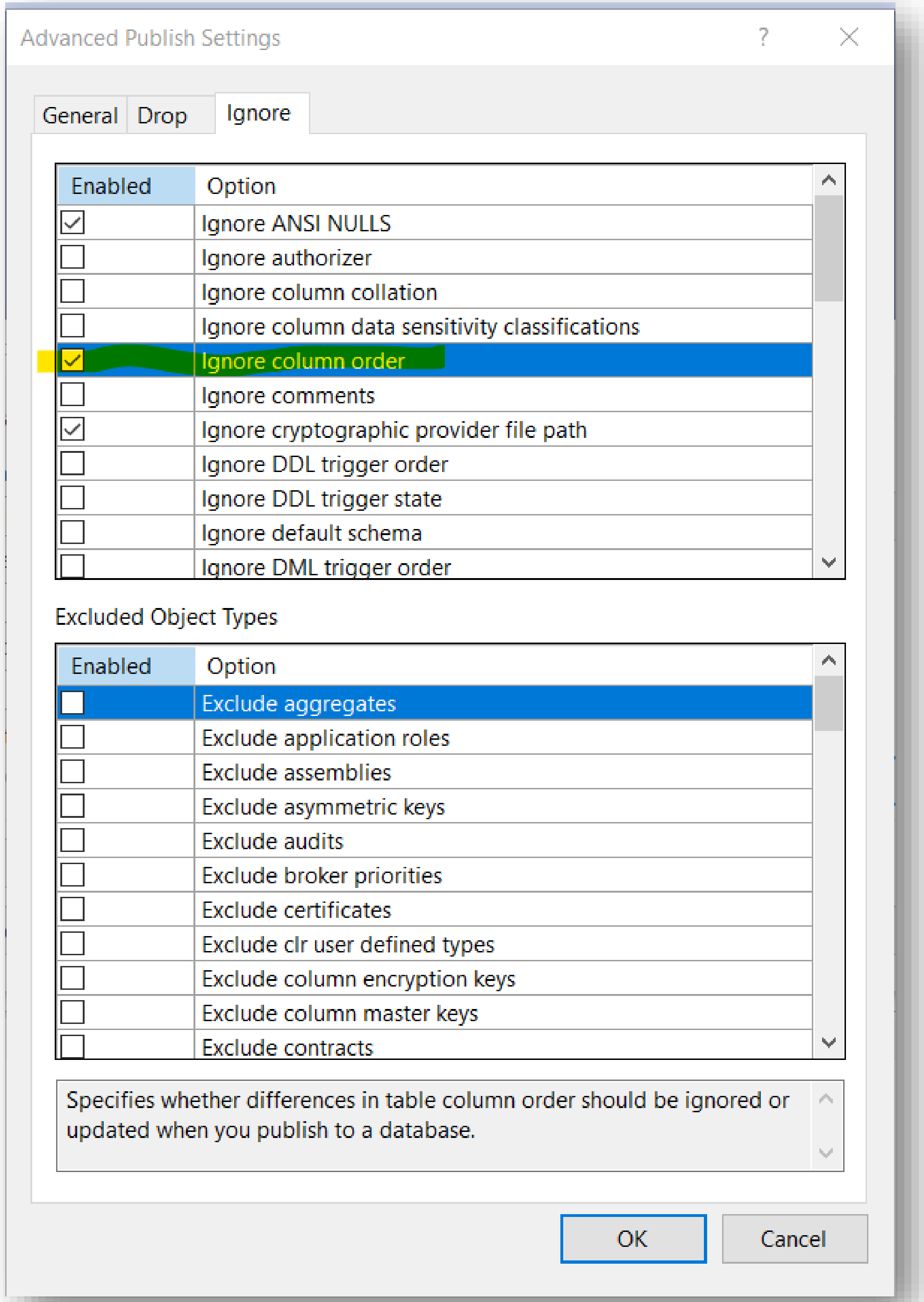
Enabled	Option
☐	Allow drop blocking assemblies
☐	Allow incompatible platform
☐	Allow unsafe row level security data movement
☐	Comment out SetVar declarations
☐	Compare using target collation
✓	Disable and reenable DDL triggers
✓	Disable indexes for data phase
☐	Disable parallelism for enabling indexes
✓	Do not alter Change Data Capture objects
✓	Do not ALTER replicated objects
✓	Do not evaluate SQLCMD variables
✓	Drop statistics not in source
☐	Enable parameterization for Always Encrypted
✓	Generate smart defaults

Automatically provides a default value when updating a table that contains data with a column that does not allow null values.

imaginet

Publish from Visual Studio

- Rt-click, Deploy
    - Create script
    - Publish directly
- Publish profiles (*.publish.xml)
- Critical settings in publish profile
    - Use Smart Defaults
    - Ignore column order



Advanced Publish Settings

General | Drop | Ignore

Enabled	Option
✓	Ignore ANSI NULLS
	Ignore authorizer
	Ignore column collation
	Ignore column data sensitivity classifications
✓	Ignore column order
	Ignore comments
✓	Ignore cryptographic provider file path
	Ignore DDL trigger order
	Ignore DDL trigger state
	Ignore default schema
	Ignore DML trigger order

Excluded Object Types

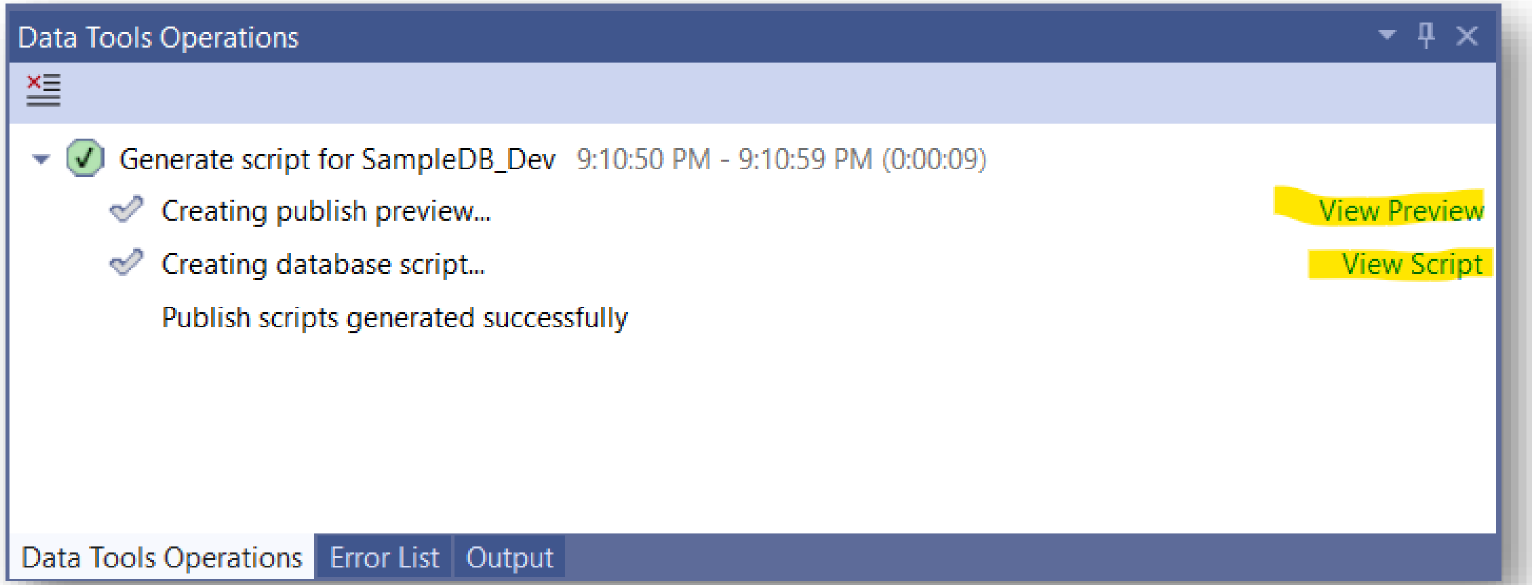
Enabled	Option
	Exclude aggregates
	Exclude application roles
	Exclude assemblies
	Exclude asymmetric keys
	Exclude audits
	Exclude broker priorities
	Exclude certificates
	Exclude clr user defined types
	Exclude column encryption keys
	Exclude column master keys
	Exclude contracts

Specifies whether differences in table column order should be ignored or updated when you publish to a database.

imaginet

Publish from Visual Studio

- Rt-click, Deploy
    - Create script
    - Publish directly
- Preview contains summary of actions and potential warnings
    - Avoid table rebuilds



Generate script for SampleDB_Dev 9:10:50 PM - 9:10:59 PM (0:00:09)
Creating publish preview...
Creating database script...
Publish scripts generated successfully

SQLPACKAGE.EXE

- Command line utility for deploying DACPAC files to SQL database
- Uses DACPAC file and publish.xml file
- Command line options
  - Generate script
  - Deploy changes
- Used by Azure DevOps deployment tasks or other CI/CD processes

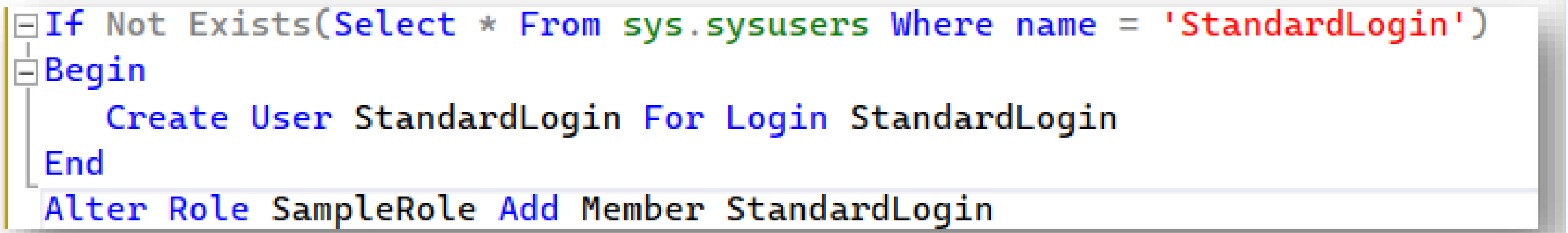
Pre-/Post-deploy scripts

- Only one per project
  - :r .\DefaultSecurity.sql
  - :r .\BackfillReferenceData.sql
  - :r .\Patch_2024-2-10.sql
- Scripts must be **idempotent**
  - Run one or more times without affecting the desired outcome

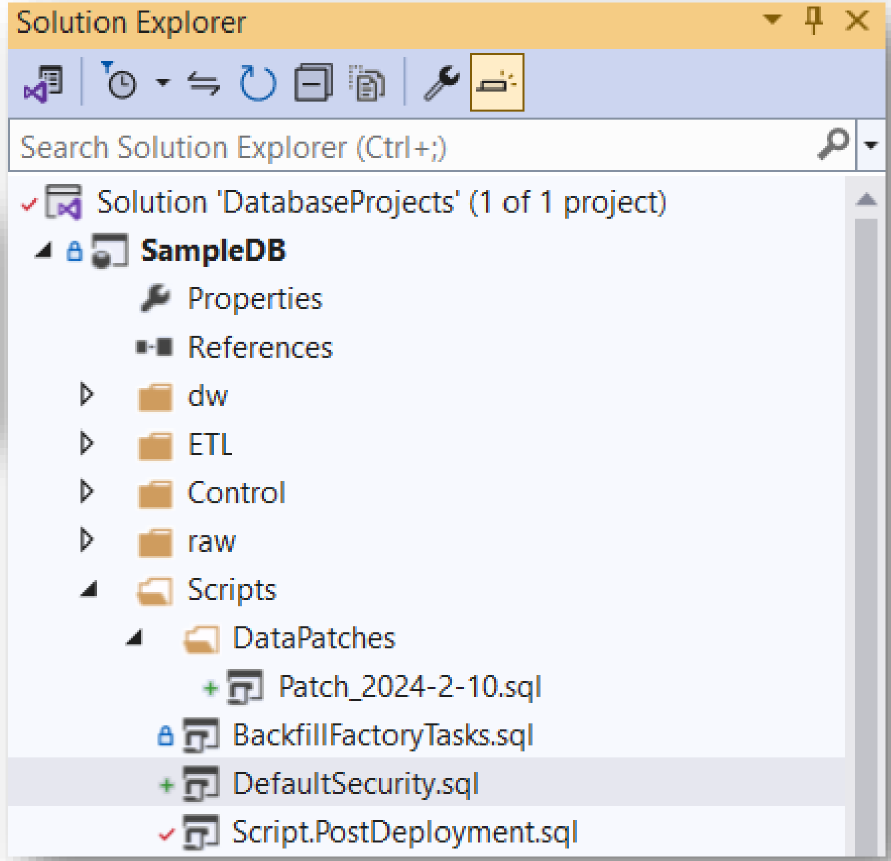
SQLCMD syntax

- :r – run file
- :setVar – set variable
- $(variables)

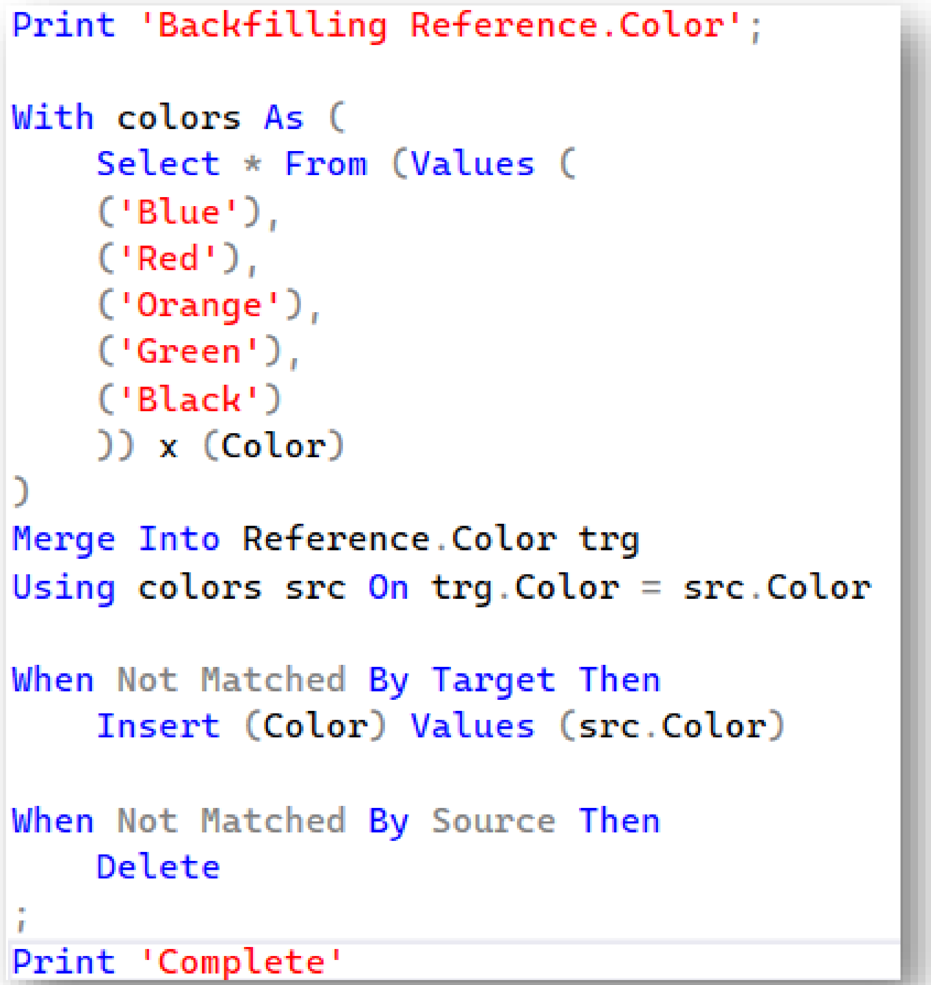
Default security



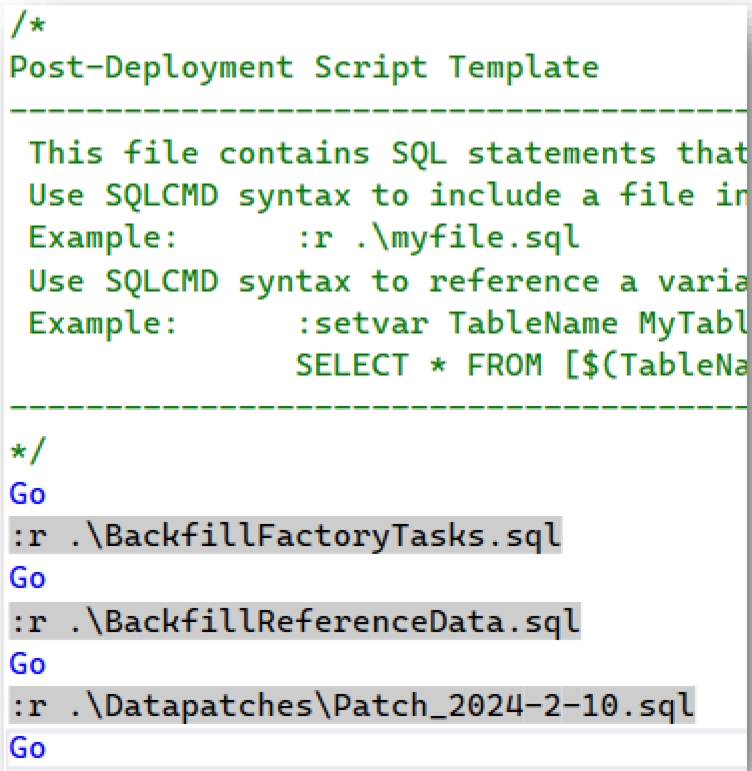
```sql
If Not Exists(Select * From sys.sysusers Where name = 'StandardLogin')
Begin
    Create User StandardLogin For Login StandardLogin
End
Alter Role SampleRole Add Member StandardLogin
```



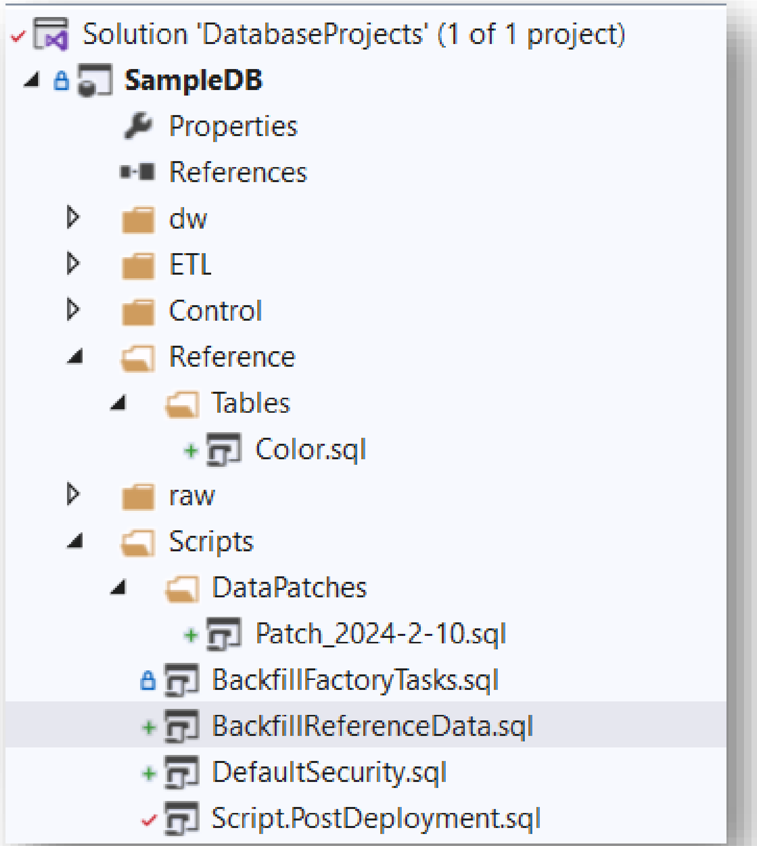
Backfilling Reference Data



```
Print 'Backfilling Reference.Color';

With colors As (
    Select * From (Values (
    ('Blue'),
    ('Red'),
    ('Orange'),
    ('Green'),
    ('Black')
    )) x (Color)
)
Merge Into Reference.Color trg
Using colors src On trg.Color = src.Color

When Not Matched By Target Then
    Insert (Color) Values (src.Color)

When Not Matched By Source Then
    Delete
;
Print 'Complete'
```



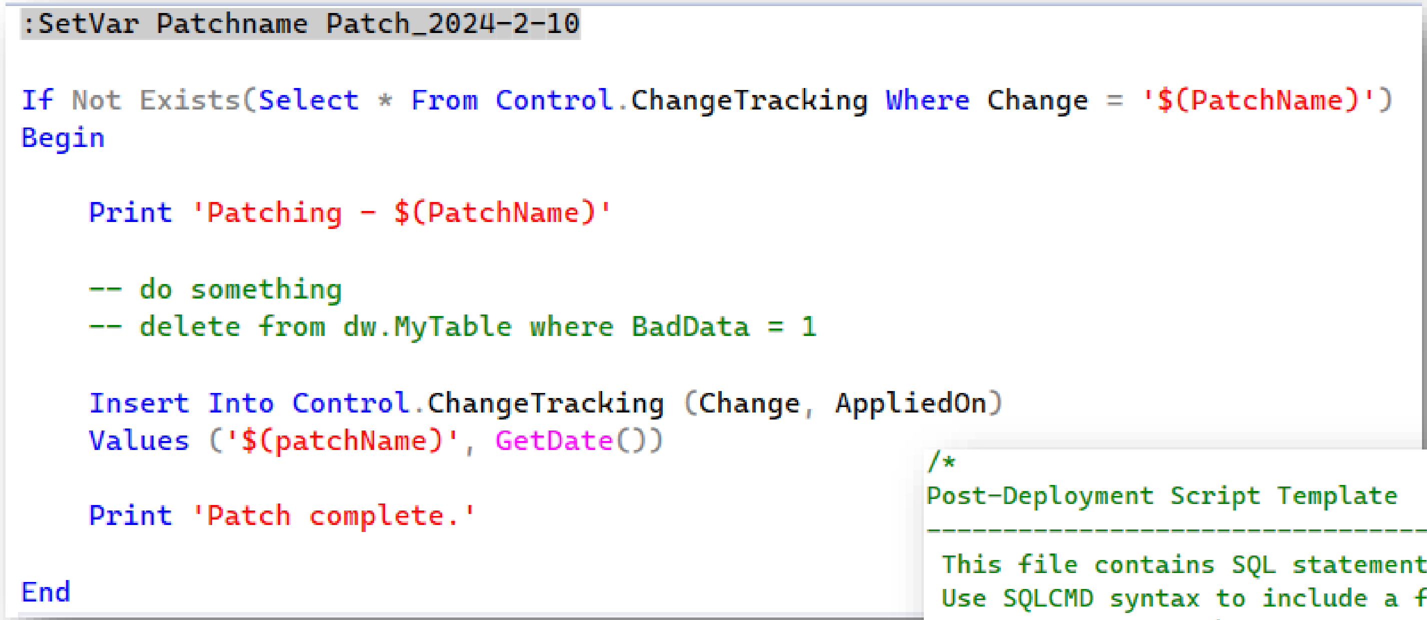
```
/*
Post-Deployment Script Template
--------------------------------------------
 This file contains SQL statements that
 Use SQLCMD syntax to include a file in
 Example:      :r .\myfile.sql
 Use SQLCMD syntax to reference a varia
 Example:      :setvar TableName MyTabl
               SELECT * FROM [$(TableNa
--------------------------------------------
*/
Go
:r .\BackfillFactoryTasks.sql
Go
:r .\BackfillReferenceData.sql
Go
:r .\Datapatches\Patch_2024-2-10.sql
Go
```



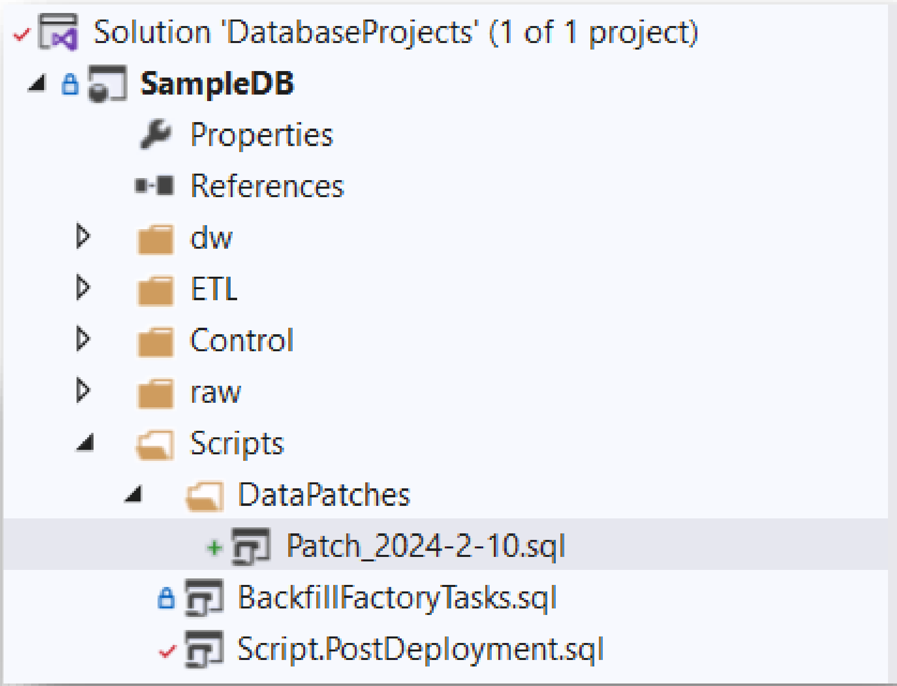
```
Solution 'DatabaseProjects' (1 of 1 project)
  SampleDB
    Properties
    References
    dw
    ETL
    Control
    Reference
      Tables
        Color.sql
    raw
    Scripts
      DataPatches
        Patch_2024-2-10.sql
      BackfillFactoryTasks.sql
      BackfillReferenceData.sql
      DefaultSecurity.sql
      Script.PostDeployment.sql
```

imaginet

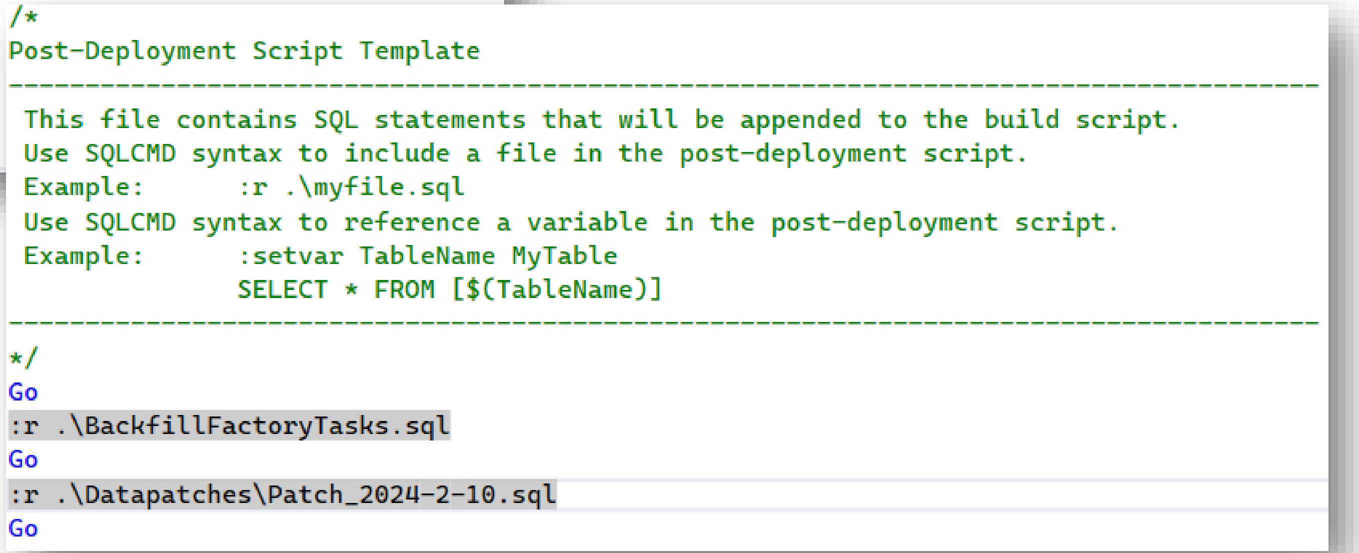
Executing Data Patches



```
:SetVar Patchname Patch_2024-2-10

If Not Exists(Select * From Control.ChangeTracking Where Change = '$(PatchName)')
Begin

    Print 'Patching - $(PatchName)'

    -- do something
    -- delete from dw.MyTable where BadData = 1

    Insert Into Control.ChangeTracking (Change, AppliedOn)
    Values ('$(patchName)', GetDate())

    Print 'Patch complete.'

End
```



```
Solution 'DatabaseProjects' (1 of 1 project)
  SampleDB
      Properties
      References
    dw
    ETL
    Control
    raw
    Scripts
      DataPatches
        Patch_2024-2-10.sql
      BackfillFactoryTasks.sql
      Script.PostDeployment.sql
```



```
/*
Post-Deployment Script Template
--------------------------------------------------------------------------------------
 This file contains SQL statements that will be appended to the build script.
 Use SQLCMD syntax to include a file in the post-deployment script.
 Example:      :r .\myfile.sql
 Use SQLCMD syntax to reference a variable in the post-deployment script.
 Example:      :setvar TableName MyTable
               SELECT * FROM [$(TableName)]
--------------------------------------------------------------------------------------
*/
Go
:r .\BackfillFactoryTasks.sql
Go
:r .\Datapatches\Patch_2024-2-10.sql
Go
```

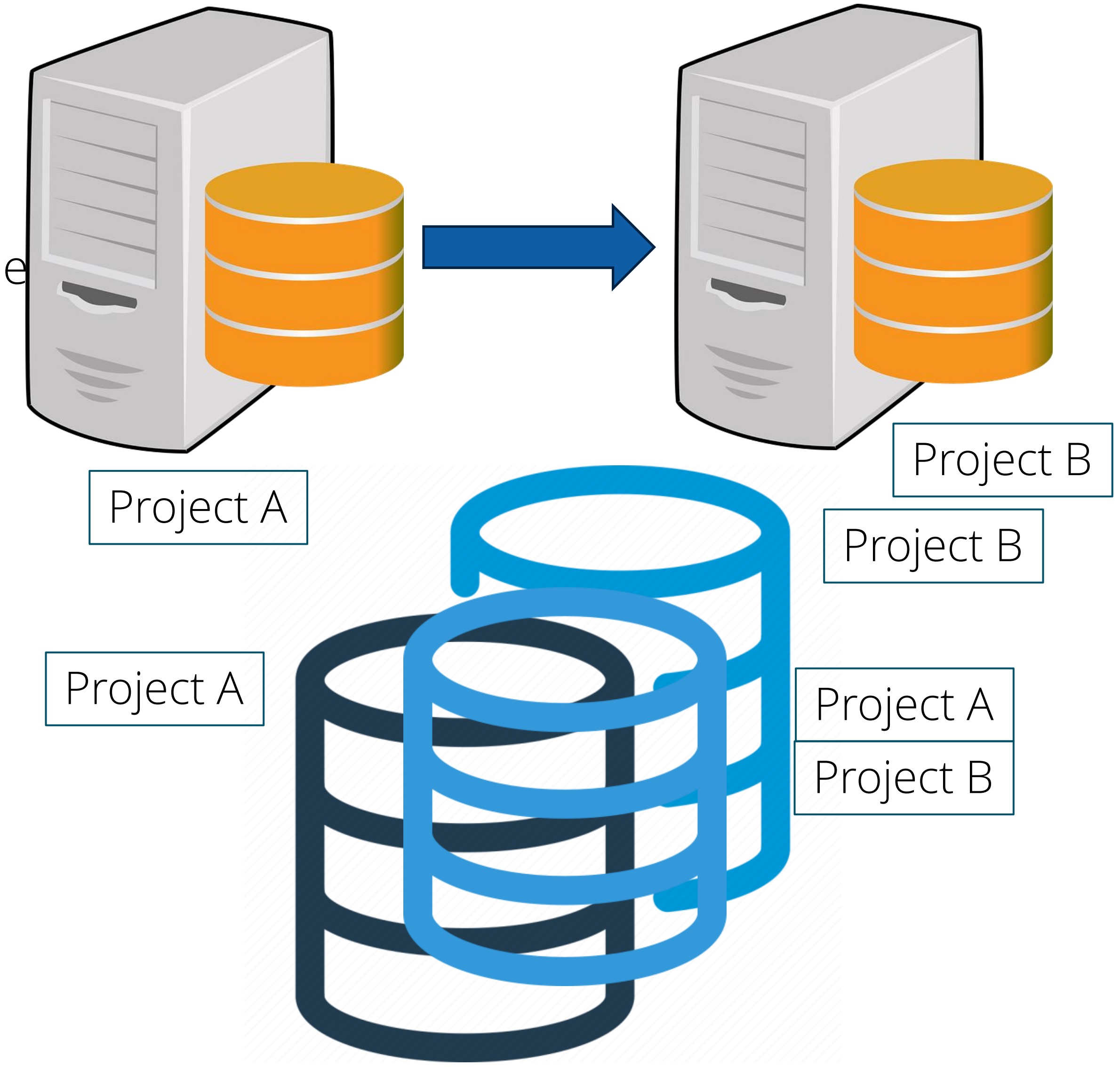
Importing from an existing database

- Databases *should* be self-contained
- Often exposes errors/problems
  - Missing objects, bad names
  - Three-part and four-part names
- Replace external references with CREATE SYNONYM and $(variables)

imaginet

Database references

- What is in or out of your control? What can you de
- System databases
  - master, msdb
- Other database projects
- DACPAC files (binary reference)
- Target:
  - Different server, different database
  - Same server, different database
  - Same server, same database



Database reference scenarios

- Cross-reference to another database outside of your control
- Same or different server, different database (dependency)
  - CREATE SYNONYM to specific tables, views or sprocs
  - Deploy only our own objects in our own database
  - Schema compare on original database
- Same server, same database (inheritance)
  - No synonyms necessary
  - Deploy our own objects in the referenced database
    - Don’t clobber other objects in the target database
    - Our own objects could get clobbered by another deployment

Database reference scenario – Unit Testing

- Database project (SUT - system under test)
- Unit test framework project contains framework objects (TDD)
- Test database project
  - References database project under test and unit test framework project
    - Both references are "same server, same database"
  - Contains test objects (sprocs)
- Deploy test project and execute unit tests in unit test environment or part of a build
  - SUT database and TDD framework objects get deployed
  - Execute unit tests for pass/fail
- Deploy SUT normally in production environments
  - Clean deploy - no TDD or unit tests included

imaginet

Using SQL Database Projects

With Visual Studio

SQL Server Data Tools

https://github.com/xhead/SqlSaturdayATL-2024

Mike Diehl, Miked@imaginet.Com
Practice Lead
Data Engineering and Business Intelligence

@xhead

/mikediehlsqlbi