

Steps to install SQL Server Data Tools (SSDT) for Visual Studio — based on below Microsoft doc's page.

<https://learn.microsoft.com/en-us/sql/ssdt/download-sql-server-data-tools-ssdt?view=sql-server-ver17&tabs=vs2026>

✓ What is SSDT

- SSDT is a set of development tools integrated into Visual Studio that lets you build and manage: relational databases (on-premises or Azure SQL), data models for Analysis Services (AS), Integration Services (IS) packages, and reports for Reporting Services (RS).
- It treats database development like application development: you use projects, version-control, build and deploy process — making it easier to maintain, deploy, and integrate with CI/CD workflows (for example, via the SqlPackage CLI).
- For more advanced or extended workloads (like SSAS, SSIS, SSRS), SSDT works together with separate extensions.

🔧 Installing SSDT with Visual Studio

If you already have Visual Studio (2022 or 2026)

1. Open the **Visual Studio Installer** (search “installer” in Windows Start).
2. For the Visual Studio version you want to add SSDT to, click **Modify**.
3. In the list of workloads, under **“Data storage and processing”**, check **“SQL Server Data Tools”**.
4. Continue and complete the installation. Visual Studio will then include SSDT and you’ll be able to create database-related projects.

If you don't have Visual Studio yet

- You can download and install Visual Studio 2026 (or 2022) from Microsoft. Then during setup you can select SSDT as part of the workloads.

<https://learn.microsoft.com/en-us/sql/ssdt/download-sql-server-data-tools-ssdt?tabs=vs2026&view=sql-server-ver17>

✳️ Additional Workloads: SSAS / SSIS / SSRS

- SSDT core — when installed via workloads — enables **relational database projects**.
- If you want to work with **Analysis Services (SSAS)**, **Integration Services (SSIS)**, or **Reporting Services (SSRS)**, you need to install additional **extensions**: open Visual Studio → go to **Extensions > Manage Extensions** (or visit the Marketplace) and install the appropriate extensions.
- For VS 2022 and 2026, these extensions are maintained separately from the core SSDT workload.

📄 Supported SQL & Platform Versions

Depending on your Visual Studio version, SSDT supports different SQL Server and Azure SQL platforms:

- **Visual Studio 2026:**
 - Relational databases: SQL Server 2016 (13.x) → SQL Server 2025 (17.x)
 - Azure SQL Database / Managed Instance, Azure Synapse Analytics (Dedicated & Serverless pools), Microsoft Fabric SQL databases and warehouses
 - Analysis Services models & Reporting Services reports: SQL Server 2016 → SQL Server 2025
 - Integration Services packages: SQL Server 2019 (15.x) → SQL Server 2025 (17.x)
- **Visual Studio 2022:**
 - Relational databases: SQL Server 2016 → SQL Server 2022
 - Azure SQL, Synapse, similar cloud platforms
 - Analysis Services & Reporting Services: SQL Server 2016 → SQL Server 2022
 - Integration Services: SQL Server 2019 → SQL Server 2022
- **Visual Studio 2019:** (still supported by SSDT) — supports earlier versions accordingly.

 **Offline Installation (if you don't have stable internet / for isolated networks)**

- SSDT supports **offline installation**. Two approaches:
 1. **Download all required files** first (as a bundle), then install on your machine.
 2. For multiple machines — use the **Visual Studio bootstrapper** via command line to deploy SSDT.
- This is useful in corporate environments, secure networks, or when bandwidth is limited.

 **Notes / Special Considerations**

- There is a “**SDK-style (preview)**” version of SSDT — this brings a newer project format (in line with .NET Core SDK-style projects). It’s separate from the “classic” SSDT. This can be relevant if you prefer or require the newer project format.
- On **Arm64 devices** (e.g. Windows 11 on ARM64), SSDT is supported in Visual Studio 2026 — but with limitations: no IntelliSense or code completion for T-SQL files, no T-SQL debugger, and no support for LocalDB.
- If working with SSAS, SSIS, or SSRS — don’t forget to install the appropriate extensions after SSDT; just installing SSDT doesn’t automatically enable those BI/ETL/reporting features.

<https://www.sqlbachamps.com/>

Step-by-step checklist for installing SSDT + SSRS + SSIS + SSAS on a fresh Windows machine.

SSDT Installation Checklist (Visual Studio 2022 / 2026)

Part 1 — Install Visual Studio

1. Download Visual Studio (Community / Professional / Enterprise).
2. Run the installer.
3. On the *Workloads* tab, check:
✓ Data storage and processing
 (This includes **SQL Server Data Tools (SSDT)** for relational database development.)
4. Complete the installation.

Part 2 — Install SSDT Components (Core SQL Tools)

1. Open **Visual Studio Installer**.
2. Click **Modify** next to your Visual Studio version.
3. Ensure **SQL Server Data Tools** is checked (under *Data storage and processing*).
4. Click **Modify / Install** to apply changes.
5. After installation, confirm that Visual Studio now offers:
 - SQL Server Database Project (.sqlproj)
 - Schema Compare
 - Data Compare
 - SQL Editor

Part 3 — Install BI Extensions (SSRS / SSAS / SSIS)

SSDT installs *database development* only.

BI/ETL tools require separate extensions.

A. Install Reporting Services (SSRS) Extension

1. Open Visual Studio.
2. Go to **Extensions > Manage Extensions**.
3. Search: **Reporting Services**
4. Install **Microsoft Reporting Services Projects**.
5. Restart Visual Studio.

New project types added:

- Report Server Project
- Report Wizard
- RDLC Report designer

B. Install Analysis Services (SSAS) Extension

1. Open **Extensions > Manage Extensions**.
2. Search: **Analysis Services**
3. Install **Microsoft Analysis Services Projects**.
4. Restart Visual Studio.

New project types added:

- Multidimensional and Tabular models
- Data mining structures (if supported)

C. Install Integration Services (SSIS) Extension

SSIS extension versions must match the SQL Server SSIS runtime you plan to deploy to.

1. Go to **Extensions > Manage Extensions**.

2. Search: **Integration Services**.
3. Install **SQL Server Integration Services Projects**.
4. Restart Visual Studio.

New project types added:

- Integration Services Project
- SSIS Package Designer
- Data Flow / Control Flow designers

Part 4 — Optional Components

✓ SqlPackage CLI

Useful for automation, CI/CD, and publishing .dacpac files.

Download from Microsoft and add to PATH.

✓ LocalDB (if needed)

- Comes optionally with Visual Studio.
- If not installed:
Open VS Installer → Individual Components → check **SQL Server Express LocalDB**.

Part 5 — Post-Installation Validation

After installation, open Visual Studio and verify:

In Create a New Project, you should see:

- **SQL Server Database Project**
- **Integration Services Project**
- **Report Server Project**
- **Analysis Services Tabular/Multi-Dimensional Project**

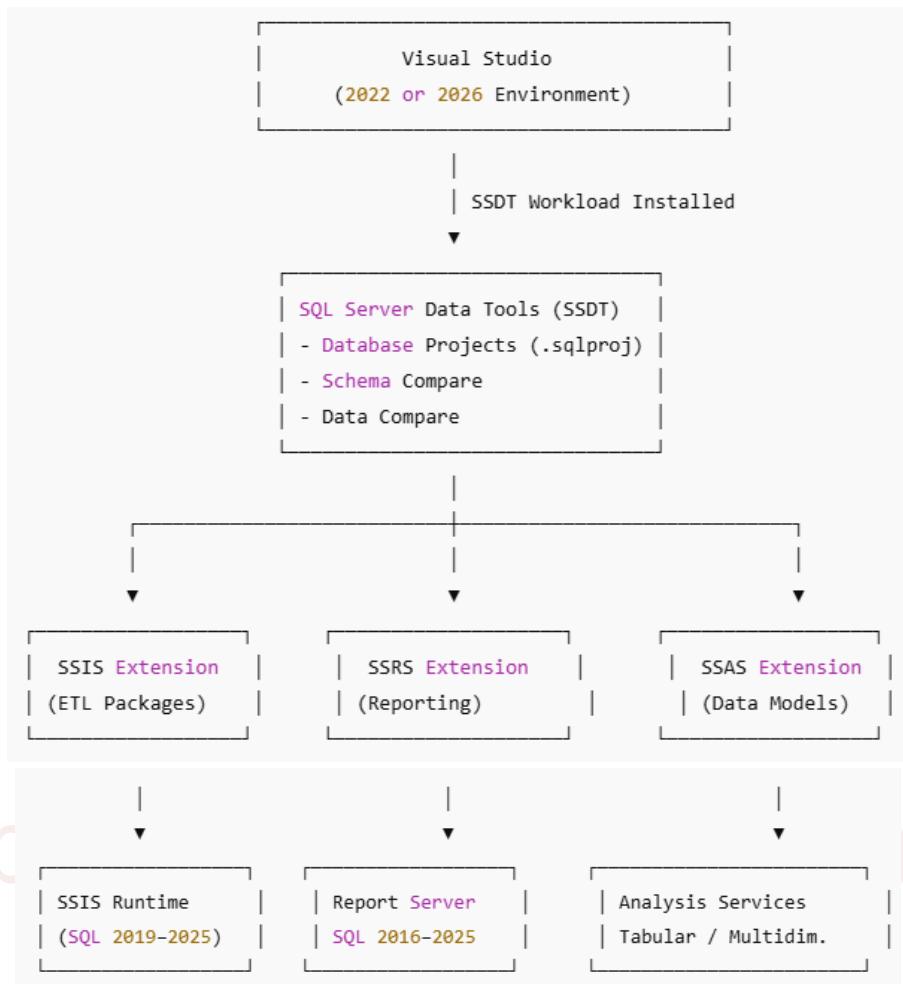
In Tools > Extensions, confirm:

- SQL Server Data Tools
- Reporting Services Projects
- Analysis Services Projects
- Integration Services Projects

Side-by-Side Comparison: VS 2022 vs VS 2026 SSDT Capabilities

Feature / Support Area	VS 2022	VS 2026
Core SSDT (relational DB projects)	✓ Fully supported	✓ Fully supported
Supported SQL Server Versions	SQL 2016 → SQL 2022	SQL 2016 → SQL 2025 + Fabric
Azure SQL DB	✓ Yes	✓ Yes
Azure Synapse Analytics	✓ Dedicated + Serverless	✓ Dedicated + Serverless
Microsoft Fabric (lakehouse/warehouse SQL)	✗ Not fully integrated	✓ Fully supported
Analysis Services Projects (SSAS)	✓ 2016–2022	✓ 2016–2025
Reporting Services Projects (SSRS)	✓ 2016–2022	✓ 2016–2025
Integration Services (SSIS)	SQL 2019 → SQL 2022	SQL 2019 → SQL 2025
ARM64 Support	Partial/limited	Improved (but IntelliSense issues remain)
SDK-Style Database Project (Preview)	✓ Available	✓ Improved & more stable
LocalDB support	✓ Yes	✗ No LocalDB on ARM64
Best Use Case	Enterprise use for SQL 2016–2022	Modern workloads including SQL 2025 & Fabric

3. SSDT + SSIS/SSRS/SSAS Architecture Diagram (Text-Based)



This shows:

- SSDT is the **core database development layer**
- SSIS / SSRS / SSAS are **separate extensions**
- Each extension connects to corresponding **SQL Server services**

PowerShell diagnostic script that will check whether **SSDT, SSIS, SSRS, and SSAS** components are installed correctly on your machine. It's safe to run and gives a clear report.

PowerShell Diagnostic Script: SSDT + SSIS/SSRS/SSAS

```
# =====
# SSDT + SSIS/SSRS/SSAS Installation Diagnostic
# =====

Write-Host "Starting SSDT + BI components diagnostic..." -ForegroundColor Cyan

#
# Function to check Visual Studio workloads
#
function Test-VSWorkload {
    param(
        [string]$WorkloadID,
        [string]$Description
    )
    $vsSetup = "${env:ProgramFiles(x86)}\Microsoft Visual Studio\Installer\vswhere.exe"
    if (Test-Path $vsSetup) {
        $result = & $vsSetup -latest -products * -requires $WorkloadID -property installationPath
        if ($result) {
            Write-Host "[✓] $Description installed at $result" -ForegroundColor Green
        } else {
            Write-Host "[✗] $Description not found" -ForegroundColor Red
        }
    } else {
        Write-Host "[✗] vswhere.exe not found. Cannot check $Description" -ForegroundColor Yellow
    }
}

#
# Function to check VSIX Extensions
#
function Test-VSIXExtension {
    param(
        [string]$ExtensionName,
        [string]$ProjectTemplate
    )
    $extensions = Get-ChildItem -Path "$env:LOCALAPPDATA\Microsoft\VisualStudio" -Recurse -ErrorAction SilentlyContinue |
        Where-Object { $_.Name -match "$ExtensionName" }

    if ($extensions) {
        Write-Host "[✓] $ExtensionName ($ProjectTemplate) installed" -ForegroundColor Green
    } else {
        Write-Host "[✗] $ExtensionName ($ProjectTemplate) not installed" -ForegroundColor Red
    }
}

#
# Check SSDT (Database Projects)
#
Test-VSWorkload -WorkloadID "Microsoft.VisualStudio.Workload.Data" -Description "SQL Server Data Tools (SSDT)"
```

```

# -----
# Check SSRS (Reporting Services)
# -----
Test-VSIXExtension -ExtensionName "ReportingServicesProjects" -ProjectTemplate "SSRS"

# -----
# Check SSAS (Analysis Services)
# -----
Test-VSIXExtension -ExtensionName "AnalysisServicesProjects" -ProjectTemplate "SSAS"

# -----
# Check SSIS (Integration Services)
# -----
Test-VSIXExtension -ExtensionName "SSISProjects" -ProjectTemplate "SSIS"

# -----
# Optional: Check SQL Server LocalDB
# -----
if (Get-Command "sqllocaldb" -ErrorAction SilentlyContinue) {
    $instances = & sqllocaldb i
    Write-Host "[✓] LocalDB installed. Instances:" -ForegroundColor Green
    $instances | ForEach-Object { Write-Host "  $_" }
} else {
    Write-Host "[✗] LocalDB not installed" -ForegroundColor Red
}

Write-Host "`nDiagnostic Complete!" -ForegroundColor Cyan

```

How it Works

1. Checks if **SSDT workload** is installed using **vswhere.exe** (part of Visual Studio Installer).
2. Checks if **SSRS, SSIS, SSAS VSIX extensions** are installed by scanning local Visual Studio extensions folders.
3. Optionally checks **LocalDB** availability.
4. Outputs a clear **green check / red cross report**.

Usage

1. Open **PowerShell as Administrator**.
2. Copy & paste the script or save as **Check-SSDT-BI.ps1**.
3. Run:
.\Check-SSDT-BI.ps1
4. Read the results: green  means installed, red  means missing.

Sample output:

```

PS C:\Windows\system32>
PS C:\Windows\system32> # -----
PS C:\Windows\system32> # Check SSDT (Database Projects)
PS C:\Windows\system32> # -----
PS C:\Windows\system32> Test-VSWorkload -WorkloadID "Microsoft.VisualStudio.Workload.Data" -Description "SQL Server Data Tools (SSDT)"
[!] SQL Server Data Tools (SSDT) installed at C:\Program Files\Microsoft Visual Studio\18\Enterprise
PS C:\Windows\system32>
PS C:\Windows\system32> # -----
PS C:\Windows\system32> # Check SSRS (Reporting Services)
PS C:\Windows\system32> # -----
PS C:\Windows\system32> Test-VSIXExtension -ExtensionName "ReportingServicesProjects" -ProjectTemplate "SSRS"
[!] ReportingServicesProjects (SSRS) not installed
PS C:\Windows\system32>
PS C:\Windows\system32> # -----
PS C:\Windows\system32> # Check SSAS (Analysis Services)
PS C:\Windows\system32> # -----
PS C:\Windows\system32> Test-VSIXExtension -ExtensionName "AnalysisServicesProjects" -ProjectTemplate "SSAS"
[!] AnalysisServicesProjects (SSAS) not installed
PS C:\Windows\system32>
PS C:\Windows\system32> # -----
PS C:\Windows\system32> # Check SSIS (Integration Services)
PS C:\Windows\system32> # -----
PS C:\Windows\system32> Test-VSIXExtension -ExtensionName "SSISProjects" -ProjectTemplate "SSIS"
[!] SSISProjects (SSIS) not installed
PS C:\Windows\system32>
PS C:\Windows\system32> # -----
PS C:\Windows\system32> # Optional: Check SQL Server LocalDB
PS C:\Windows\system32> # -----
PS C:\Windows\system32> if (Get-Command "sqllocaldb" -ErrorAction SilentlyContinue) {
>>     $instances = & sqllocaldb i
>>     Write-Host "[?] LocalDB installed. Instances:" -ForegroundColor Green
>>     $instances | ForEach-Object { Write-Host "    $_" }
>> } else {
>>     Write-Host "[?] LocalDB not installed" -ForegroundColor Red
>> }
[!] LocalDB installed. Instances:
    MSSQLLocalDB
PS C:\Windows\system32>
PS C:\Windows\system32> Write-Host "`nDiagnostic Complete!" -ForegroundColor Cyan

Diagnostic Complete!
PS C:\Windows\system32>
```

PowerShell diagnostic script that not only checks whether SSDT, SSRS, SSIS, and SSAS are installed, but also reports the **exact installed version numbers** where possible.

Enhanced SSDT + SSIS/SSRS/SSAS Diagnostic Script (with Versions)

```
# =====
# Enhanced SSDT + BI Components Diagnostic
# Checks Installation + Version Numbers
# =====

Write-Host "Starting Enhanced SSDT + BI Components Diagnostic..." -ForegroundColor Cyan

# -----
# Function: Check Visual Studio and SSDT Version
# -----
function Get-VSWorkloadVersion {
    param(
        [string]$WorkloadID,
        [string]$Description
    )

    $vswhere = "${env:ProgramFiles(x86)}\Microsoft Visual Studio\Installer\vswhere.exe"
    if (-not (Test-Path $vswhere)) {
        Write-Host "[X] vswhere.exe not found. Cannot check $Description" -ForegroundColor Yellow
        return
    }

    $vsInstall = & $vswhere -latest -products * -requires $WorkloadID -property installationPath
    if ($vsInstall) {
        $devenvPath = Join-Path $vsInstall "Common7\IDE\devenv.exe"
        if (Test-Path $devenvPath) {
            $version = (Get-Item $devenvPath).VersionInfo.ProductVersion
            Write-Host "[✓] $Description installed at $vsInstall (Version: $version)" -ForegroundColor Green
        } else {
            Write-Host "[✓] $Description installed at $vsInstall (Version not found)" -ForegroundColor Green
        }
    } else {
        Write-Host "[X] $Description not installed" -ForegroundColor Red
    }
}

# -----
# Function: Check VSIX Extension + Version
# -----
function Get-VSIXExtensionVersion {
    param(
        [string]$ExtensionName,
        [string]$ProjectTemplate
    )

    $extensionsFolder = "$env:LOCALAPPDATA\Microsoft\VisualStudio"
    $matches = Get-ChildItem -Path $extensionsFolder -Recurse -ErrorAction SilentlyContinue |
        Where-Object { $_.Name -match $ExtensionName -and $_.Extension -eq ".vsixmanifest" }

    if ($matches) {
```

```

foreach ($match in $matches) {
    try {
        [xml]$xml = Get-Content $match.FullName
        $version = $xml.PackageManifest.Metadata.Identity.Version
        Write-Host "[✓] $ExtensionName ($ProjectTemplate) installed, Version: $version" -ForegroundColor Green
    } catch {
        Write-Host "[✗] $ExtensionName ($ProjectTemplate) installed, Version not found" -ForegroundColor Green
    }
}
} else {
    Write-Host "[✗] $ExtensionName ($ProjectTemplate) not installed" -ForegroundColor Red
}
}

# -----
# Check SSDT Workload
# -----
Get-VSWorkloadVersion -WorkloadID "Microsoft.VisualStudio.Workload.Data" -Description "SQL Server Data Tools (SSDT)"

# -----
# Check SSRS Extension
# -----
Get-VSIXExtensionVersion -ExtensionName "ReportingServicesProjects" -ProjectTemplate "SSRS"

# -----
# Check SSAS Extension
# -----
Get-VSIXExtensionVersion -ExtensionName "AnalysisServicesProjects" -ProjectTemplate "SSAS"

# -----
# Check SSIS Extension
# -----
Get-VSIXExtensionVersion -ExtensionName "SSISProjects" -ProjectTemplate "SSIS"

# -----
# Optional: Check LocalDB Version
# -----
if (Get-Command "sqllocaldb" -ErrorAction SilentlyContinue) {
    $instances = & sqllocaldb i
    Write-Host "[✓] LocalDB installed. Instances:" -ForegroundColor Green
    $instances | ForEach-Object {
        $version = & sqllocaldb v $_
        Write-Host " $_ (Version: $version)"
    }
} else {
    Write-Host "[✗] LocalDB not installed" -ForegroundColor Red
}

Write-Host "`nEnhanced Diagnostic Complete!" -ForegroundColor Cyan

```

What This Script Does

1. **SSDT Workload Check:** Uses vswhere.exe to detect installation path and product version.
2. **VSIX Extensions Check:** Detects SSRS, SSIS, SSAS extensions and tries to extract their version from .vsixmanifest.
3. **LocalDB Check:** Lists installed LocalDB instances with their versions.
4. **Color-coded output:** Green  = installed, Red  = missing, Yellow  = partial info.

How to Use

1. Save the script as Check-SSDT-BI-Enhanced.ps1.
2. Open **PowerShell as Administrator**.
3. Run:
.\Check-SSDT-BI-Enhanced.ps1
4. Review the results — it will report **installation status + version numbers** for each component.

Sample Output:

```
PS C:\Windows\system32>
PS C:\Windows\system32> # -----
PS C:\Windows\system32> # Check SSDT Workload
PS C:\Windows\system32> # -----
PS C:\Windows\system32> Get-VSWorkloadVersion -WorkloadID "Microsoft.VisualStudio.Workload.Data" -Description "SQL Server Data Tools (SSDT)"
[!] SQL Server Data Tools (SSDT) installed at C:\Program Files\Microsoft Visual Studio\18\Enterprise (Version: 18.0.11222.15)
PS C:\Windows\system32>
PS C:\Windows\system32> # -----
PS C:\Windows\system32> # Check SSRS Extension
PS C:\Windows\system32> # -----
PS C:\Windows\system32> Get-VSIXExtensionVersion -ExtensionName "ReportingServicesProjects" -ProjectTemplate "SSRS"
[!] ReportingServicesProjects (SSRS) not installed
PS C:\Windows\system32>
PS C:\Windows\system32> # -----
PS C:\Windows\system32> # Check SSAS Extension
PS C:\Windows\system32> # -----
PS C:\Windows\system32> Get-VSIXExtensionVersion -ExtensionName "AnalysisServicesProjects" -ProjectTemplate "SSAS"
[!] AnalysisServicesProjects (SSAS) not installed
PS C:\Windows\system32>
PS C:\Windows\system32> # -----
PS C:\Windows\system32> # Check SSIS Extension
PS C:\Windows\system32> # -----
PS C:\Windows\system32> Get-VSIXExtensionVersion -ExtensionName "SSISProjects" -ProjectTemplate "SSIS"
[!] SSISProjects (SSIS) not installed
PS C:\Windows\system32>
PS C:\Windows\system32> # -----
PS C:\Windows\system32> # Optional: check LocalDB Version
PS C:\Windows\system32> #
PS C:\Windows\system32> if ((Get-Command "sqllocaldb" -ErrorAction SilentlyContinue) {
>>     $instances = & sqllocaldb i
>>     Write-Host "[?] LocalDB installed. Instances:" -ForegroundColor Green
>>     $instances | ForEach-Object {
>>         $version = & sqllocaldb v $_
>>         Write-Host "    $_ (Version: $version)"
>>     }
>> } else {
>>     Write-Host "[?] LocalDB not installed" -ForegroundColor Red
>> }
[!] LocalDB installed. Instances:
MSSQLLocalDB (Version: Microsoft (R) SQL Server Express LocalDB Command Line Tool - Version 17.0.2254.1 - Copyright (c) Microsoft Corporation)
```

GUI-based PowerShell tool for diagnosing **SSDT, SSRS, SSIS, and SSAS** installations. It uses **Windows Forms** to show a clear, interactive report with version numbers.

PowerShell GUI Diagnostic Tool for SSDT + BI Components

```
Add-Type -AssemblyName System.Windows.Forms
Add-Type -AssemblyName System.Drawing

# -----
# Create Form
# -----
$form = New-Object System.Windows.Forms.Form
$form.Text = "SSDT + BI Components Diagnostic"
$form.Size = New-Object System.Drawing.Size(700,500)
$form.StartPosition = "CenterScreen"

# -----
# Create RichTextBox for output
# -----
$richBox = New-Object System.Windows.Forms.RichTextBox
$richBox.Multiline = $true
$richBox.ReadOnly = $true
$richBox.Dock = "Fill"
$richBox.Font = New-Object System.Drawing.Font("Consolas",10)
$richBox.BackColor = [System.Drawing.Color]::White
$form.Controls.Add($richBox)

# -----
# Function to write colored text
# -----
function Write-OutputBox {
    param (
        [string]$Message,
        [string]$Color = "Black"
    )

    $richBox.SelectionColor = [System.Drawing.Color]::$Color
    $richBox.AppendText("`$Message`r`n")
    $richBox.SelectionColor = [System.Drawing.Color]::Black
}

# -----
# Detect VS workloads (SSDT)
# -----
function Get-VSWorkloadVersion {
    param([string]$WorkloadID,[string]$Description)

    $vswhere = "${env:ProgramFiles(x86)}\Microsoft Visual Studio\Installer\vswhere.exe"
    if (-not (Test-Path $vswhere)) {
        Write-OutputBox "[△] vswhere.exe not found. Cannot check $Description" "Red"
        return
    }

    $vsInstall = & $vswhere -latest -products * -requires $WorkloadID -property installationPath
    if ($vsInstall) {
```

```

$devenvPath = Join-Path $vsInstall "Common7\IDE\devenv.exe"
if (Test-Path $devenvPath) {
    $version = (Get-Item $devenvPath).VersionInfo.ProductVersion
    Write-OutputBox "[✓] $Description installed at $vsInstall (Version: $version)" "Green"
} else {
    Write-OutputBox "[✗] $Description installed at $vsInstall (Version not found)" "Green"
}
} else {
    Write-OutputBox "[✗] $Description not installed" "Red"
}
}

# -----
# Detect VSIX extensions (SSIS/SSRS/SSAS)
# -----
function Get-VSIXExtensionVersion {
    param([string]$ExtensionName,[string]$ProjectTemplate)

    $extensionsFolder = "$env:LOCALAPPDATA\Microsoft\VisualStudio"
    $matches = Get-ChildItem -Path $extensionsFolder -Recurse -Filter "*.{vsixmanifest}" -ErrorAction SilentlyContinue |
        Where-Object { $_.Name -match "$ExtensionName" }

    if ($matches) {
        foreach ($match in $matches) {
            try {
                [xml]$xml = Get-Content $match.FullName
                $version = $xml.PackageManifest.Metadata.Identity.Version
                Write-OutputBox "[✓] $ExtensionName ($ProjectTemplate) installed, Version: $version" "Green"
            } catch {
                Write-OutputBox "[✗] $ExtensionName ($ProjectTemplate) installed, version unreadable" "Orange"
            }
        }
    } else {
        Write-OutputBox "[✗] $ExtensionName ($ProjectTemplate) not found" "Red"
    }
}

# -----
# Check SQL LocalDB
# -----
function Check-LocalDB {
    if (Get-Command "sqllocaldb" -ErrorAction SilentlyContinue) {
        $instances = & sqllocaldb i
        Write-OutputBox "[✓] LocalDB installed. Instances:" "Green"

        foreach ($instance in $instances) {
            $version = & sqllocaldb v $instance
            Write-OutputBox " - $instance (Version: $version)"
        }
    } else {
        Write-OutputBox "[✗] LocalDB not installed" "Red"
    }
}

# -----
# Button to Run Diagnostics

```

```

# -----
$button = New-Object System.Windows.Forms.Button
$button.Text = "Run Diagnostics"
$button.Dock = "Top"
$button.Height = 40

$button.Add_Click({
    $richBox.Clear()
    Write-OutputBox "Starting SSDT + BI Components Diagnostic..." `n" "Blue"

    # SSDT Workload
    Get-VSWorkloadVersion -WorkloadID "Microsoft.VisualStudio.Workload.Data" -Description "SQL Server Data Tools (SSDT)"

    # SSRS extension
    Get-VSIXExtensionVersion -ExtensionName "ReportingServicesProjects" -ProjectTemplate "SSRS"

    # SSAS extension
    Get-VSIXExtensionVersion -ExtensionName "AnalysisServicesProjects" -ProjectTemplate "SSAS"

    # SSIS extension
    Get-VSIXExtensionVersion -ExtensionName "SSIS" -ProjectTemplate "SSIS"

    # LocalDB
    Check-LocalDB

    Write-OutputBox ``nDiagnostic Complete!" "Blue"
})

$form.Controls.Add($button)

# -----
# Show Form
# -----
[void]$form.ShowDialog()

```

How It Works

1. Opens a GUI window with a **text box** and a **Run Diagnostics button**.
2. Clicking the button performs:
 - o SSDT workload check + version
 - o SSRS, SSAS, SSIS extensions + versions
 - o LocalDB instances + versions
3. Outputs **results in the GUI** with clear symbols:
 - o  Installed
 - o  Missing
 - o  Partial info

Usage Instructions

1. Save the script as: SSDT-BI-DiagnosticGUI.ps1.
2. Open **PowerShell as Administrator**.
3. Run:

\SSDT-BI-DiagnosticGUI.ps1
4. Click **Run Diagnostics** in the GUI window.
5. Review the results immediately in the textbox.

Sample output:

```

Administrator: Windows PowerShell
>> # SSAS extension
>> Get-VSIXExtensionVersion -ExtensionName "AnalysisServicesProjects" -ProjectTemplate "SSAS"
>>
>> # SSIS extension
>> Get-VSIXExtensionVersion -ExtensionName "SSIS" -ProjectTemplate "SSIS"
>>
>> # LocalDB
>> Check-LocalDB
>>
>> Write-OutputBox "nDiagnostic C
>> })
PS C:\Windows\system32>
PS C:\Windows\system32> $form.Controls.
PS C:\Windows\system32> # -----
PS C:\Windows\system32> # Show Form
PS C:\Windows\system32> #
PS C:\Windows\system32> [void]$form.Show()
[+] LocalDB installed. Instances:
  - MSSQLLocalDB (Version: Microsoft (R) SQL Server Express LocalDB Command Line Tool Version 17.0.925.4
Copyright (c) Microsoft Corporation. All rights reserved. Usage: SqlLocalDB operation [parameters...]
Operations:      -?      Prints this information      create|c ["instance name" [version-number] [-s]]
Creates a new LocalDB instance with a specified name and version. If the [version-number] parameter is
omitted, it defaults to the latest LocalDB version installed in the system.      -s starts the new
LocalDB instance after it's created      delete|d ["instance name"]      Deletes the LocalDB instance with the
specified name      start|s ["instance name"]      Starts the LocalDB instance with the specified name
stop|p ["instance name" [-i|-k]]      Stops the LocalDB instance with the specified name, after current
queries finish      -i request LocalDB instance shutdown with NOWAIT option      -k kills LocalDB instance
process without contacting it      share|h ["owner SID or account"] "private name" "shared name"      Shares
the specified private instance using the specified shared name.      If the user SID or account name is
omitted, it defaults to current user.      unshare|u ["shared name"]      Stops the sharing of the specified
shared LocalDB instance.      info|i      Lists all existing LocalDB instances owned by the current user
and all shared LocalDB instances.      info|i "instance name"      Prints the information about the specified
LocalDB instance.      versions|v      Lists all LocalDB versions installed on the computer.      trace|t
on|off      Turns tracing on and off      SqlLocalDB treats spaces as delimiters. It is necessary to surround
instance names that contain spaces and special characters with quotes. For example:      SqlLocalDB create
"My LocalDB Instance"      The instance name can sometimes be omitted, as indicated above, or specified as "".
In this case, the reference is to the default LocalDB instance "MSSQLLocalDB". )
Diagnostic Complete!

```

GUI diagnostic tool: **SSDT + BI GUI Diagnostic Tool**

```
Add-Type -AssemblyName System.Windows.Forms
Add-Type -AssemblyName System.Drawing

$form = New-Object System.Windows.Forms.Form
$form.Text = "SSDT + BI Components Diagnostic"
$form.Size = New-Object System.Drawing.Size(800,600)
$form.StartPosition = "CenterScreen"
$form.BackColor = [System.Drawing.Color]::FromArgb(30,30,30)

$textBox = New-Object System.Windows.Forms.TextBox
$textBox.Multiline = $true
$textBox.ScrollBars = "Vertical"
$textBox.ReadOnly = $true
$textBox.Dock = "Fill"
$textBox.Font = New-Object System.Drawing.Font("Consolas",10)
$textBox.BackColor = [System.Drawing.Color]::FromArgb(40,40,40)
$textBox.ForeColor = [System.Drawing.Color]::White
$form.Controls.Add($textBox)

$panel = New-Object System.Windows.Forms.Panel
$panel.Dock = "Top"
$panel.Height = 50
$panel.BackColor = [System.Drawing.Color]::FromArgb(50,50,50)
$form.Controls.Add($panel)

$script:OutputResults = @()

function Write-OutputBox($message) {
    $script:OutputResults += $message
    if ($form.InvokeRequired) {
        $form.Invoke({ $textBox.AppendText("$message`r`n") }) | Out-Null
    } else {
        $textBox.AppendText("$message`r`n")
    }
}

function Get-VSWorkloadVersion {
    param([string]$WorkloadID,[string]$Description)
    try {
        $vswhere = Join-Path "${env:ProgramFiles(x86)}" "Microsoft Visual Studio\Installer\vswhere.exe"
        if (-not (Test-Path $vswhere)) {
            Write-OutputBox "[△] vswhere.exe not found. Cannot check $Description"
            return
        }

        $vsInstall = & $vswhere -latest -products * -requires $WorkloadID -property installationPath
        if ($vsInstall) {
            $devenvPath = Join-Path $vsInstall "Common7\IDE\devenv.exe"
            if (Test-Path $devenvPath) {
                $version = (Get-Item $devenvPath).VersionInfo.ProductVersion
                Write-OutputBox "[✓] $Description installed at $vsInstall (Version: $version)"
            } else {
        }
    }
}
```

```

        Write-OutputBox "[✓] $Description installed at $vsInstall (Version not found)"
    }
} else {
    Write-OutputBox "[✗] $Description not installed"
}
} catch {
    Write-OutputBox "[✗] Error checking $Description: $_"
}
}

function Get-VSIXExtensionVersion {
param([string]$ExtensionName,[string]$ProjectTemplate)
try {
    $extensionsFolder = "$env:LOCALAPPDATA\Microsoft\VisualStudio"
    $matches = Get-ChildItem -Path $extensionsFolder -Recurse -ErrorAction SilentlyContinue |
        Where-Object { $_.Name -match "$ExtensionName" -and $_.Extension -eq ".vsixmanifest" }

    if ($matches) {
        foreach ($match in $matches) {
            try {
                [xml]$xml = Get-Content $match.FullName
                $version = $xml.PackageManifest.Metadata.Identity.Version
                Write-OutputBox "[✓] $ExtensionName ($ProjectTemplate) installed, Version: $version"
            } catch {
                Write-OutputBox "[✗] $ExtensionName ($ProjectTemplate) installed, Version not found"
            }
        }
    } else {
        Write-OutputBox "[✗] $ExtensionName ($ProjectTemplate) not installed"
    }
} catch {
    Write-OutputBox "[✗] Error checking $ExtensionName ($ProjectTemplate): $_"
}
}

function Check-LocalDB {
try {
    if (Get-Command "sqllocaldb" -ErrorAction SilentlyContinue) {
        $instances = & sqllocaldb i
        Write-OutputBox "[✓] LocalDB installed. Instances:"
        foreach ($instance in $instances) {
            $version = & sqllocaldb v $instance
            Write-OutputBox "  $instance (Version: $version)"
        }
    } else {
        Write-OutputBox "[✗] LocalDB not installed"
    }
} catch {
    Write-OutputBox "[✗] Error checking LocalDB: $_"
}
}

# -----
# Button: Run Diagnostics only
# -----

```

```
$btnRun = New-Object System.Windows.Forms.Button
$btnRun.Text = "Run Diagnostics"
$btnRun.Width = 150
$btnRun.Left = 10
$btnRun.Top = 10
$btnRun.BackColor = [System.Drawing.Color]::FromArgb(70,70,70)
$btnRun.ForeColor = [System.Drawing.Color]::White
$btnRun.Add_Click({
    $textBox.Clear()
    $script:OutputResults = @()
    Write-OutputBox "Starting SSDT + BI Components Diagnostic...`n"

    Get-VSWorkloadVersion -WorkloadID "Microsoft.VisualStudio.Workload.Data" -Description "SQL Server Data Tools (SSDT)"
    Get-VSIXExtensionVersion -ExtensionName "ReportingServicesProjects" -ProjectTemplate "SSRS"
    Get-VSIXExtensionVersion -ExtensionName "AnalysisServicesProjects" -ProjectTemplate "SSAS"
    Get-VSIXExtensionVersion -ExtensionName "SSISProjects" -ProjectTemplate "SSIS"
    Check-LocalDB

    Write-OutputBox "`nDiagnostic Complete!"
})
$panel.Controls.Add($btnRun)

[void]$form.ShowDialog()
```

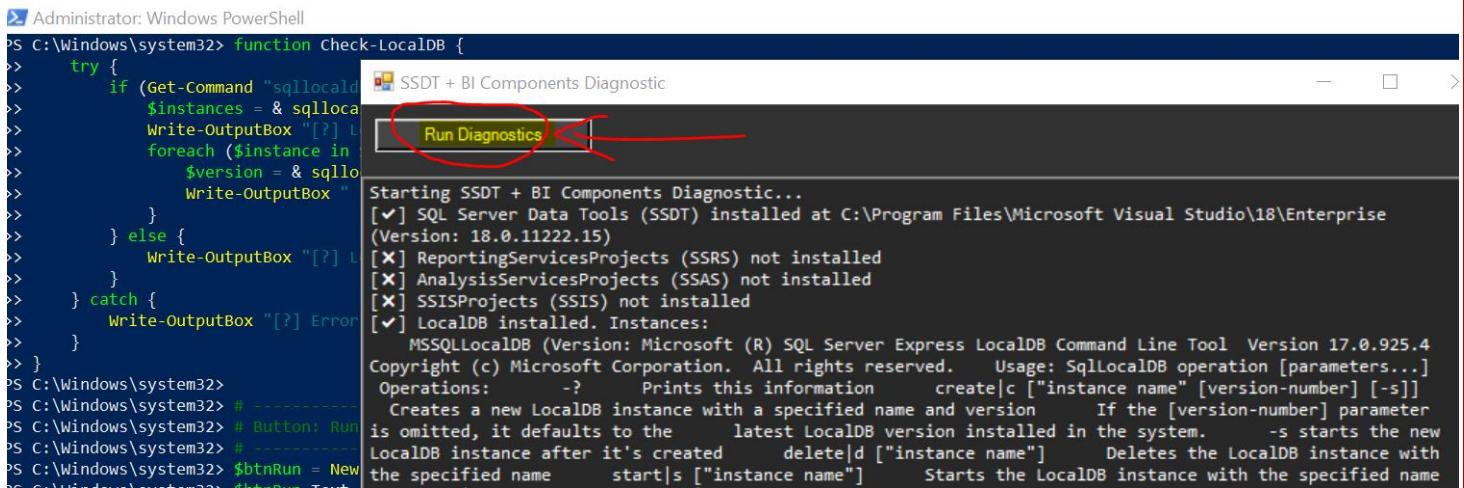
Key Features

1. **GUI-based:** TextBox displays results interactively.
2. **Run Diagnostics Button:** Checks SSDT, SSRS, SSIS, SSAS, LocalDB + versions.

Instructions

1. Save as: SSDT-BI-Diagnostic-GUI.ps1.
2. Open **PowerShell as Administrator**.
3. Run the script:
.\SSDT-BI-Diagnostic-GUI.ps1
5. Click **Run Diagnostics** button.

Sample Output:



```
Administrator: Windows PowerShell
PS C:\Windows\system32> function Check-LocalDB {
>>>     try {
>>>         if (Get-Command "sqllocaldb") {
>>>             $instances = & sqllocaldb list
>>>             Write-OutputBox "[?] LocalDB instances"
>>>             foreach ($instance in $instances) {
>>>                 $version = & sqllocaldb info $instance | select-object -expandproperty version
>>>                 Write-OutputBox "[$version] $instance"
>>>             }
>>>         } else {
>>>             Write-OutputBox "[?] LocalDB not installed"
>>>         }
>>>     } catch {
>>>         Write-OutputBox "[?] Error"
>>>     }
>>> }
PS C:\Windows\system32>
PS C:\Windows\system32> # -----
PS C:\Windows\system32> # Button: Run
PS C:\Windows\system32> # -----
PS C:\Windows\system32> $btnRun = New-
```

SSDT + BI GUI Diagnostic with below details:

1. **Header & Timestamp**
2. **Colored status:**
 - Green = Installed
 - Red = Missing
 - Black = Info / general messages

Powershell Code:

```
# -----
# Load Required Assemblies
# -----
Add-Type -AssemblyName System.Windows.Forms
Add-Type -AssemblyName System.Drawing
# System.Drawing.Printing is included in System.Drawing.dll, no need to add separately

# -----
# Create Form
# -----
$form = New-Object System.Windows.Forms.Form
$form.Text = "SSDT + BI Components Diagnostic"
$form.Size = New-Object System.Drawing.Size(900,600)
$form.StartPosition = "CenterScreen"

# -----
# RichTextBox for Output
# -----
$textBox = New-Object System.Windows.Forms.RichTextBox
$textBox.Multiline = $true
$textBox.ScrollBars = "Vertical"
$textBox.ReadOnly = $true
$textBox.Dock = "Fill"
$textBox.Font = New-Object System.Drawing.Font("Consolas",10)
$form.Controls.Add($textBox)

# -----
# Panel for Buttons
# -----
$panel = New-Object System.Windows.Forms.Panel
$panel.Dock = "Top"
$panel.Height = 50
$form.Controls.Add($panel)

# -----
# Global Output Array
# -----
$global:OutputResults = @()

# -----
# Helper Functions
# -----
function Write-OutputBox {
    param([string]$message, [string]$status="info")
    switch ($status) {
```

```

"ok" { $color = [System.Drawing.Color]::Green }
"fail" { $color = [System.Drawing.Color]::Red }
default { $color = [System.Drawing.Color]::Black }
}
$textBox.SelectionStart = $textBox.TextLength
$textBox.SelectionLength = 0
$textBox.SelectionColor = $color
$textBox.AppendText("$message`r`n")
$textBox.SelectionColor = $textBox.ForeColor
$global:OutputResults += @{Text=$message; Status=$status}
}

function Get-VSWorkloadVersion {
    param([string]$WorkloadID,[string]$Description)
    $vswhere = "${env:ProgramFiles(x86)}\Microsoft Visual Studio\Installer\vswhere.exe"
    if (-not (Test-Path $vswhere)) {
        Write-OutputBox "[△] vswhere.exe not found. Cannot check $Description" "fail"
        return
    }
    $vsInstall = & $vswhere -latest -products * -requires $WorkloadID -property installationPath
    if ($vsInstall) {
        $devenvPath = Join-Path $vsInstall "Common7\IDE\devenv.exe"
        if (Test-Path $devenvPath) {
            $version = (Get-Item $devenvPath).VersionInfo.ProductVersion
            Write-OutputBox "[✓] $Description installed at $vsInstall (Version: $version)" "ok"
        } else {
            Write-OutputBox "[✓] $Description installed at $vsInstall (Version not found)" "ok"
        }
    } else {
        Write-OutputBox "[✗] $Description not installed" "fail"
    }
}

function Get-VSIXExtensionVersion {
    param([string]$ExtensionName,[string]$ProjectTemplate)
    $extensionsFolder = "$env:LOCALAPPDATA\Microsoft\VisualStudio"
    $matches = Get-ChildItem -Path $extensionsFolder -Recurse -ErrorAction SilentlyContinue |
        Where-Object { $_.Name -match "$ExtensionName" -and $_.Extension -eq ".vsixmanifest" }
    if ($matches) {
        foreach ($match in $matches) {
            try {
                [xml]$xml = Get-Content $match.FullName
                $version = $xml.PackageManifest.Metadata.Identity.Version
                Write-OutputBox "[✓] $ExtensionName ($ProjectTemplate) installed, Version: $version" "ok"
            } catch {
                Write-OutputBox "[✓] $ExtensionName ($ProjectTemplate) installed, Version not found" "ok"
            }
        }
    } else {
        Write-OutputBox "[✗] $ExtensionName ($ProjectTemplate) not installed" "fail"
    }
}

function Check-LocalDB {
    if (Get-Command "sqllocaldb" -ErrorAction SilentlyContinue) {
        $instances = & sqllocaldb i
        Write-OutputBox "[✓] LocalDB installed. Instances:" "ok"
        foreach ($instance in $instances) {
            $version = & sqllocaldb v $instance
    }
}

```

```

        Write-OutputBox " $instance (Version: $version)" "ok"
    }
} else {
    Write-OutputBox "[X] LocalDB not installed" "fail"
}
}

# -----
# Run Diagnostics Button
# -----
$btnRun = New-Object System.Windows.Forms.Button
$btnRun.Text = "Run Diagnostics"
$btnRun.Width = 150
$btnRun.Left = 10
$btnRun.Top = 10
$btnRun.Add_Click({
    $textBox.Clear()
    $global:OutputResults = @()
    $timestamp = Get-Date -Format "yyyy-MM-dd HH:mm:ss"
    Write-OutputBox "```n==> SSDT + BI Components Diagnostic ==`n" "info"
    Write-OutputBox "Timestamp: $timestamp`n" "info"

    Get-VSWorkloadVersion -WorkloadID "Microsoft.VisualStudio.Workload.Data" -Description "SQL Server Data Tools (SSDT)"
    Get-VSIXExtensionVersion -ExtensionName "ReportingServicesProjects" -ProjectTemplate "SSRS"
    Get-VSIXExtensionVersion -ExtensionName "AnalysisServicesProjects" -ProjectTemplate "SSAS"
    Get-VSIXExtensionVersion -ExtensionName "SSISProjects" -ProjectTemplate "SSIS"
    Check-LocalDB

    Write-OutputBox "```n==> Diagnostic Complete ==`n" "info"
})
$panel.Controls.Add($btnRun)
# -----
# Show Form
# -----
[void]$form.ShowDialog()

```

Sample output:

