erwin Data Modeler

Git Support

Release 12.0

Legal Notices

This Documentation, which includes embedded help systems and electronically distributed materials (hereinafter referred to as the “Documentation”), is for your informational purposes only and is subject to change or withdrawal by Quest Software, Inc and/or its affiliates at any time. This Documentation is proprietary information of Quest Software, Inc and/or its affiliates and may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of Quest Software, Inc and/or its affiliates

If you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all Quest Software, Inc and/or its affiliates copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to Quest Software, Inc and/or its affiliates that all copies and partial copies of the Documentation have been returned to Quest Software, Inc and/or its affiliates or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, QUEST SOFTWARE, INC. PROVIDES THIS DOCUMENTATION “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL QUEST SOFTWARE, INC. BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF QUEST SOFTWARE, INC. IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is Quest Software, Inc and/or its affiliates.

Provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2022 Quest Software, Inc and/or its affiliates All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Contact erwin

Understanding your Support

Review support maintenance programs and offerings.

Registering for Support

Access the erwin support site and click **Sign in** or **Sign up** to register for product support.

Accessing Technical Support

For your convenience, erwin provides easy access to "One Stop" support for all editions of erwin Data Modeler, and includes the following:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- erwin Support policies and guidelines
- Other helpful resources appropriate for your product

For information about other erwin products, visit http://erwin.com/products.

Provide Feedback

If you have comments or questions, or feedback about erwin product documentation, you can send a message to techpubs@erwin.com.

erwin Data Modeler News and Events

Visit www.erwin.com to get up-to-date news, announcements, and events. View video demos and read up on customer success stories and articles by industry experts.

Contents

Git Support 6

Connecting to Git Repositories 7

Committing Forward Engineering Scripts 11

Scenario 1: Committing New or Full FE Scripts 11

Scenario 2: Committing Alter Scripts 20

Git Support

Starting erwin Data Modeler (DM) 12.0, you can connect erwin DM to Git repositories via Mart Server. This enables you to push Forward Engineering (FE) scripts for a Mart Model to GitLab or GitHub. You cannot store FE scripts or DDL on a Mart Server but only erwin models. With Git support you can adopt DevOps principles as you can commit FE scripts in Git repositories. Working with these repositories help you in:

- DevOps adoption
- collaboration with team members
- version control
- workflow management
- data integrity

Pushing FE scripts to a Git repository involves:

1. Connecting erwin DM to Mart Server
2. Connecting erwin DM to a Git repository
3. Opening a Mart Model and committing FE scripts

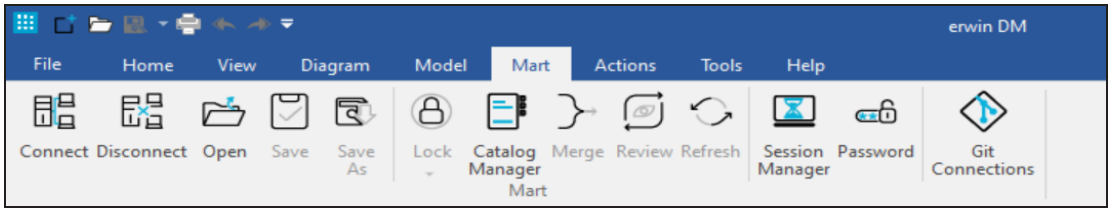
Connecting to Git Repositories

A Git repository may be hosted on GitLab or GitHub. For a successful connection to these repositories, you need to ensure that:

- erwin DM is connected to erwin Mart Server. For more information on connecting erwin DM to Mart Server, refer to the Connect to Mart topic.
- you have created the required personal access token. To know how to create personal access tokens for GitLab, refer to the GitLab documentation. To know how to create personal access tokens for GitHub, refer to the GitHub documentation.

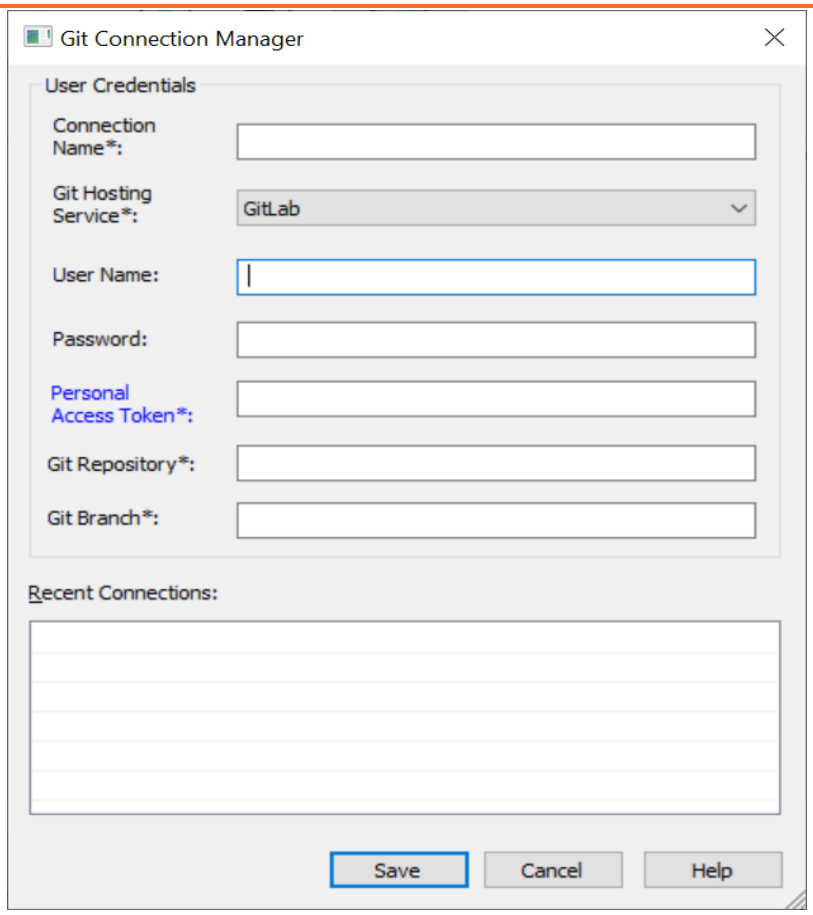
Once, these prerequisites are in place, to connect Git repositories to erwin DM, follow these steps:

1. On the ribbon, click **Mart**.



2. Click **Git Connections**.

   The Git Connection Manager page appears.



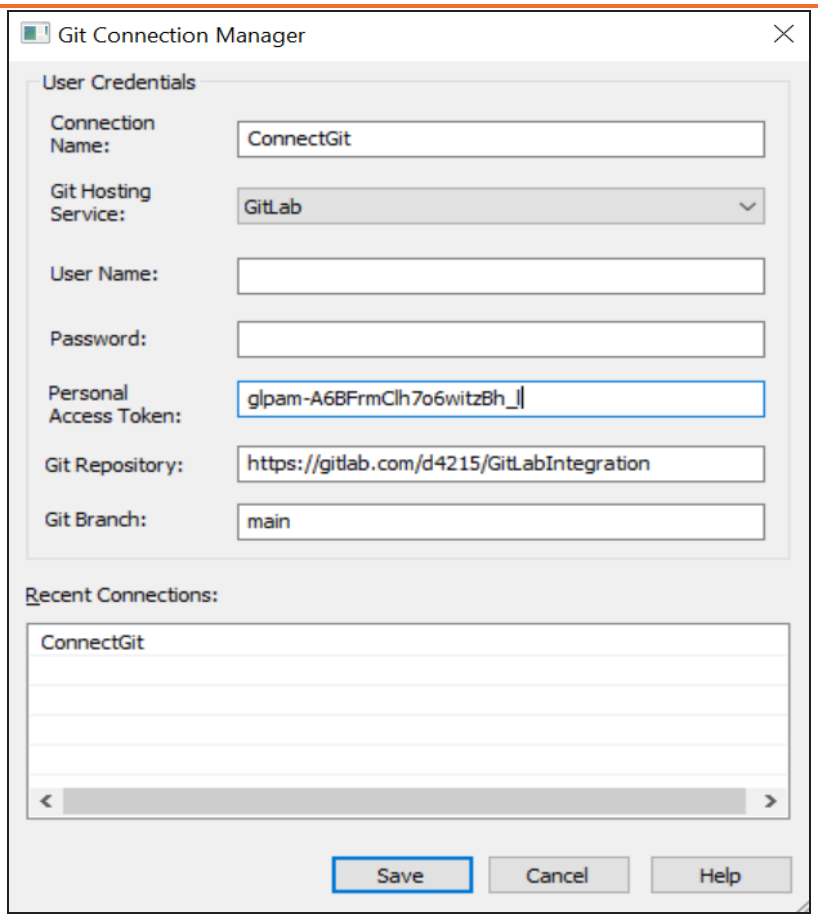
3. Enter appropriate values in the fields. Refer to the following table for field descriptions.

Field Name	Description	Additional Information
Connection Name	Specifies a user defined connection name	For example, ConnectGit. You can create multiple connections one for each Git repository.
Git Hosting Service	Specifies the Git hosting service to which erwin DM connects	**GitLab**: Indicates that erwin DM connects to GitLab **GitHub**: Indicates that erwin DM connects to GitHub
User Name	Specifies the username to	This field is not mandatory.

Field Name	Description	Additional Information
	log on to the Git hosting service	
Password	Specifies the password to log on to the Git hosting service	This field is not mandatory.
Personal Access Token	Specifies the personal access token to connect to the Git hosting service	
Git Repository	Specifies the URL of a Git repository where you want to push the forward engineering script	For example, https://-gitlab.com/d4215/GitLabIntegration
Git Branch	Specifies the branch that is used to push the forward engineering script	For example, main.

4. Click **Save**.

   On successful connection, the connection name appears under Recent Connections.



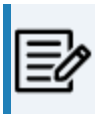
Once you are connected to a Git repository, you can commit FE scripts.

Committing Forward Engineering Scripts

There are two scenarios in which you commit Forward Engineering (FE) scripts to a Git repository:

- **Scenario 1: Committing new or full FE scripts**:

  Use the Forward Engineer Schema Generation Wizard to commit a physical database schema or FE script from a Mart Model.

  To avoid script files from being overwritten, ensure that you use unique file names.

- **Scenario 2: Committing alter scripts**:

  Use the Forward Engineer Alter Script Schema Generation Wizard to commit an alter script after you make changes to a Mart Model. You can commit an alter script in two ways:

  - **Commit and append an alter script to an existing script file**
  - **Commit and create a new alter script file in the Git repository**

  For more information, refer to the Scenario 2: Committing Alter Scripts topic.

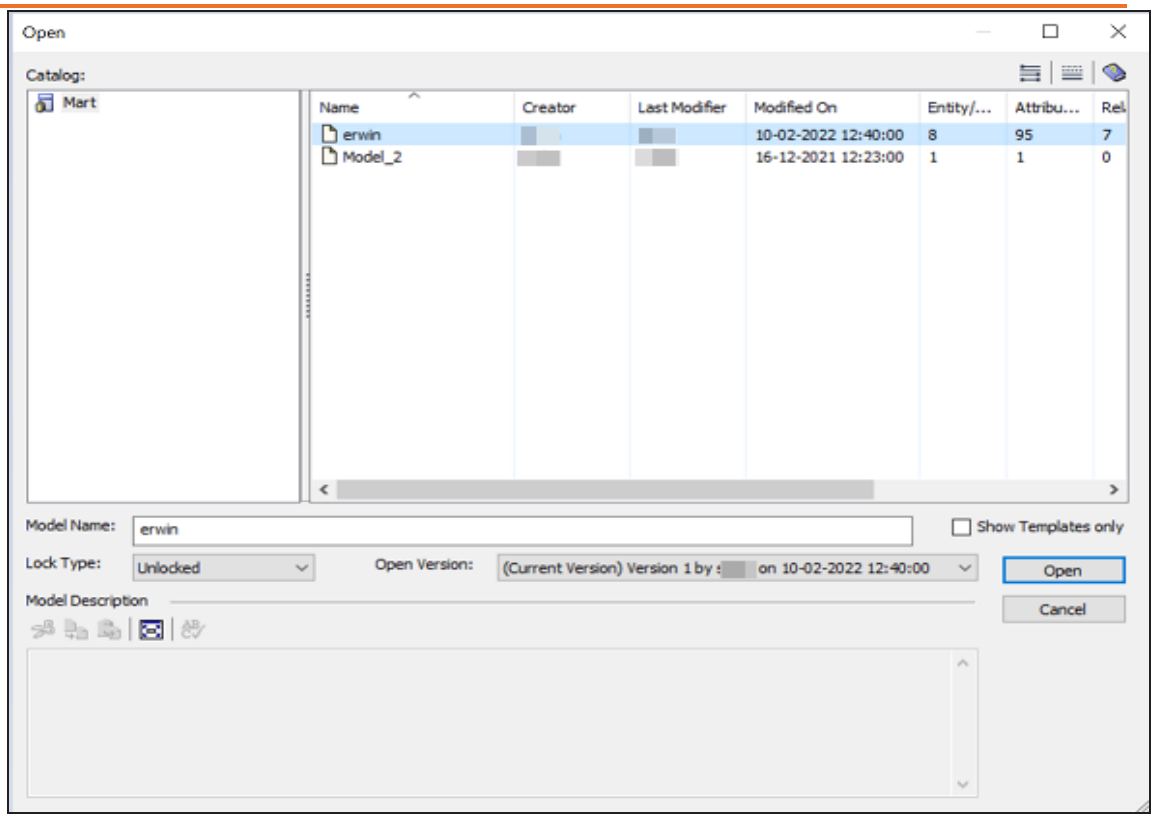
Scenario 1: Committing New or Full FE Scripts

The Forward Engineer Schema Generation Wizard generates a physical database schema or Forward Engineering (FE) script. For a Mart Model, you can push the FE script to a Git repository.

To commit new or full FE scripts to Git repositories, follow these steps:

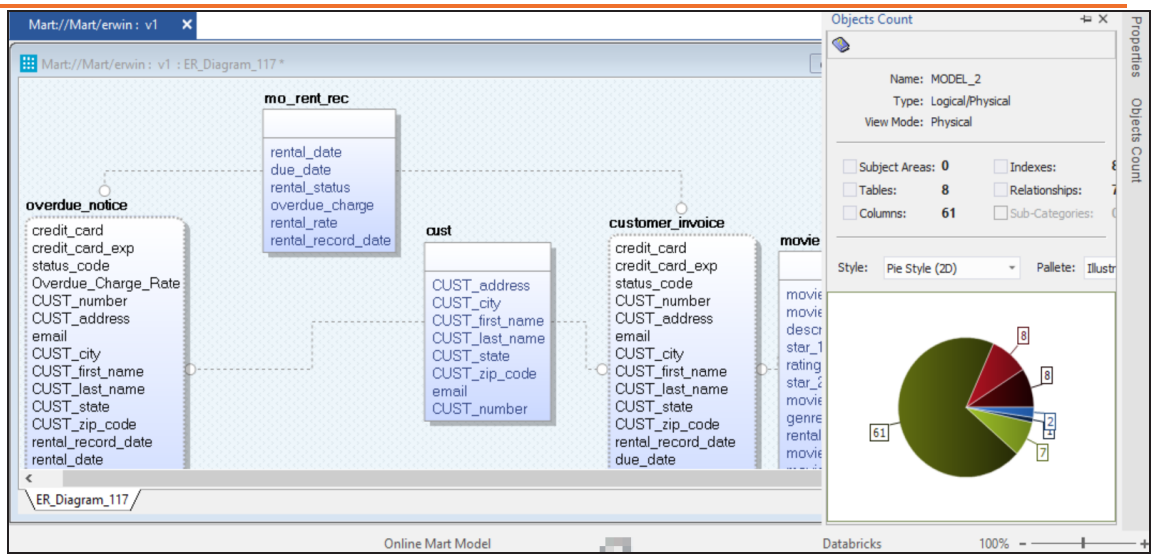
1. On the ribbon, go to **Mart** > **Open**.

   The Open page appears.



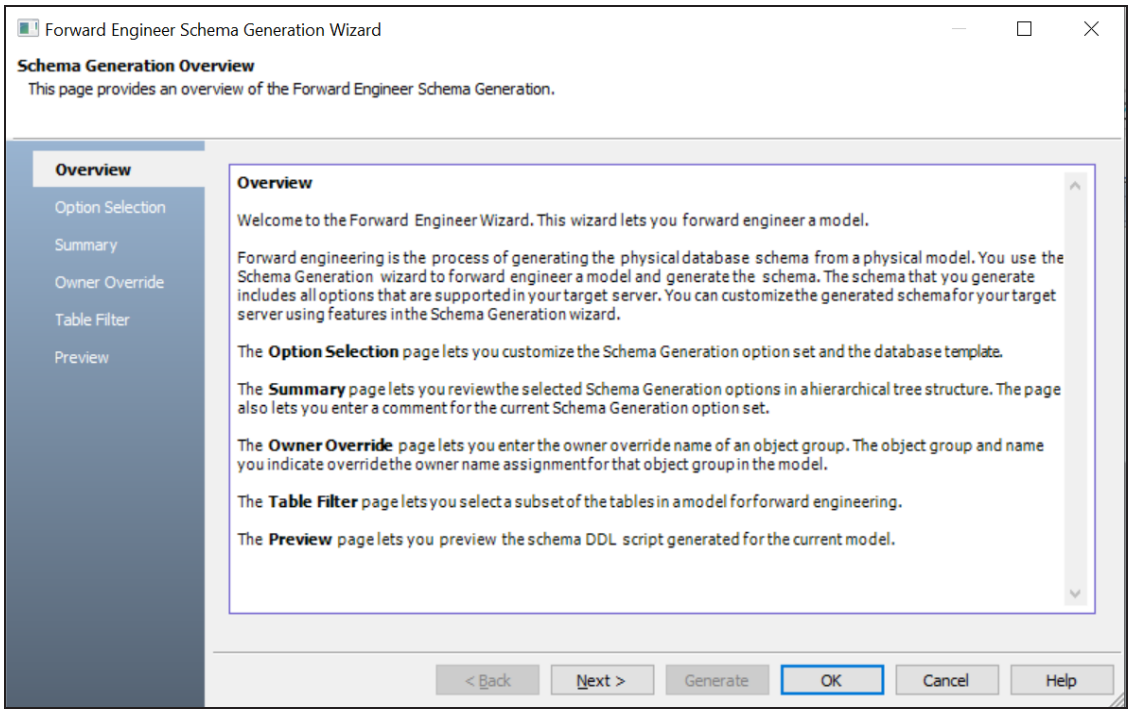
2. Select a model, and then click **Open**.

   The Mart Model opens.



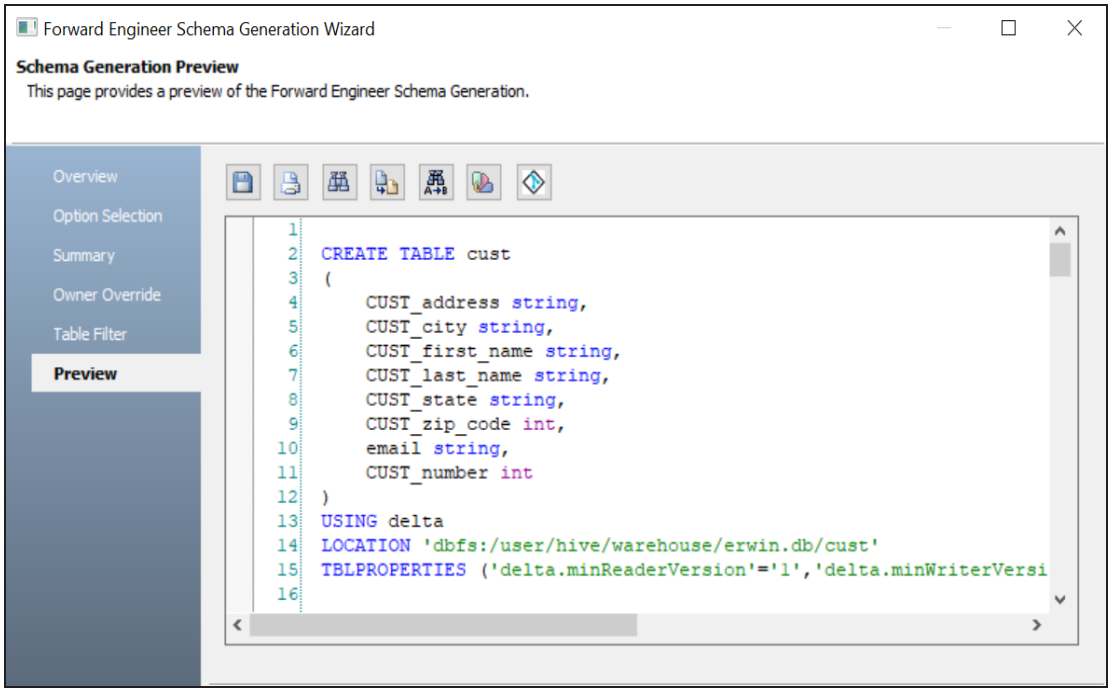
3. Go to **Actions** > **Schema**.

   The Forward Engineer Schema Generation Wizard appears.



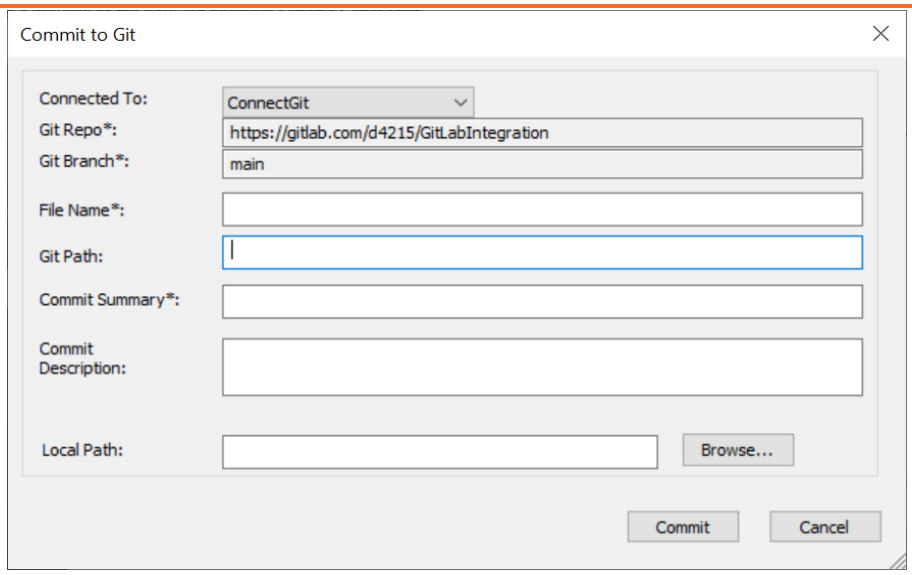
4. On the **Forward Engineer Schema Generation Wizard**, click the **Preview** section.

The FE script appears. For example, in the following image the Preview section displays FE script of a Databricks database. For more information on generating FE scripts, refer to the Forward Engineering/Schema Generation for Databases topic.



5. Click .

   The Commit to Git screen appears.



6. Enter appropriate values in the fields. Fields marked with an asterisk (*) are mandatory. Refer to the following table for field descriptions.

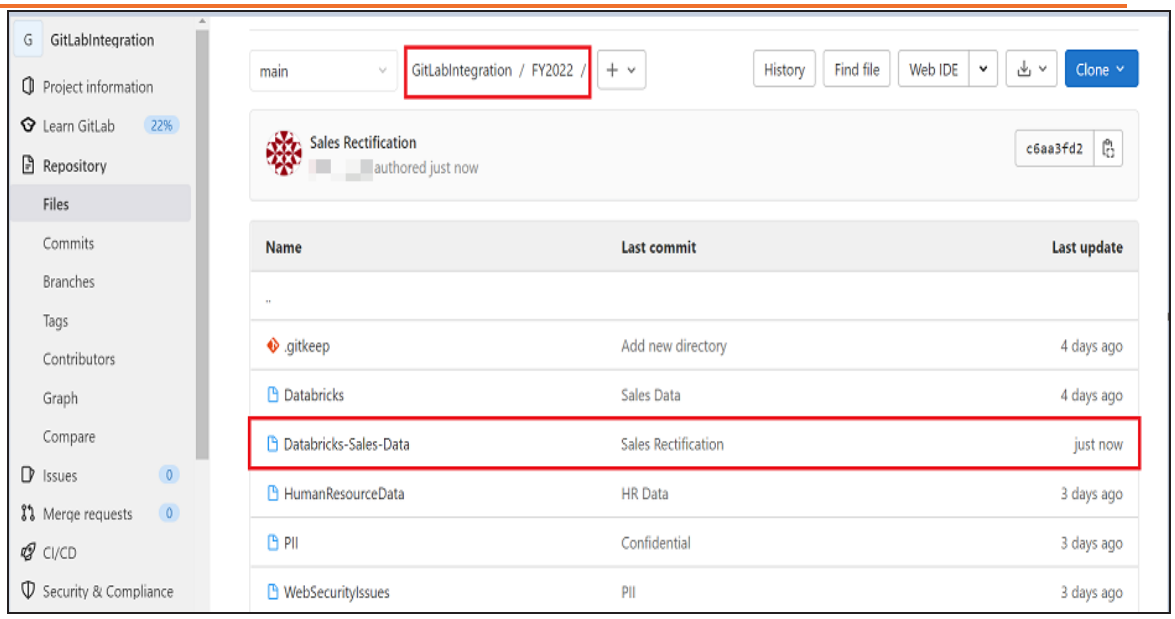
Field Name	Description	Additional Information
Connected To	Specifies the connection that connects erwin DM to a Git repository	For example, ConnectGit.
Git Repo	Specifies the Git repository configured for the connection	For example, https://-gitlab.com/d4215/GitLabIntegration is set for the ConnectGit connection. This field autopopulates based on the repository configured in the Git Connection Manager.
Git Branch	Specifies the Git branch configured for the connection	For example, main is set for the ConnectGit connection. This field autopopulates based on the repository configured in the Git Connection Manager.
File Name	Specifies the user-defined name of the FE script file	For example, Databricks-Sales-Data.sql

Field Name	Description	Additional Information
	being committed to a Git repository	To avoid script files from being overwritten, ensure that you use unique file names.
Git Path	Specifies the location in the Git repository where the FE script is committed	For example, FY2022/ The FE script is committed to the FY2022 folder inside the root folder of your Git repository.
Commit Summary	Specifies the summary of the push commit	For example, Sales Rectification.
Commit Description	Specifies the description of the push commit	For example, Forward schema script includes December sales data.
Local Path	Specifies the location on your local machine where the FE script is saved	C:\Users\SO\Documents\Databricks

7. Click **Commit**.

   The FE script file is saved on the local path and committed to the Git repository.

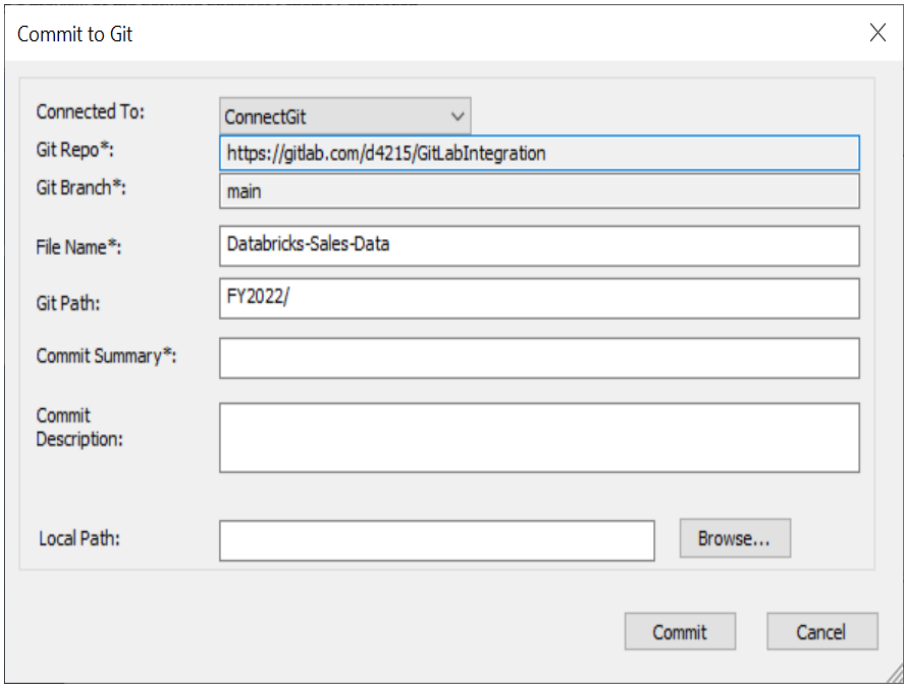
   For example, in the following image, FE script is committed to a GitLab repository in a file, Databricks-Sales-Data, with a commit summary, Sales Rectification using the main branch.



You can click the file to review its content. For example, in the following image, Databricks-Sales-Data's content is visible.

You can use FE Schema Generation Wizard to commit FE script using the same connection again. The Commit to Git screen autopopulates the previously set values in File Name and Git Path.

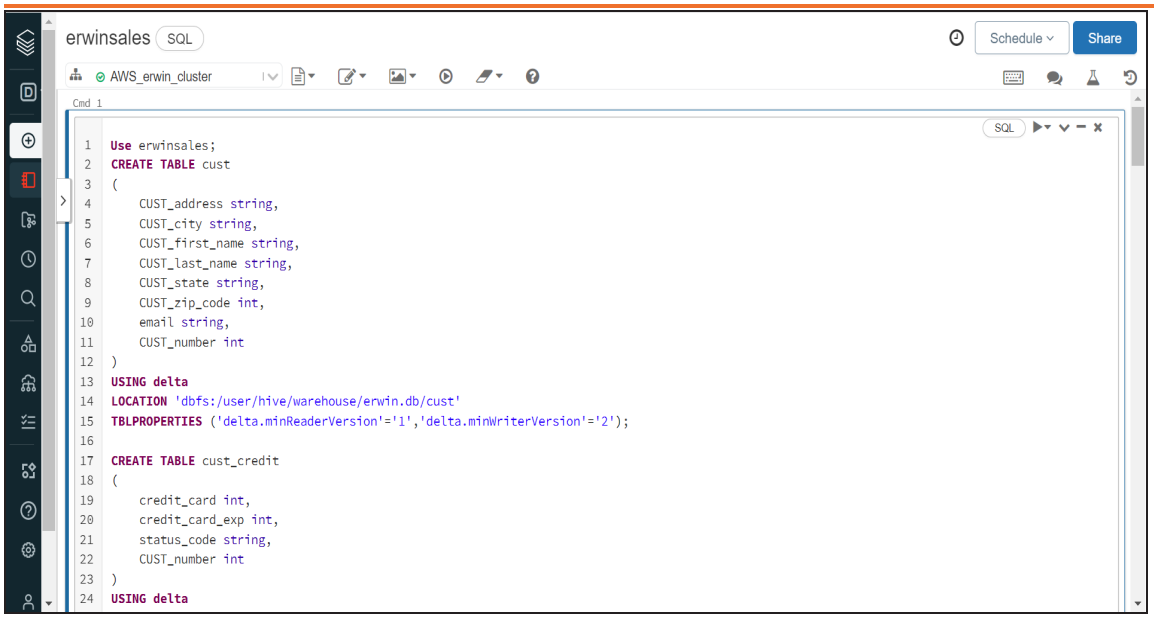
For example, in the following image File Name is set to Databricks-Sales-Data and Git Path is set to FY2022/.



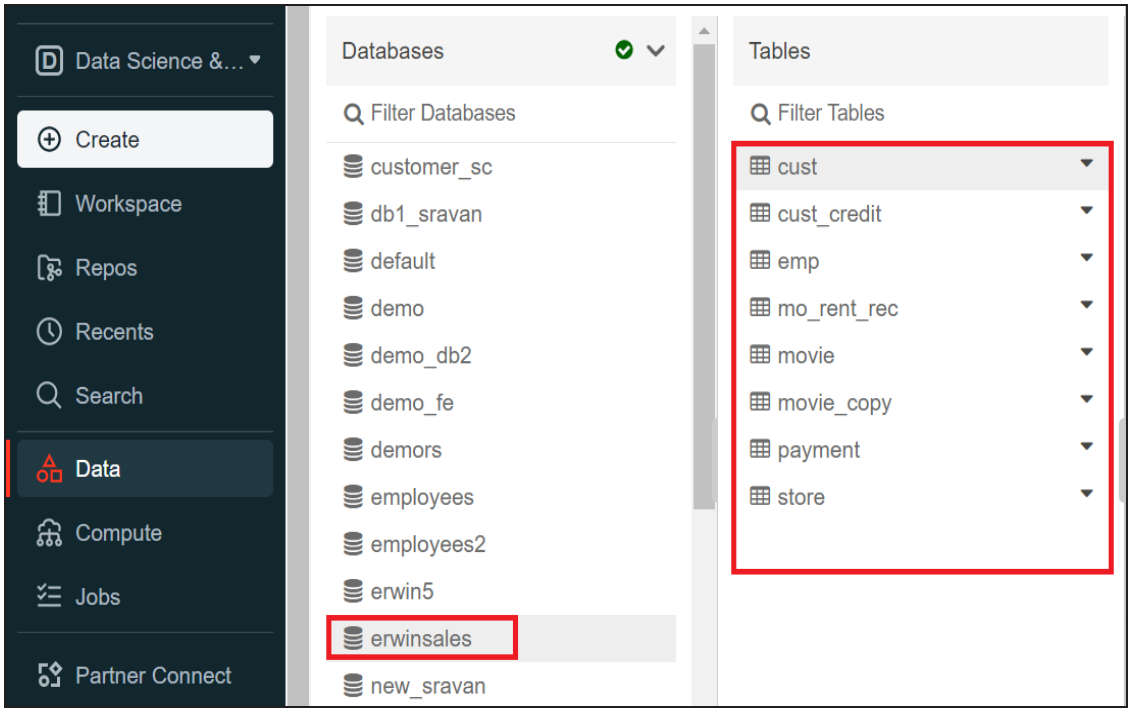
Committing the FE script again with the same File Name and Git Path overwrites the previous file in the Git repository.

Once the FE script is committed, you can run it on your database to generate and verify the physical schema.

For example, in the following Databricks database, the FE script copied from the Git repository is run.



After running the FE script, the required database objects are created. You can access these objects from the database. For example, the following tables can be accessed in a Databricks database.



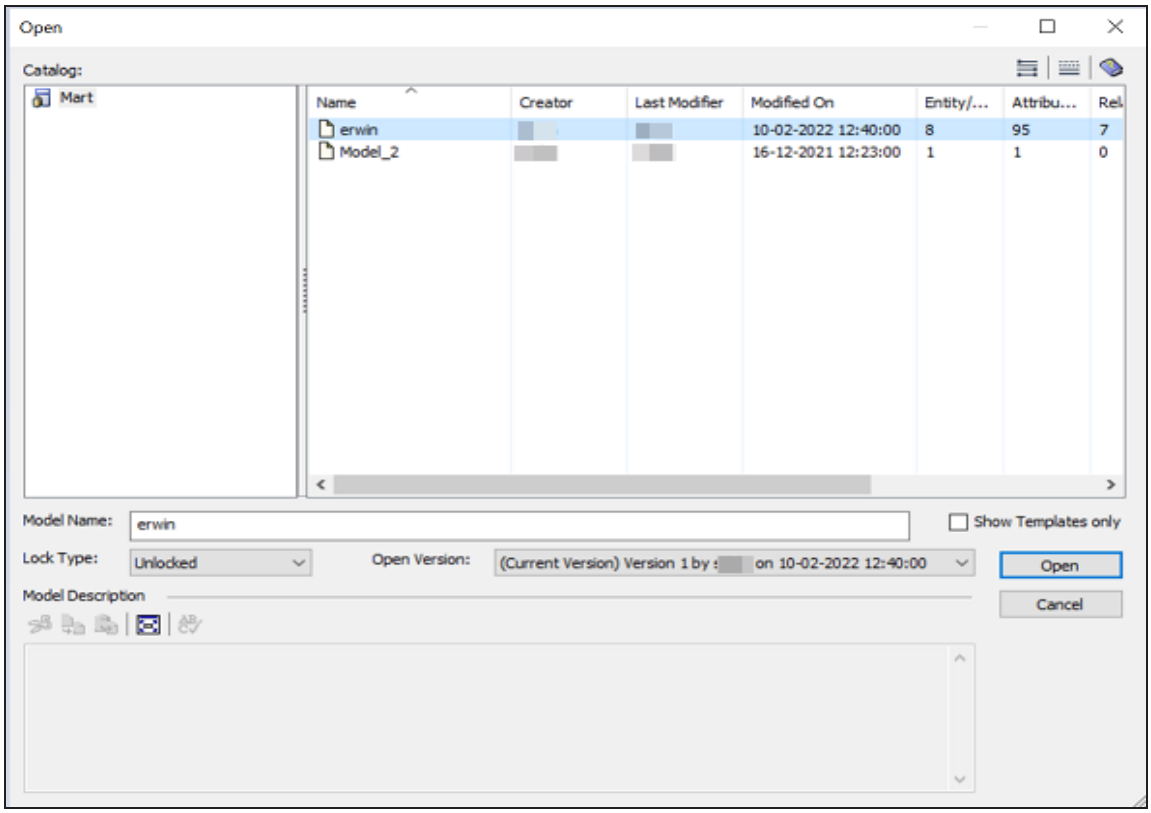
Scenario 2: Committing Alter Scripts

The Forward Engineer Alter Schema Generation Wizard generates an alter script for a database after you make changes to a model. For a Mart Model, you can push the alter script to a Git repository.

To commit alter scripts to Git repositories, follow these steps:

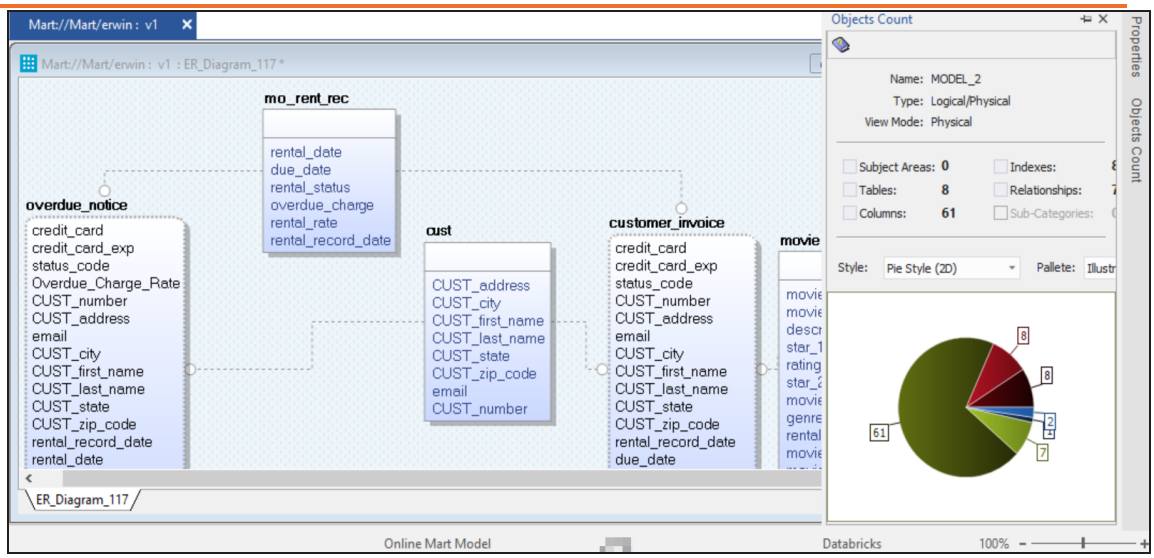
1. On the ribbon, go to **Mart** > **Open**

   The Open page appears.



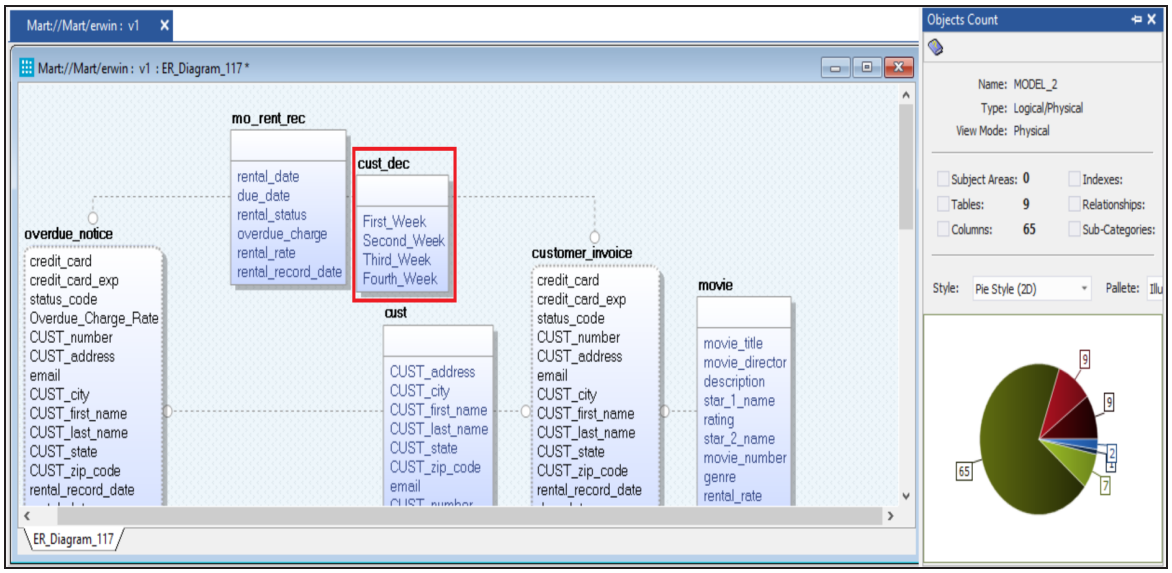
2. Select a model, and then click **Open**.

   The Mart Model opens.



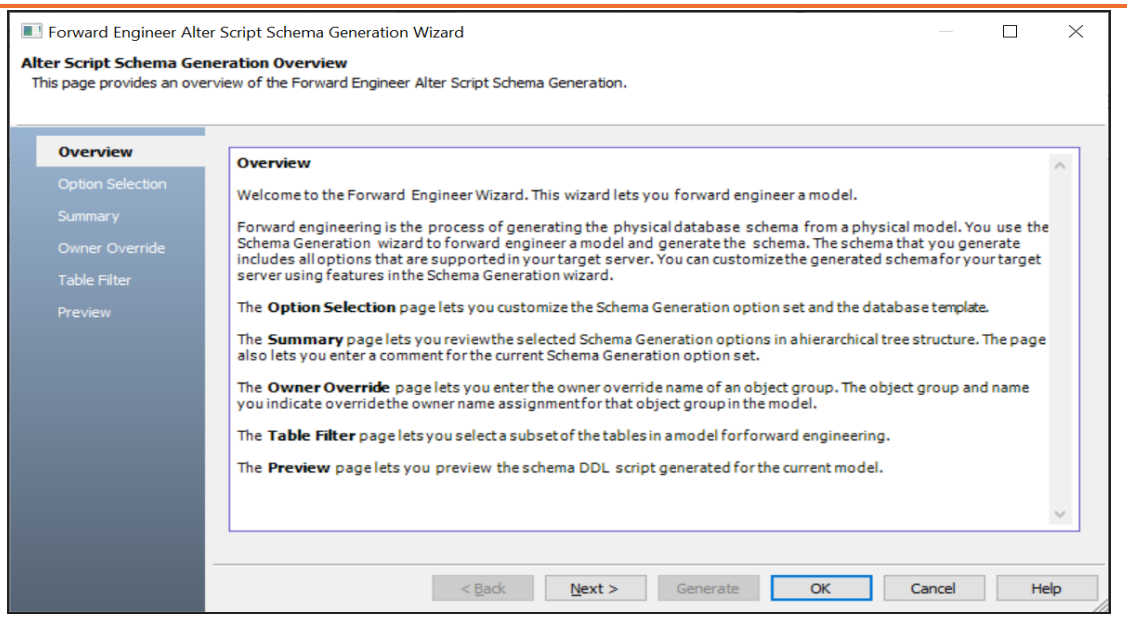
3. Make the required changes in the model.

   For example, in the following model, a new table, cust_dec with four columns is added.



4. Go to **Actions** > **Alter Script**.

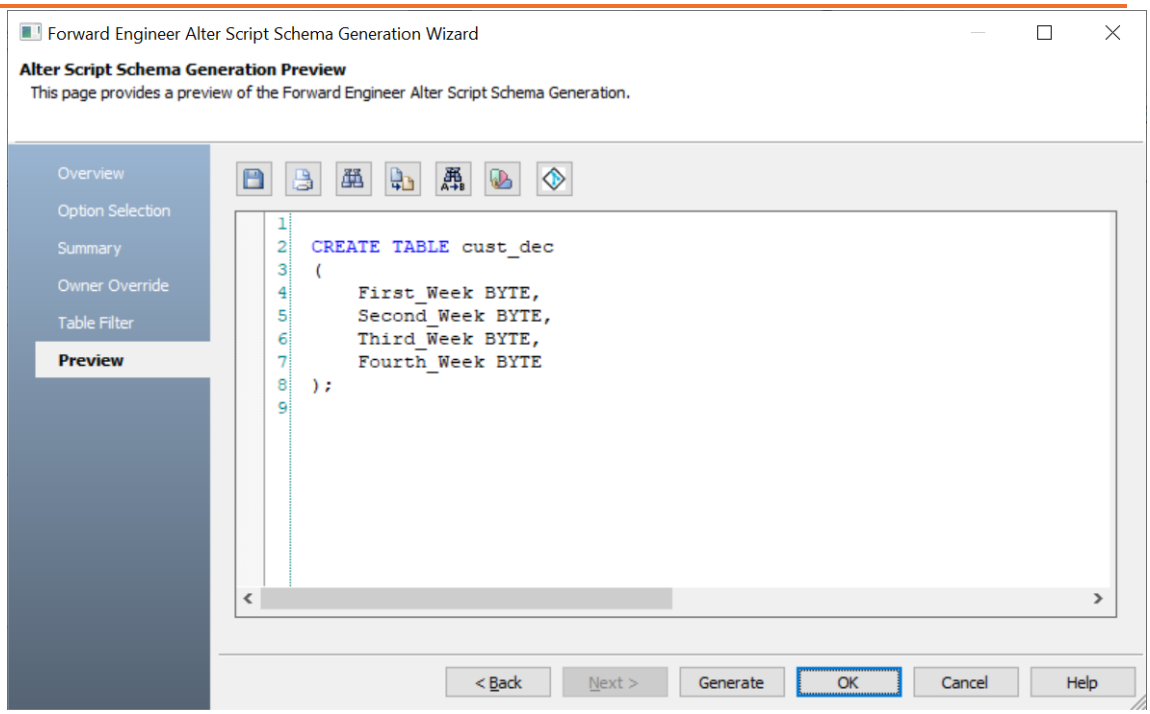
   The Forward Engineer Alter Script Schema Generation Wizard appears.



5. On the **Forward Engineer Alter Schema Generation Wizard**, click the **Preview** section.

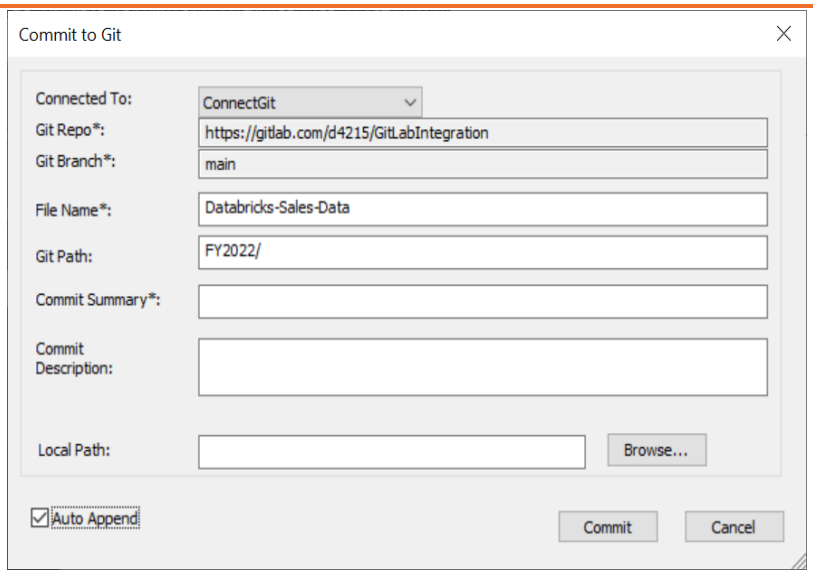
   The alter script appears. For more information on generating alter scripts, refer to the Generating Alter Script for Databases topic.

   For example, in the following image the Preview section displays an alter script of a Databricks database.



6. Click .

   The Commit to Git screen appears. The File Name and Git Path values autopopulates with the values configured in the previous commit. You can update the File Name and Git Path as per the requirement.



7. Enter appropriate values in the fields. Fields marked with an asterisk (*) are mandatory. Refer to the following table for field descriptions.

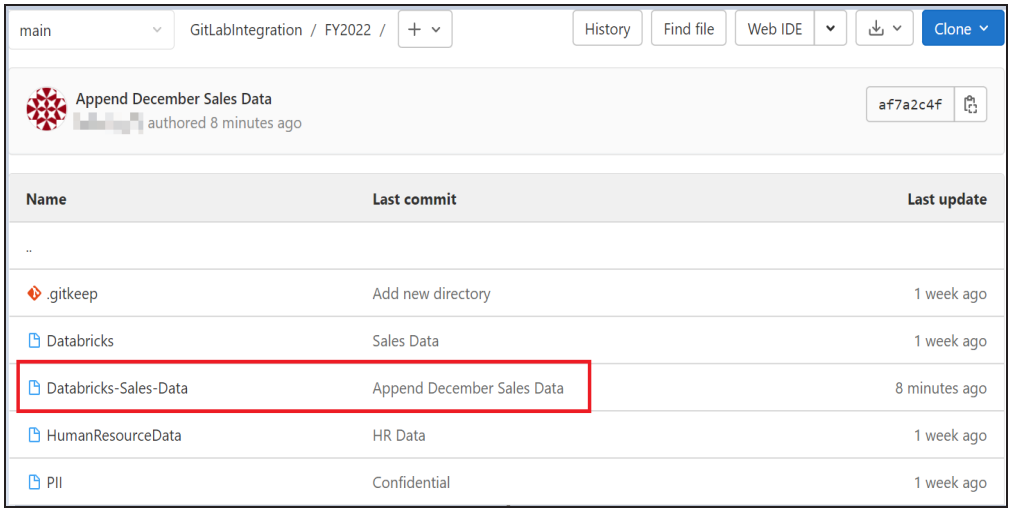
Field Name	Description	Additional Information
Connected To	Specifies the connection that connects erwin DM to a Git repository	For example, ConnectGit.
Git Repo	Specifies the Git repository configured for Connection	For example, https://-gitlab.com/d4215/GitLabIntegration is set for the ConnectGit connection. This field autopopulates based on the repository configured in the Git Connection Manager.
Git Branch	Specifies the Git branch that was set for connection in the Git Connection Manager	For example, main is set for the ConnectGit connection. This field autopopulates based on the repository configured in the Git Connection Manager.

Field Name	Description	Additional Information
File Name	Specifies the user-defined name of the FE script file being committed to a Git repository	For example, Databricks-Sales-Data.sql
Git Path	Specifies the location in the Git repository where the FE script is committed	For example, FY2022/ The FE script is committed to the FY2022 folder inside the root folder of your Git repository.
Commit Summary	Specifies the summary of the push commit	For example, Append December Sales.
Commit Description	Specifies the description of the push commit	For example, Forward schema script includes December sales data.
Local Path	Specifies the location on your local machine where the Alter script is saved	C:\Users\SO\Documents\Databricks
Auto Append	Specifies whether the alter script is appended to the file set in File Name and Git Path	By default, the Auto Append check box is selected. To create a new script file, clear the Auto Append check box and set the File Name and File Path belonging to an existing file. A new file with the following naming convention: <File Name>_YYYY-MM-DD_HH-MM-SS is created. Ensure that you use this check box consistently every time you commit an alter script.

8. Click **Commit**.

   The alter script file is saved on the local path and committed to the Git repository.

For example, in the following image, an alter script file is committed to a GitLab repository and appended to an existing file, Databricks-Sales-Data, with a commit summary, Append December Sales using the main branch.



You can click the file to review its content. For example, in the following image, Databricks-Sales-Data contains the alter script.

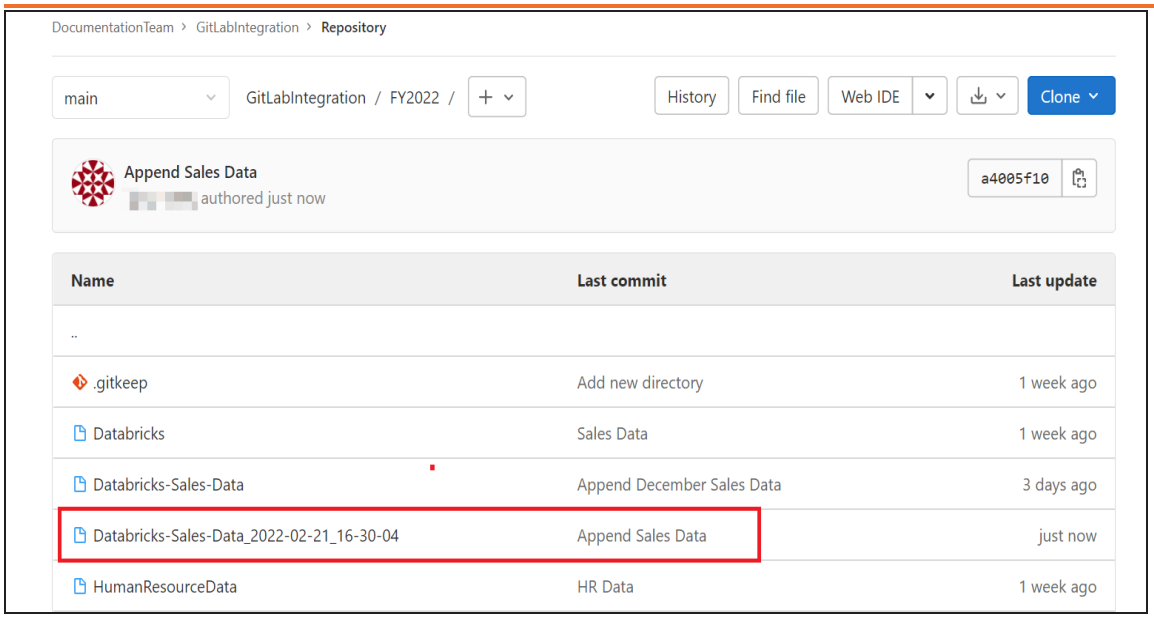
```
        status_code string,
        CUST_number int
)
USING delta
LOCATION 'dbfs:/user/hive/warehouse/erwin.db/cust_credit'
TBLPROPERTIES ('delta.minReaderVersion'='1','delta.minWriterVersion'='2');

CREATE TABLE cust_dec
(
        First_Week BYTE,
        Second_Week BYTE,
        Third_Week BYTE,
        Fourth_Week BYTE
);

CREATE TABLE emp
(
        EMP_first_name string,
        EMP_address string,
        EMP_phone int,
        EMP_address_2 string,
        email string,
        salary int,
        hire_date timestamp,
        soc_sec_number int,
        EMP_number string
```

Clearing the Auto Append check box and setting the File Name and File Path belonging to an existing file creates a new file with the following naming convention: <File Name>_YYYY-MM-DD_HH-MM-SS.

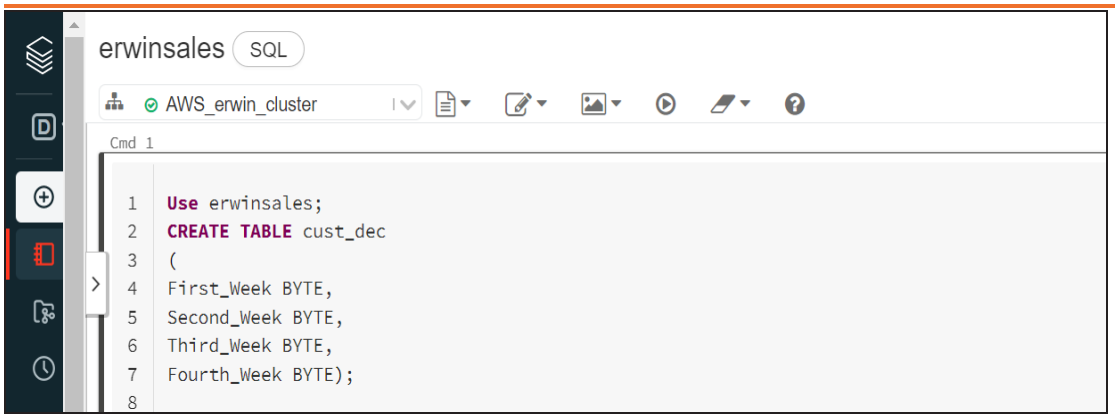
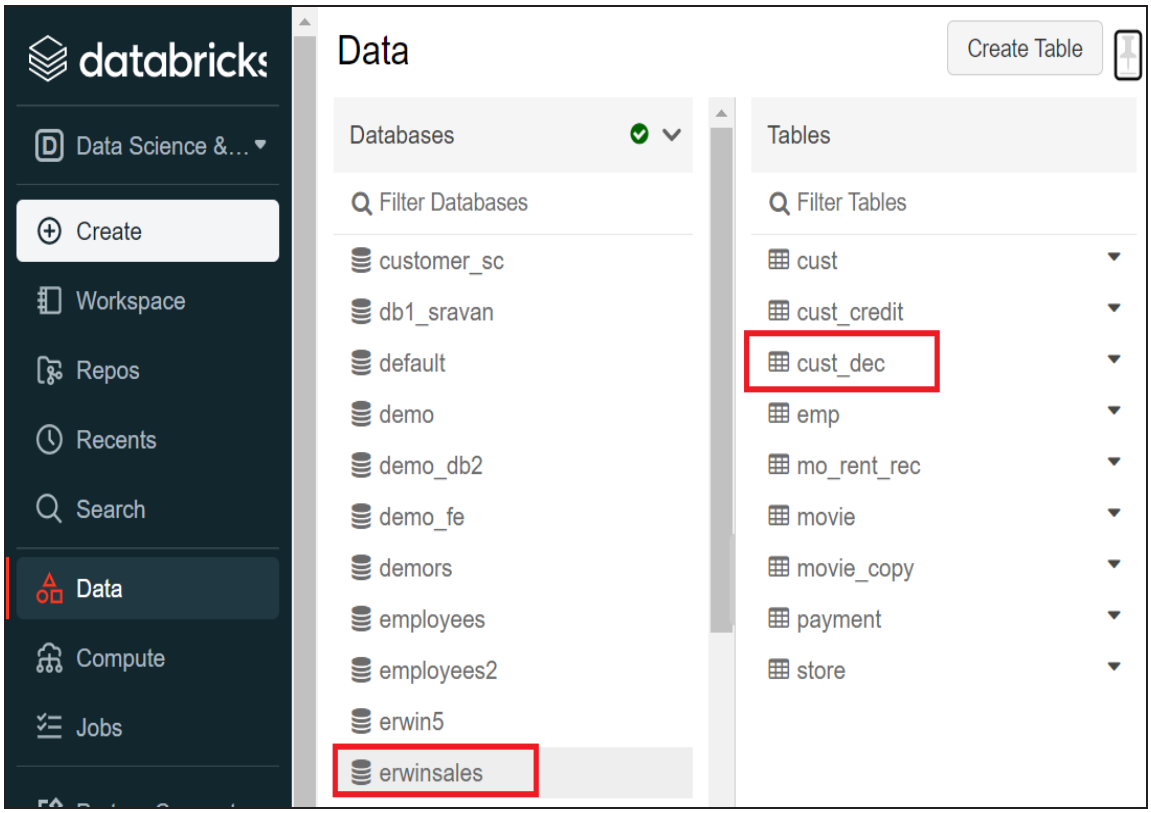
For example, in the following image, a file is created with a time stamp in a Git repository.



This file contains only the alter script.

Use the committed FE script to generate a physical schema in your database. To generate schema, copy the FE script from your Git repository and run the script in the database.

For example, in the following Databricks database, the FE script copied from the Git repository is run.



The cust_dec table is created in a Databricks database.