An Oracle White Paper
October 2013

Oracle Data Miner (Extension of SQL Developer 4.0)

Generate a PL/SQL script for workflow deployment

Denny Wong
Oracle Data Mining Technologies
10 Van de Graff Drive
Burlington, MA 01803
USA
denny.wong@oracle.com

Contents

Introduction

Integrating data mining with an end user application has been one of the more challenging assignments for developers of advanced analytics. With Data Miner 4.0, this effort has been made considerably easier to accomplish. Data analysts and application developers can now work together to develop advanced analytics and easily imbed them into applications.

Data analysts can use the Data Miner workflows to define, build and test their analytic methodologies. When they are satisfied with the results, they can use the new script generation feature to hand off a set of SQL scripts to the application developer. The application developer will be able to take these standard Oracle SQL scripts and integrate them easily into their applications. Since all of the results of Oracle Data Miner are database objects, there is no concern about moving data and models from non Oracle database systems.

Although the primary audience for this paper are application developers; data analysts, database administrators and IT management can benefit from reading it as well.

Objectives/Benefits:

- Application Developers
  - Describes the structure of the generated scripts and run time behavior.
  - Shows how the generated script can be scheduled to run in the database.
- Data Analysts
  - Describes how to generate a SQL script for all or part of a workflow.
- Database Administrators and IT Management
  - Provides an understanding of the entire process and how it all runs within the Oracle Database environment.

The business benefit will be in the ability to quickly integrate advanced analytics into existing applications without incurring the high cost and complexity of integrating additional non-database platforms

Importing the Workflow

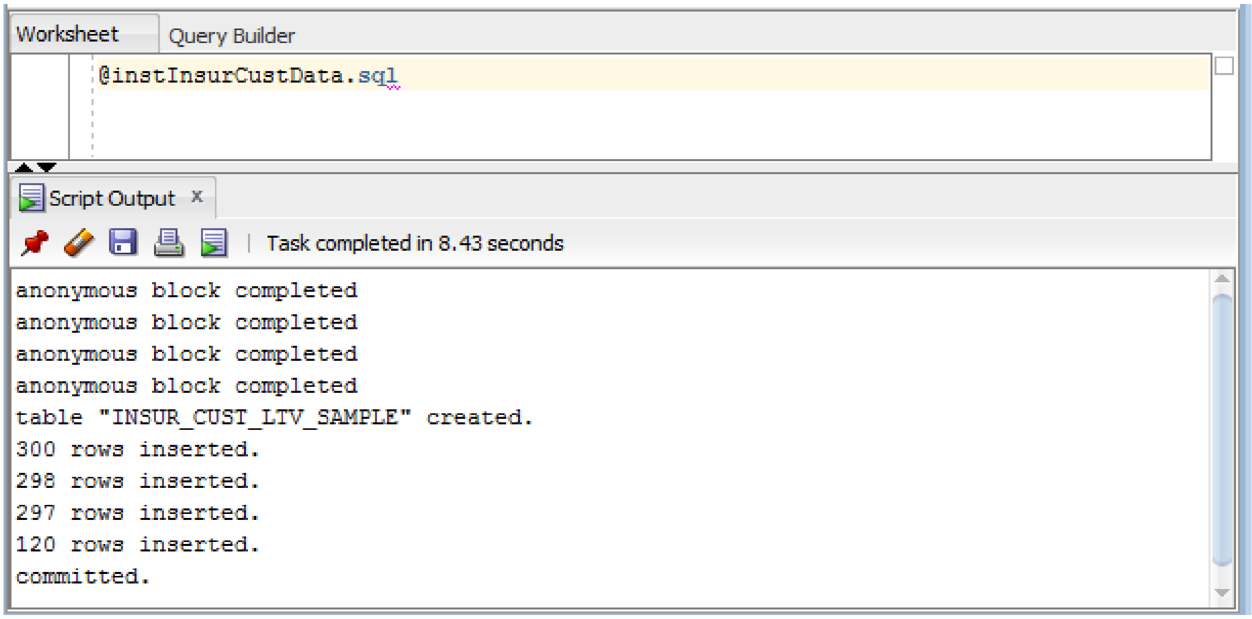
Create the User

This white paper demo requires a Data Miner installation and a Data Miner user account. Please refer to the Oracle By Example Tutorials to review how to create a Data Miner user account and install the Data Miner Repository.

Load the Table

The example in this white paper involves a demo database table INSUR_CUST_LTV_SAMPLE, which comes with the Data Miner Repository installation by default. If it is not loaded in your user account, you can install it manually using the instInsurCustData.sql SQL script (available in the companion .zip download for this white paper).

The instInsurCustData.sql SQL script can be invoked in the SQL Worksheet of SQL Developer:



```
@instInsurCustData.sql
```

Task completed in 8.43 seconds

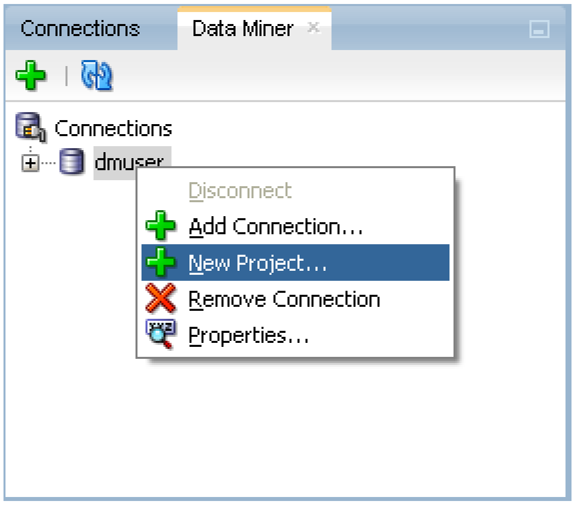
```
anonymous block completed
anonymous block completed
anonymous block completed
anonymous block completed
table "INSUR_CUST_LTV_SAMPLE" created.
300 rows inserted.
298 rows inserted.
297 rows inserted.
120 rows inserted.
committed.
```

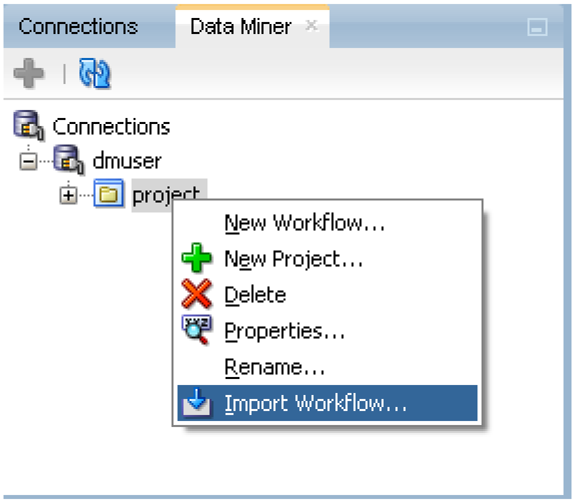
Make sure that the script is run from within the user schema.

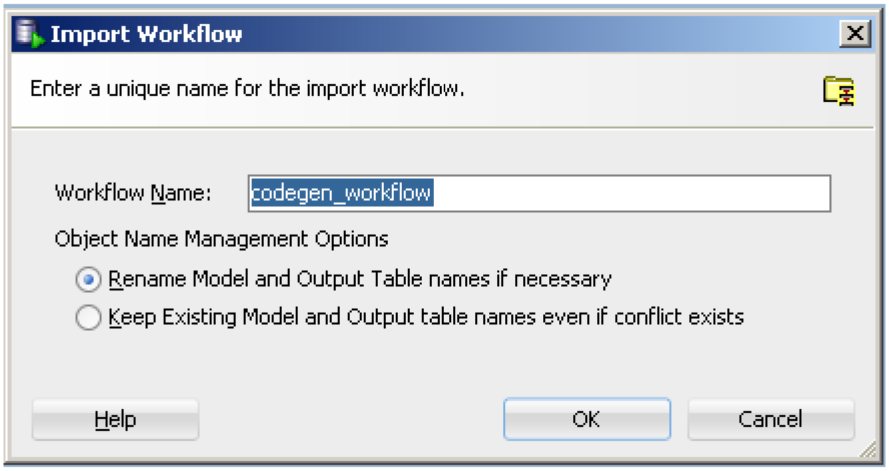
Import the Workflow

The Oracle Data Miner interface provides a mechanism for exporting and importing workflows that capture complex data mining methodologies. The next step in this example is to create a project and import a pre-defined workflow into existing Data Miner user account. The workflow, codegen_workflow.xml (available in the companion .zip download for this white paper) will be imported into the user account as shown below.

To create the project and import the workflow, right-click in the Data Miner navigator as shown in the two images below. The first step will create a new project, and the second will perform the import of the pre-defined workflow, codegen_workflow.xml.







Import Workflow

Enter a unique name for the import workflow.

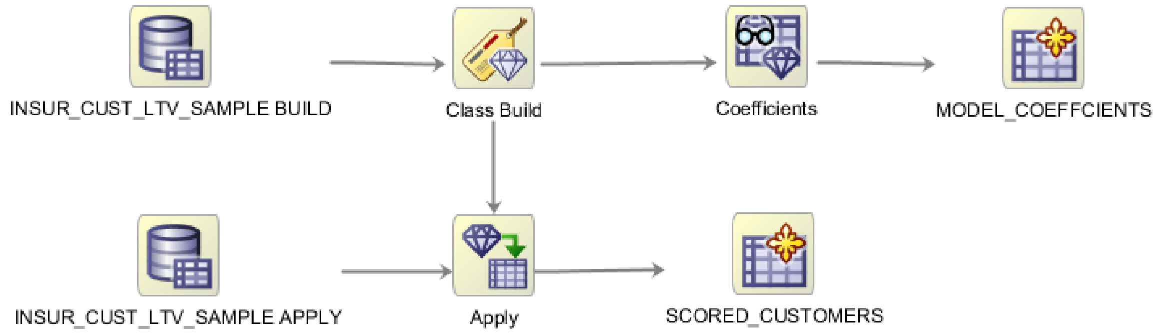
Workflow Name: codegen_workflow

Object Name Management Options

● Rename Model and Output Table names if necessary

○ Keep Existing Model and Output table names even if conflict exists

Once imported, the workflow should look like the picture below:

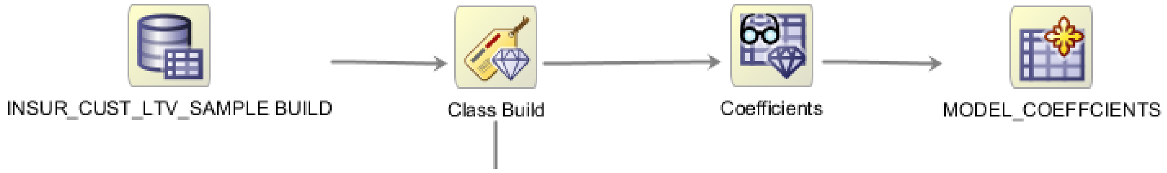


Workflow Overview

The workflow is comprised of two distinct processes contained within a single lineage: modeling (top) and scoring (bottom). Both processes use the demo data INSUR_CUST_LTV_SAMPLE as input data source.

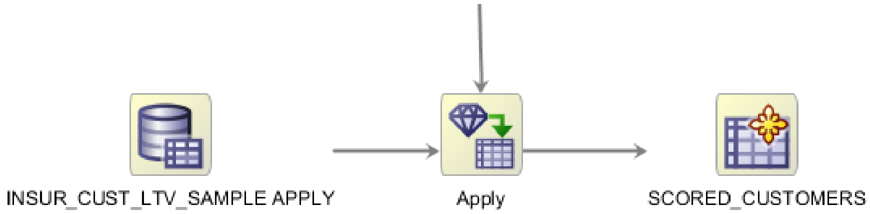
Modeling

The modeling process is used for building a classification SVM model to predict whether the customer will buy insurance or not. The model coefficients are persisted to a database table for viewing and this table may provide a basis for application integration.



Scoring

The scoring process is used to make predictions for the customer data using the SVM model created by the modeling lineage. The prediction result is persisted to a database view for viewing. This view always reflects the predictions of the current input data. For example, if the input table is refreshed with new data, this view will automatically capture the predictions of the new data. Moreover, this view may provide a basis for application integration.



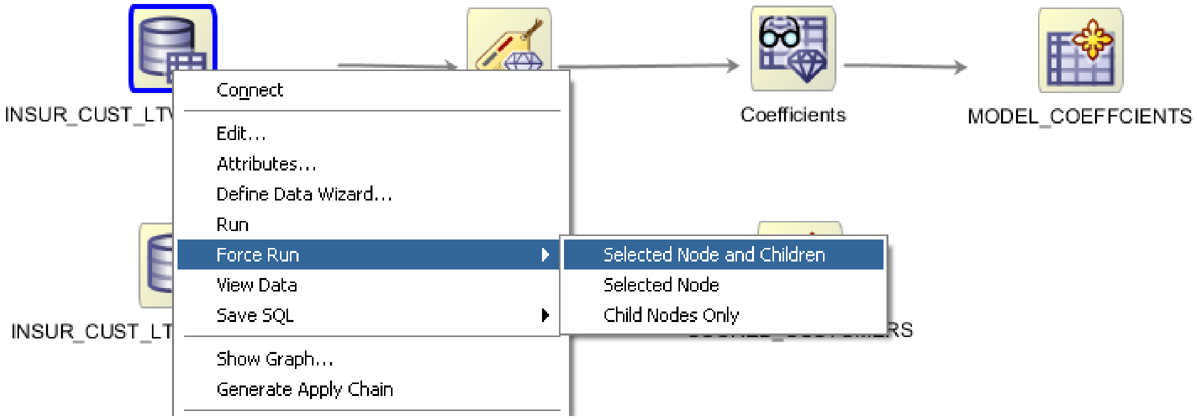
Deployment Use Case

For this demo, we assume a Data Analyst defines the above workflow for model build and score purposes. When he is satisfied with the results, he then uses the new script generation feature to hand off a set of SQL scripts to the Application Developer for deployment. The Developer deploys the scripts to the target or production database, where he can schedule the scripts to run periodically, so that the model will be rebuilt with fresh customer data. In this demo, you will see how to use both SQL Developer/Oracle Scheduler and Oracle Enterprise Manager to schedule the scripts to run on the database. Although not described in this paper, the output results generated by the scripts can easily be integrated into end user reporting applications like Oracle Business Intelligence Enterprise Edition. The output results are described below to give you some ideas on what insights they could provide for the end user.

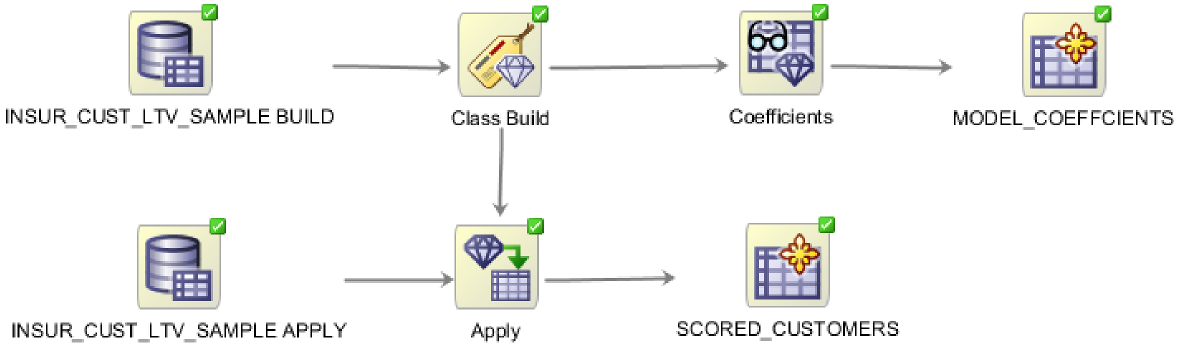
- MODEL_COEFFICIENTS – underlying database table can be queried to provide insight on what predictors are used to determine whether the user will buy insurance or not.
- SCORED_CUSTOMERS – underlying database view can be queried to display whether a customer will purchase insurance or not, and the likelihood of the prediction. This view always reflects the real time prediction of current customer data.

Workflow Run

The workflow must be run before it can be deployed. Right-click the INSUR_CUST_LTV_SAMPLE BUILD node and select Force Run | Selected Node and Children from the menu to run the whole workflow.



Once workflow run completed, the workflow should look like the picture below:



Generating Workflow Script Files

The next step in this demo is to generate a SQL script from the workflow.

Deploy Script Options

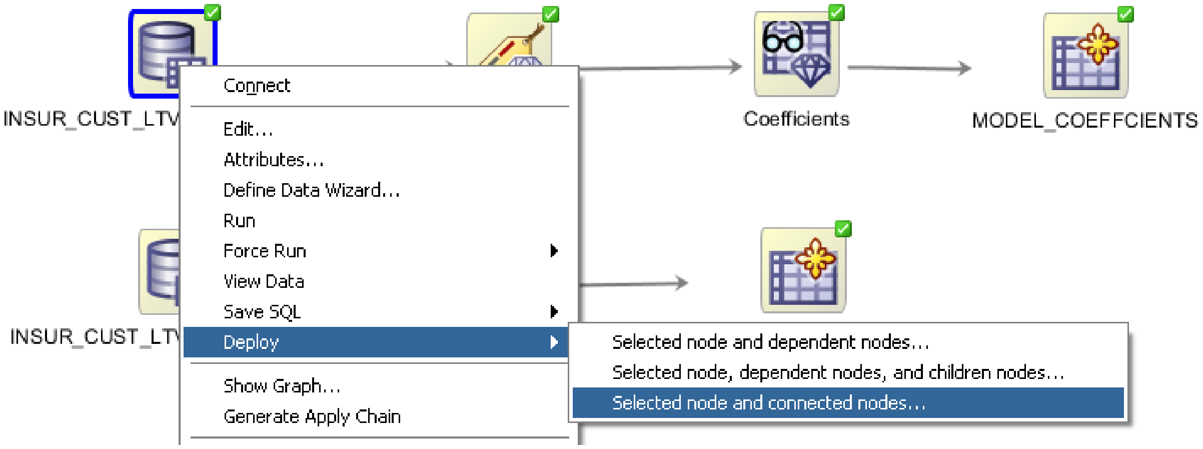
There are three deploy options of how the script can be generated:

- Selected node and dependent nodes
  - Generate script for the selected node(s) and all its parent nodes
  - For example, if the Apply node is selected, a script will be generated for these nodes: INSUR_CUST_LTV_SAMPLE BUILD, Class Build, INSUR_CUST_LTV_SAMPLE APPLY, and Apply.
- Selected node, dependent nodes, and children nodes
  - Generate script for the selected node(s) and all its parent and children nodes

    - For example, if the Apply node is selected, a script will be generated for these nodes: INSUR_CUST_LTV_SAMPLE BUILD, Class Build, INSUR_CUST_LTV_SAMPLE APPLY, Apply, and SCORED_CUSTOMERS.
- Selected node and connected nodes
    - Generate script for the selected node(s) and all nodes that are connected to this node.
    - For example, if the Apply node is selected, a script will be generated for all nodes in the workflow.

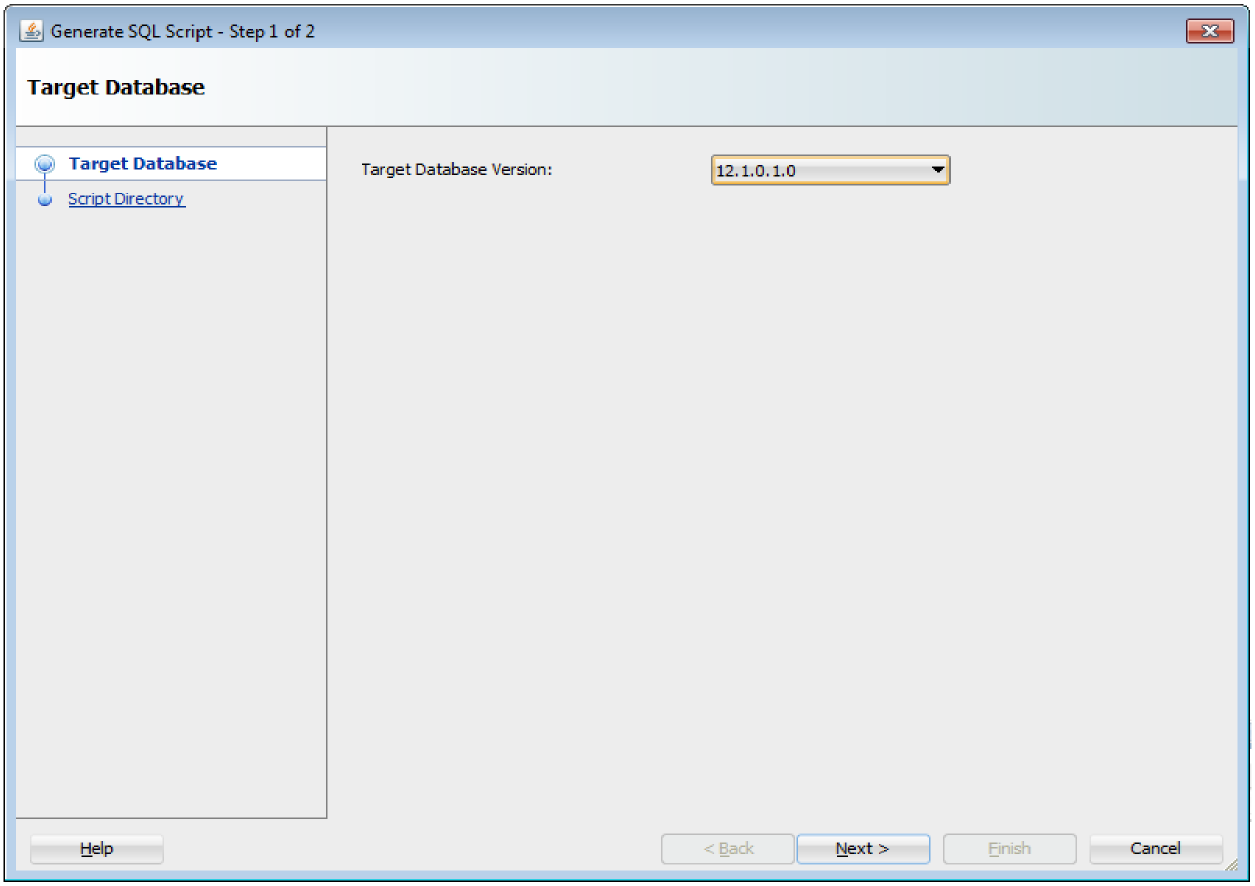
Alternatively, to generate a script for the entire workflow, you can multi-select all nodes in the workflow and select any of above options from the selected node.

To generate a script for the demo workflow, right-click the INSUR_CUST_LTV_SAMPLE BUILD node and select the Deploy | Selected node and connected nodes item from the menu.



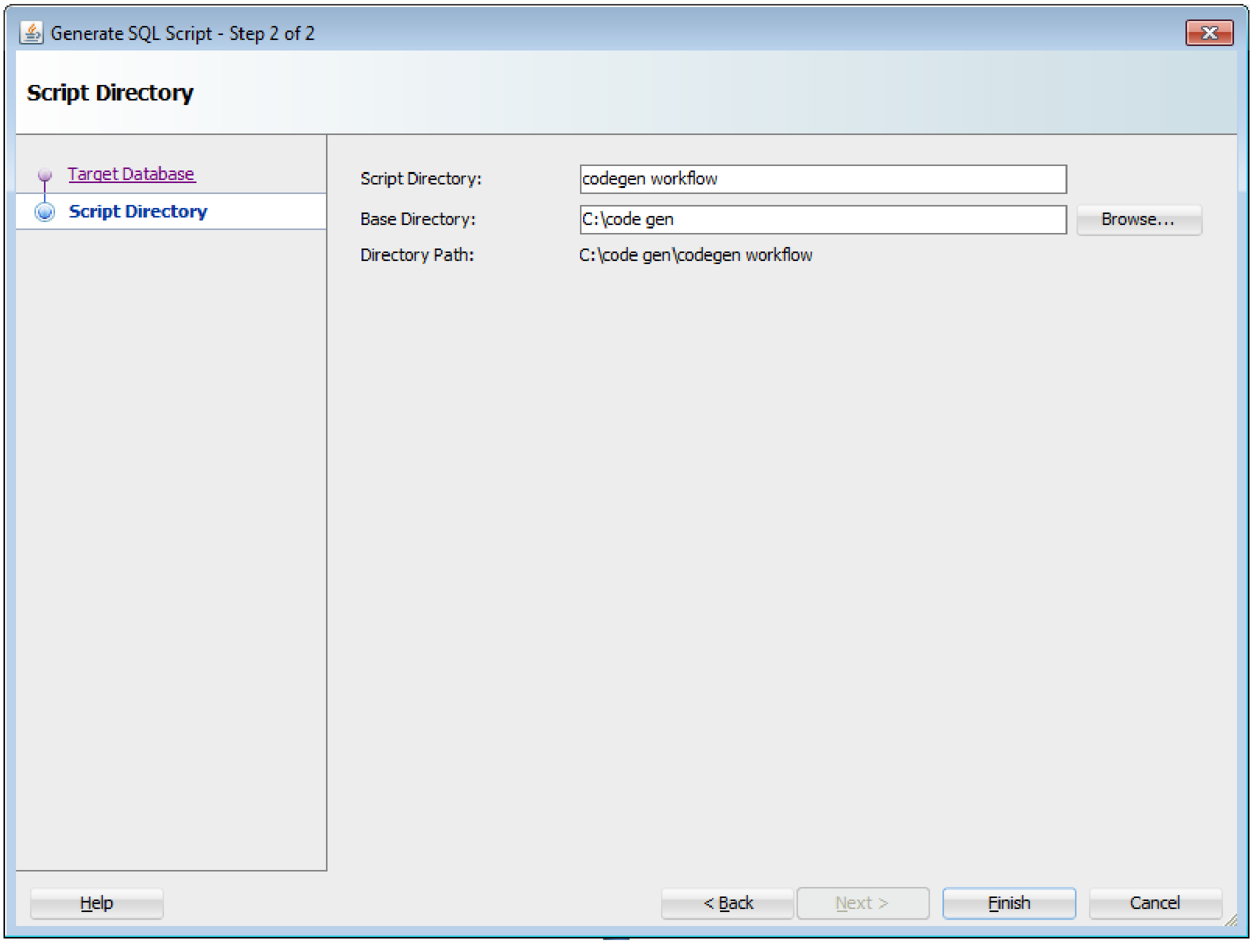
Generate Script UI Wizard

After you select the deploy option, the Generate SQL Script wizard launches. You must select Target Database Version, so that the generated script will be compatible for that database version. If the workflow nodes included in the script operation contain features that are not compatible with the target database, a warning dialog will be displayed along with unsupported feature information.



Target Database Version: 12.1.0.1.0

Next you need to specify the Script Directory where the scripts will be saved. The Script Directory name defaults to the workflow name. A new directory will be created to store all the generated scripts.



Generate SQL Script - Step 2 of 2

Script Directory

Script Directory: codegen workflow

Base Directory: C:\code gen

Directory Path: C:\code gen\codegen workflow

Click Finish to generate the script.

Script Files Specifications

The generated script consists of the following script files:

Master Script: <Workflow name>_Run.sql

The Master Script invokes all the required node level scripts in the appropriate order. It performs some validation, such as testing to determine if the version of the script is compatible with the version of the Data Miner Repository that is installed. It creates a workflow master table that contains entries for all the underlying objects created by the workflow script.

Cleanup script: <Workflow name>_Drop.sql

The Cleanup Script drops all objects created by running the master script. It drops hidden objects (such as the table generated for Explore Data) as well as public objects (such as Model Names created by Build nodes, and Tables created by a Create Table node).

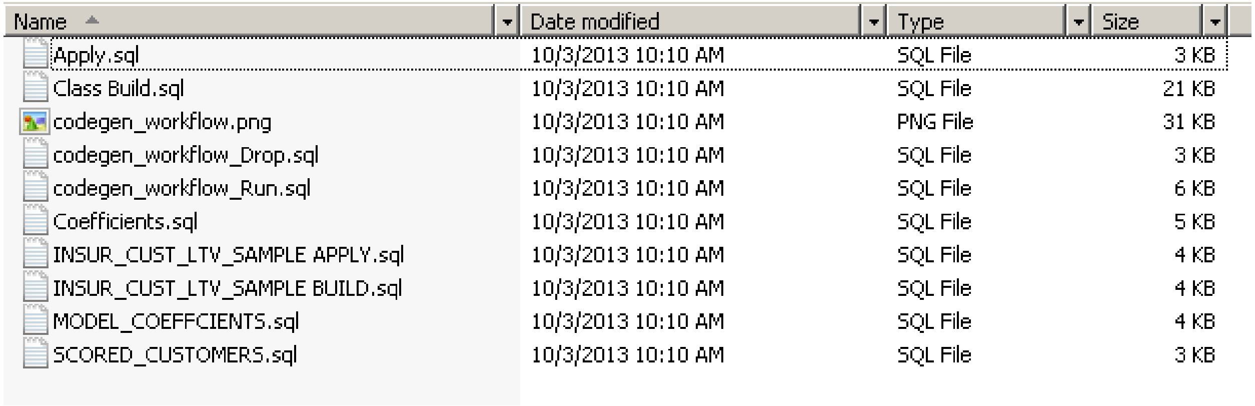
Workflow Image: <Workflow name>.png

An image (.png file) of the workflow at the time the script was generated. The entire workflow is displayed.

Node Script: <Node name>.sql

A Node Script performs node specific operation (such as model creation in Build node). A node script is generated for each node that participates in the script generation. The node script should be invoked by the master script only; it should not be invoked directly. Refer to the Appendix for output result for each node type.

For the demo workflow, the following script files are generated:



Name	Date modified	Type	Size
Apply.sql	10/3/2013 10:10 AM	SQL File	3 KB
Class Build.sql	10/3/2013 10:10 AM	SQL File	21 KB
codegen_workflow.png	10/3/2013 10:10 AM	PNG File	31 KB
codegen_workflow_Drop.sql	10/3/2013 10:10 AM	SQL File	3 KB
codegen_workflow_Run.sql	10/3/2013 10:10 AM	SQL File	6 KB
Coefficients.sql	10/3/2013 10:10 AM	SQL File	5 KB
INSUR_CUST_LTV_SAMPLE APPLY.sql	10/3/2013 10:10 AM	SQL File	4 KB
INSUR_CUST_LTV_SAMPLE BUILD.sql	10/3/2013 10:10 AM	SQL File	4 KB
MODEL_COEFFCIENTS.sql	10/3/2013 10:10 AM	SQL File	4 KB
SCORED_CUSTOMERS.sql	10/3/2013 10:10 AM	SQL File	3 KB

Master script: codegen_workflow_Run.sql.
Cleanup script: codegen_workflow_Drop.sql.
Workflow image: codegen_workflow.png.
Node scripts: Apply.sql, Class Build.sql, Coefficients.sql, INSUR_CUST_LTV_SAMPLE APPLY.sql, INSUR_CUST_LTV_SAMPLE BUILD.sql, and MODEL_COEFFCIENTS.sql, and SCORED_CUSTOMERS.sql.

Running Workflow Script Files

Variable Definitions

The Node level scripts have variable definitions that provide object names for the public objects created by the scripts. The Master script is responsible for invoking all the underlying node level scripts in proper order, so all variable definitions must be defined in the Master script.

The following variables are supported:

- Variables that allows you to change the name of the object names that are input to the Node level scripts, such as tables/views and models. By default, these names are the original table/view and model names.
- Variable that allows you to change the name of the Control table (see below). By default, this name is the workflow name.
- Variable that indicates if named objects should be deleted first before they are generated by the script.

The following object types can be named using these definitions:

- Models referenced in Build Nodes or in a Model Node
- Tables referenced in Create Table or Update Table Nodes

For the demo workflow, the following variable definitions are generated in the Master script:

```
-- Subsitution Variable Definition Section: Override default object names here
DEFINE WORKFLOW_OUTPUT = 'codegen_workflow'
-- Drop user named objects (e.g. model, output table)?  TRUE or FALSE
DEFINE DROP_EXISTING_OBJECTS = 'TRUE'
-- From Node: "INSUR_CUST_LTV_SAMPLE BUILD"
DEFINE DATA_SOURCE_1 = '"DMUSER"."INSUR_CUST_LTV_SAMPLE"'
-- From Node: "INSUR_CUST_LTV_SAMPLE APPLY"
DEFINE DATA_SOURCE_2 = '"DMUSER"."INSUR_CUST_LTV_SAMPLE"'
-- From Node: "Class Build"
DEFINE MODEL_1 = '"DMUSER"."CLAS_SVM_MODEL"'
-- From Node: "MODEL_COEFFCIENTS"
DEFINE CREATE_TABLE_2 = '"DMUSER"."MODEL_COEFFCIENTS"'
-- From Node: "SCORED_CUSTOMERS"
DEFINE CREATE_VIEW_3 = '"DMUSER"."SCORED_CUSTOMERS_V"'
```

The Control table name variable, WORKFLOW_OUTPUT defaults to the workflow name. The drop variable, DROP_EXISTING_OBJECTS defaults to TRUE, which indicates all existing named objects should be removed before they are generated. The Node specific object variables default to their original names in the workflow. For example, the generated Node variable DATA_SOURCE_1 allows user to override the input table used in the INSUR_CUST_LTV_SAMPLE BUILD Data Source node. It is expected that the input data sources have the same columns as the originally referenced input data sources. Missing column names could result in a run time failure.

Control Table

When the Master script is run, it first creates the Control table using the name specified in the control table name variable. The purposes of the Control table are the followings:

- Generated objects, such as views, models, text specifications, are registered in this table.
- Logical nodes in the workflow are able to look up their input objects and register their output objects in the control table.
- The Cleanup script uses the table to determine what objects need to be dropped.
- For advanced users, the Control table provides internal name of objects that are not readily accessible via the workflows today. For example, users can find the model test result tables by viewing the Control table.
- By using different control file names along with different output variable names, the generated script can be used concurrently to generate and manage different results. This may be useful if the input data sources continue different sets of data that you wish to mine independently. In this use case, the application would be responsible for saving the name of the control table so that it can be utilized when rerunning or dropping the generated results.

Table Structure

The Control table is defined as following structure:

```
CREATE TABLE "&WORKFLOW_OUTPUT"
(
 NODE_ID VARCHAR2(30) NOT NULL,
 NODE_NAME VARCHAR2(30) NOT NULL,
 NODE_TYPE VARCHAR2(30) NOT NULL,
 MODEL_ID VARCHAR2(30),
 MODEL_NAME VARCHAR2(65),
 MODEL_TYPE VARCHAR2(35),
 OUTPUT_NAME VARCHAR2(30) NOT NULL,
 OUTPUT_TYPE VARCHAR2(30) NOT NULL,
 ADDITIONAL_INFO VARCHAR2(65),
 CREATION_TIME TIMESTAMP(6) NOT NULL,
 COMMENTS VARCHAR2(4000 CHAR)
)
```

Column Descriptions

The followings describe how the columns in the table are used:

Column Name	Description	Examples
NODE_ID	Workflow Node id. This uniquely identifies each node.	10001, 10002
NODE_NAME	Workflow Node name.	Class Build, MINING_DATA_BUILD_V
NODE_TYPE	Node type is the category of node type that the node is based on.	Data Source, Class Build
MODEL_ID	Workflow Model id. This uniquely identifies each model referenced within a workflow.	10101, 10102
MODEL_NAME	Name of Model.	CLAS_GLM_1_6
MODEL_TYPE	Model type is the algorithm type used by the model.	Generalized Linear Model
OUTPUT_NAME	The name of the output. These will be internally generated names unless the names are under the control of the user.	Table/View Name, Model Name, Text object names Such as: ODMR$15_37_21_839599RMAFRXI - table name "DMUSER"."CLAS_GLM_1_6" - fully qualified model name
OUTPUT_TYPE	Qualifies what type of object.	Table, View, Model
ADDITIONAL_INFO	Used to qualify what the purpose of the object is in context of the script execution.	Target class for test lift result
CREATION_TIME	Object creation time.	11-DEC-12 03.37.25.935193000 PM (format determined by locale)
COMMENTS	Comment to qualify the object's role in the context of the script execution.	Output Data (displayed for nodes like Data Source) Data Usage (displayed for the view passed into model build) Weights Setting (displayed for a weights table passed into model build) Build Setting (displayed for a build settings table passed into model build) Model (displayed for a Model object)

To run the deployed workflow, invoke the Master script in the SQL Plus:

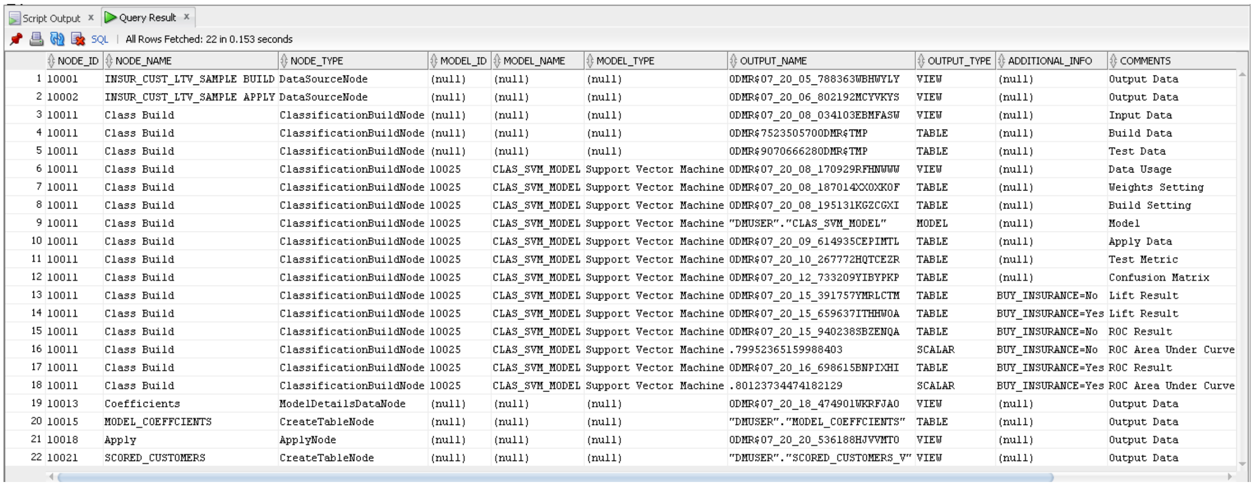
```
>@" C:\code gen\codegen workflow\codegen_workflow_Run.sql"
```

For subsequent run, invoke the Cleanup script first to delete previously generated objects, and then run the Master script:

```
>@" C:\code gen\codegen workflow\codegen_workflow_Drop.sql"
>@" C:\code gen\codegen workflow\codegen_workflow_Run.sql"
```

After the script is run successfully, you can query the Control table to examine the generated objects:

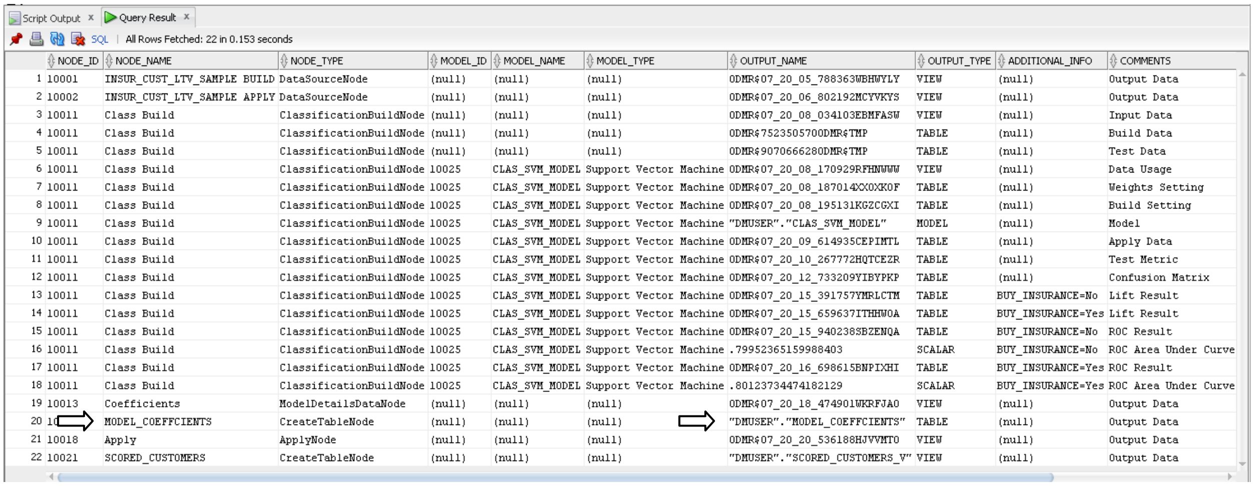
```
>select * from "codegen_workflow"
```



All Rows Fetched: 22 in 0.153 seconds

	NODE_ID	NODE_NAME	NODE_TYPE	MODEL_ID	MODEL_NAME	MODEL_TYPE	OUTPUT_NAME	OUTPUT_TYPE	ADDITIONAL_INFO	COMMENTS
1	10001	INSUR_CUST_LTV_SAMPLE BUILD	DataSourceNode	(null)	(null)	(null)	ODMR$07_20_05_788363WBHWYLY	VIEW	(null)	Output Data
2	10002	INSUR_CUST_LTV_SAMPLE APPLY	DataSourceNode	(null)	(null)	(null)	ODMR$07_20_06_802192MCYVKYS	VIEW	(null)	Output Data
3	10011	Class Build	ClassificationBuildNode	(null)	(null)	(null)	ODMR$07_20_08_034103EBMFASW	VIEW	(null)	Input Data
4	10011	Class Build	ClassificationBuildNode	(null)	(null)	(null)	ODMR$7523505700DMR$TMP	TABLE	(null)	Build Data
5	10011	Class Build	ClassificationBuildNode	(null)	(null)	(null)	ODMR$9070666280DMR$TMP	TABLE	(null)	Test Data
6	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	ODMR$07_20_08_170929RFHNWWW	VIEW	(null)	Data Usage
7	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	ODMR$07_20_08_187014XX0XK0F	TABLE	(null)	Weights Setting
8	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	ODMR$07_20_08_195131KGZCGXI	TABLE	(null)	Build Setting
9	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	"DMUSER"."CLAS_SVM_MODEL"	MODEL	(null)	Model
10	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	ODMR$07_20_09_614935CEPIMTL	TABLE	(null)	Apply Data
11	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	ODMR$07_20_10_267772HQTCEZR	TABLE	(null)	Test Metric
12	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	ODMR$07_20_12_733209YIBYPKP	TABLE	(null)	Confusion Matrix
13	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	ODMR$07_20_15_391757YMRLCTM	TABLE	BUY_INSURANCE=No	Lift Result
14	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	ODMR$07_20_15_659637ITHHW0A	TABLE	BUY_INSURANCE=Yes	Lift Result
15	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	ODMR$07_20_15_940238SBZENQA	TABLE	BUY_INSURANCE=No	ROC Result
16	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	.79952365159988403	SCALAR	BUY_INSURANCE=No	ROC Area Under Curve
17	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	ODMR$07_20_16_698615BNPIXHI	TABLE	BUY_INSURANCE=Yes	ROC Result
18	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	.80123734474182129	SCALAR	BUY_INSURANCE=Yes	ROC Area Under Curve
19	10013	Coefficients	ModelDetailsDataNode	(null)	(null)	(null)	ODMR$07_20_18_474901WKRFJA0	VIEW	(null)	Output Data
20	10015	MODEL_COEFFCIENTS	CreateTableNode	(null)	(null)	(null)	"DMUSER"."MODEL_COEFFCIENTS"	TABLE	(null)	Output Data
21	10018	Apply	ApplyNode	(null)	(null)	(null)	ODMR$07_20_20_536188HJVVMT0	VIEW	(null)	Output Data
22	10021	SCORED_CUSTOMERS	CreateTableNode	(null)	(null)	(null)	"DMUSER"."SCORED_CUSTOMERS_V"	VIEW	(null)	Output Data

For example, the Create Table Node, MODEL_COEFFICIENTS, produced an output table MODEL_COEFFCIENTS that persisted the coefficient data extracted from the generated SVM model.

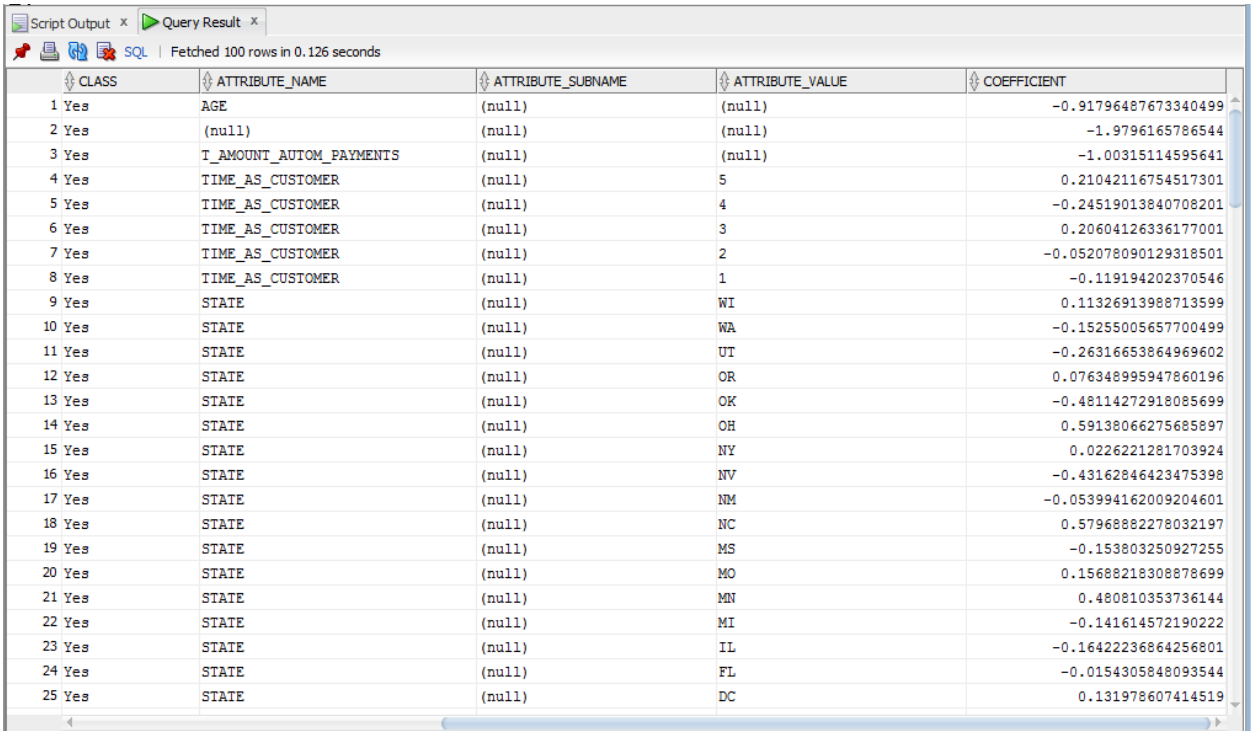


All Rows Fetched: 22 in 0.153 seconds

	NODE_ID	NODE_NAME	NODE_TYPE	MODEL_ID	MODEL_NAME	MODEL_TYPE	OUTPUT_NAME	OUTPUT_TYPE	ADDITIONAL_INFO	COMMENTS
1	10001	INSUR_CUST_LTV_SAMPLE BUILD	DataSourceNode	(null)	(null)	(null)	ODMR$07_20_05_788363WBHWYLY	VIEW	(null)	Output Data
2	10002	INSUR_CUST_LTV_SAMPLE APPLY	DataSourceNode	(null)	(null)	(null)	ODMR$07_20_06_802192MCYVKYS	VIEW	(null)	Output Data
3	10011	Class Build	ClassificationBuildNode	(null)	(null)	(null)	ODMR$07_20_08_034103EBMFASW	VIEW	(null)	Input Data
4	10011	Class Build	ClassificationBuildNode	(null)	(null)	(null)	ODMR$7523505700DMR$TMP	TABLE	(null)	Build Data
5	10011	Class Build	ClassificationBuildNode	(null)	(null)	(null)	ODMR$9070666280DMR$TMP	TABLE	(null)	Test Data
6	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	ODMR$07_20_08_170929RFHNWWW	VIEW	(null)	Data Usage
7	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	ODMR$07_20_08_187014XX0XK0F	TABLE	(null)	Weights Setting
8	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	ODMR$07_20_08_195131KGZCGXI	TABLE	(null)	Build Setting
9	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	"DMUSER"."CLAS_SVM_MODEL"	MODEL	(null)	Model
10	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	ODMR$07_20_09_614935CEPIMTL	TABLE	(null)	Apply Data
11	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	ODMR$07_20_10_267772HQTCEZR	TABLE	(null)	Test Metric
12	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	ODMR$07_20_12_733209YIBYPKP	TABLE	(null)	Confusion Matrix
13	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	ODMR$07_20_15_391757YMRLCTM	TABLE	BUY_INSURANCE=No	Lift Result
14	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	ODMR$07_20_15_659637ITHHW0A	TABLE	BUY_INSURANCE=Yes	Lift Result
15	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	ODMR$07_20_15_940238SBZENQA	TABLE	BUY_INSURANCE=No	ROC Result
16	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	.79952365159988403	SCALAR	BUY_INSURANCE=No	ROC Area Under Curve
17	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	ODMR$07_20_16_698615BNPIXHI	TABLE	BUY_INSURANCE=Yes	ROC Result
18	10011	Class Build	ClassificationBuildNode	10025	CLAS_SVM_MODEL	Support Vector Machine	.80123734474182129	SCALAR	BUY_INSURANCE=Yes	ROC Area Under Curve
19	10013	Coefficients	ModelDetailsDataNode	(null)	(null)	(null)	ODMR$07_20_18_474901WKRFJA0	VIEW	(null)	Output Data
20	1	MODEL_COEFFCIENTS	CreateTableNode	(null)	(null)	(null)	"DMUSER"."MODEL_COEFFCIENTS"	TABLE	(null)	Output Data
21	10018	Apply	ApplyNode	(null)	(null)	(null)	ODMR$07_20_20_536188HJVVMT0	VIEW	(null)	Output Data
22	10021	SCORED_CUSTOMERS	CreateTableNode	(null)	(null)	(null)	"DMUSER"."SCORED_CUSTOMERS_V"	VIEW	(null)	Output Data

To examine the coefficient data, you can query the output table:

> select * from "DMUSER"."MODEL_COEFFCIENTS"



	CLASS	ATTRIBUTE_NAME	ATTRIBUTE_SUBNAME	ATTRIBUTE_VALUE	COEFFICIENT
1	Yes	AGE	(null)	(null)	-0.91796487673340499
2	Yes	(null)	(null)	(null)	-1.9796165786544
3	Yes	T_AMOUNT_AUTOM_PAYMENTS	(null)	(null)	-1.00315114595641
4	Yes	TIME_AS_CUSTOMER	(null)	5	0.21042116754517301
5	Yes	TIME_AS_CUSTOMER	(null)	4	-0.24519013840708201
6	Yes	TIME_AS_CUSTOMER	(null)	3	0.20604126336177001
7	Yes	TIME_AS_CUSTOMER	(null)	2	-0.052078090129318501
8	Yes	TIME_AS_CUSTOMER	(null)	1	-0.119194202370546
9	Yes	STATE	(null)	WI	0.11326913988713599
10	Yes	STATE	(null)	WA	-0.15255005657700499
11	Yes	STATE	(null)	UT	-0.26316653864969602
12	Yes	STATE	(null)	OR	0.076348995947860196
13	Yes	STATE	(null)	OK	-0.48114272918085699
14	Yes	STATE	(null)	OH	0.59138066275685897
15	Yes	STATE	(null)	NY	0.0226221281703924
16	Yes	STATE	(null)	NV	-0.43162846423475398
17	Yes	STATE	(null)	NM	-0.053994162009204601
18	Yes	STATE	(null)	NC	0.57968882278032197
19	Yes	STATE	(null)	MS	-0.153803250927255
20	Yes	STATE	(null)	MO	0.15688218308878699
21	Yes	STATE	(null)	MN	0.480810353736144
22	Yes	STATE	(null)	MI	-0.141614572190222
23	Yes	STATE	(null)	IL	-0.16422236864256801
24	Yes	STATE	(null)	FL	-0.0154305848093544
25	Yes	STATE	(null)	DC	0.131978607414519

Scheduling Workflow Script Files

We will use Oracle Database to schedule the SQL script files to run on the target or production database. All generated SQL script files need to be deployed to the target or production database host machine where they are accessible by the database instance, and these files should be stored together in the same directory. In addition, Data Miner repository is required to run the script files because some node scripts make use of repository provided services at runtime. Moreover, Data Miner user account is required to run the script files because it has necessary grants to the repository provided services. In the following sections, we will show you how to use both SQL Developer and Oracle Enterprise Manager web application to schedule the master script to run.

Add full directory path in the master file

A full directory path needs to be added to each node script invocation in the master script, so that these individual node script files can be called by the master script during runtime.

In the master script, codegen_workflow_Run.sql, find the following node script invocations:

```
-- Workflow run
@@"INSUR_CUST_LTV_SAMPLE BUILD.sql";
@@"INSUR_CUST_LTV_SAMPLE APPLY.sql";
@@"Class Build.sql";
@@"Coefficients.sql";
@@"MODEL_COEFFCIENTS.sql";
@@"Apply.sql";
@@"SCORED_CUSTOMERS.sql";
```

Then add the full directory path where the scripts will be stored in the target or production database host machine. In this example, we assume the script files will be deployed to the /home/workspace directory.

```
-- Workflow run
@@"/home/workspace/INSUR_CUST_LTV_SAMPLE BUILD.sql";
@@"/home/workspace/INSUR_CUST_LTV_SAMPLE APPLY.sql";
@@"/home/workspace/Class Build.sql";
@@"/home/workspace/Coefficients.sql";
@@"/home/workspace/MODEL_COEFFCIENTS.sql";
@@"/home/workspace/Apply.sql";
@@"/home/workspace/SCORED_CUSTOMERS.sql";
```

SQL Developer

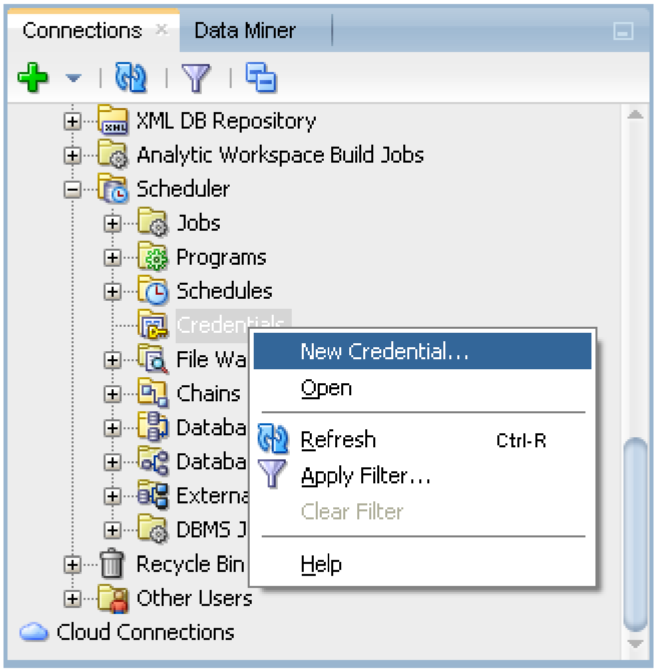
SQL Developer provides a graphical interface for developers to define Scheduler Jobs. A SQLPlus script job is required to run the SQL script files, but the Script job type is only supported in the 12c database. The Job definition defines the master script invocation as a script file using a full file path. The user can decide on whether the Job should be run on a schedule or on demand. The Job execution can be monitored within the application. The result will either be a success or a reported failure.

The following system privileges are required for the account that runs the scheduled SQL script files:

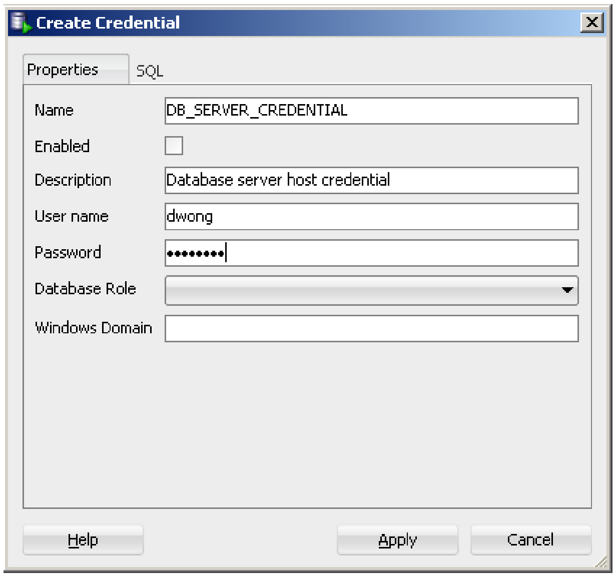
- CREATE CREDENTIAL
- CREATE EXTERNAL JOB

A credential is an Oracle Scheduler object that has a user name and password pair stored in a dedicated database object. A SQLPlus script job uses a host credential to authenticate itself with a database instance or the operating system so that the SQLPlus executable can run. In addition, the job may point to a connect credential that contains a database credential, which is used to connect SQLPlus to the database before running the script.

To create credentials, right-click the Credentials item in the Connections navigator as shown below.



First we will create the database host credential, and then the database credential.



Create Credential

Name: DB_SERVER_CREDENTIAL

Enabled: □

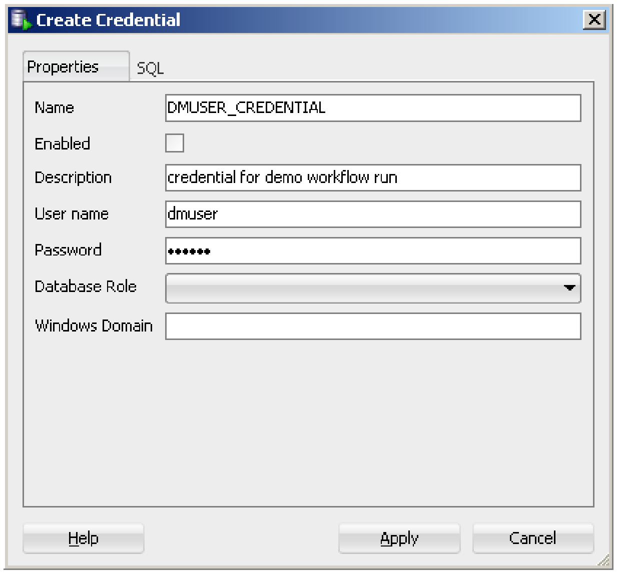
Description: Database server host credential

User name: dwong

Password: ••••••••

Database Role:

Windows Domain:



Create Credential

Name: DMUSER_CREDENTIAL

Enabled: □

Description: credential for demo workflow run

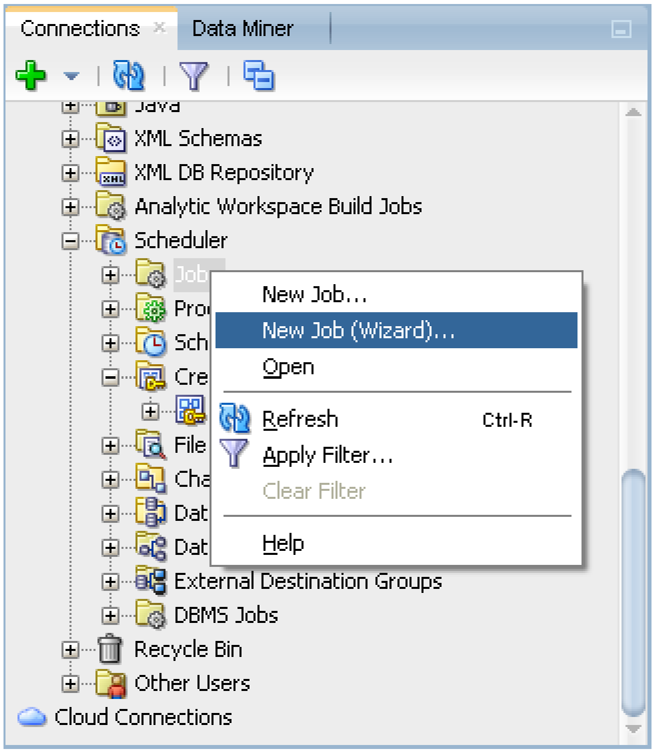
User name: dmuser

Password: ••••••

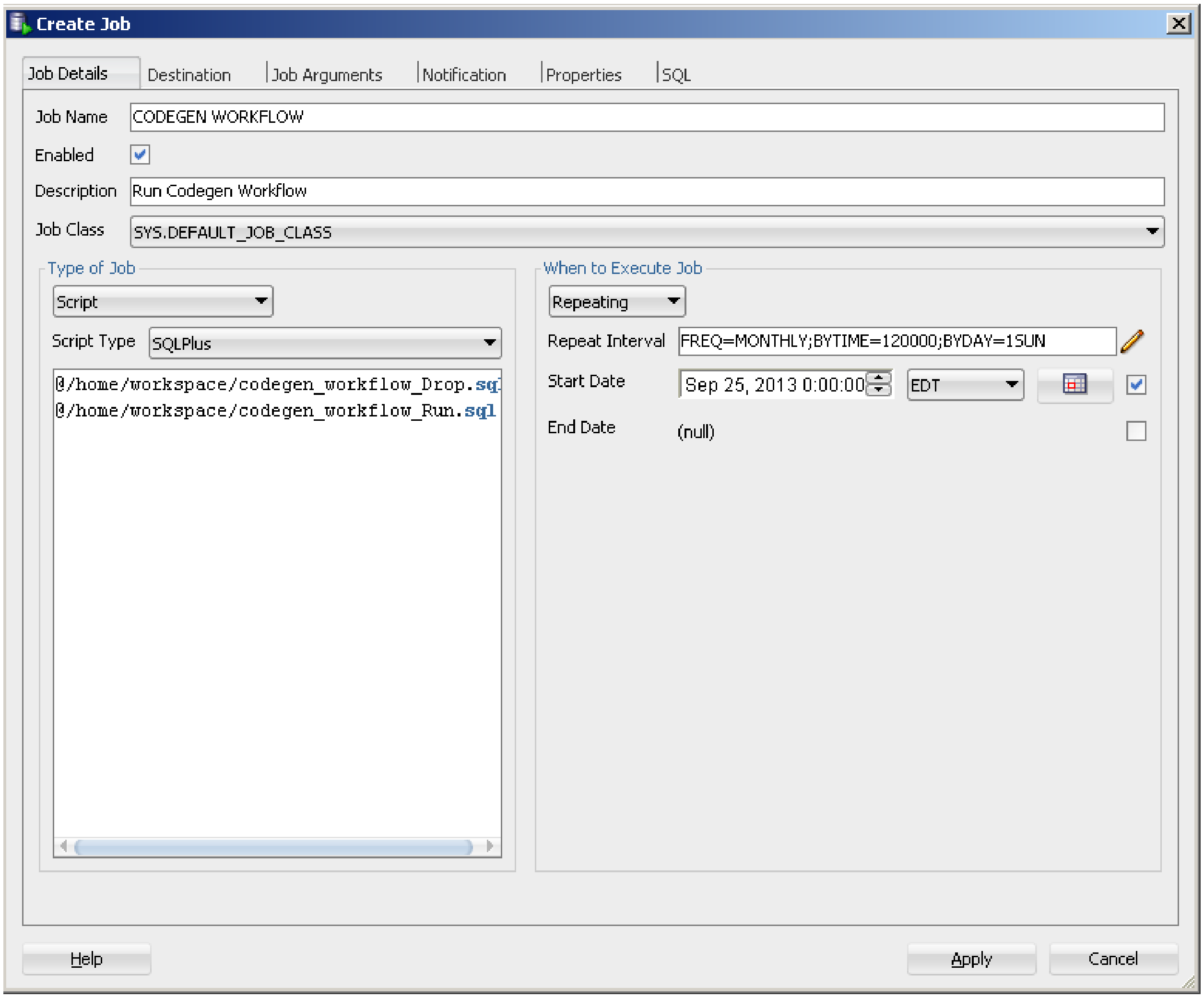
Database Role:

Windows Domain:

Next we will use the Job wizard to define a new Scheduler job. Click the Job item in the Connections navigator to launch the wizard.



In the Job Details step, enter the following Job name and description. Select the Script job type and SQLPlus script type, and enter the full path names of the cleanup and master scripts in the script window. You can specify if the job will be run one time only or repeating. The following shows the job will be scheduled to run on the first Sunday of each month at midnight.



Job Name: CODEGEN WORKFLOW

Enabled: ☑

Description: Run Codegen Workflow

Job Class: SYS.DEFAULT_JOB_CLASS

Type of Job: Script

Script Type: SQLPlus

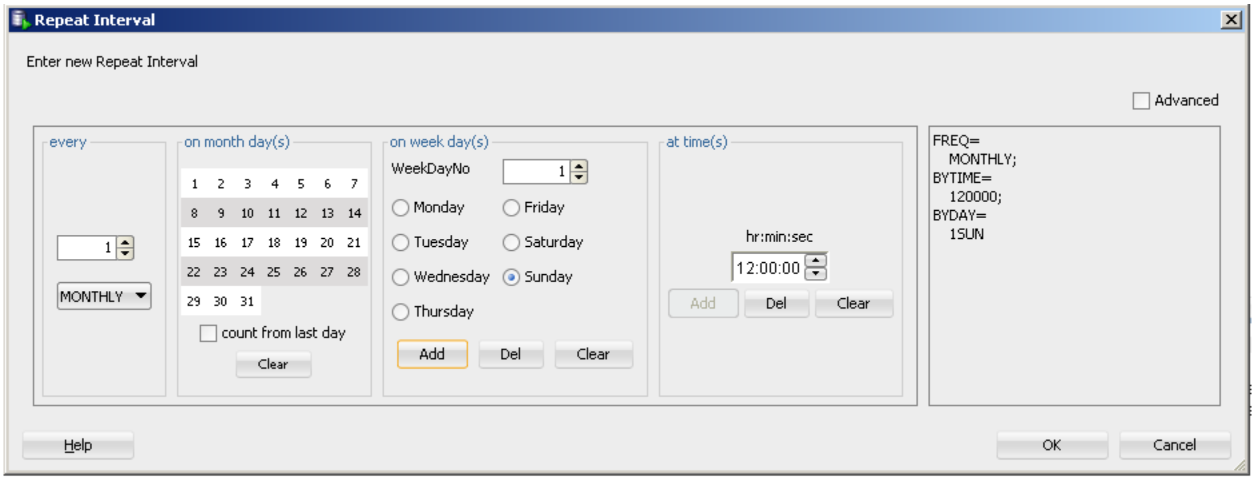
```
@/home/workspace/codegen_workflow_Drop.sq
@/home/workspace/codegen_workflow_Run.sql
```

When to Execute Job: Repeating

Repeat Interval: FREQ=MONTHLY;BYTIME=120000;BYDAY=1SUN

Start Date: Sep 25, 2013 0:00:00 EDT ☑

End Date: (null) □



Enter new Repeat Interval

□ Advanced

every: 1 MONTHLY

on month day(s): □ count from last day

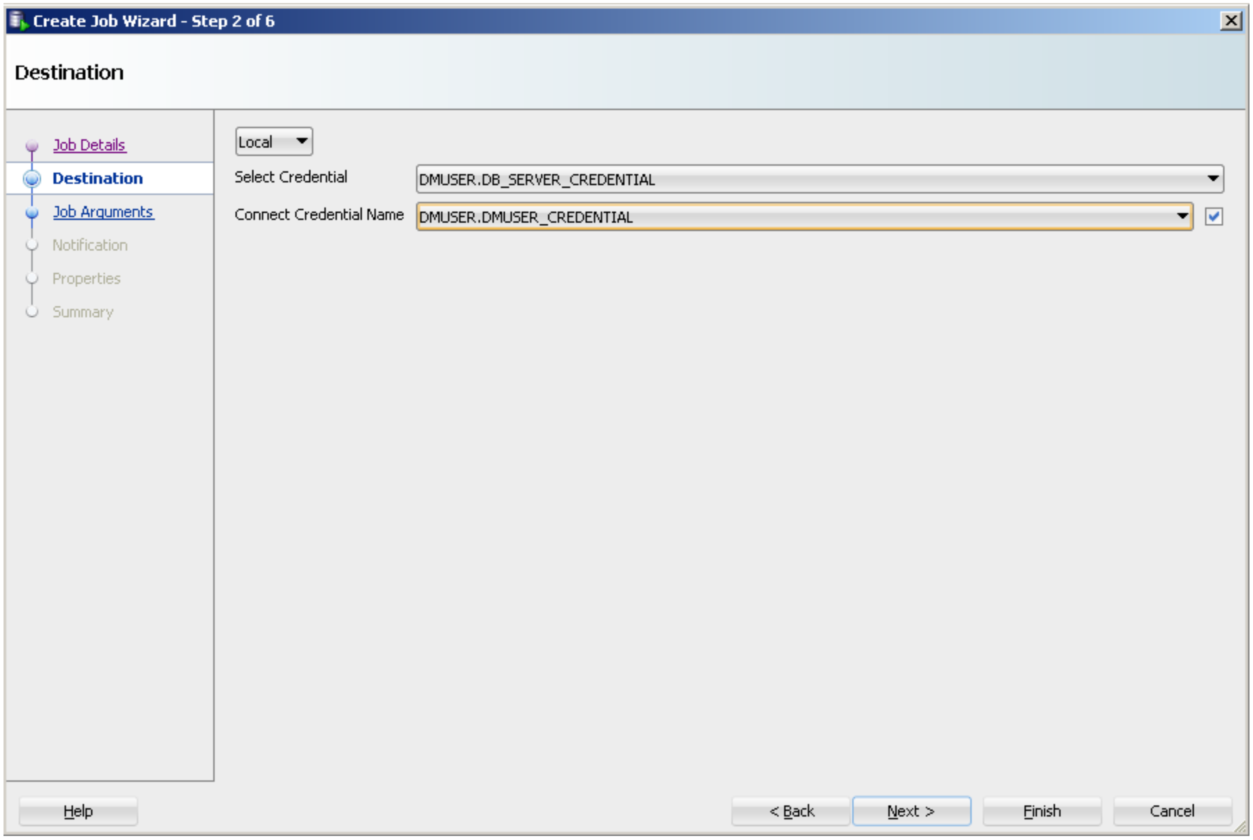
on week day(s): WeekDayNo 1

○ Monday ○ Friday
○ Tuesday ○ Saturday
○ Wednesday ● Sunday
○ Thursday

at time(s): hr:min:sec 12:00:00

FREQ=
MONTHLY;
BYTIME=
120000;
BYDAY=
1SUN

In the Destination step, select the Local database and the credentials created above.



Create Job Wizard - Step 2 of 6

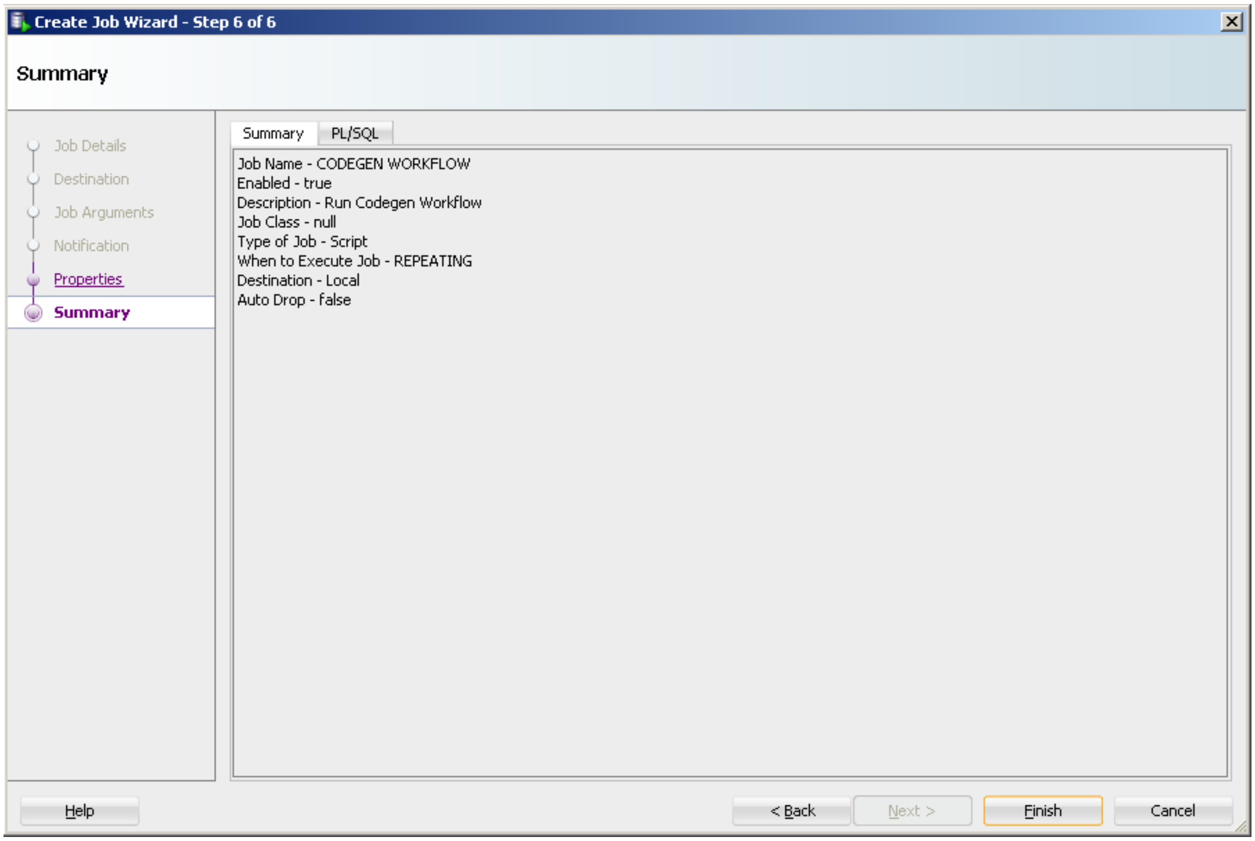
Destination

Local

Select Credential: DMUSER.DB_SERVER_CREDENTIAL

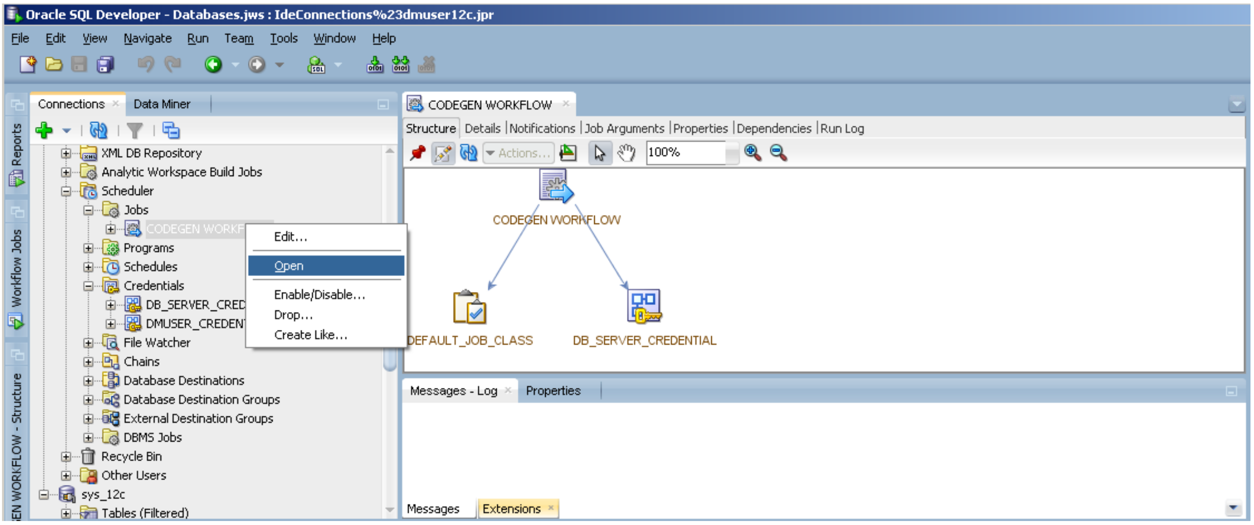
Connect Credential Name: DMUSER.DMUSER_CREDENTIAL ☑

In the Notification step, you can set up an email notification based on the job status. For this to work, email server should be set up in the host machine. For now, we will skip to the Summary step and click Finish to create the job.

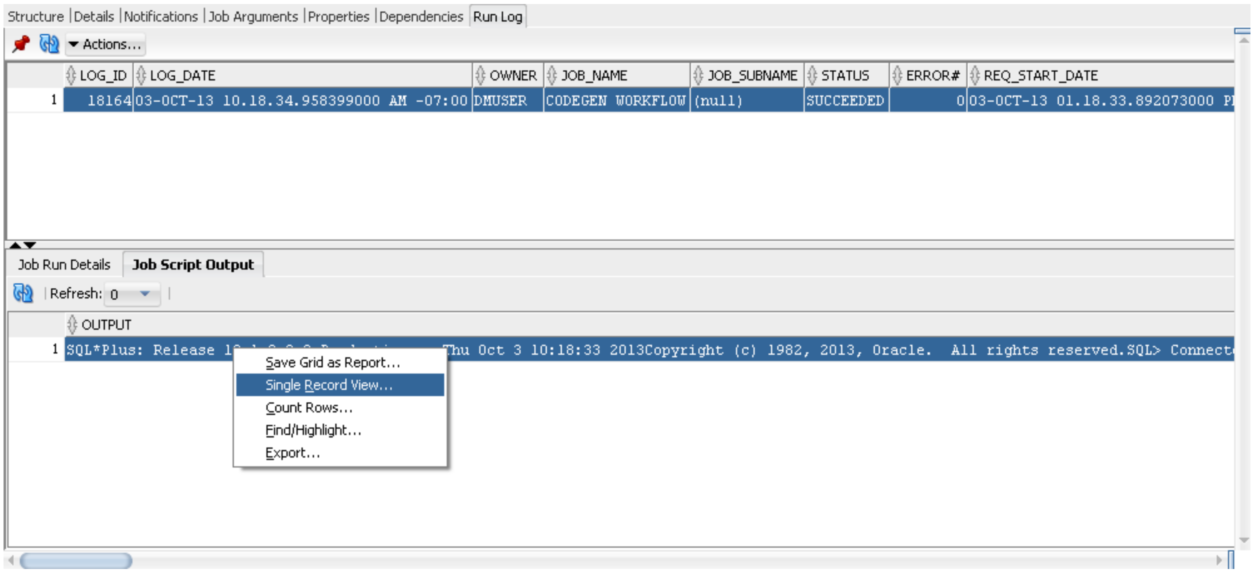


Job Name - CODEGEN WORKFLOW
Enabled - true
Description - Run Codegen Workflow
Job Class - null
Type of Job - Script
When to Execute Job - REPEATING
Destination - Local
Auto Drop - false

Once the job is created, you can monitor it within the SQL Developer. To monitor the job, right-click the job name, CODEGEN WORKFLOW, in the Jobs folder in the Connections navigator to open the job viewer as shown below.



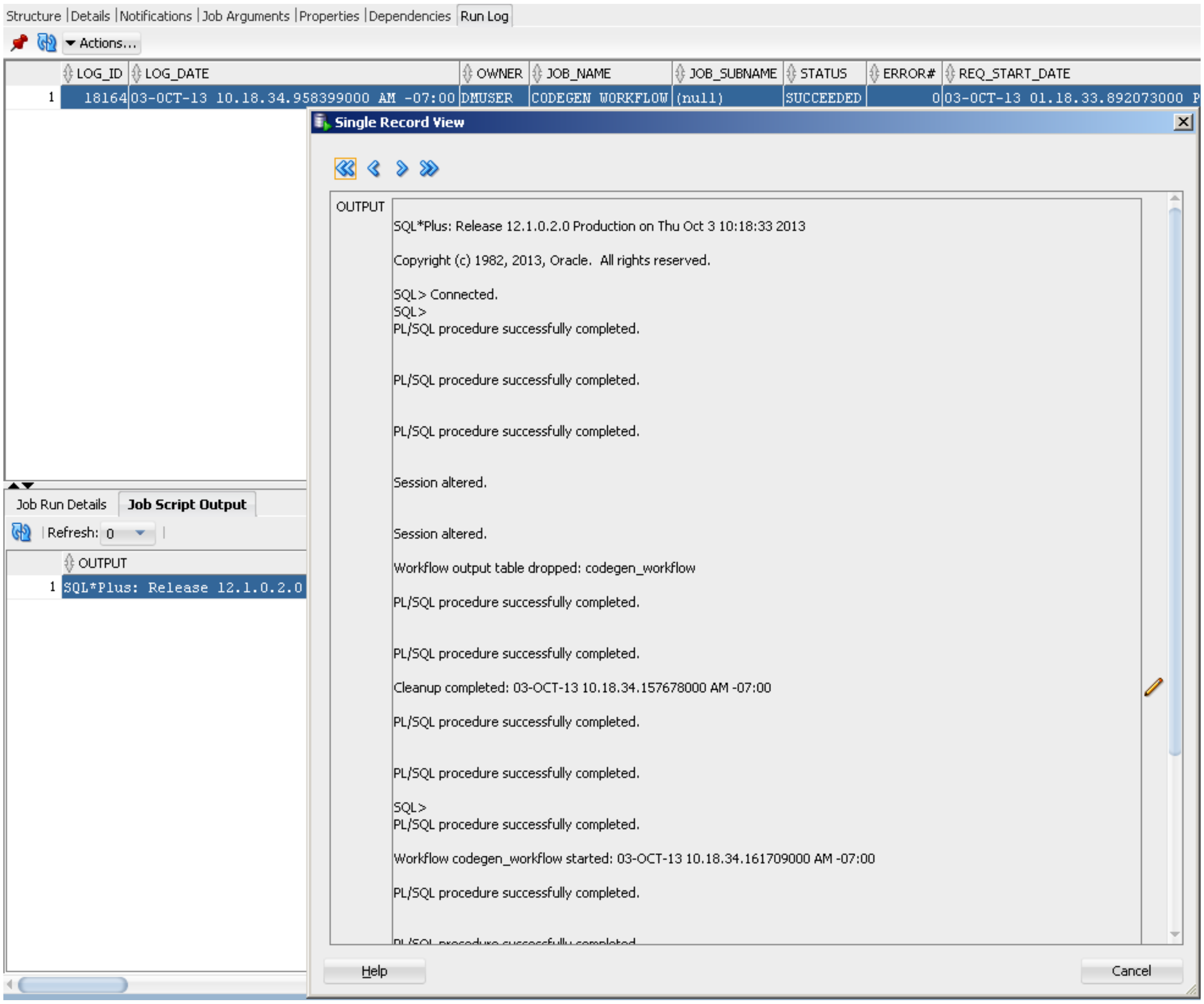
Click the Run Log tab to see the status of the job. Here you can see when the job started running, how long it took to complete, CPU usage, etc. In case of failure, you should see an error message.



	LOG_ID	LOG_DATE	OWNER	JOB_NAME	JOB_SUBNAME	STATUS	ERROR#	REQ_START_DATE
1	18164	03-OCT-13 10.18.34.958399000 AM -07:00	DMUSER	CODEGEN WORKFLOW	(null)	SUCCEEDED	0	03-OCT-13 01.18.33.892073000 PI

	OUTPUT
1	SQL*Plus: Release 1 Thu Oct 3 10:18:33 2013Copyright (c) 1982, 2013, Oracle. All rights reserved.SQL> Connect

To examine the script run result, click the Job Script Output tab in the bottom window, select the result in the OUTPUT column and right-click to bring up the context menu. Then select the Single Row View to bring up the viewer with the script run result.



Structure | Details | Notifications | Job Arguments | Properties | Dependencies | Run Log
Actions...
Single Record View
Job Run Details
Job Script Output
Refresh: 0
Help
Cancel

	LOG_ID	LOG_DATE	OWNER	JOB_NAME	JOB_SUBNAME	STATUS	ERROR#	REQ_START_DATE
1	18164	03-OCT-13 10.18.34.958399000 AM -07:00	DMUSER	CODEGEN WORKFLOW	(null)	SUCCEEDED	0	03-OCT-13 01.18.33.892073000 P

	OUTPUT
1	SQL*Plus: Release 12.1.0.2.0

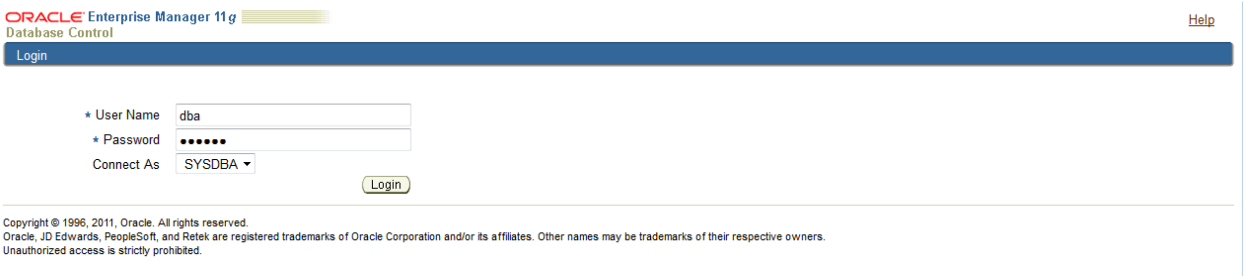
OUTPUT

```
SQL*Plus: Release 12.1.0.2.0 Production on Thu Oct 3 10:18:33 2013

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

SQL> Connected.
SQL>
PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


Session altered.


Session altered.

Workflow output table dropped: codegen_workflow

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

Cleanup completed: 03-OCT-13 10.18.34.157678000 AM -07:00

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

SQL>
PL/SQL procedure successfully completed.

Workflow codegen_workflow started: 03-OCT-13 10.18.34.161709000 AM -07:00

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

Oracle Enterprise Manager Jobs

Oracle Enterprise Manager allows DBAs to define Jobs through the application. The Job definition defines the master script invocation as a script file using a full file path. The user can decide on whether the Job should be run on a schedule or on demand. The job execution can be monitored in the application. The result will either be a success or a reported failure.

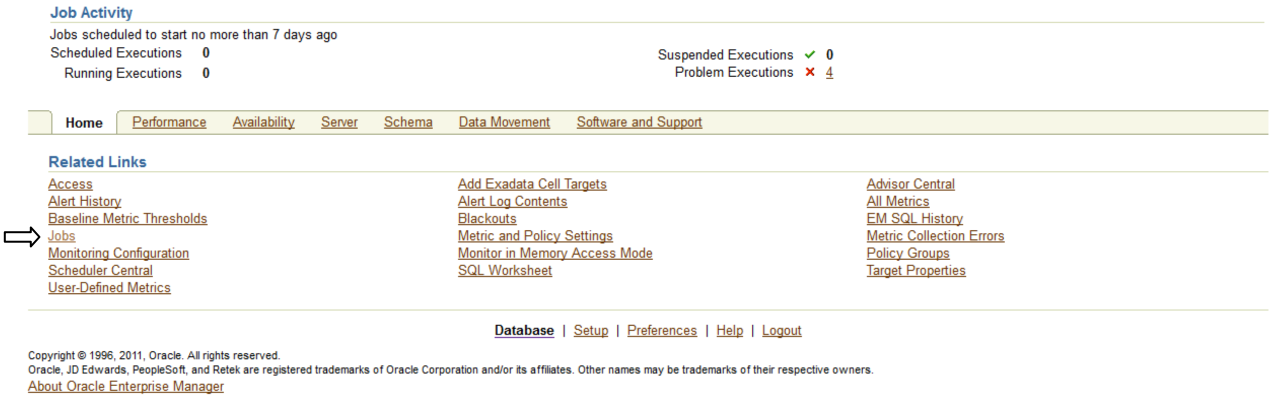
User needs to log in to the Oracle Enterprise Manager using the DBA account.



* User Name: dba
* Password: ••••••
Connect As: SYSDBA

Copyright © 1996, 2011, Oracle. All rights reserved.
Oracle, JD Edwards, PeopleSoft, and Retek are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.
Unauthorized access is strictly prohibited.

Once logged in, use the Jobs link in the Job Activity section at the bottom of the page to launch the Job creation page.



Job Activity

Jobs scheduled to start no more than 7 days ago

Scheduled Executions 0

Running Executions 0

Suspended Executions ✓ 0

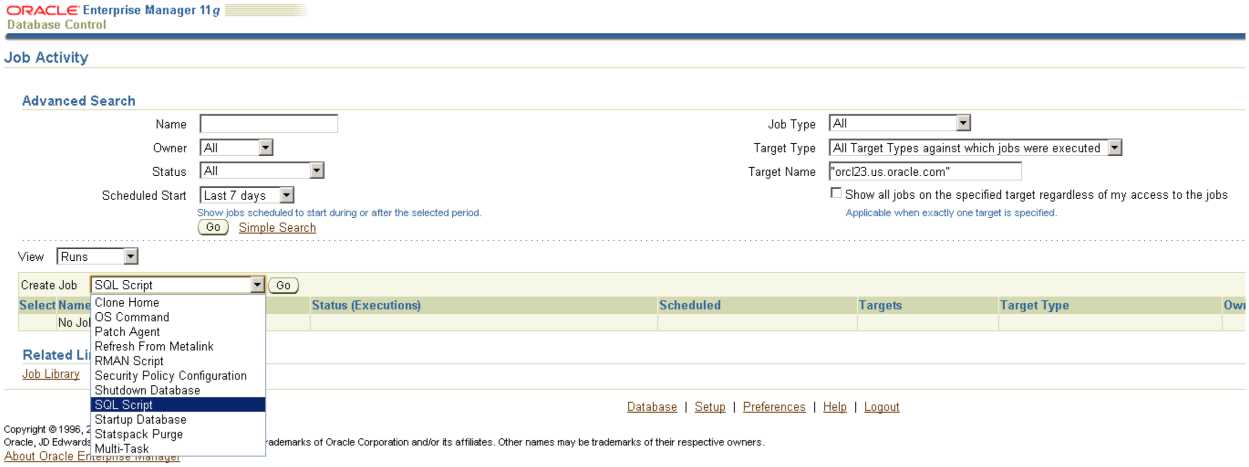
Problem Executions ✗ 4

Related Links

Access	Add Exadata Cell Targets	Advisor Central
Alert History	Alert Log Contents	All Metrics
Baseline Metric Thresholds	Blackouts	EM SQL History
Jobs	Metric and Policy Settings	Metric Collection Errors
Monitoring Configuration	Monitor in Memory Access Mode	Policy Groups
Scheduler Central	SQL Worksheet	Target Properties
User-Defined Metrics		

Copyright © 1996, 2011, Oracle. All rights reserved.
Oracle, JD Edwards, PeopleSoft, and Retek are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.
About Oracle Enterprise Manager

Select the SQL Script type in the Create Job drop-down list and click Go. This will take you to the Create Job page where you can define the new job.



ORACLE Enterprise Manager 11*g*
Database Control

Job Activity

Advanced Search

Name

Owner All

Status All

Scheduled Start Last 7 days

Show jobs scheduled to start during or after the selected period.

Job Type All

Target Type All Target Types against which jobs were executed

Target Name "orcl23.us.oracle.com"

□ Show all jobs on the specified target regardless of my access to the jobs

Applicable when exactly one target is specified.

View Runs

Create Job SQL Script

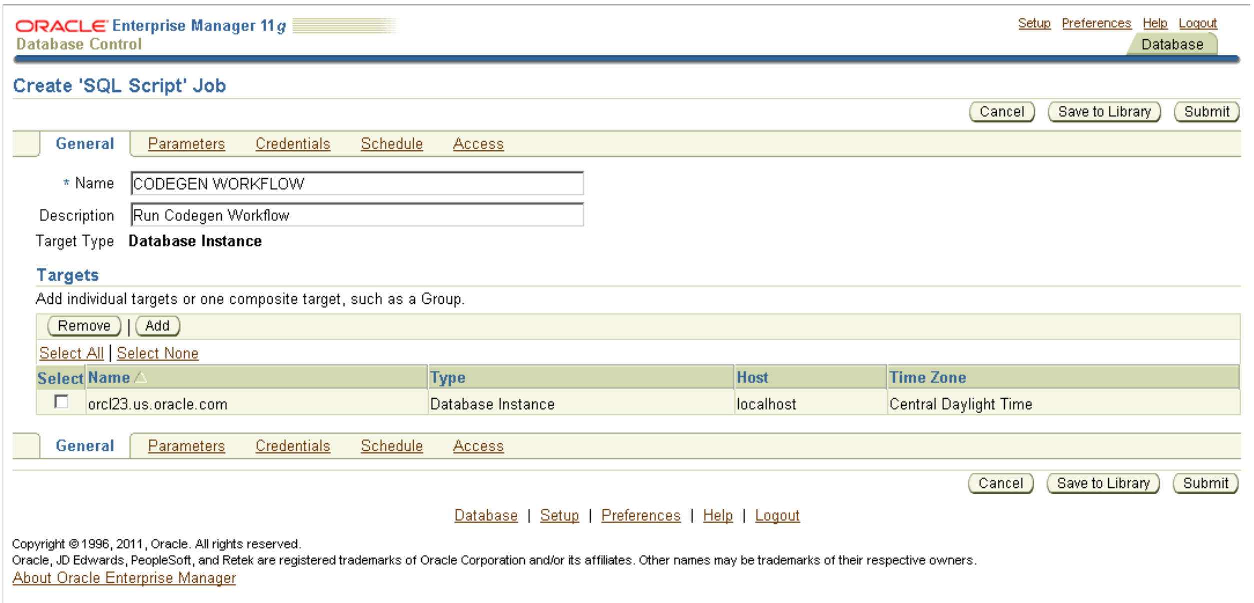
Select	Name	Status (Executions)	Scheduled	Targets	Target Type	Ow
	No Jo					

Related Li

Job Library

Copyright © 1996, 2
Oracle, JD Edwards ...rademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.
About Oracle Enterprise Manager

In the General tab, enter the following Job name and description and add a target database where the job will be run.


ORACLE Enterprise Manager 11g
Database Control
Setup Preferences Help Logout
Database

Create 'SQL Script' Job

Cancel | Save to Library | Submit
General | Parameters | Credentials | Schedule | Access

* Name: CODEGEN WORKFLOW

Description: Run Codegen Workflow

Target Type: **Database Instance**

Targets

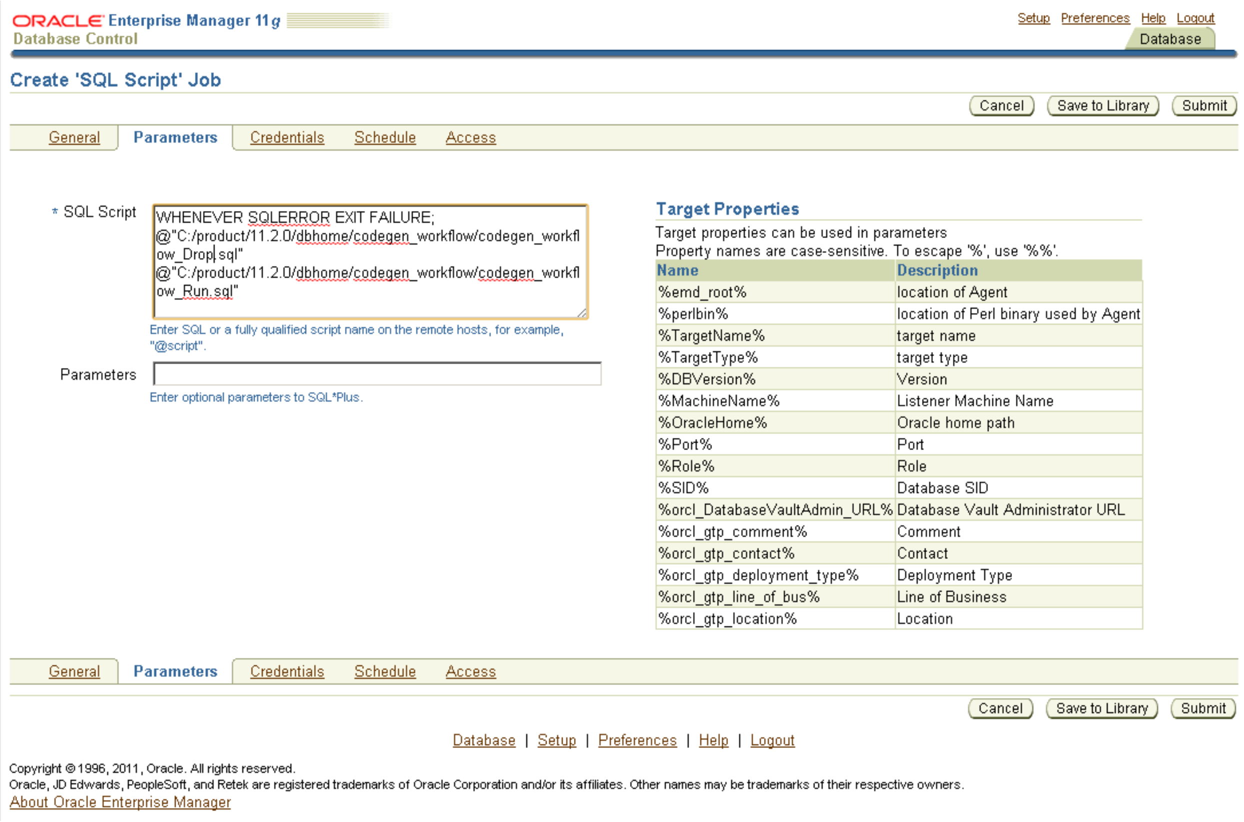
Add individual targets or one composite target, such as a Group.

Remove | Add
Select All | Select None

Select	Name	Type	Host	Time Zone
☐	orcl23.us.oracle.com	Database Instance	localhost	Central Daylight Time

General | Parameters | Credentials | Schedule | Access
Cancel | Save to Library | Submit
Database | Setup | Preferences | Help | Logout

In the Parameters tab, enter the full path names of the cleanup and master scripts in the SQL Script edit control.


ORACLE Enterprise Manager 11g
Database Control
Setup Preferences Help Logout
Database

Create 'SQL Script' Job

Cancel | Save to Library | Submit
General | Parameters | Credentials | Schedule | Access

* SQL Script

```
WHENEVER SQLERROR EXIT FAILURE;
@"C:/product/11.2.0/dbhome/codegen_workflow/codegen_workflow_Drop.sql"
@"C:/product/11.2.0/dbhome/codegen_workflow/codegen_workflow_Run.sql"
```

Enter SQL or a fully qualified script name on the remote hosts, for example, "@script".

Parameters

Enter optional parameters to SQL*Plus.

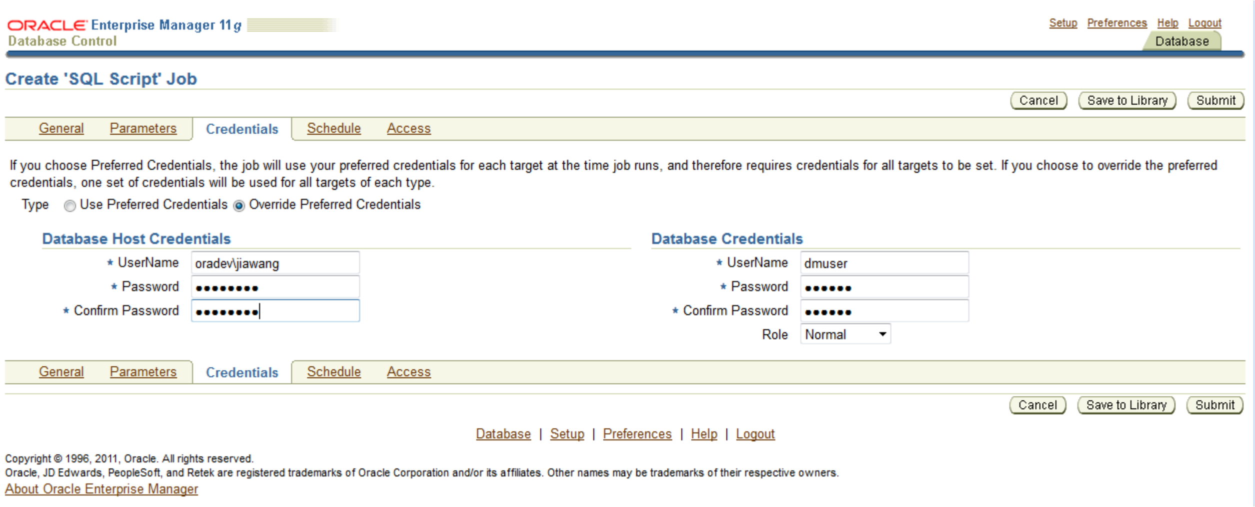
Target Properties

Target properties can be used in parameters
Property names are case-sensitive. To escape '%', use '%%'.

Name	Description
%emd_root%	location of Agent
%perlbin%	location of Perl binary used by Agent
%TargetName%	target name
%TargetType%	target type
%DBVersion%	Version
%MachineName%	Listener Machine Name
%OracleHome%	Oracle home path
%Port%	Port
%Role%	Role
%SID%	Database SID
%orcl_DatabaseVaultAdmin_URL%	Database Vault Administrator URL
%orcl_gtp_comment%	Comment
%orcl_gtp_contact%	Contact
%orcl_gtp_deployment_type%	Deployment Type
%orcl_gtp_line_of_bus%	Line of Business
%orcl_gtp_location%	Location

General | Parameters | Credentials | Schedule | Access
Cancel | Save to Library | Submit
Database | Setup | Preferences | Help | Logout

In the Credentials tab, enter the database host credential and the database credential.



Create 'SQL Script' Job

If you choose Preferred Credentials, the job will use your preferred credentials for each target at the time job runs, and therefore requires credentials for all targets to be set. If you choose to override the preferred credentials, one set of credentials will be used for all targets of each type.

Type ○ Use Preferred Credentials ● Override Preferred Credentials

Database Host Credentials

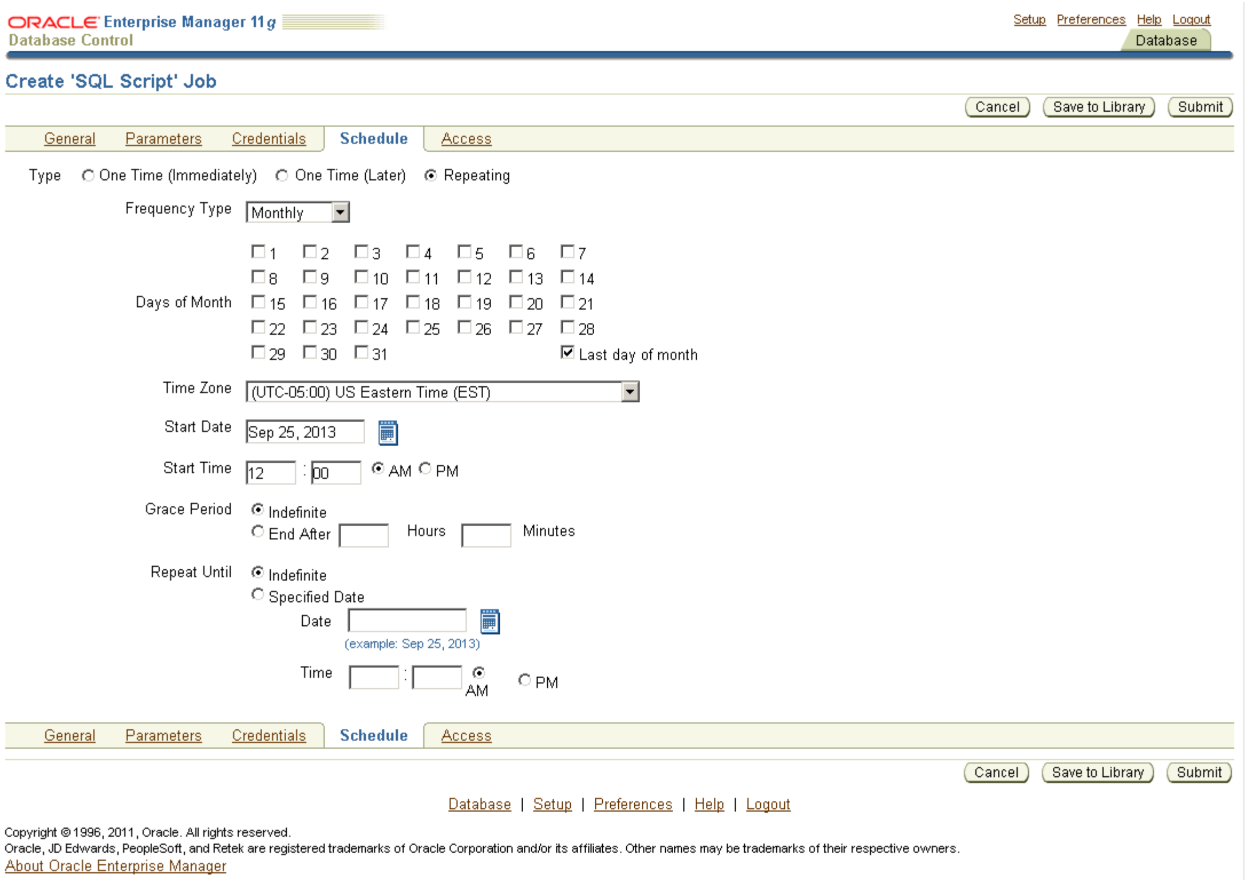
* UserName	oradev\jiawang
* Password	••••••••
* Confirm Password	••••••••

Database Credentials

* UserName	dmuser
* Password	••••••
* Confirm Password	••••••
Role	Normal

About Oracle Enterprise Manager

In the Schedule tab, you can specify if the job will be run one time only or repeating. The following shows the job will be scheduled to run on the last day of each month at midnight.



Create 'SQL Script' Job

Type ○ One Time (Immediately) ○ One Time (Later) ● Repeating

Frequency Type: Monthly

Days of Month:

□ 1 □ 2 □ 3 □ 4 □ 5 □ 6 □ 7
□ 8 □ 9 □ 10 □ 11 □ 12 □ 13 □ 14
□ 15 □ 16 □ 17 □ 18 □ 19 □ 20 □ 21
□ 22 □ 23 □ 24 □ 25 □ 26 □ 27 □ 28
□ 29 □ 30 □ 31 ☑ Last day of month

Time Zone: (UTC-05:00) US Eastern Time (EST)

Start Date: Sep 25, 2013

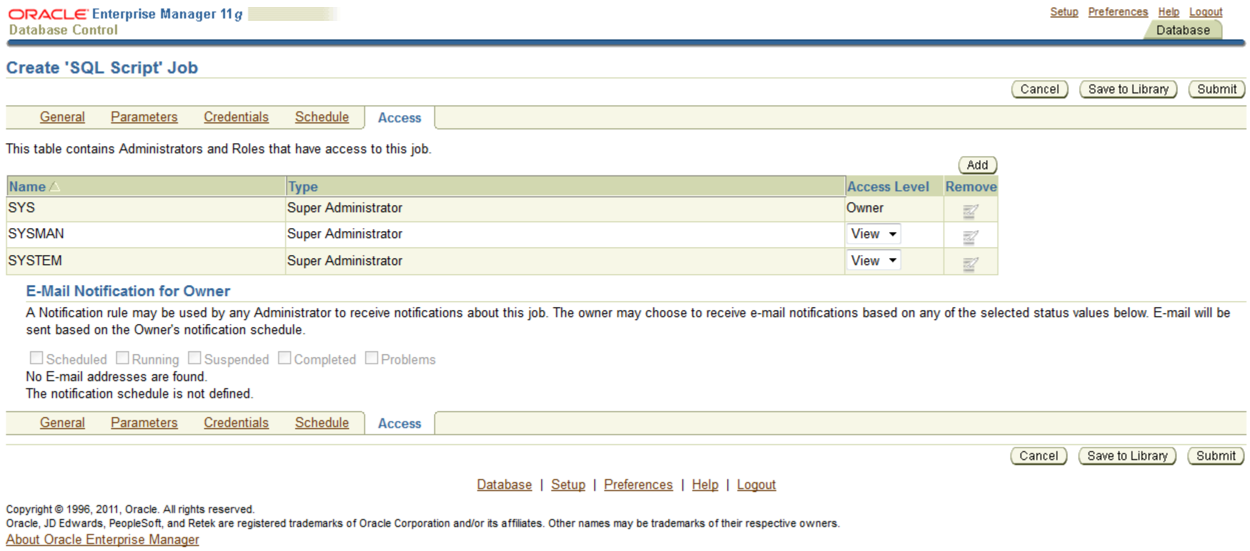
Start Time: 12 : 00 ● AM ○ PM

Grace Period: ● Indefinite
○ End After ___ Hours ___ Minutes

Repeat Until: ● Indefinite
○ Specified Date
Date ___
(example: Sep 25, 2013)
Time ___ : ___ ● AM ○ PM

About Oracle Enterprise Manager

In the Access tab, you can set up an email notification based on the job status. For this to work, email server should be set up in the host machine.


ORACLE Enterprise Manager 11g
Database Control
Setup Preferences Help Logout
Database
Cancel
Save to Library
Submit
General Parameters Credentials Schedule Access
Add

Create 'SQL Script' Job

This table contains Administrators and Roles that have access to this job.

Name	Type	Access Level	Remove
SYS	Super Administrator	Owner	
SYSMAN	Super Administrator	View	
SYSTEM	Super Administrator	View	

E-Mail Notification for Owner

A Notification rule may be used by any Administrator to receive notifications about this job. The owner may choose to receive e-mail notifications based on any of the selected status values below. E-mail will be sent based on the Owner's notification schedule.

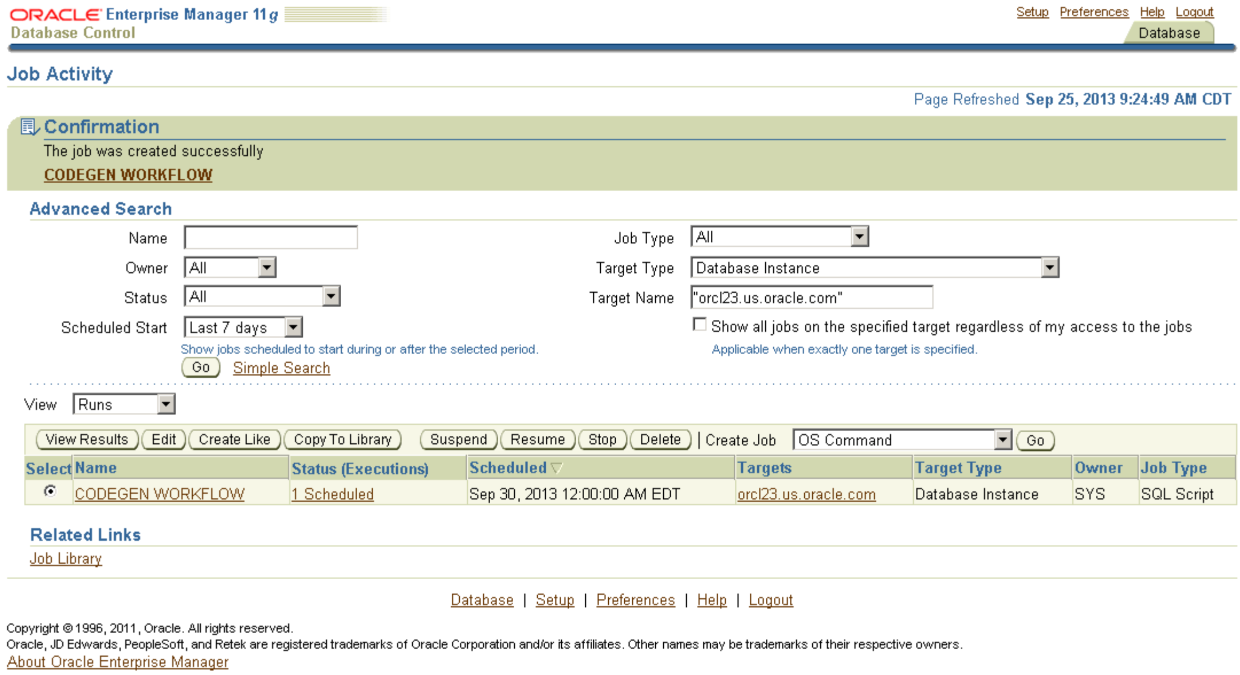
□ Scheduled □ Running □ Suspended □ Completed □ Problems

No E-mail addresses are found.
The notification schedule is not defined.

General Parameters Credentials Schedule Access
Cancel
Save to Library
Submit
Database | Setup | Preferences | Help | Logout

About Oracle Enterprise Manager

Once you have entered all the settings, click Submit to create the job.



Job Activity

Page Refreshed **Sep 25, 2013 9:24:49 AM CDT**

Confirmation
The job was created successfully
CODEGEN WORKFLOW

Advanced Search

Name		Job Type	All
Owner	All	Target Type	Database Instance
Status	All	Target Name	"orcl23.us.oracle.com"
Scheduled Start	Last 7 days		□ Show all jobs on the specified target regardless of my access to the jobs

Show jobs scheduled to start during or after the selected period.
Applicable when exactly one target is specified.

View: Runs

| Create Job: OS Command | Go

Select	Name	Status (Executions)	Scheduled	Targets	Target Type	Owner	Job Type
●	CODEGEN WORKFLOW	1 Scheduled	Sep 30, 2013 12:00:00 AM EDT	orcl23.us.oracle.com	Database Instance	SYS	SQL Script

Related Links

Job Library

About Oracle Enterprise Manager

Conclusions

This white paper shows how easy it is to deploy Data Miner workflow to the target or production database using the SQL script generation feature. Furthermore, Oracle Enterprise Manager in the target database can be used to schedule the scripts to run in any desirable time interval. Given that Oracle's data mining and advanced analytics operate natively inside the Oracle Database, mining insights and predictions also remain inside the database so can be accessed by SQL queries from OBIEE. Predictive model results can be called interactively using OBIEE reports and dashboards. With Oracle, all the data mining and advanced analysis and results stay inside the database providing a simpler architecture, better security, better scalability and a single source of truth.

Appendix

The following table describes the objects that are generated by each node script.

Script Type	Output
Create Table Node	• Creates a table or view that persists the input data
Data Explore Node	• Creates a table that contains statistical data of the input data
Aggregate Node Anomaly Detection Query Node (12c) Apply Node Apply Text Node Clustering Query Node (12c) Data Source Node Feature Extraction Query Node (12c) Filter Columns Details Node Filter Rows Node Join Node Model Details Node Prediction Query Node (12c) Sample Node SQL Query Node Transform Node	• Creates a view reflecting the output of the node
Filter Columns Node	• Creates a view reflecting the output of the Column Filter node (like other transform type nodes) • If the Attribute Importance setting is specified, a table is generated containing the AI result
Build Text Node	• For each text transformation the following objects are created ○ Feature Table ○ Oracle Text Objects (Policy, Lexer, Stoplist) • Creates a view reflecting the output of the Build Text node. This is essentially the same output as the Apply Text node
Classification Build Node	• Create a Build and Test (if necessary) data • For each imbedded text transformation (12c), Oracle Text Objects are created (Policy, Lexer, Stoplist) • A model is created for each model build specification • GLM Model Row Diagnostics Table (if row diagnostics turned on) • Each Model Test will have one table generated for each of the following test results ○ Test Metric ○ Confusion Matrix ○ ROC (binary classification only)

	- Each Model Test will have one table for each of the following tests per target value (up to 100 maximum target values) o Lift
Regression Build Node	- Create a Build and Test (if necessary) data - For each imbedded text transformation (12c), Oracle Text Objects are created (Policy, Lexer, Stoplist) - A model is created for each model build specification - GLM Model Row Diagnostics Table (if row diagnostics turned on) - Each Model Test will have one table generated for each of the following test results o Test Metric o Residual Plot data
Anomaly Detection Build Clustering Build Feature Extraction Build	- For each imbedded text transformation (12c), Oracle Text Objects are created (Policy, Lexer, Stoplist) - A model is created for each model build specification
Association Build	- A model is created for each model build specification
Test Node (Classification)	- Same specifications as Class Build except no models are created
Test Node (Regression)	- Same specifications as Regression Build except no models are created
Model Node Text Reference Node Graph Node	- No code generated