PhUSE US Connect 2019

Paper SD07

Building SQL Scripts Using Google Sheets, R and R Shiny

Huiming Tu, FHL Consulting, Chesterbrook, PA, USA
Hanming Tu, TuCai Consulting, Mullica Hill, NJ, USA

ABSTRACT

Creating a complex data model requires not only careful design but also visual tools. Some of the commercial tools are quite expensive as well. This paper will explore free and ready available tools such as Google Sheets, R and R shiny to design a framework to easily generate SQL scripts for relational Oracle database and NoSQL Mongo database. Once the scripts are generated, you can use the scripts to create your databases or import into your favorite data modeling tool to further develop or visualize the data model.

INTRODUCTION

The correct data model and consistent data modeling is of great importance for business users to make quick and well informed decisions. The entities and relationships of data need to be defined and structured to ensure best results. The life science industry took over 10 years to develop data standards such as SDTM from CDISC but the Janus data model used to store standardized clinical data is still not widely used or flexible to accommodate deviation and complexity of real world studies.

It is not easy to convert our understanding of this world into a digital model. This paper explores a simple model for data modeling elements and an easy way to produce data model using free tools such as Google sheets, R and R shiny. This paper will include the following sections:

1. Data model and data modeling
2. Simple data model for data modeling
3. Google sheets as data store
4. R and R shiny as tool
5. PL/SQL scripts for Oracle database
6. JSON scripts for MongoDB
7. Visualization of data models
8. Conclusion

DATA MODEL AND DATA MODELING

According to wikipedia, a data model is an abstract model that organizes elements of data and standardizes how they relate to one another and to properties of the real world entities. For instance, CDISC Study Data Tabulation Model (SDTM) is a data model (standard) for organizing and formatting data to streamline processes in collection, management, analysis and reporting. Traditionally, data models have been built during the analysis and design phases of a project to ensure that the requirements for a new application are fully understood. Data models can also be invoked later in the data lifecycle to rationalize data designs that were originally created by programmers on an ad hoc basis. It is evolving over time and may need a convenient and easy tool or system to manage it.

Data modeling is an important skill for data scientists or others involved with data analysis and is the process of documenting a complex software system design and creating entity and relationship (ER) diagram. There are some very sophisticated systems developed just to create the ER digram. Here we are going to explore a simple way to document the data entity and relationship using some free tools.

SIMPLE DATA MODEL FOR DATA MODELLING

THE SIMPLE DATA MODEL

Here is a simple data model for storing metadata to be used for data modelling. We can use the metadata to generate PL/SQL or JSON files for creating tables or collections; then we can use the generated code to create diagram or actual tables or collections.

variable	seq	label	type	req	length	kt	fk	desc
table	0		collection					Collection of table variables
id	1	Variable ID	numeric	T		pk		Unique identifier for table variable
db	2	Database Name	string	T	50			Database name
collection	3	Collection Name	string	T	50			Collection / Table name
variable	4	Variable Name	string	T	50			Variable name
seq	5	Variable Sequence	numeric	T				Variable sequence
label	6	Variable Label	string		100			Variable label
hidden	7	Hidden Variable	string		5			Whether to hide the variable from displaying
type	8	Variable Type	string	T	20			Variable type
req	9	Required?	string		5			Whether the variable is required(T), permissible (P) or optional (O, blank)
ref_def	10	Definition Reference	string		20			Definition referenced by this variable
default	11	Default Value	string		100			Default value
mode	12	Variable Mode	string		20			Variable mode such as 64 bit or 32 bit or string array, etc.
unit	13	Value Unit	string		50			Variable unit
min	14	Min Value	string		100			Min value
max	15	Max Value	string		100			Max value
length	16	Variable Length	numeric					Variable Length
kt	17	Key Type	string		20			Key type such primary key (pk), foreign key (fk), etc.
fk	18	Foreign Key	string		200			Foreign key in the format of collection.variable
cardinality	19	Cardinality	string		20			Cardinality such one-to-one ([1:1]), one-to-many ([1:n]), zero-to-many ([0:n]), etc
opts	20	Optional Values	string		200			Optional values
typedef	21	Type Definition	string		20			Variable type definition used to validate the variable
req_msg	22	Requirement Message	string		200			Message displayed with validating rule

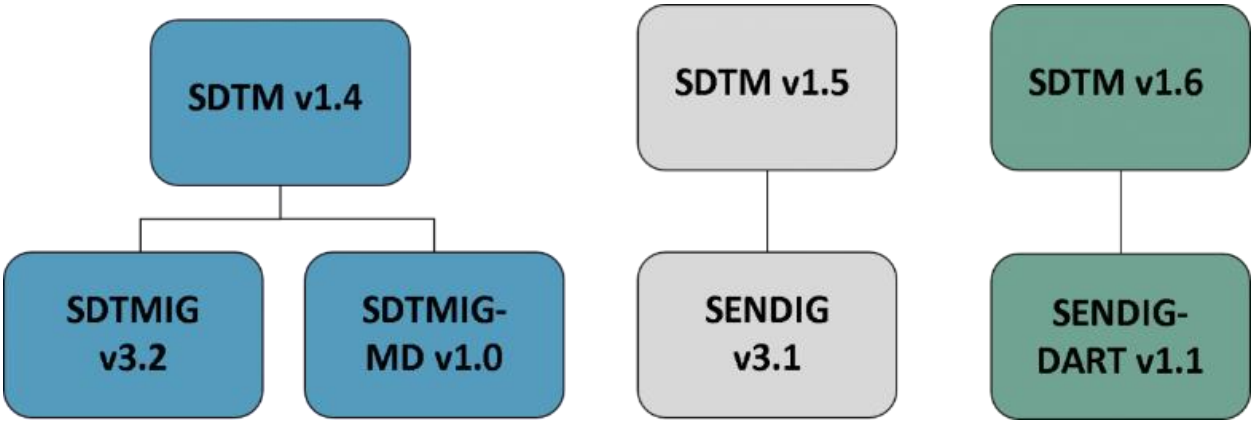
desc	23	Description	string		1000			Description of the definition

MODELLING A METADATA MODEL FOR SDTM

We use CDISC SDTM standards as an example to demonstrate how to use Google sheets, R and R Shiny to create a data model for storing the metadata of the SDTM standards. This data model has three entities:

Standard Version (m_std_versions)

The CDISC SDTM has many different versions. The m_std_versions wil contains the version information.



Standard Domain (m_std_domains)

This domain entity contains the classes and domains for each version.

Standard Variable (m_std_variables)

The variable entity contains all the variables for each domain.

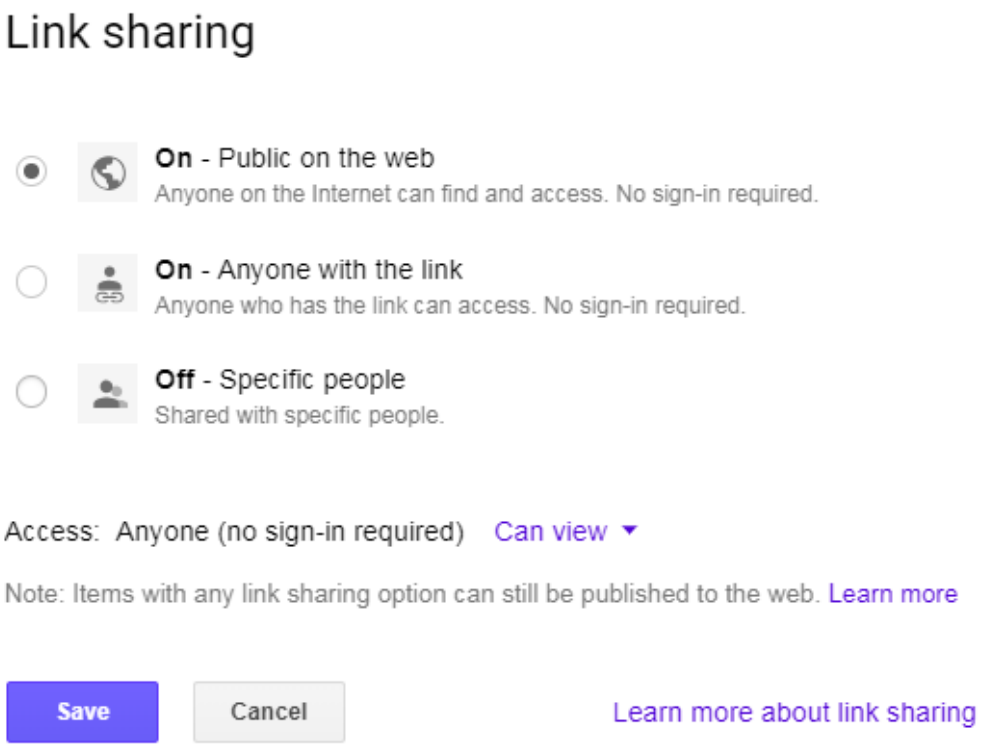
All the metadata for these entities and their relationships are stored in Google Sheet: https://docs.google.com/spreadsheets/d/1ZTycKUaMhh4gbgNg814zBqszsrqN6kHbvAvjCwYwm20.

GOOGLE SHEETS AS DATA STORE

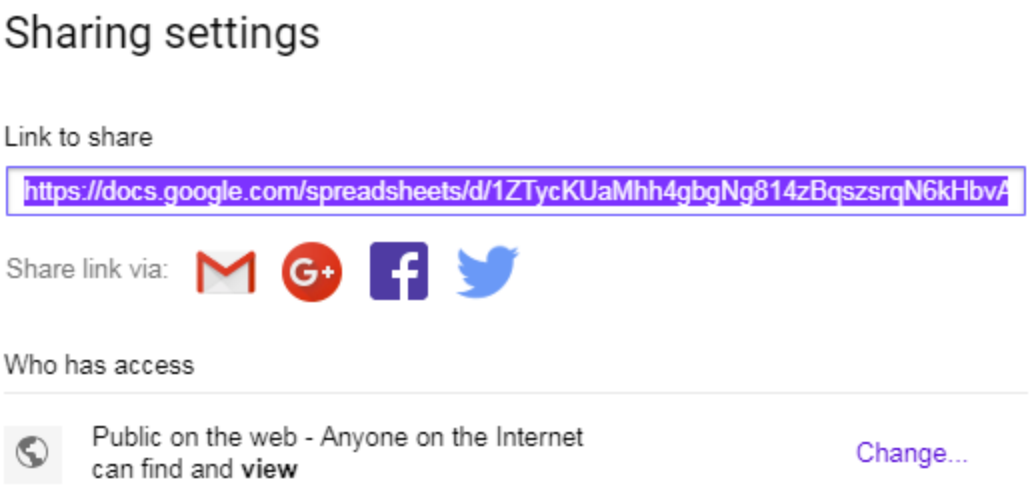
Where can we store the metadata? After trying a few different options, we chose Google Sheets. Google Spreadsheets is a Web-based application that allows users to create, update and modify spreadsheets and share the data live online. There are two steps to make your Google sheets shared and published to the public:

STEP 1: SHARING THE GOOGLE SHEET

1. Open the metadata Google Sheet. Mouse over the Share button. It shows “Private to only me”.
2. Click “SHARE” and then click “Advanced”
3. Click “On - Public on the web” under the Link Sharing



4. Click on “Save”



5. Mouse over the “SHARE” and it should display “Public on the web”

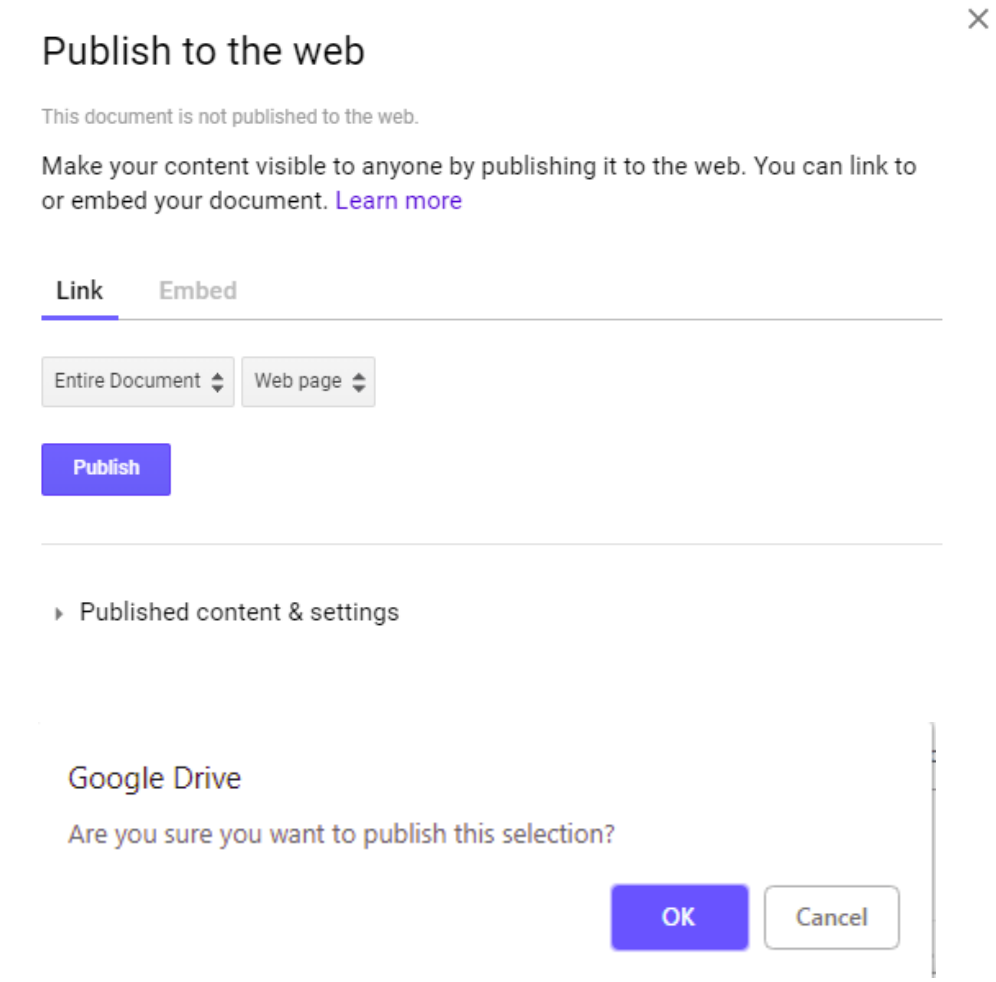
STEP 2: PUBLISH THE GOOGLE SHEET

To make a document, spreadsheet, or presentation available for a large audience to see, publish the file. After you publish your file you can send a new URL to anyone or embed into your website.

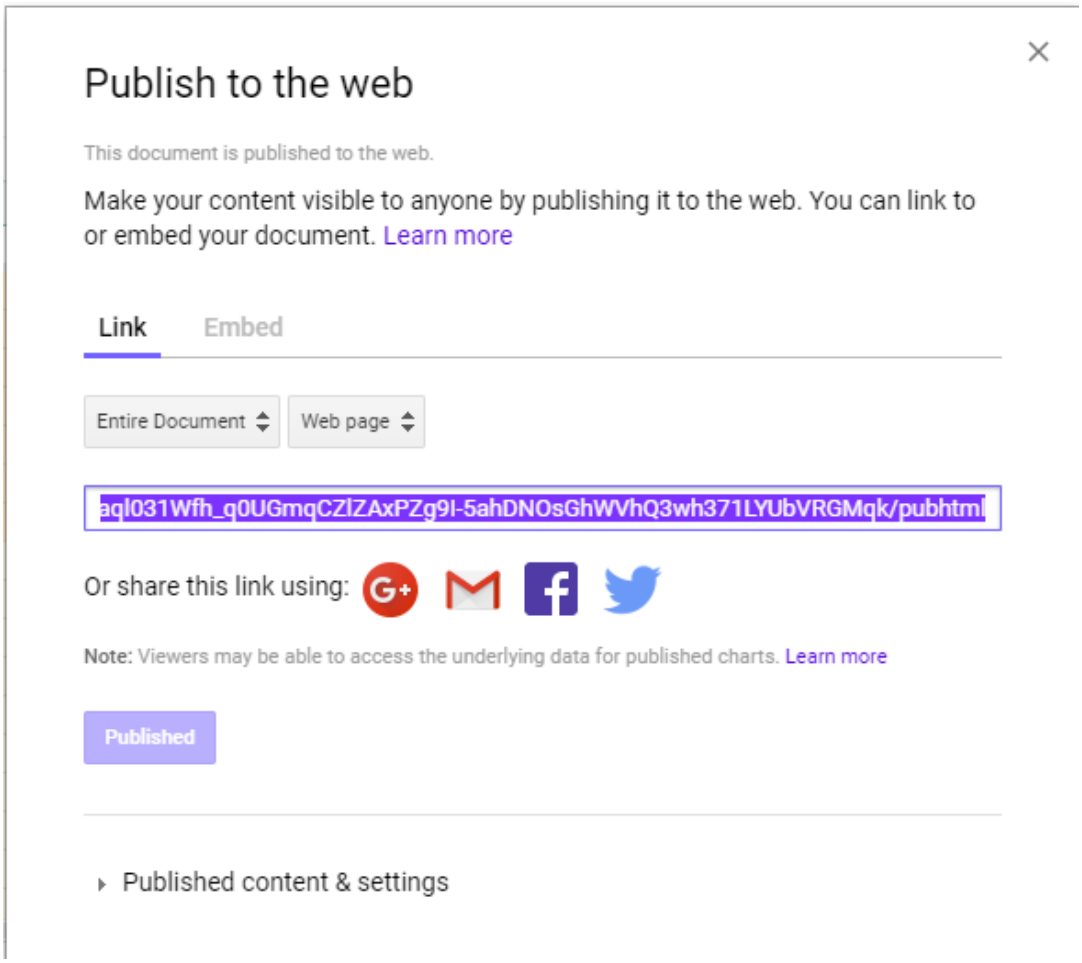
Publish the file

1. In Google Docs, Sheets, or Slides, open a file.
2. At the top, click File > Publish to the web.
3. Choose a publishing option:
   - Spreadsheet: Publish the entire spreadsheet or individual sheets. You can also choose a publishing format.
   - Presentation: Choose how quickly to advance the slides.
4. Click Publish.
5. Copy the URL and send it to anyone you'd like to see the file. Or, embed it into your website.

Publish to the Web



Published Link



https://docs.google.com/spreadsheets/d/e/2PACX-1vQJjAgs5Zd_VeTR37Pdj-Saql031Wfh_q0UGmqCZlZAxPZg9I-5ahDNOsGhWVhQ3wh371LYUbVRGMqk/pubhtml

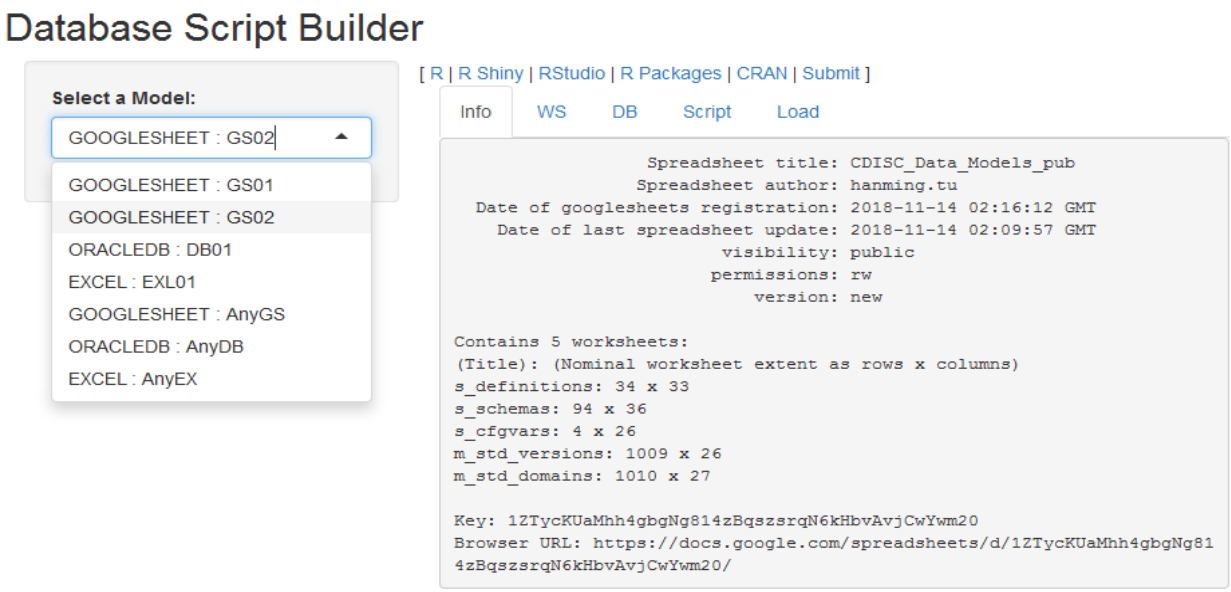
Do not publish anything that you do not want others to see. This link will let everyone in the World be able to see your data.

R AND R SHINY AS A TOOL

We use R and R Shiny built a tool called Database Script Builder to use the data model metadata to generate SQL and JSON codes.

DATABASE SCRIPT BUILDER UI

Here is the user interface (UI) of the database script builder:



WAYS TO PROVIDE METADATA MODELS

The DB script builder can read published Google sheets and Microsoft Excel files. It has two ways for you to provide the metadata sheets

Configure the YML file

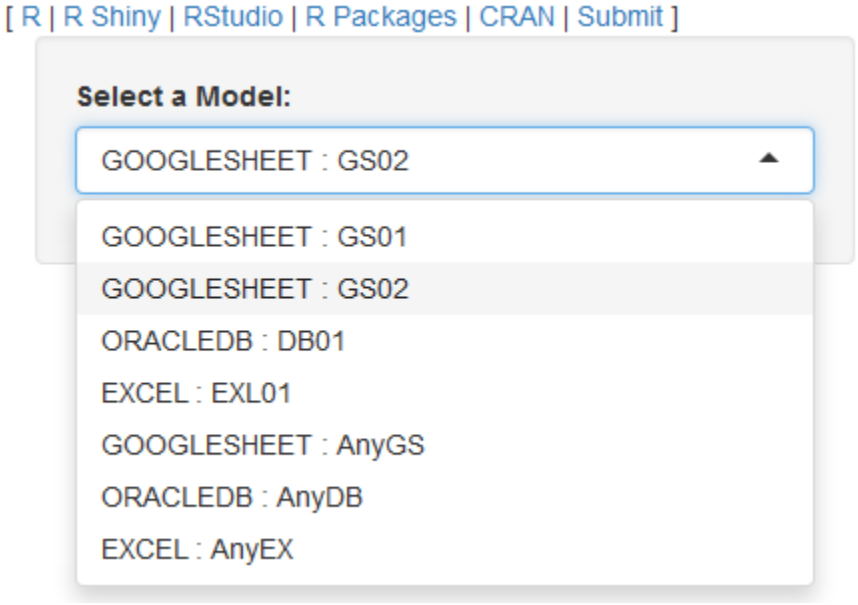
You can configure the source models in the YML file. Here is an example:

```
Keywords: mongoDB, configuration
Script:
  name    : bldsql.yml
  title   : Configuration file for bldSQL web application
  desc    : >
          This is the configuration file for bldSQL web application.
          It contains the connection information to the mongoDB.
  version: 0.1.0
Language:
  name    : YML
  version: x.x.x
Environment:
  system: Linux or Window 2010
  os_version: OEL 5.8, Window 7
  desc: This is built in Window 7 environment with mongoDB 3.4 window x64 version.
  debug:
    msg_lvl: 3
    log_lvl: 1
    write2log: FALSE
models:
  GS02:
    typ: googlesheet
    src: 'https://docs.google.com/spreadsheets/d'
    val: '1d6teWmhjrrgsGJzjuk4idjs0c-un1N26pd4bYZu-ZIg'
    def: '1ZTycKUaMhh4gbgNg814zBqszsrqN6kHbvAvjCwYwm20'
  EXL01:
    typ: excel
    src: "C:/Users/htu/gDrive/BuzDocs/Ashanda/Models"
    val: sys_values.xlsx
```

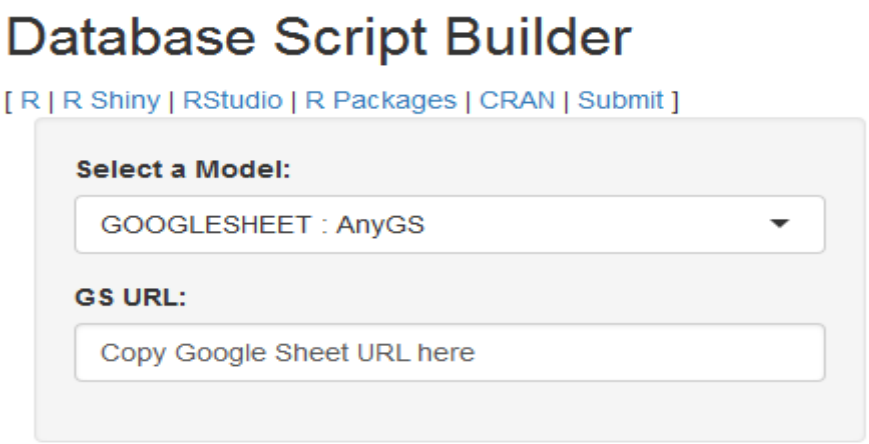
```
    def: sj_design.xlsx
  AnyGS:
    typ: googlesheet
    src: 'https://docs.google.com/spreadsheets/d'
    val: $val
    def: $def
  AnyDB:
    typ: oracledb
    src: $cs
    val: $val
    def: $def
  AnyEX:
    typ: excel
    src: $path
    val: $val
    def: $def
dbs:
  mongo_mac:
    typ: mongo
    imp: "/Users/htu/Applications/mongodb264/bin/mongoimport"
    prn: "/Users/htu/Applications/mongodb264/bin/mongo"
    svr: localhost
    port: 27017
    user:
    pwd:
    outdir : /Users/htu/Desktop/myGithub/pkgs/ashanda/trunk/scripts
  mongo_pc:
    typ: mongo
    imp: "C:/Program Files/MongoDB/Server/3.4/bin/mongoimport.exe"
    prn: "C:/Program Files/MongoDB/Server/3.4/bin/mongo.exe"
    svr: localhost
    port: 27017
    user: htu_root
    pwd: xxxxx
    outdir: "C:/myCodes/pkgs/ashanda/trunk/scripts"
  oracle_pc:
    typ: oracle
    imp: "C:/myApps/oracle/product/12.1.0/client_1/BIN/sqlldr.exe"
    prn: "C:/myApps/oracle/product/12.1.0/client_1/BIN/sqlplus.exe"
    svr: xe
    port: 1215
    usr: fbp_admin
    pwd: fbp2admin
    outdir: "C:/myCodes/pkgs/ashanda/trunk/scripts"
Outputs:
  datasets: out1, out2, out3
  v1: Date - scription execution date and time
  v2: User - user who executes the script
# end of file
```

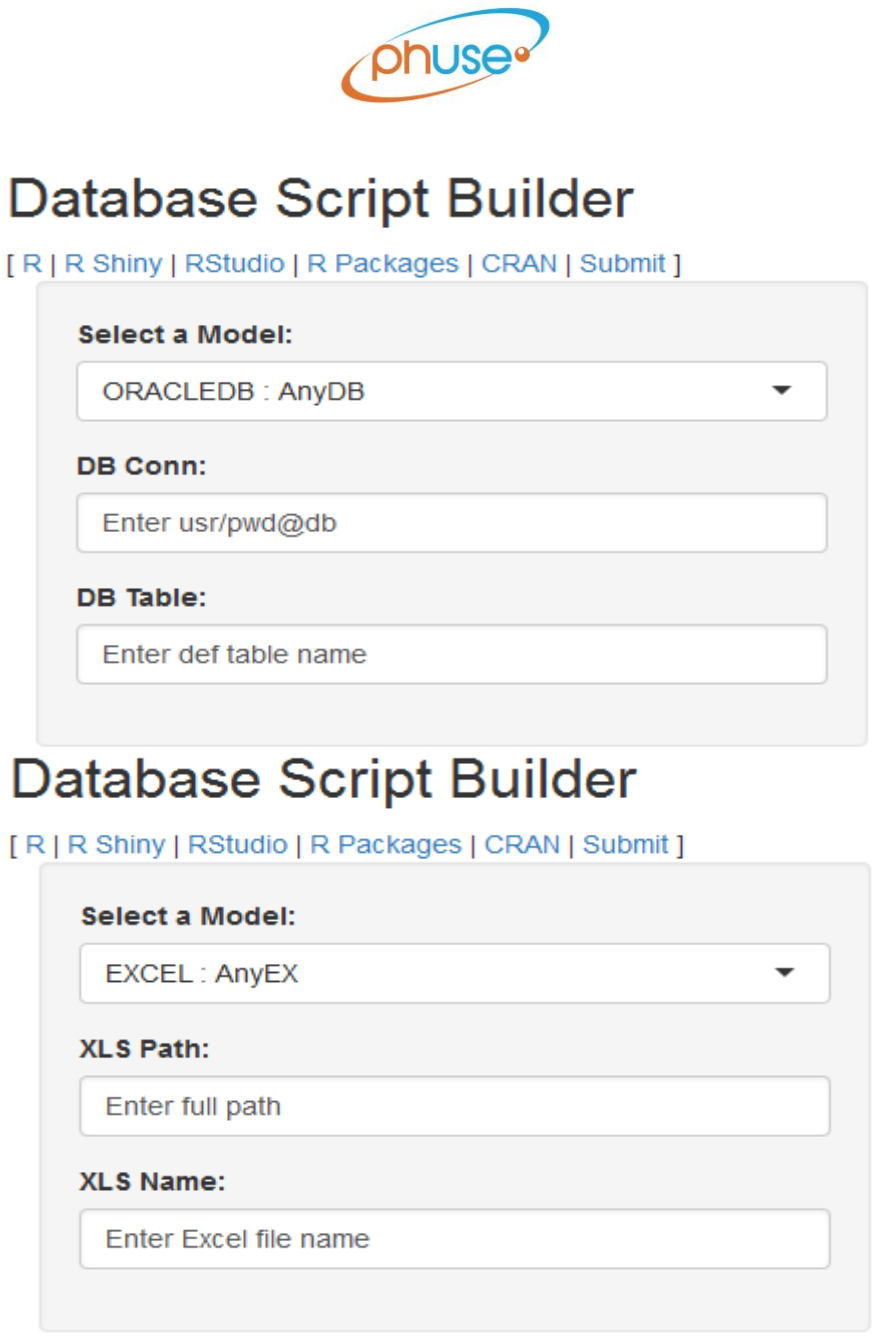
Provide through the UI

Here shows the pre-configured models:



Here are the three type of sources that could have the model definitions:





In these model metadata sources such as Google sheet, MS Excel or Oracle database, this R Shiny app expects the **s_schemas**, **s_definitions** and **s_cfgvars** tabs in Google sheets and MS Excel or tables in Oracle database.

FUNCTIONAL TABS

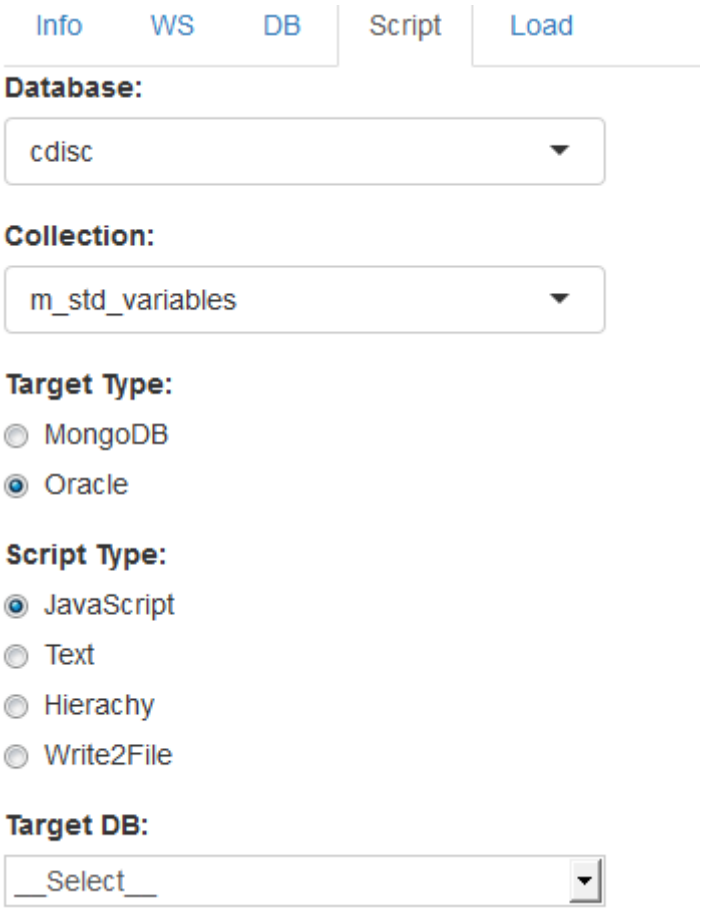
The database script builder has the following functional tabs:

- **Info**: displays the information about the model source.
- **WS**: This is worksheet tab which allows you to view the content of each worksheet in the Google sheets, MS Excel or Oracle DB.
- **DB**: This is the database tab which allows you to view all the table/collection definitions in the model.
- **Script**: allows you to generate Oracle SQL or MongoDB JSON scripts.
- **Load**: allows you to load metadata into target database or insert the data into Google Sheets or MS sheets.

SQL SCRIPTS FOR ORACLE DATABASE

The following screenshot shows the user interface for generating MongoDB JSON or Oracle PL/SQL scripts. Here are the steps for you to proceed:

1. Select a database if you have defined multiple databases in your schema definition table; it defaults to the first database name.
2. Select a collection/table name
3. Choose a target database type: MongoDB or Oracle
4. The script type only impact the script generated for MongoDB
5. If you selected a target database, it will try to connect to the target database and execute the generated scripts.



The Oracle PL/SQL scripts generated for the data model for hosting CDISC SDTM standards are listed in the subsequent sections.

SCRIPT FOR CREATING M_STD_VERSIONS

```
PROMPT Drop objects in  cdisc ...
-----**************************************************************************-----
DROP TABLE m_std_versions CASCADE CONSTRAINTS;
DROP SEQUENCE m_std_versions_sq ;

PROMPT Create objects in  cdisc ...
-----**************************************************************************-----

 PROMPT Creating table m_std_versions ...
--------------------------------------------------------------------------------
-- create objects --
CREATE TABLE m_std_versions (
```

```
    "V_ID"          NUMBER           PRIMARY KEY,
    "P_VID"         NUMBER          ,
    "SDO"           VARCHAR2(100)    NOT NULL,
    "CLASS"         VARCHAR2(50)     NOT NULL,
    "NAME"          VARCHAR2(20)     NOT NULL,
    "VERSION"       VARCHAR2(20)     NOT NULL,
    "DT_RELEASED"   DATE             NOT NULL,
    "DT_ENFORCED"   DATE            ,
    "NOTE"          VARCHAR2(4000)
);

COMMENT ON TABLE m_std_versions IS
  'Collection of standard versions';

 COMMENT ON COLUMN m_std_versions."V_ID" IS
  'Unique identifier for standard or model version';
 COMMENT ON COLUMN m_std_versions."P_VID" IS
  'Parent ID for version id';
 COMMENT ON COLUMN m_std_versions."SDO" IS
  'Standard development organization';
 COMMENT ON COLUMN m_std_versions."CLASS" IS
  'Standard class';
 COMMENT ON COLUMN m_std_versions."NAME" IS
  'Model name';
 COMMENT ON COLUMN m_std_versions."VERSION" IS
  'Model version';
 COMMENT ON COLUMN m_std_versions."DT_RELEASED" IS
  'Date released';
 COMMENT ON COLUMN m_std_versions."DT_ENFORCED" IS
  'Date enforced';
 COMMENT ON COLUMN m_std_versions."NOTE" IS
  'Note or description';

PROMPT Creating sequence m_std_versions_sq...
CREATE SEQUENCE m_std_versions_sq
  START WITH 1
  INCREMENT BY 1
  NOCACHE NOCYCLE;
-- Ended for m_std_versions
```

SCRIPT FOR CREATING M_STD_DOMAINS

```
PROMPT Drop objects in  cdisc ...
-----************************************************************************-----
DROP TABLE m_std_domains CASCADE CONSTRAINTS;
DROP SEQUENCE m_std_domains_sq ;

PROMPT Create objects in  cdisc ...
-----************************************************************************-----

 PROMPT Creating table m_std_domains ...
-------------------------------------------------------------------------------
-- create objects --
CREATE TABLE m_std_domains (
```

```
    "D_ID"          NUMBER          PRIMARY KEY,
    "V_ID"          NUMBER          NOT NULL,
    "CLASS_NAME"    VARCHAR2(NA)    NOT NULL,
    "DOMAIN_NAME"   VARCHAR2(NA)    NOT NULL,
    "DOMAIN_ABBR"   VARCHAR2(NA)    NOT NULL,
    "NOTE"          VARCHAR2(NA)
);

COMMENT ON TABLE m_std_domains IS
  'Collection of domains';

 COMMENT ON COLUMN m_std_domains."D_ID" IS
  'Unique identifier for domain name';
 COMMENT ON COLUMN m_std_domains."V_ID" IS
  'Version ID linked to m_std_versions.v_id';
 COMMENT ON COLUMN m_std_domains."CLASS_NAME" IS
  'Class name';
 COMMENT ON COLUMN m_std_domains."DOMAIN_NAME" IS
  'Domain name';
 COMMENT ON COLUMN m_std_domains."DOMAIN_ABBR" IS
  'Domain abbreviation';
 COMMENT ON COLUMN m_std_domains."NOTE" IS
  'Note or description';

PROMPT Creating sequence m_std_domains_sq...
CREATE SEQUENCE m_std_domains_sq
  START WITH 1
  INCREMENT BY 1
  NOCACHE NOCYCLE;
-- Ended for m_std_domains

PROMPT Altering objects for cdisc...
-----**************************************************************************-----


PROMPT Altering table (FK) m_std_domains...

ALTER TABLE m_std_domains
  ADD CONSTRAINT fk_m_std_domains_3_v_id
  FOREIGN KEY (v_id)
  REFERENCES m_std_versions(v_id);

-- Ended for m_std_domains
```

SCRIPT FOR CREATING M_STD_VARIABLES

```
PROMPT Drop objects in  cdisc ...
-----**************************************************************************-----
DROP TABLE m_std_variables CASCADE CONSTRAINTS;
DROP SEQUENCE m_std_variables_sq ;

PROMPT Create objects in  cdisc ...
-----**************************************************************************-----
```

```
 PROMPT Creating table m_std_variables ...
-------------------------------------------------------------------------------
-- create objects --
CREATE TABLE m_std_variables (
   "VAR_ID"         NUMBER           PRIMARY KEY,
   "SEQ"            NUMBER           NOT NULL,
   "CLASS"          VARCHAR2(50)     NOT NULL,
   "DOMAIN"         VARCHAR2(30)    ,
   "VARIABLE"       VARCHAR2(30)     NOT NULL,
   "VAR_NAME"       VARCHAR2(30)     NOT NULL,
   "VAR_LABEL"      VARCHAR2(500)    NOT NULL,
   "TYPE"           VARCHAR2(20)     NOT NULL,
   "CT_FORMAT"      VARCHAR2(50)    ,
   "ROLE"           VARCHAR2(50)    ,
   "NOTE"           VARCHAR2(4000)  ,
   "CORE"           VARCHAR2(20)    ,
   "LENGTH"         NUMBER          ,
   "STD_VERSION"    VARCHAR2(50)    ,
   "D_ID"           NUMBER
);

COMMENT ON TABLE m_std_variables IS
  'Collection of variables';

 COMMENT ON COLUMN m_std_variables."VAR_ID" IS
  'Unique identifier for variables';
 COMMENT ON COLUMN m_std_variables."SEQ" IS
  'Variable sequence or position';
 COMMENT ON COLUMN m_std_variables."CLASS" IS
  'Class name';
 COMMENT ON COLUMN m_std_variables."DOMAIN" IS
  'Domain name';
 COMMENT ON COLUMN m_std_variables."VARIABLE" IS
  'Variable name without domain prefix';
 COMMENT ON COLUMN m_std_variables."VAR_NAME" IS
  'Variable name';
 COMMENT ON COLUMN m_std_variables."VAR_LABEL" IS
  'Variable label';
 COMMENT ON COLUMN m_std_variables."TYPE" IS
  'Data type such as char, num, etc.';
 COMMENT ON COLUMN m_std_variables."CT_FORMAT" IS
  'Controlled Terms or Format such as STENRF, ISO 8601, etc.';
 COMMENT ON COLUMN m_std_variables."ROLE" IS
  'Variable role such as Synonym Qualifier, Timing,';
 COMMENT ON COLUMN m_std_variables."NOTE" IS
  'CDISC Notes (for domains) Description (for General Classes)';
 COMMENT ON COLUMN m_std_variables."CORE" IS
  'Core: req, exp, perm.';
 COMMENT ON COLUMN m_std_variables."LENGTH" IS
  'Max length of the variable';
 COMMENT ON COLUMN m_std_variables."STD_VERSION" IS
  'Standard version';
 COMMENT ON COLUMN m_std_variables."D_ID" IS
  'Domain ID linked to m_std_domains.d_id';

PROMPT Creating sequence m_std_variables_sq...
CREATE SEQUENCE m_std_variables_sq
```

```
  START WITH 1
  INCREMENT BY 1
  NOCACHE NOCYCLE;
-- Ended for m_std_variables

PROMPT Altering objects for cdisc...
-----************************************************************************-----

PROMPT Altering table (FK) m_std_variables...

ALTER TABLE m_std_variables
  ADD CONSTRAINT fk_m_std_variables_16_d_id
  FOREIGN KEY (d_id)
  REFERENCES m_std_domains(d_id);

-- Ended for m_std_variables
```

SCRIPT FOR CREATING ALL IN A BATCH

Here is the script to create all the Oracle SQL scripts in a batch in R:

```
> library(bldsql)
> ofn <- 'C:/Users/htu/gDrive/mySoft/AI/abp1/scripts/ai_03_crt_stddb.sql'
> bld_dbs(ofn, src_mdl = "GS02", tgt="Oracle", out.header="wrt")
```

The batch mode will generate one file containing all the scripts to create tables.

JSON SCRIPTS FOR MONGODB

The MongoDB JSON scripts are generated and list in the following table:

m_std_versions	m_std_domains	m_std_variables

db = db.getSiblingDB('cdisc');
db.createCollection("m_std_versions",
{
"capped": false,
"validator": {
"table": {
"$type": "collection"
},
"v_id": {
"$exists": true,
"$type": "int"
},
"p_vid": {
"$type": "int"
},
"sdo": {
	db = db.getSiblingDB('cdisc');
	db.createCollection("m_std_domains",
	{
	"capped": false,
	"validator": {
	"table": {
	"$type": "collection"
	},
	"d_id": {
	"$exists": true,
	"$type": "int"
	},
	"v_id": {
	"$exists": true,
	"$type": "int"
	},
		db = db.getSiblingDB('cdisc');
		db.createCollection("m_std_variables",
		{
		"capped": false,
		"validator": {
		"table": {
		"$type": "collection"
		},
		"var_id": {
		"$exists": true,
		"$type": "int"
		},
		"seq": {
		"$exists": true,
		"$type": "int"
		},

```
          "$exists": true,
          "$type": "string"
        },
        "class": {
          "$exists": true,
          "$type": "string"
        },
        "name": {
          "$exists": true,
          "$type": "string"
        },
        "version": {
          "$exists": true,
          "$type": "string"
        },
        "dt_released": {
          "$exists": true,
          "$type": "date"
        },
        "dt_enforced": {
          "$type": "date"
        },
        "note": {
          "$type": "string"
        },
      },
      "validationLevel":
"strict",
      "validationAction":
"error"
}
);
```

```
        "class_name": {
          "$exists": true,
          "$type": "string"
        },
        "domain_name": {
          "$exists": true,
          "$type": "string"
        },
        "domain_abbr": {
          "$exists": true,
          "$type": "string"
        },
        "note": {
          "$type": "string"
        },
      },
      "validationLevel":
"strict",
      "validationAction":
"error"
}
);
```

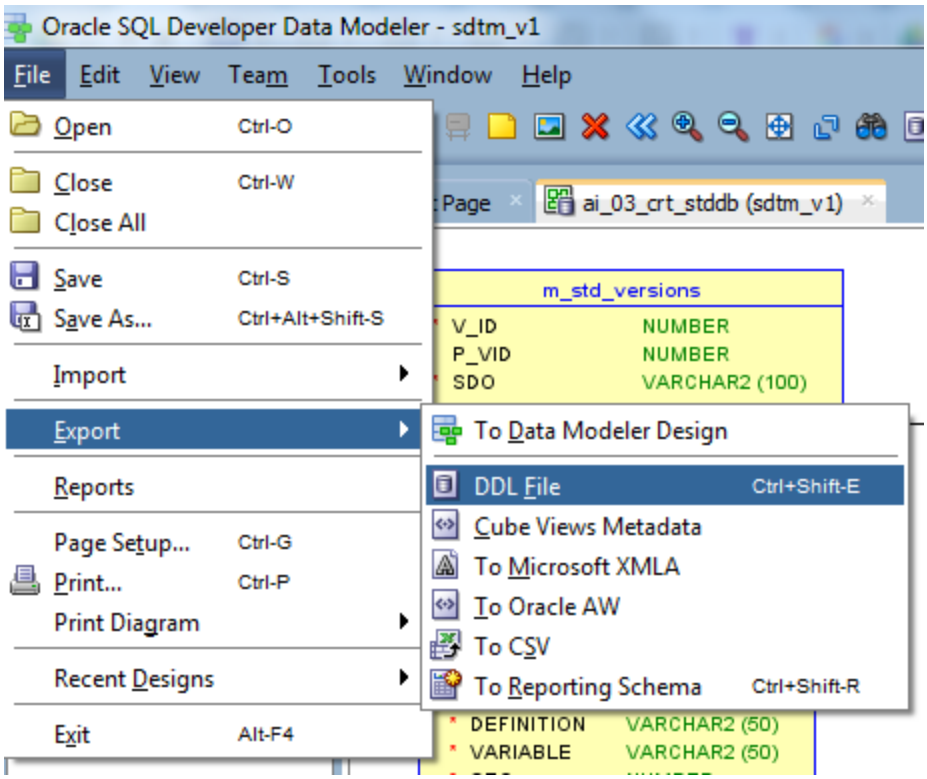
```
        "class": {
          "$exists": true,
          "$type": "string"
        },
        "domain": {
          "$type": "string"
        },
        "variable": {
          "$exists": true,
          "$type": "string"
        },
        "var_name": {
          "$exists": true,
          "$type": "string"
        },
        "var_label": {
          "$exists": true,
          "$type": "string"
        },
        "type": {
          "$exists": true,
          "$type": "string"
        },
        "ct_format": {
          "$type": "string"
        },
        "role": {
          "$type": "string"
        },
        "note": {
          "$type": "string"
        },
        "core": {
          "$type": "string"
        },
        "length": {
          "$type": "int"
        },
        "std_version": {
          "$type": "string"
        },
        "d_id": {
          "$type": "int"
        },
      },
      "validationLevel":
"strict",
      "validationAction":
"error"
}
);
```

VISUALIZATION OF DATA MODELS

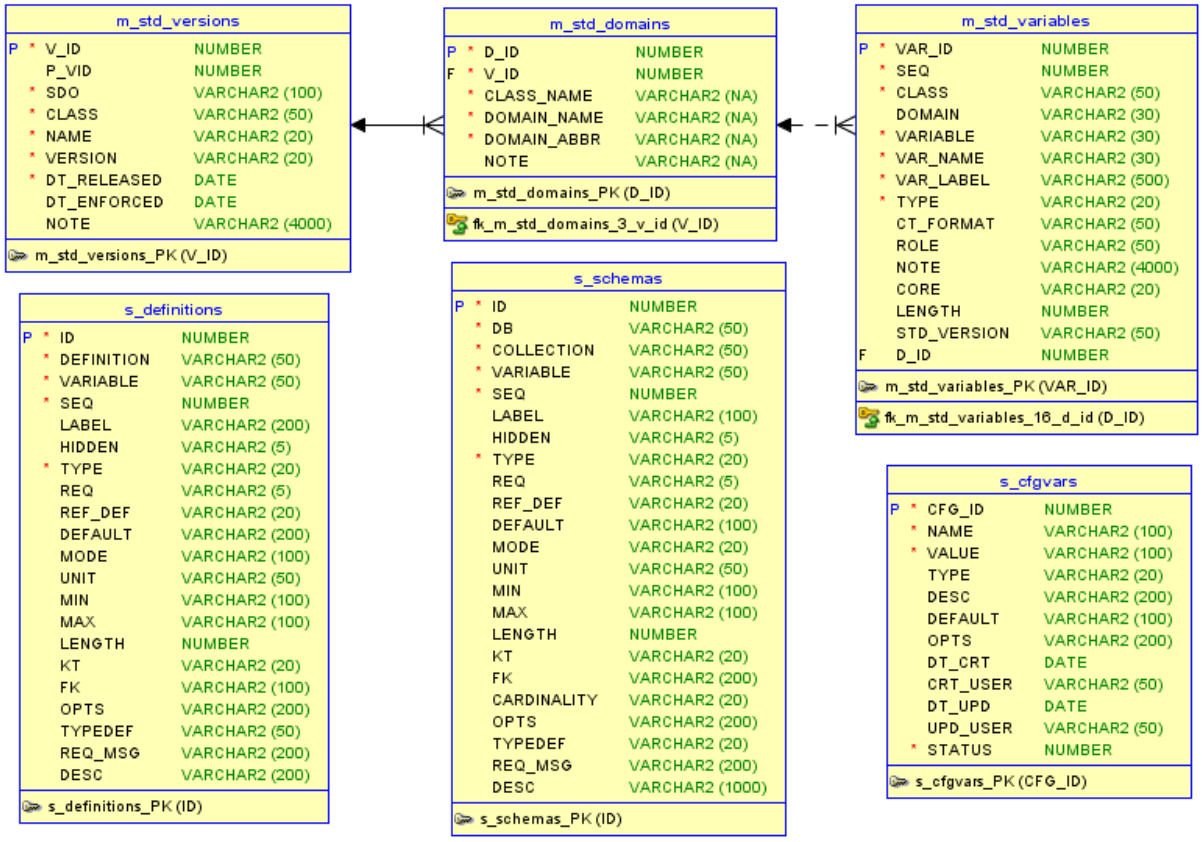
Once we created the single data definition language (DDL) file containing all the tables and relationships from the section of “script for creating all in a batch”, we can import the codes into Oracle data modeler to display the physical

model in ER diagram. Oracle data modeler is a free tool for data modeling. You can use the import the tables and relationships to further develop your data model.

The following screenshot shows how to start importing a DDL file:



The following picture shows the ER diagram:



We can use this method to further develop the SDTM into a database model to store study data and create a SDTM data warehouse.

CONCLUSION

This paper shows that it is possible to use all free and readily available tools to develop and document data models and generates codes to create tables in Oracle database and collections in NoSQL database such as MongoDB.

Google Sheets can be used as a quick data store to store and share your data. R and R Shiny is an open source language and can be used to develop simple user interface and perform many tasks. There are many free packages enabling you to do many complicated things such as reading from Google Sheets and MS Excel, connecting to Oracle database, and producing codes in different languages.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Author Name: Huiming Tu
Company: FHL Consulting
Address: 34 Independence Pl
City / Postcode: Chesterbrook, PA 19087
Work Phone: 484-463-1198
Fax: N/A
Email: huiming.tu@fhlconsultingus.com
Web:

Author Name: Hanming Tu

Company: TuCai Consulting
Address: 617 Hancock Drive
City / Postcode: Mullica Hill, NJ 08062
Work Phone: 484-881-2384
Fax: N/A
Email: hanming.tu@gmail.com
Web: