2022 - Data Analytics for Immersive Environments - CA4 - RDBMS & Linear Regression Project

Part 1 - Generate and Populate Database

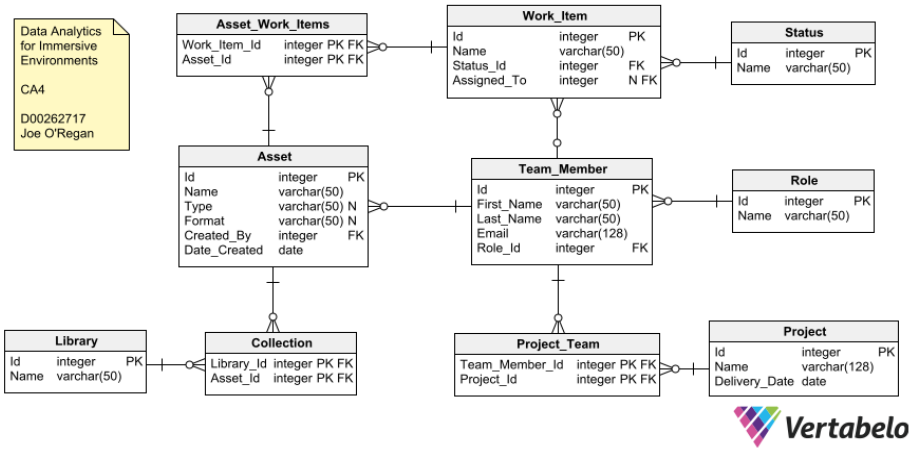
Joe O'Regan

2023-01-11

Repo Link

https://github.com/joeaoregan/2022_DAIE_CA4_JOR1

ER Diagram



Create Tables

Create or recreate the sqlite file

Check if the sqlite file exists already and if it does delete it. Then create an in-memory RSQLite database.

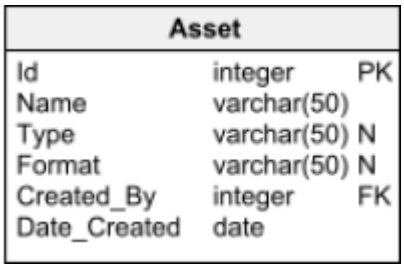
```
# if the sqlite file exists already delete it
if (file.exists("daie_ca4_data.sqlite"))
  file.remove("daie_ca4_data.sqlite")

# create the sqlite file
conn <- dbConnect(RSQLite::SQLite(), "daie_ca4_data.sqlite")
```

Create the tables

Asset

Show each table cropped from the main ER Diagram using magick library. Only dimensions and offset change for each cropped table, so this code is shown once.

```
er_img <- image_read("daie_ca4_er_diagram.png")
# cropped image dimension offset by location (in pixels)
# (newXdimension x newYdimension + Xlocation + Ylocation)
asset_img = image_scale(image_crop(er_img, "181x117+184+155"),
                        ifelse(is_html_output(), scale_html, scale_pdf))
asset_img
```



Describes a single electronic resource produced by one or more work items

- **Id:** Unique identifier for each row
- **Name:** Description of the asset
- **Type:** Short description of type of asset (Not mandatory)
- **Format:** Short description of format of asset (Not mandatory)
- **Created_By:** Team member who created the asset
- **Date_Created:** Date the asset was created

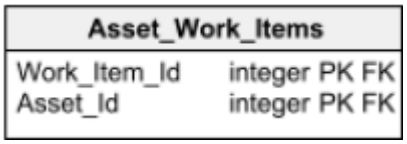
Primary Key(s): Id (auto increments)

Foreign Key(s): Created_By (Team_Member.Id)

Create the Asset table. Automatically increment the primary key. Id, Name, Created_By, and Date_created are mandatory fields.

```
# create tables
dbExecute(conn, "CREATE TABLE Asset (
    Id integer NOT NULL CONSTRAINT Asset_pk PRIMARY KEY AUTOINCREMENT,
    Name varchar(50) NOT NULL,
    Type varchar(50),
    Format varchar(50),
    Created_By integer NOT NULL,
    Date_Created date NOT NULL,
    CONSTRAINT Asset_Team_Member FOREIGN KEY (Created_By)
    REFERENCES Team_Member (Id)
);")
```

Asset_Work_Items



Assets are made up of work items

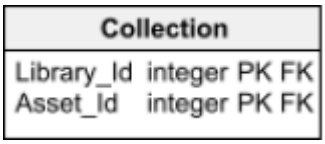
- **Work_Item_Id:** Id for work items used to create the asset
- **Asset_Id:** Id of the asset

Primary Key(s): Composite. Work_Item_Id + Asset_Id

Foreign Key(s): Work_Item_Id (Work_Item.Id), Asset_Id (Asset.Id)

```
dbExecute(conn, "CREATE TABLE Asset_Work_Items (
    Work_Item_Id integer NOT NULL,
    Asset_Id integer NOT NULL,
    CONSTRAINT Asset_Work_Items_pk PRIMARY KEY (Work_Item_Id,Asset_Id),
    CONSTRAINT Asset_Work_Item_Work_Item FOREIGN KEY (Work_Item_Id)
    REFERENCES Work_Item (Id),
    CONSTRAINT Asset_Work_Items_Asset FOREIGN KEY (Asset_Id)
    REFERENCES Asset (Id)
);")
```

Collection



Collection of assets

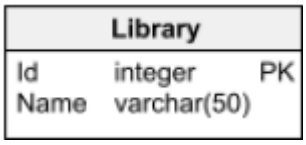
- **Library_Id:** Unique reference for a collection of assets
- **Asset_Id:** Unique reference of the asset that make up a collection

Primary Key(s): Composite. Library_Id + Asset_Id

Foreign Key(s): Library_Id (Library.Id), Asset_Id (Asset.Id)

```
dbExecute(conn, "CREATE TABLE Collection (
    Library_Id integer NOT NULL,
    Asset_Id integer NOT NULL,
    CONSTRAINT Collection_Id PRIMARY KEY (Library_Id,Asset_Id),
    CONSTRAINT Collection_Asset FOREIGN KEY (Asset_Id)
    REFERENCES Asset (Id),
    CONSTRAINT Collection_Library FOREIGN KEY (Library_Id)
    REFERENCES Library (Id)
);")
```

Library



Describes a collection of available assets

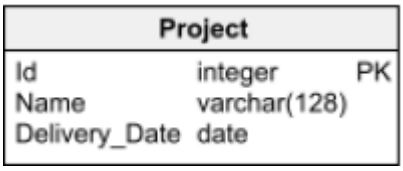
- **Id:** Unique reference for each collection in Library
- **Name:** Description of library collection

Primary Key(s): Id

Foreign Key(s): Id (Collection.Library_Id)

```
dbExecute(conn, "CREATE TABLE Library (
    Id integer NOT NULL CONSTRAINT Library_pk PRIMARY KEY AUTOINCREMENT,
    Name varchar(50) NOT NULL
);")
```

Project



Describes a project being produced, timeline, and team involved

- **Id:** Unique reference for each project
- **Name:** Description of Project
- **Delivery_Date:** Date project is due to finish

Primary Key(s): Id

Foreign Key(s): Id (Project_Team.Project_Id). Each Project has multiple Team Members.

```
dbExecute(conn, "CREATE TABLE Project (
    Id integer NOT NULL CONSTRAINT Project_pk PRIMARY KEY,
    Name varchar(128) NOT NULL,
    Delivery_Date date NOT NULL
);")
```

Project_Team

Project_Team	
Team_Member_Id	integer PK FK
Project_Id	integer PK FK

Each project has team members assigned to it

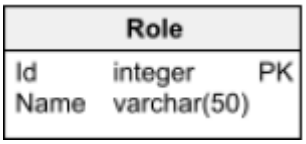
- **Team_Member_Id:** Unique reference for each Team Member
- **Project_Id:** Id of Project

Primary Key(s): Composite. Team_Member_Id + Project_Id

Foreign Key(s): Team_Member_Id (Team_Member.Id), Project_Id (Project.Id). Each Project has multiple Team Members. Each team member can work on 1 or more projects.

```
dbExecute(conn, "CREATE TABLE Project_Team (
    Team_Member_Id integer NOT NULL,
    Project_Id integer NOT NULL,
    CONSTRAINT Project_Team_Id PRIMARY KEY (Project_Id,Team_Member_Id),
    CONSTRAINT Team_Team_Member FOREIGN KEY (Team_Member_Id)
    REFERENCES Team_Member (Id),
    CONSTRAINT Team_Project FOREIGN KEY (Project_Id)
    REFERENCES Project (Id)
);")
```

Role



Each team member has a role (and only 1 role as I already used 10 tables)

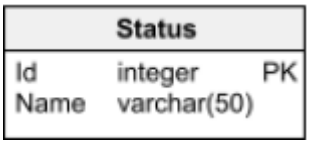
- **Id:** Unique reference for role
- **Name:** Role description

Primary Key(s): Id

Foreign Key(s): Id (Team_Member.Role_Id). Each team member has 1 role for projects.

```
dbExecute(conn, "CREATE TABLE Role (
    Id integer NOT NULL CONSTRAINT Role_pk PRIMARY KEY AUTOINCREMENT,
    Name varchar(50) NOT NULL
);")
```

Status



Work items have a progress status

- **Id:** Unique reference for status
- **Name:** Status description

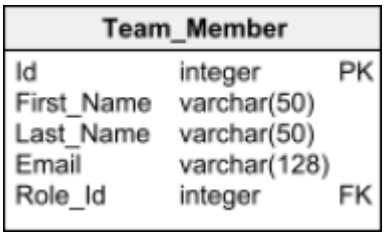
Primary Key(s): Id

Foreign Key(s): Id (Work_Item.Stats_Id). Each work item has a status (To Do, In Progress, Review, Done).

```
dbExecute(conn, "CREATE TABLE Status (
    Id integer NOT NULL CONSTRAINT Status_pk PRIMARY KEY AUTOINCREMENT,
    Name varchar(50) NOT NULL
);")
```

Team_Member

Describes an individual on the team



- **Id:** Unique reference for team member
- **First_Name:** Team Member first name
- **Last_Name:** Team Member last name
- **Email:** Team Member email
- **Role_Id:** Team members role for projects

Primary Key(s): Id

Foreign Key(s): Id (Project_Team.Team_Member_Id, Work_Item.Assigned_To), Role_Id (Role.Id). Each Team Member has a role for projects. Each project team has team members. Each work item can be assigned to a team member.

```
dbExecute(conn, "CREATE TABLE Team_Member (
    Id integer NOT NULL CONSTRAINT Team_Member_pk PRIMARY KEY AUTOINCREMENT,
    First_Name varchar(50) NOT NULL,
    Last_Name varchar(50) NOT NULL,
    Email varchar(128) NOT NULL,
    Role_Id integer NOT NULL,
    CONSTRAINT Team_Member_Email_AK UNIQUE (Email),
    CONSTRAINT Team_Member_Role FOREIGN KEY (Role_Id)
    REFERENCES Role (Id)
);")
```

Work_Item

Work_Item		
Id	integer	PK
Name	varchar(50)	
Status_Id	integer	FK
Assigned_To	integer	N FK

Describes smallest parcel of work resulting in a testable output

- **Id:** Unique reference for work item.
- **Name:** Description of work item.
- **Status_Id:** Work item has as status (To Do, In Progress, Review, Done).
- **Assigned_To:** Work items can be assigned to team members.

Primary Key(s): Id

Foreign Key(s): Id (Asset_Work_Items.Work_Item.Id), Status_Id (Status.Id), Assigned_To (Team_Member.Id). Each work item can be assigned to a team member. Each work Item has a status. Assets are made of work items.

```
dbExecute(conn, "CREATE TABLE Work_Item (
    Id integer NOT NULL CONSTRAINT Work_Item_pk PRIMARY KEY AUTOINCREMENT,
    Name varchar(50) NOT NULL,
    Status_Id integer NOT NULL,
    Assigned_To integer NOT NULL,
    CONSTRAINT Work_Item_Statuses FOREIGN KEY (Status_Id)
    REFERENCES Status (Id),
    CONSTRAINT Work_Item_Team_Member FOREIGN KEY (Assigned_To)
    REFERENCES Team_Member (Id)
);")
```

Show tables in database

```
dbListTables(conn)
```

```
##  [1] "Asset"            "Asset_Work_Items" "Collection"       "Library"
##  [5] "Project"          "Project_Team"     "Role"             "Status"
##  [9] "Team_Member"      "Work_Item"        "sqlite_sequence"
```

Data format function

Function to format tables for HTML and PDF output. Display tables using knitr library's kable function and kableExtra to format tables.

```
data_format.function <- function(data, capt) {
  data %>%
    {
      #if (is_html_output()) { # if the output is HTML add class attribute
      #  kbl(., table.attr='class="table-striped table-hover"', caption = capt)}
      #else if (is_latex_output()) { # if the output is PDF ignore class attribute
        kbl(., caption = capt)
      #}
    } %>% # pdf output keep tables in position
      #kable_styling("striped", ifelse(is_html_output(),"hover","hold_position"),
      kable_styling(bootstrap_options = c("striped", "hover", "condensed", "HOLD_position"),
                    latex_options = c("striped","HOLD_position"),
                    stripe_color="#AAFFAA",
                    htmltable_class="table-striped table-hover") %>%
        row_spec(0, background = "#1AE81A")
}
```

Insert Data

Status

```
dbListFields(conn, "Status")
```

```
## [1] "Id"   "Name"
```

Insert table data (using DBI library dbExecute function):

```
dbExecute(conn, "INSERT INTO Status ('Name') VALUES ('To Do');")
dbExecute(conn, "INSERT INTO Status ('Name') VALUES ('In Progress');")
dbExecute(conn, "INSERT INTO Status ('Name') VALUES ('Review');")
dbExecute(conn, "INSERT INTO Status ('Name') VALUES ('Done');")
```

Display table data (using knitr library kable function):

```
status_data <- dbGetQuery(conn, "SELECT * FROM Status;")
data_format.function(status_data, "Status Table")
```

Table 1: Status Table

Id	Name
1	To Do
2	In Progress
3	Review
4	Done

Role

```
dbListFields(conn, "Role")
```

```
## [1] "Id"   "Name"
```

Insert table data:

```
dbExecute(conn, "INSERT INTO Role ('Name') VALUES ('Project Manager');")
dbExecute(conn, "INSERT INTO Role ('Name') VALUES ('Programmer');")
dbExecute(conn, "INSERT INTO Role ('Name') VALUES ('Tester');")
dbExecute(conn, "INSERT INTO Role ('Name') VALUES ('Artist');")
dbExecute(conn, "INSERT INTO Role ('Name') VALUES ('3D Modeller');")
dbExecute(conn, "INSERT INTO Role ('Name') VALUES ('Environment Modeller');")
dbExecute(conn, "INSERT INTO Role ('Name') VALUES ('Animator');")
dbExecute(conn, "INSERT INTO Role ('Name') VALUES ('Shading Artist');")
dbExecute(conn, "INSERT INTO Role ('Name') VALUES ('Concept Artist');")
```

Display table data:

```
role_data <- dbGetQuery(conn, "SELECT * FROM Role;")
data_format.function(role_data, "Role Table")
```

Table 2: Role Table

Id	Name
1	Project Manager
2	Programmer
3	Tester
4	Artist
5	3D Modeller
6	Environment Modeller
7	Animator
8	Shading Artist
9	Concept Artist

Team_Member

```
dbListFields(conn, "Team_Member")
```

```
## [1] "Id"         "First_Name" "Last_Name"  "Email"      "Role_Id"
```

Insert table data:

```
dbExecute(conn, "INSERT INTO Team_Member ('First_Name', 'Last_Name', 'Email', 'Role_Id')
          VALUES ('Joe', 'O''Regan', 'joe.oregan@daie.ca4', 2);")
dbExecute(conn, "INSERT INTO Team_Member ('First_Name', 'Last_Name', 'Email', 'Role_Id')
          VALUES ('Derp', 'McDerp', 'derpmcderp@daie.ca4', 1);")
dbExecute(conn, "INSERT INTO Team_Member ('First_Name', 'Last_Name', 'Email', 'Role_Id')
          VALUES ('Herpderp', 'Derpderpenson', 'hd.derpderpenson@daie.ca4', 3);")
dbExecute(conn, "INSERT INTO Team_Member ('First_Name', 'Last_Name', 'Email', 'Role_Id')
          VALUES ('Herpa', 'Derpderp', 'herpa.derpderp@daie.ca4', 4);")
dbExecute(conn, "INSERT INTO Team_Member ('First_Name', 'Last_Name', 'Email', 'Role_Id')
          VALUES ('De', 'Rpderp', 'de.rpderp@daie.ca4', 5);")
dbExecute(conn, "INSERT INTO Team_Member ('First_Name', 'Last_Name', 'Email', 'Role_Id')
          VALUES ('Pred', 'Prehpred', 'predprehpred@daie.ca4', 6);")
dbExecute(conn, "INSERT INTO Team_Member ('First_Name', 'Last_Name', 'Email', 'Role_Id')
          VALUES ('Derpa', 'Derpa', 'derpaderpa@daie.ca4', 9);")
dbExecute(conn, "INSERT INTO Team_Member ('First_Name', 'Last_Name', 'Email', 'Role_Id')
          VALUES ('Herpa', 'Derpa', 'herpaderpa@daie.ca4', 8);")
dbExecute(conn, "INSERT INTO Team_Member ('First_Name', 'Last_Name', 'Email', 'Role_Id')
          VALUES ('HerpaDerpa', 'McDerpa', 'herpaderpa.mcderpa@daie.ca4', 7);")
dbExecute(conn, "INSERT INTO Team_Member ('First_Name', 'Last_Name', 'Email', 'Role_Id')
          VALUES ('Joe', 'Derp', 'j.derp@daie.ca4', 2);")
dbExecute(conn, "INSERT INTO Team_Member ('First_Name', 'Last_Name', 'Email', 'Role_Id')
          VALUES ('Jon', 'Herpaderp', 'jherpaderp@daie.ca4', 7);")
dbExecute(conn, "INSERT INTO Team_Member ('First_Name', 'Last_Name', 'Email', 'Role_Id')
          VALUES ('Joblot', 'O''Stuff', 'joblot.ostuff@daie.ca4', 3);")
dbExecute(conn, "INSERT INTO Team_Member ('First_Name', 'Last_Name', 'Email', 'Role_Id')
          VALUES ('Joderp', 'Herpderpenson', 'j.herpderpenson@daie.ca4', 4);")
dbExecute(conn, "INSERT INTO Team_Member ('First_Name', 'Last_Name', 'Email', 'Role_Id')
          VALUES ('Jo', 'McQueryfiller', 'j.mcqueryfiller@daie.ca4', 1);")
```

Display table data:

```
team_member_data <- dbGetQuery(conn, "SELECT * FROM Team_Member;")
data_format.function(team_member_data, "Team Member Table")
```

Table 3: Team Member Table

Id	First_Name	Last_Name	Email	Role_Id
1	Joe	O'Regan	joe.oregan@daie.ca4	2
2	Derp	McDerp	derpmcderp@daie.ca4	1
3	Herpderp	Derpderpenson	hd.derpderpenson@daie.ca4	3
4	Herpa	Derpderp	herpa.derpderp@daie.ca4	4
5	De	Rpderp	de.rpderp@daie.ca4	5
6	Pred	Prehpred	predprehpred@daie.ca4	6
7	Derpa	Derpa	derpaderpa@daie.ca4	9
8	Herpa	Derpa	herpaderpa@daie.ca4	8
9	HerpaDerpa	McDerpa	herpaderpa.mcderpa@daie.ca4	7
10	Joe	Derp	j.derp@daie.ca4	2
11	Jon	Herpaderp	jherpaderp@daie.ca4	7
12	Joblot	O'Stuff	joblot.ostuff@daie.ca4	3
13	Joderp	Herpderpenson	j.herpderpenson@daie.ca4	4
14	Jo	McQueryfiller	j.mcqueryfiller@daie.ca4	1

Work_Item

```
dbListFields(conn, "Work_Item")
```

```
## [1] "Id"          "Name"        "Status_Id"   "Assigned_To"
```

Insert table data:

```
dbExecute(conn, "INSERT INTO Work_Item ('Name', 'Status_Id', 'Assigned_To')
          VALUES ('Art Thingy', 2, 4);")
dbExecute(conn, "INSERT INTO Work_Item ('Name', 'Status_Id', 'Assigned_To')
          VALUES ('Art Test Thingy', 3, 3);")
dbExecute(conn, "INSERT INTO Work_Item ('Name', 'Status_Id', 'Assigned_To')
          VALUES ('Environment Model Thingy', 2, 6);")
dbExecute(conn, "INSERT INTO Work_Item ('Name', 'Status_Id', 'Assigned_To')
          VALUES ('Art Concept Thingy', 4, 7);")
dbExecute(conn, "INSERT INTO Work_Item ('Name', 'Status_Id', 'Assigned_To')
          VALUES ('Art Shading Thingy', 4, 8);")
dbExecute(conn, "INSERT INTO Work_Item ('Name', 'Status_Id', 'Assigned_To')
          VALUES ('Random 3D Model', 2, 5);")
dbExecute(conn, "INSERT INTO Work_Item ('Name', 'Status_Id', 'Assigned_To')
          VALUES ('3D Model Test Obj', 2, 3);")
dbExecute(conn, "INSERT INTO Work_Item ('Name', 'Status_Id', 'Assigned_To')
          VALUES ('Random Blueprint', 2, 1);")
dbExecute(conn, "INSERT INTO Work_Item ('Name', 'Status_Id', 'Assigned_To')
          VALUES ('Blueprint Thingy', 2, 2);")
dbExecute(conn, "INSERT INTO Work_Item ('Name', 'Status_Id', 'Assigned_To')
          VALUES ('Blueprint Test', 3, 2);")
dbExecute(conn, "INSERT INTO Work_Item ('Name', 'Status_Id', 'Assigned_To')
          VALUES ('Art Test', 1, 12);")
dbExecute(conn, "INSERT INTO Work_Item ('Name', 'Status_Id', 'Assigned_To')
          VALUES ('Query Model Thingy', 3, 11);")
dbExecute(conn, "INSERT INTO Work_Item ('Name', 'Status_Id', 'Assigned_To')
          VALUES ('Another Test', 4, 13);")
```

Display table data:

```
work_item_data <- dbGetQuery(conn, "SELECT * FROM Work_Item;")
data_format.function(work_item_data, "Work Item Table")
```

Table 4: Work Item Table

Id	Name	Status_Id	Assigned_To
1	Art Thingy	2	4
2	Art Test Thingy	3	3
3	Environment Model Thingy	2	6
4	Art Concept Thingy	4	7
5	Art Shading Thingy	4	8
6	Random 3D Model	2	5
7	3D Model Test Obj	2	3
8	Random Blueprint	2	1
9	Blueprint Thingy	2	2
10	Blueprint Test	3	2
11	Art Test	1	12
12	Query Model Thingy	3	11
13	Another Test	4	13

Project

```
dbListFields(conn, "Project")
```

```
## [1] "Id"            "Name"          "Delivery_Date"
```

Insert table data:

```
dbExecute(conn, "INSERT INTO Project ('Name', 'Delivery_Date')
          VALUES ('Art Proj', '2023-01-11');")
dbExecute(conn, "INSERT INTO Project ('Name', 'Delivery_Date')
          VALUES ('DAIE CA4', '2023-01-20');")
dbExecute(conn, "INSERT INTO Project ('Name', 'Delivery_Date')
          VALUES ('New Project', '2023-01-24');")
dbExecute(conn, "INSERT INTO Project ('Name', 'Delivery_Date')
          VALUES ('Old Project', '2022-12-14');")
dbExecute(conn, "INSERT INTO Project ('Name', 'Delivery_Date')
          VALUES ('Christmas 2022 Project', '2022-12-25');")
dbExecute(conn, "INSERT INTO Project ('Name', 'Delivery_Date')
          VALUES ('Date Range Project', '2023-01-17');")
dbExecute(conn, "INSERT INTO Project ('Name', 'Delivery_Date')
          VALUES ('Another Date Range Project', '2023-02-01');")
dbExecute(conn, "INSERT INTO Project ('Name', 'Delivery_Date')
          VALUES ('Project Filler', '2023-03-01');")
dbExecute(conn, "INSERT INTO Project ('Name', 'Delivery_Date')
          VALUES ('Derp Project', '2023-03-17');")
dbExecute(conn, "INSERT INTO Project ('Name', 'Delivery_Date')
          VALUES ('Hmmm I Ran Out of Names', '2023-01-20');")
```

Display table data:

```
project_data <- dbGetQuery(conn, "SELECT * FROM Project;")
data_format.function(project_data, "Project Table")
```

Table 5: Project Table

Id	Name	Delivery_Date
1	Art Proj	2023-01-11
2	DAIE CA4	2023-01-20
3	New Project	2023-01-24
4	Old Project	2022-12-14
5	Christmas 2022 Project	2022-12-25
6	Date Range Project	2023-01-17
7	Another Date Range Project	2023-02-01
8	Project Filler	2023-03-01
9	Derp Project	2023-03-17
10	Hmmm I Ran Out of Names	2023-01-20

Project_Team

```
dbListFields(conn, "Project_Team")
```

```
## [1] "Team_Member_Id" "Project_Id"
```

Insert table data:

```
dbExecute(conn, "INSERT INTO Project_Team ('Team_Member_Id', 'Project_Id') VALUES (1, 1);")
dbExecute(conn, "INSERT INTO Project_Team ('Team_Member_Id', 'Project_Id') VALUES (2, 1);")
dbExecute(conn, "INSERT INTO Project_Team ('Team_Member_Id', 'Project_Id') VALUES (3, 1);")
dbExecute(conn, "INSERT INTO Project_Team ('Team_Member_Id', 'Project_Id') VALUES (4, 1);")
dbExecute(conn, "INSERT INTO Project_Team ('Team_Member_Id', 'Project_Id') VALUES (5, 1);")
dbExecute(conn, "INSERT INTO Project_Team ('Team_Member_Id', 'Project_Id') VALUES (2, 2);")
dbExecute(conn, "INSERT INTO Project_Team ('Team_Member_Id', 'Project_Id') VALUES (6, 2);")
dbExecute(conn, "INSERT INTO Project_Team ('Team_Member_Id', 'Project_Id') VALUES (7, 2);")
dbExecute(conn, "INSERT INTO Project_Team ('Team_Member_Id', 'Project_Id') VALUES (8, 2);")
dbExecute(conn, "INSERT INTO Project_Team ('Team_Member_Id', 'Project_Id') VALUES (9, 2);")
dbExecute(conn, "INSERT INTO Project_Team ('Team_Member_Id', 'Project_Id') VALUES (2, 3);")
dbExecute(conn, "INSERT INTO Project_Team ('Team_Member_Id', 'Project_Id') VALUES (10, 3);")
dbExecute(conn, "INSERT INTO Project_Team ('Team_Member_Id', 'Project_Id') VALUES (11, 3);")
dbExecute(conn, "INSERT INTO Project_Team ('Team_Member_Id', 'Project_Id') VALUES (14, 4);")
dbExecute(conn, "INSERT INTO Project_Team ('Team_Member_Id', 'Project_Id') VALUES (12, 4);")
dbExecute(conn, "INSERT INTO Project_Team ('Team_Member_Id', 'Project_Id') VALUES (13, 5);")
dbExecute(conn, "INSERT INTO Project_Team ('Team_Member_Id', 'Project_Id') VALUES (14, 6);")
dbExecute(conn, "INSERT INTO Project_Team ('Team_Member_Id', 'Project_Id') VALUES (14, 7);")
```

Display table data:

```
project_team_data <- dbGetQuery(conn, "SELECT * FROM Project_Team;")
data_format.function(project_team_data, "Project Team Table")
```

Table 6: Project Team Table

Team_Member_Id	Project_Id
1	1
2	1
3	1
4	1
5	1
2	2
6	2
7	2
8	2
9	2
2	3
10	3
11	3
14	4
12	4
13	5
14	6
14	7

Asset

```
dbListFields(conn, "Asset")
```

```
## [1] "Id"           "Name"         "Type"         "Format"       "Created_By"
## [6] "Date_Created"
```

Insert table data:

```
dbExecute(conn, "INSERT INTO Asset ('Name', 'Type', 'Format', 'Created_By', 'Date_Created')
          VALUES ('Random Blueprint Asset', 'Combination of Blueprints', 'Zip file', 1, '2023-01-11');")
dbExecute(conn, "INSERT INTO Asset ('Name', 'Created_By', 'Date_Created')
          VALUES ('Random Art Asset', 4, '2023-01-10');")
dbExecute(conn, "INSERT INTO Asset ('Name', 'Created_By', 'Date_Created')
          VALUES ('Art Asset Thingy', 4, '2023-01-10');")
dbExecute(conn, "INSERT INTO Asset ('Name', 'Type', 'Created_By', 'Date_Created')
          VALUES ('Environment Asset Thingy', 'Tree for use in Environment',4, '2023-01-02');")
```

Display table data:

```
asset_data <- dbGetQuery(conn, "SELECT * FROM Asset;")
data_format.function(asset_data, "Asset Table")
```

Table 7: Asset Table

Id	Name	Type	Format	Created_By	Date_Created
1	Random Blueprint Asset	Combination of Blueprints	Zip file	1	2023-01-11
2	Random Art Asset	NA	NA	4	2023-01-10
3	Art Asset Thingy	NA	NA	4	2023-01-10
4	Environment Asset Thingy	Tree for use in Environment	NA	4	2023-01-02

Asset_Work_Items

```
dbListFields(conn, "Asset_Work_Items")
```

```
## [1] "Work_Item_Id" "Asset_Id"
```

Insert table data:

```
dbExecute(conn, "INSERT INTO Asset_Work_Items ('Work_Item_Id', 'Asset_Id') VALUES (8, 1);")
dbExecute(conn, "INSERT INTO Asset_Work_Items ('Work_Item_Id', 'Asset_Id') VALUES (9, 1);")
dbExecute(conn, "INSERT INTO Asset_Work_Items ('Work_Item_Id', 'Asset_Id') VALUES (10, 1);")
dbExecute(conn, "INSERT INTO Asset_Work_Items ('Work_Item_Id', 'Asset_Id') VALUES (1, 2);")
dbExecute(conn, "INSERT INTO Asset_Work_Items ('Work_Item_Id', 'Asset_Id') VALUES (2, 2);")
dbExecute(conn, "INSERT INTO Asset_Work_Items ('Work_Item_Id', 'Asset_Id') VALUES (4, 3);")
dbExecute(conn, "INSERT INTO Asset_Work_Items ('Work_Item_Id', 'Asset_Id') VALUES (5, 3);")
dbExecute(conn, "INSERT INTO Asset_Work_Items ('Work_Item_Id', 'Asset_Id') VALUES (3, 4);")
dbExecute(conn, "INSERT INTO Asset_Work_Items ('Work_Item_Id', 'Asset_Id') VALUES (6, 4);")
dbExecute(conn, "INSERT INTO Asset_Work_Items ('Work_Item_Id', 'Asset_Id') VALUES (7, 4);")
```

Display table data:

```
asset_work_items_data <- dbGetQuery(conn, "SELECT * FROM Asset_Work_Items;")
data_format.function(asset_work_items_data, "Asset Work Items Table")
```

Table 8: Asset Work Items Table

Work_Item_Id	Asset_Id
8	1
9	1
10	1
1	2
2	2
4	3
5	3
3	4
6	4
7	4

Collection

```
dbListFields(conn, "Collection")
```

```
## [1] "Library_Id" "Asset_Id"
```

Insert table data:

```
dbExecute(conn, "INSERT INTO Collection ('Library_Id', 'Asset_Id') VALUES (1, 1);")
dbExecute(conn, "INSERT INTO Collection ('Library_Id', 'Asset_Id') VALUES (2, 2);")
dbExecute(conn, "INSERT INTO Collection ('Library_Id', 'Asset_Id') VALUES (2, 3);")
dbExecute(conn, "INSERT INTO Collection ('Library_Id', 'Asset_Id') VALUES (3, 4);")
```

Display table data:

```
collection_data <- dbGetQuery(conn, "SELECT * FROM Collection;")
data_format.function(collection_data, "Collection Table")
```

Table 9: Collection Table

Library_Id	Asset_Id
1	1
2	2
2	3
3	4

Library

```
dbListFields(conn, "Library")
```

```
## [1] "Id"   "Name"
```

Insert table data:

```
dbExecute(conn, "INSERT INTO Library ('Name') VALUES ('Programming');")
dbExecute(conn, "INSERT INTO Library ('Name') VALUES ('Models');")
dbExecute(conn, "INSERT INTO Library ('Name') VALUES ('Scenery');")
dbExecute(conn, "INSERT INTO Library ('Name') VALUES ('Characters');")
```

Display table data:

```
library_data <- dbGetQuery(conn, "SELECT * FROM Library;")
data_format.function(library_data, "Library Table")
```

Table 10: Library Table

Id	Name
1	Programming
2	Models
3	Scenery
4	Characters

Disconnect Database

```
dbDisconnect(conn)
```