Eric Olson
Final Project: Kerbal Space Program Database
CS340_400 – Summer 2015 – Sarbaziazad

Project Outline

The Kerbal Space Program is a video game in which the depicted race, Kerbals, start a space program. The player can design various rockets, space probes, and landers. They can take on contracts to complete missions, and can visit the various planets, and moons within their solar system. The game lends it self well to a database project, given the various interconnectedness of the entities. One can track via a database which Kerbals have been assigned which Missions, and visited which planets/moons.

Database Detail

The database to support this project will consist of eight tables. Five tables of entities, and three many to many relationship tables will be used. Four of the five entities will be considered basic entities these include Kerbals, Ships, Planets, Moons. The fifth entity is missions, missions build the interconnectivity between the other tables. The three many to many relationships are as follows: Mission/Kerbal, Mission/Planet, Mission/Moon. The database will also feature two many to one relationships between Missions/Ship, and secondly between Moons and the planet they orbit.

Kerbal Table

The Kerbal table will have four attributes:

- id - Auto incrementing primary key.
- **name** - Name of Kerbal, can not be null, or blank.
- **courage** - Numeric courage rating of Kerbal 1-9. Can not be null.
- **stupidity** - Numeric stupidity rating of Kerbal 1-9. Can not be null.

Ship Table

The Ship table will have five attributes:

- id - Auto incrementing primary key.
- **name** - Name of ship, can not be null, or blank.
- **seats** - Number of seats on ship. Can not be null, can be 0.
- **stages**- Number of stages of rocket/lander. Can not be null, can be 0
- **lander** - Boolean value indicating if ship can land on a planet/moon.   Not Null.

Planet Table

The Planet table will have six attributes:

- **id** - Auto incrementing primary key.
- **name** - Name of Planet, can not be null, or blank.
- **radius** - A Float value representing the radius of the planet in km. Not Null.
- **inclination** - A Float value representing the orbital inclination of the planet in degrees. Not Null.
- **gravity** - A Float value representing the strength of gravity on the planet in m/s^2. Not Null.
- **atmosphere** - A Boolean value indicating the presence of an atmosphere. Not Null.

Moon Table

While Moons and Planets have many of the same characteristics, I wished to track which planet each moon orbited. To keep them both in the same table, and provide this relationship would have broken the rules of normal form, and were therefore seperated. The Moon table will have 6 attributes:

- **id** - Auto incrementing primary key.
- **name** - Name of Moon, can not be null, or blank.
- **radius** - A Float value representing the radius of the moon in km. Not Null.
- **gravity** - A Float value representing the strength of gravity on the moon in m/s^2. Not Null.
- **atmosphere** - A Boolean value indicating the presence of an atmosphere. Not Null.
- ***orbits*** - A foreign key referencing 'id' of the Planet Table. This supports the many to 1 relationship between moons and a planet. Required, not null.

Mission Table

The Mission table will have 4 attributes:

- **id** - Auto incrementing primary key.
- **name** - Name of Mission, can not be null, or blank.
- **first_launch_year** - Year of first launch, integer value. Not required.
- ***ship_id*** - A foreign key referencing 'id' of Ship table. This supports the many to 1 relationship between missions and ships. Not required, can be Null.

Kerbal_Mission Table

The Kerbal_Mission table will support the many to many relationship between Kerbals and Missions. The table will consist of 2 attributes, together the primary key:

- ***kerbal_id*** - Foreign key referencing 'id' from Kerbal table. Required, not null.
- ***mission_id*** - Foreign key referencing 'id' from Mission table. Required, not null.

Mission_Planet Table

The Mission_Planet table will support the many to many relationship between Missions and Planets. The table will consist of 2 attributes, together the primary key:

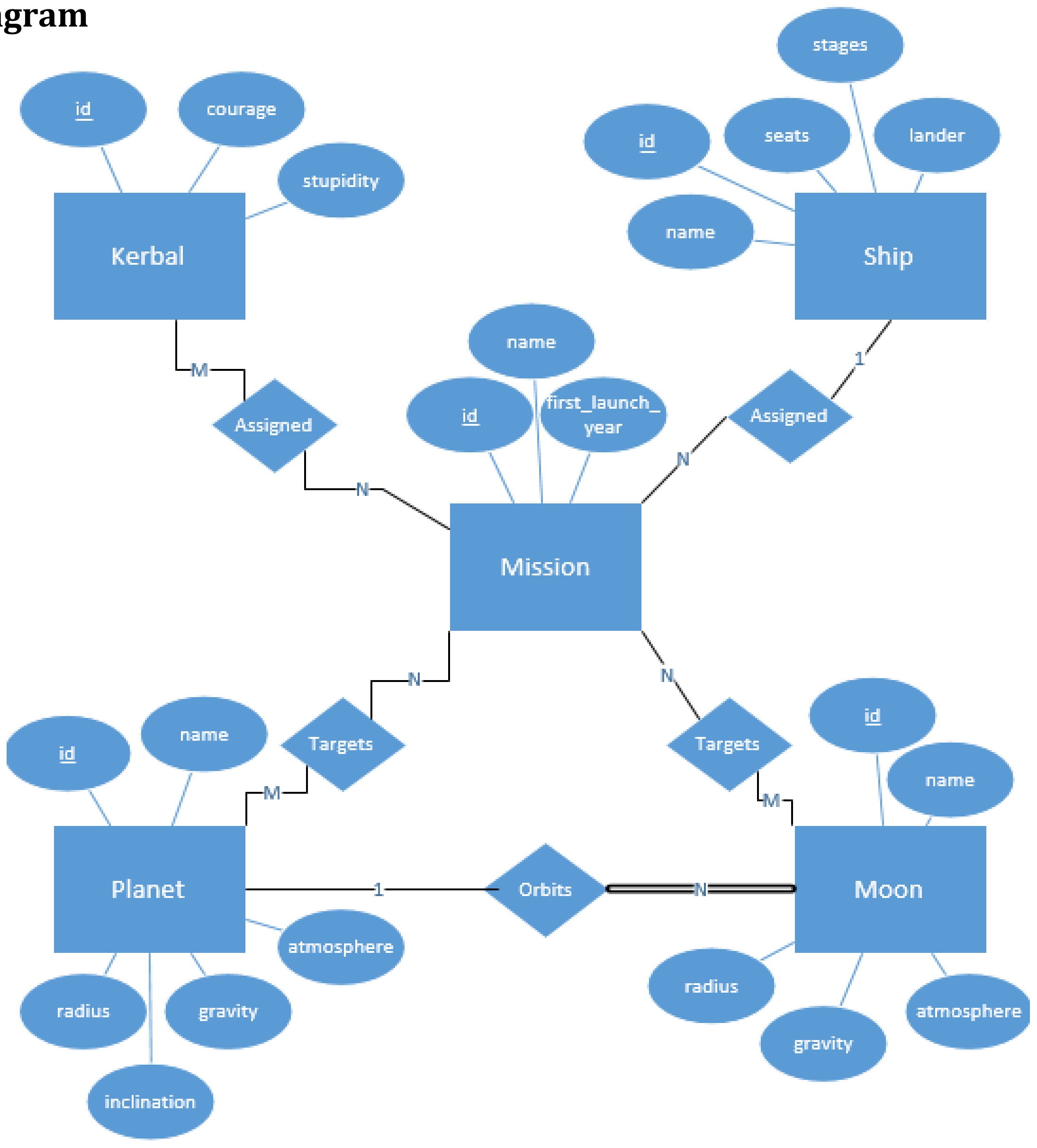
- ***mission_id*** - Foreign key referencing 'id' from Mission table. Required, not null.
- ***planet_id*** - Foreign key referencing 'id' from Planet table. Required, not null.

Mission_Moon Table

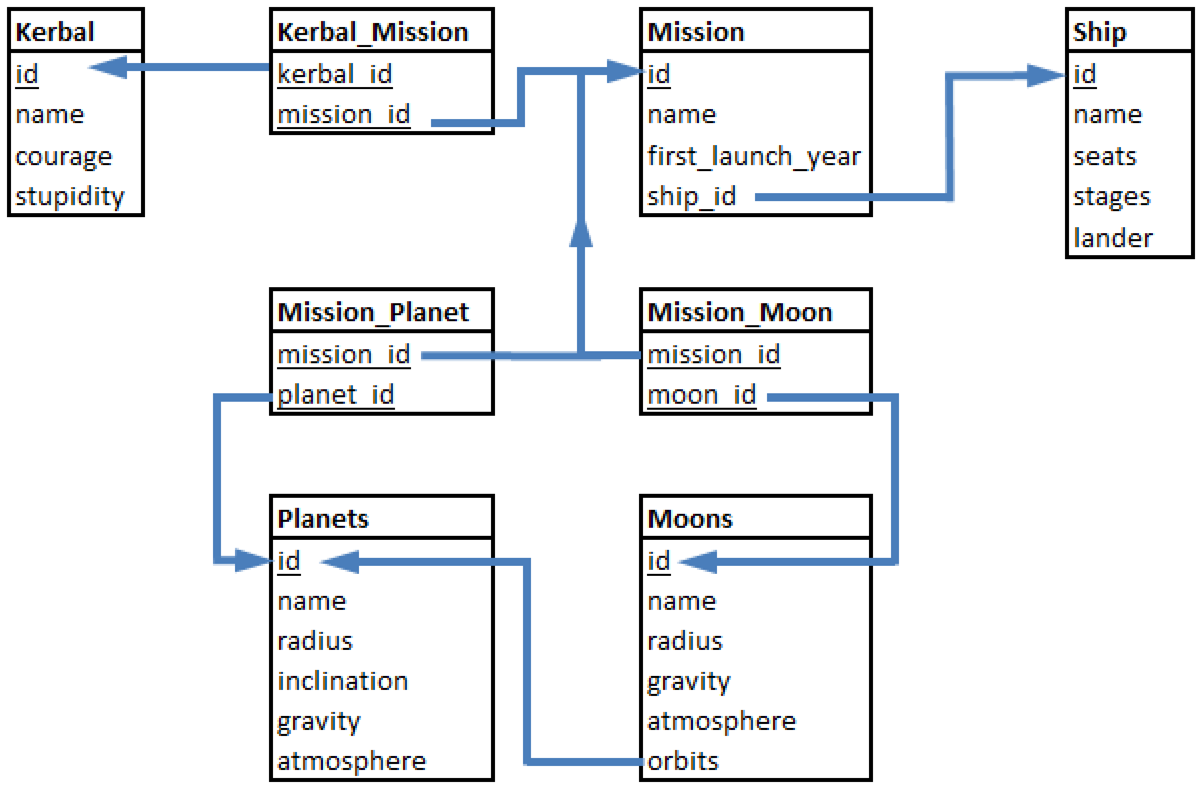
The Mission_Moon table will support the many to many relationship between Missions and Moons. The table will consist of 2 attributes, together the primary key:

- ***mission_id*** - Foreign key referencing 'id' from Mission table. Required, not null.
- ***moon_id*** - Foreign key referencing 'id' from Moon table. Required, not null.

ER Diagram



Database Scheme

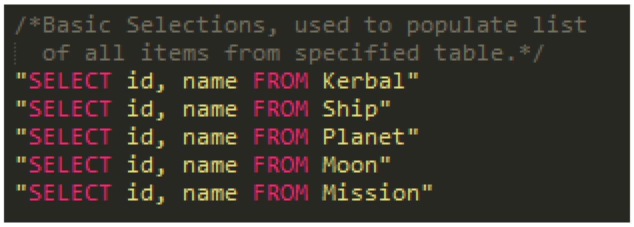


General Use Queries

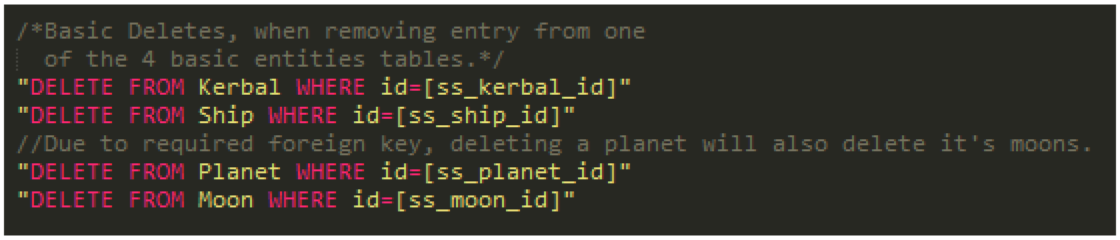
The queries that created the tables can be found in Appendix A, at the end of this paper. The sections that follow will be groupings of SQL queries used in the project, which are also available in the PHP files submitted with this project. I have attempted to include comments in each grouping to provide some context and explanation where warranted. This additional info is included as notes in the code. Where the query uses variable data, I have listed the data in []. There are three types of variable data used, and a key is listed below:

- **[ui_data] ui = User Input.** This is data typed directly by the user into a input field on the GUI.
- **[us_data] us = User Selected.** This generally refers to the user selecting from a list or dropdown of possible entities. While the user is selecting by name, the system has the id stored in the background associated with their choice.
- **[ss_data] ss = System Stored.** Here the user is working in an area or selected a button that is only related to a single entity, the id for that entity was stored in the background.

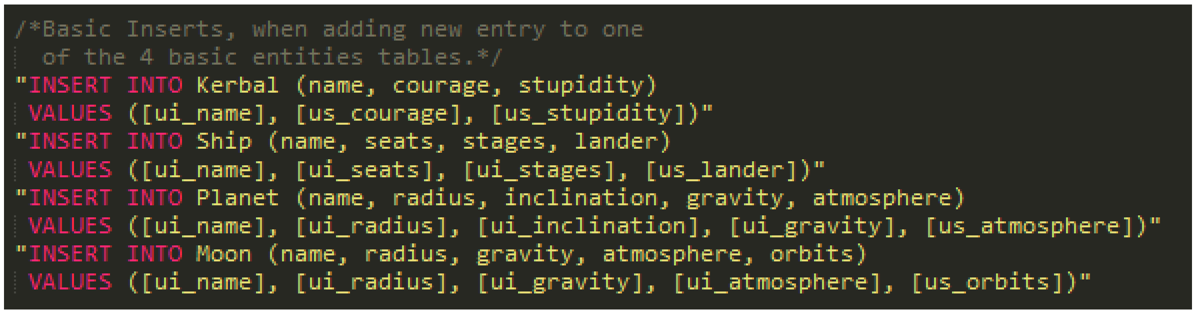
Basic Additions/Deletions/Selections:



```
/*Basic Selections, used to populate list
  of all items from specified table.*/
"SELECT id, name FROM Kerbal"
"SELECT id, name FROM Ship"
"SELECT id, name FROM Planet"
"SELECT id, name FROM Moon"
"SELECT id, name FROM Mission"
```



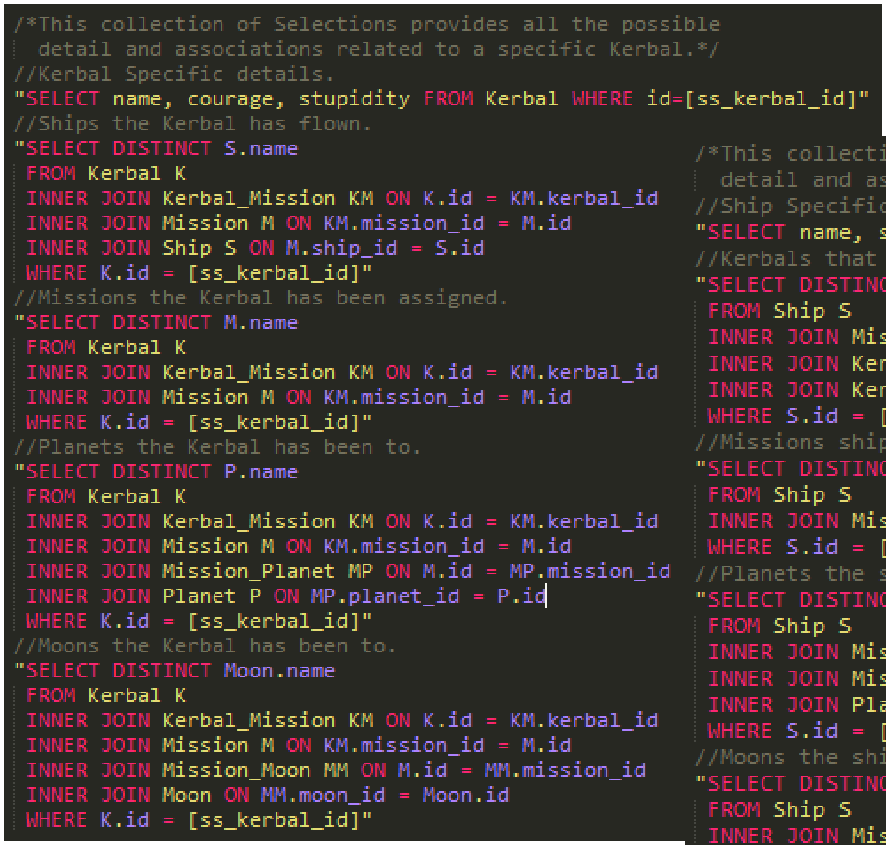
```
/*Basic Deletes, when removing entry from one
  of the 4 basic entities tables.*/
"DELETE FROM Kerbal WHERE id=[ss_kerbal_id]"
"DELETE FROM Ship WHERE id=[ss_ship_id]"
//Due to required foreign key, deleting a planet will also delete it's moons.
"DELETE FROM Planet WHERE id=[ss_planet_id]"
"DELETE FROM Moon WHERE id=[ss_moon_id]"
```



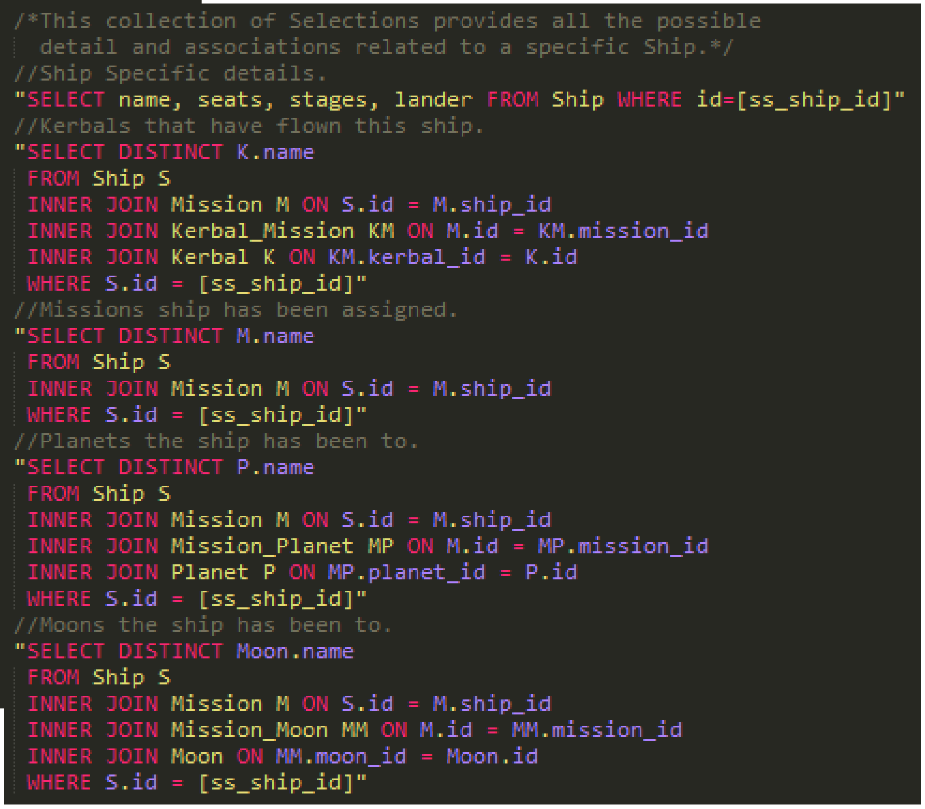
```
/*Basic Inserts, when adding new entry to one
  of the 4 basic entities tables.*/
"INSERT INTO Kerbal (name, courage, stupidity)
 VALUES ([ui_name], [us_courage], [us_stupidity])"
"INSERT INTO Ship (name, seats, stages, lander)
 VALUES ([ui_name], [ui_seats], [ui_stages], [us_lander])"
"INSERT INTO Planet (name, radius, inclination, gravity, atmosphere)
 VALUES ([ui_name], [ui_radius], [ui_inclination], [ui_gravity], [us_atmosphere])"
"INSERT INTO Moon (name, radius, gravity, atmosphere, orbits)
 VALUES ([ui_name], [ui_radius], [ui_gravity], [ui_atmosphere], [us_orbits])"
```

Details and Associations (4 Groups):

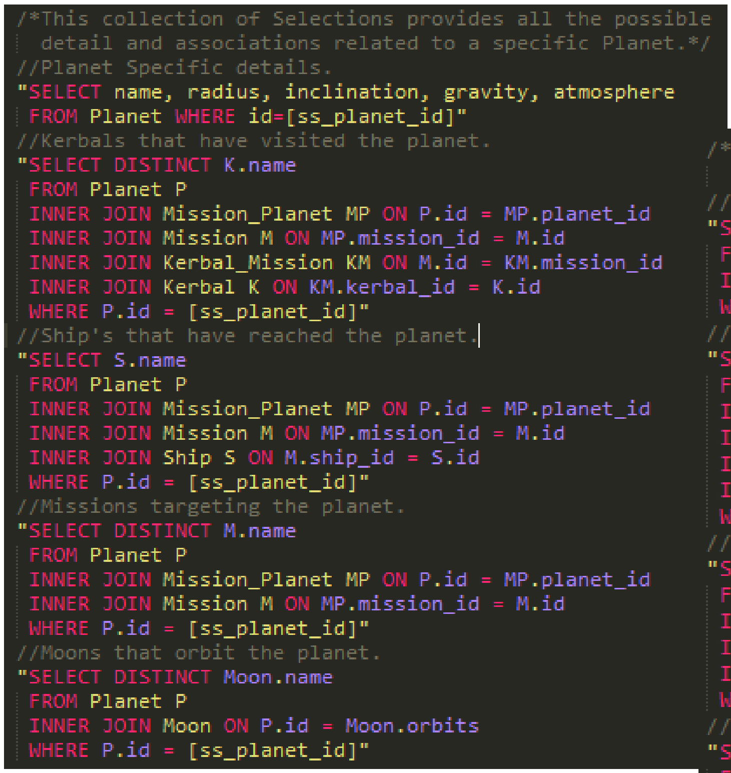
Distinct was used in these groupings. While a Kerbal may visit a planet, for example, on a number of different missions, but in service of a list of planets visited we only need to list it once.



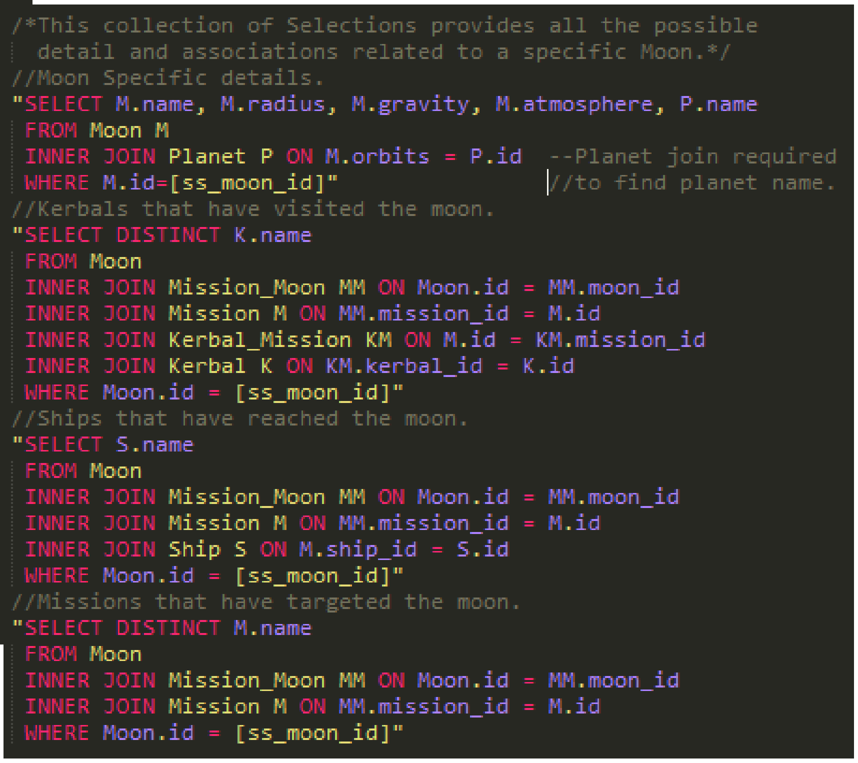
```
/*This collection of Selections provides all the possible
  detail and associations related to a specific Kerbal.*/
//Kerbal Specific details.
"SELECT name, courage, stupidity FROM Kerbal WHERE id=[ss_kerbal_id]"
//Ships the Kerbal has flown.
"SELECT DISTINCT S.name
 FROM Kerbal K
 INNER JOIN Kerbal_Mission KM ON K.id = KM.kerbal_id
 INNER JOIN Mission M ON KM.mission_id = M.id
 INNER JOIN Ship S ON M.ship_id = S.id
 WHERE K.id = [ss_kerbal_id]"
//Missions the Kerbal has been assigned.
"SELECT DISTINCT M.name
 FROM Kerbal K
 INNER JOIN Kerbal_Mission KM ON K.id = KM.kerbal_id
 INNER JOIN Mission M ON KM.mission_id = M.id
 WHERE K.id = [ss_kerbal_id]"
//Planets the Kerbal has been to.
"SELECT DISTINCT P.name
 FROM Kerbal K
 INNER JOIN Kerbal_Mission KM ON K.id = KM.kerbal_id
 INNER JOIN Mission M ON KM.mission_id = M.id
 INNER JOIN Mission_Planet MP ON M.id = MP.mission_id
 INNER JOIN Planet P ON MP.planet_id = P.id
 WHERE K.id = [ss_kerbal_id]"
//Moons the Kerbal has been to.
"SELECT DISTINCT Moon.name
 FROM Kerbal K
 INNER JOIN Kerbal_Mission KM ON K.id = KM.kerbal_id
 INNER JOIN Mission M ON KM.mission_id = M.id
 INNER JOIN Mission_Moon MM ON M.id = MM.mission_id
 INNER JOIN Moon ON MM.moon_id = Moon.id
 WHERE K.id = [ss_kerbal_id]"
```



```
/*This collection of Selections provides all the possible
  detail and associations related to a specific Ship.*/
//Ship Specific details.
"SELECT name, seats, stages, lander FROM Ship WHERE id=[ss_ship_id]"
//Kerbals that have flown this ship.
"SELECT DISTINCT K.name
 FROM Ship S
 INNER JOIN Mission M ON S.id = M.ship_id
 INNER JOIN Kerbal_Mission KM ON M.id = KM.mission_id
 INNER JOIN Kerbal K ON KM.kerbal_id = K.id
 WHERE S.id = [ss_ship_id]"
//Missions ship has been assigned.
"SELECT DISTINCT M.name
 FROM Ship S
 INNER JOIN Mission M ON S.id = M.ship_id
 WHERE S.id = [ss_ship_id]"
//Planets the ship has been to.
"SELECT DISTINCT P.name
 FROM Ship S
 INNER JOIN Mission M ON S.id = M.ship_id
 INNER JOIN Mission_Planet MP ON M.id = MP.mission_id
 INNER JOIN Planet P ON MP.planet_id = P.id
 WHERE S.id = [ss_ship_id]"
//Moons the ship has been to.
"SELECT DISTINCT Moon.name
 FROM Ship S
 INNER JOIN Mission M ON S.id = M.ship_id
 INNER JOIN Mission_Moon MM ON M.id = MM.mission_id
 INNER JOIN Moon ON MM.moon_id = Moon.id
 WHERE S.id = [ss_ship_id]"
```



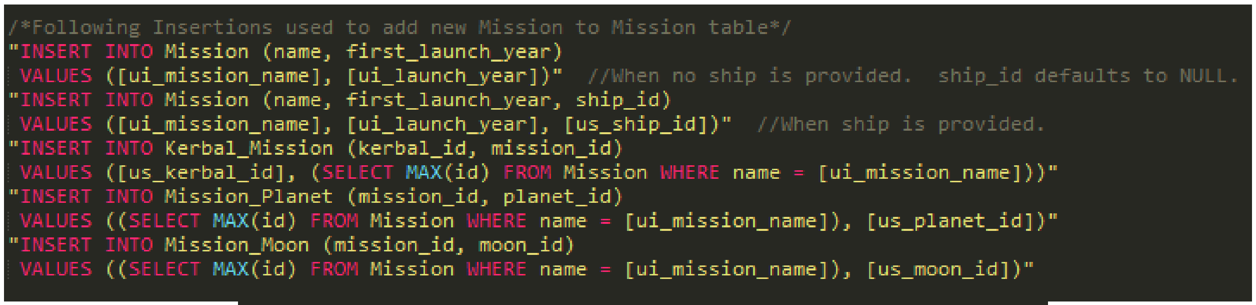
```
/*This collection of Selections provides all the possible
  detail and associations related to a specific Planet.*/
//Planet Specific details.
"SELECT name, radius, inclination, gravity, atmosphere
 FROM Planet WHERE id=[ss_planet_id]"
//Kerbals that have visited the planet.
"SELECT DISTINCT K.name
 FROM Planet P
 INNER JOIN Mission_Planet MP ON P.id = MP.planet_id
 INNER JOIN Mission M ON MP.mission_id = M.id
 INNER JOIN Kerbal_Mission KM ON M.id = KM.mission_id
 INNER JOIN Kerbal K ON KM.kerbal_id = K.id
 WHERE P.id = [ss_planet_id]"
//Ship's that have reached the planet.
"SELECT S.name
 FROM Planet P
 INNER JOIN Mission_Planet MP ON P.id = MP.planet_id
 INNER JOIN Mission M ON MP.mission_id = M.id
 INNER JOIN Ship S ON M.ship_id = S.id
 WHERE P.id = [ss_planet_id]"
//Missions targeting the planet.
"SELECT DISTINCT M.name
 FROM Planet P
 INNER JOIN Mission_Planet MP ON P.id = MP.planet_id
 INNER JOIN Mission M ON MP.mission_id = M.id
 WHERE P.id = [ss_planet_id]"
//Moons that orbit the planet.
"SELECT DISTINCT Moon.name
 FROM Planet P
 INNER JOIN Moon ON P.id = Moon.orbits
 WHERE P.id = [ss_planet_id]"
```



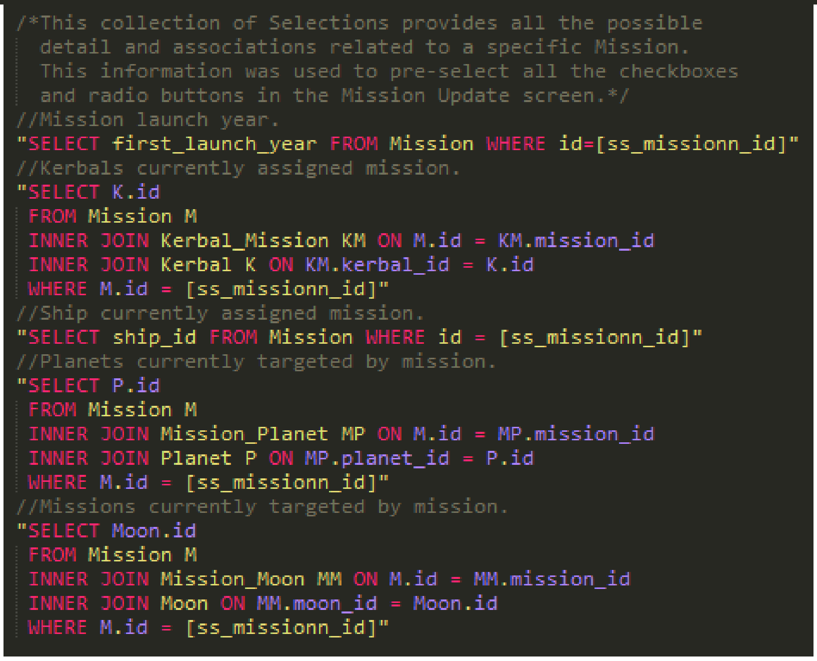
```
/*This collection of Selections provides all the possible
  detail and associations related to a specific Moon.*/
//Moon Specific details.
"SELECT M.name, M.radius, M.gravity, M.atmosphere, P.name
 FROM Moon M
 INNER JOIN Planet P ON M.orbits = P.id  --Planet join required
 WHERE M.id=[ss_moon_id]"                //to find planet name.
//Kerbals that have visited the moon.
"SELECT DISTINCT K.name
 FROM Moon
 INNER JOIN Mission_Moon MM ON Moon.id = MM.moon_id
 INNER JOIN Mission M ON MM.mission_id = M.id
 INNER JOIN Kerbal_Mission KM ON M.id = KM.mission_id
 INNER JOIN Kerbal K ON KM.kerbal_id = K.id
 WHERE Moon.id = [ss_moon_id]"
//Ships that have reached the moon.
"SELECT S.name
 FROM Moon
 INNER JOIN Mission_Moon MM ON Moon.id = MM.moon_id
 INNER JOIN Mission M ON MM.mission_id = M.id
 INNER JOIN Ship S ON M.ship_id = S.id
 WHERE Moon.id = [ss_moon_id]"
//Missions that have targeted the moon.
"SELECT DISTINCT M.name
 FROM Moon
 INNER JOIN Mission_Moon MM ON Moon.id = MM.moon_id
 INNER JOIN Mission M ON MM.mission_id = M.id
 WHERE Moon.id = [ss_moon_id]"
```

<u>Mission Management:</u>

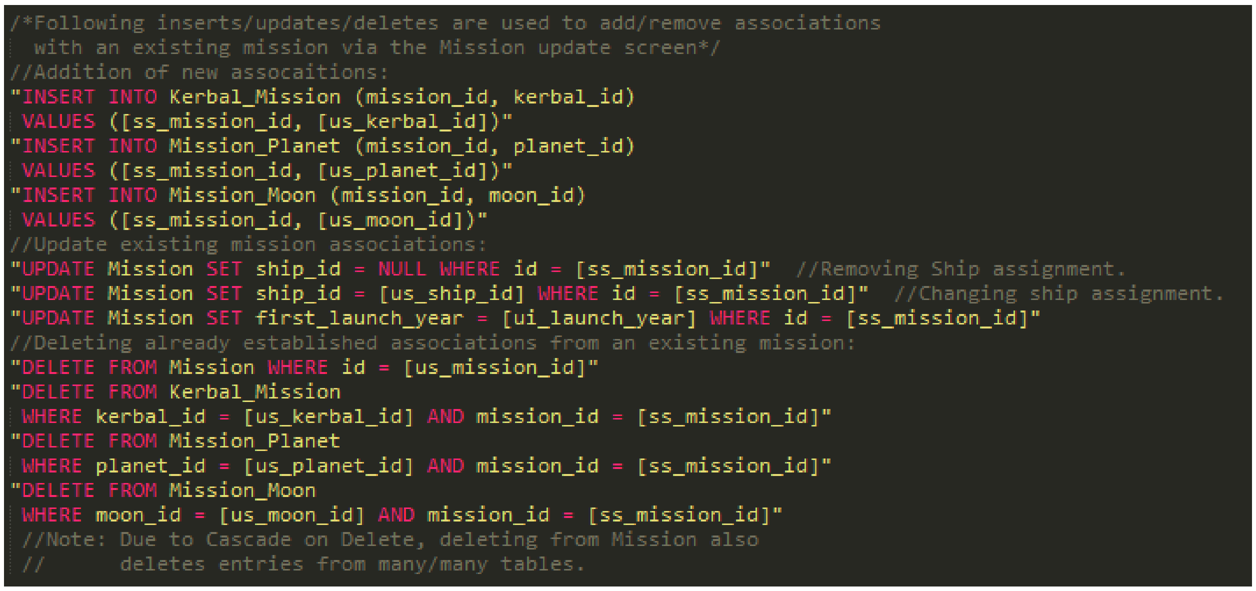
The queries using MAX() follow the addition of a new Mission. While at the time of Mission creation we have all the associated entity id's, we do not have the actual mission id until the first INSERT query completes. Since missions could theoretically share a name, I used Max to ensure I was adding associations to only the last added mission (the one I just added in the prior INSERT).



```
/*Following Insertions used to add new Mission to Mission table*/
"INSERT INTO Mission (name, first_launch_year)
 VALUES ([ui_mission_name], [ui_launch_year])"  //When no ship is provided.  ship_id defaults to NULL.
"INSERT INTO Mission (name, first_launch_year, ship_id)
 VALUES ([ui_mission_name], [ui_launch_year], [us_ship_id])"  //When ship is provided.
"INSERT INTO Kerbal_Mission (kerbal_id, mission_id)
 VALUES ([us_kerbal_id], (SELECT MAX(id) FROM Mission WHERE name = [ui_mission_name]))"
"INSERT INTO Mission_Planet (mission_id, planet_id)
 VALUES ((SELECT MAX(id) FROM Mission WHERE name = [ui_mission_name]), [us_planet_id])"
"INSERT INTO Mission_Moon (mission_id, moon_id)
 VALUES ((SELECT MAX(id) FROM Mission WHERE name = [ui_mission_name]), [us_moon_id])"
```



```
/*This collection of Selections provides all the possible
  detail and associations related to a specific Mission.
  This information was used to pre-select all the checkboxes
  and radio buttons in the Mission Update screen.*/
//Mission launch year.
"SELECT first_launch_year FROM Mission WHERE id=[ss_missionn_id]"
//Kerbals currently assigned mission.
"SELECT K.id
 FROM Mission M
 INNER JOIN Kerbal_Mission KM ON M.id = KM.mission_id
 INNER JOIN Kerbal K ON KM.kerbal_id = K.id
 WHERE M.id = [ss_missionn_id]"
//Ship currently assigned mission.
"SELECT ship_id FROM Mission WHERE id = [ss_missionn_id]"
//Planets currently targeted by mission.
"SELECT P.id
 FROM Mission M
 INNER JOIN Mission_Planet MP ON M.id = MP.mission_id
 INNER JOIN Planet P ON MP.planet_id = P.id
 WHERE M.id = [ss_missionn_id]"
//Missions currently targeted by mission.
"SELECT Moon.id
 FROM Mission M
 INNER JOIN Mission_Moon MM ON M.id = MM.mission_id
 INNER JOIN Moon ON MM.moon_id = Moon.id
 WHERE M.id = [ss_missionn_id]"
```

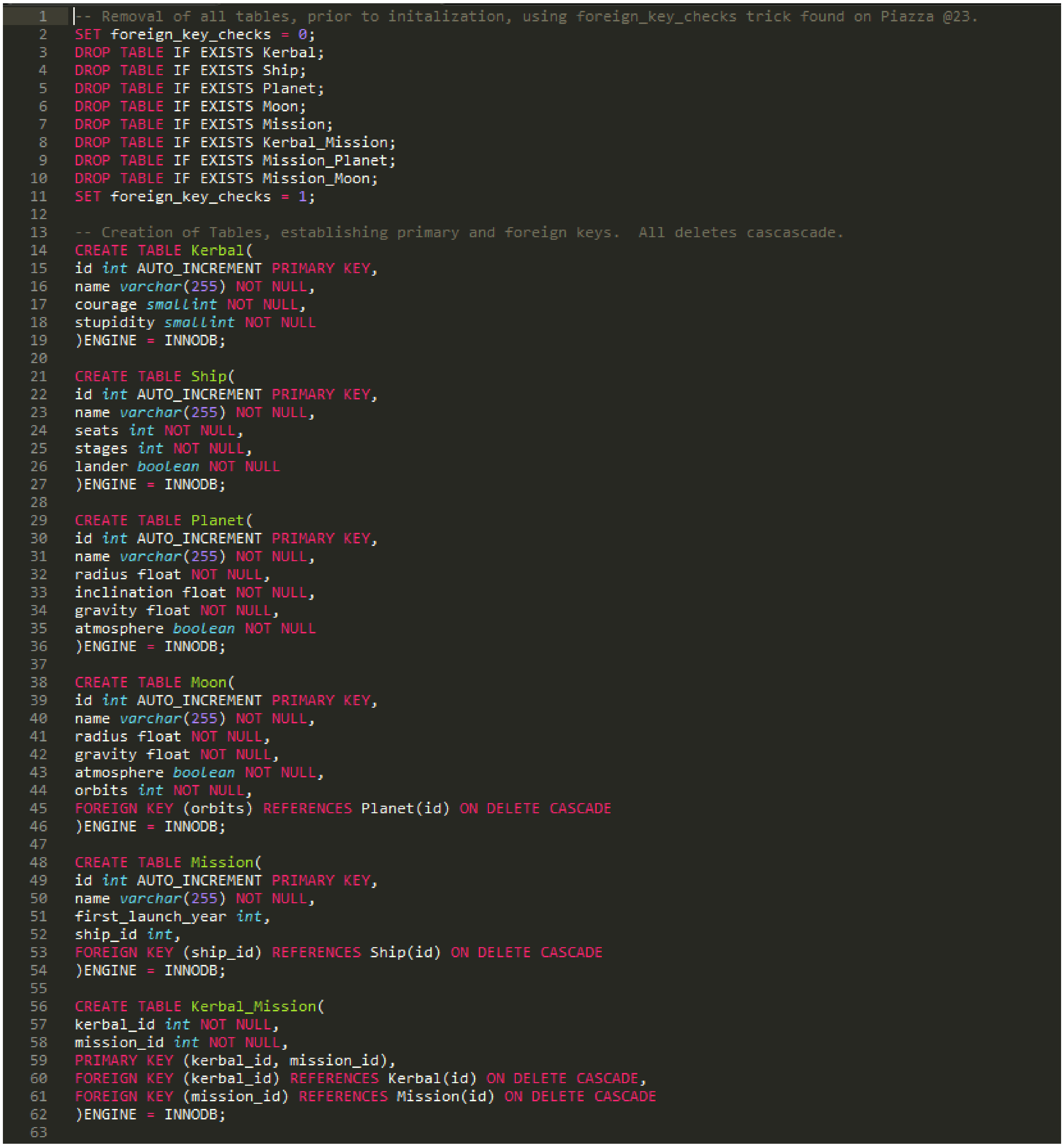

```
/*Following inserts/updates/deletes are used to add/remove associations
  with an existing mission via the Mission update screen*/
//Addition of new assocaitions:
"INSERT INTO Kerbal_Mission (mission_id, kerbal_id)
 VALUES ([ss_mission_id, [us_kerbal_id])"
"INSERT INTO Mission_Planet (mission_id, planet_id)
 VALUES ([ss_mission_id, [us_planet_id])"
"INSERT INTO Mission_Moon (mission_id, moon_id)
 VALUES ([ss_mission_id, [us_moon_id])"
//Update existing mission associations:
"UPDATE Mission SET ship_id = NULL WHERE id = [ss_mission_id]"  //Removing Ship assignment.
"UPDATE Mission SET ship_id = [us_ship_id] WHERE id = [ss_mission_id]"  //Changing ship assignment.
"UPDATE Mission SET first_launch_year = [ui_launch_year] WHERE id = [ss_mission_id]"
//Deleting already established associations from an existing mission:
"DELETE FROM Mission WHERE id = [us_mission_id]"
"DELETE FROM Kerbal_Mission
 WHERE kerbal_id = [us_kerbal_id] AND mission_id = [ss_mission_id]"
"DELETE FROM Mission_Planet
 WHERE planet_id = [us_planet_id] AND mission_id = [ss_mission_id]"
"DELETE FROM Mission_Moon
 WHERE moon_id = [us_moon_id] AND mission_id = [ss_mission_id]"
 //Note: Due to Cascade on Delete, deleting from Mission also
 //      deletes entries from many/many tables.
```

Live Production Site

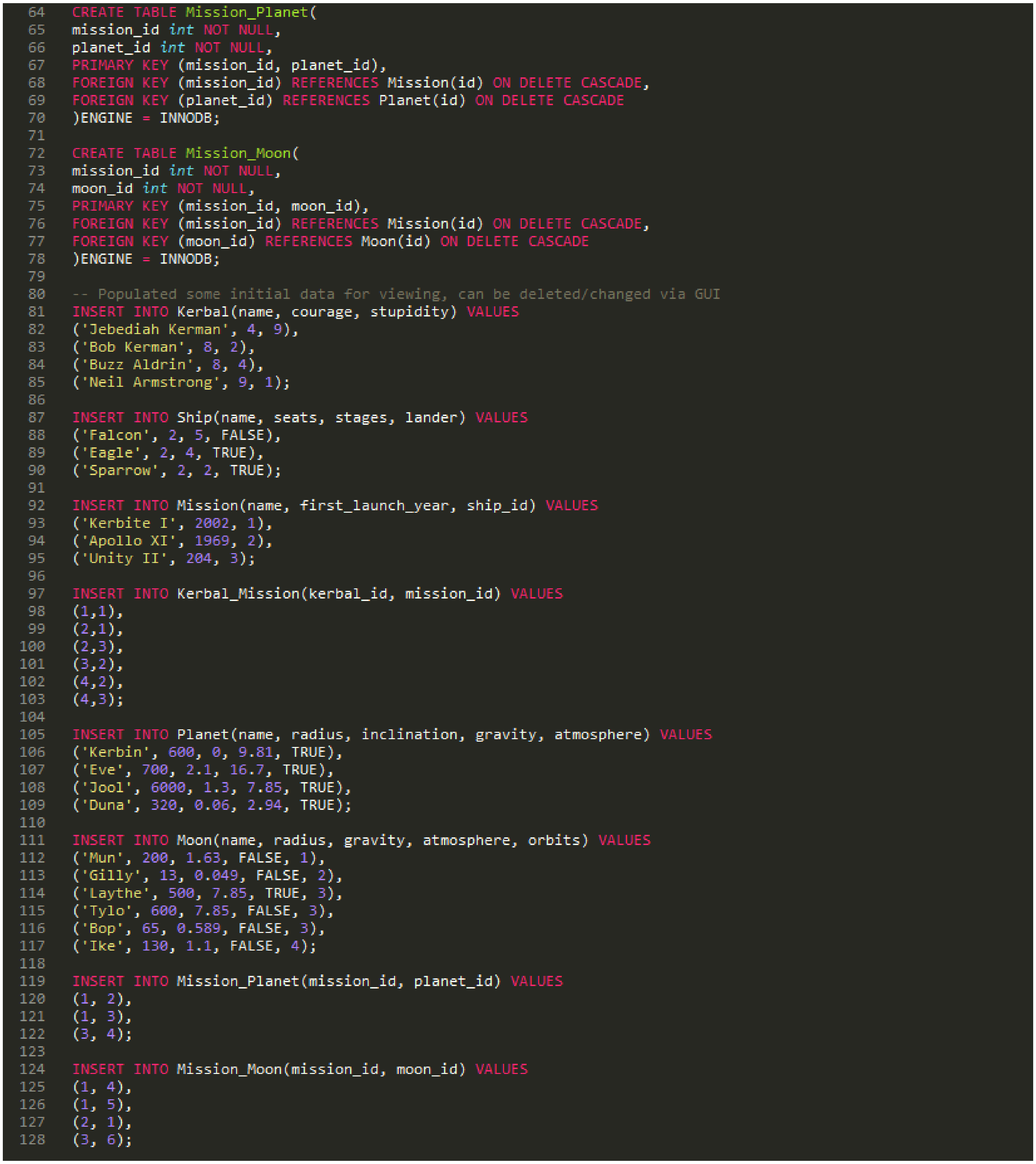
Attached to this submission are the HTML, JavaScript, PHP and CSS files used to support the live site. The site itself can be accessed at the URL below. The database has been created via the SQL file also provided in the submission, but can also be seen in the Appendix A that follows. The code has was developed and tested on Chrome, but has also been tested on Firefox, and IE version 11.

Live Site:
http://web.engr.oregonstate.edu/~olsoeric/CS340/Kerbal.html

Appendix A: Table Creation SQL Code



```sql
-- Removal of all tables, prior to initalization, using foreign_key_checks trick found on Piazza @23.
SET foreign_key_checks = 0;
DROP TABLE IF EXISTS Kerbal;
DROP TABLE IF EXISTS Ship;
DROP TABLE IF EXISTS Planet;
DROP TABLE IF EXISTS Moon;
DROP TABLE IF EXISTS Mission;
DROP TABLE IF EXISTS Kerbal_Mission;
DROP TABLE IF EXISTS Mission_Planet;
DROP TABLE IF EXISTS Mission_Moon;
SET foreign_key_checks = 1;

-- Creation of Tables, establishing primary and foreign keys.  All deletes cascascade.
CREATE TABLE Kerbal(
id int AUTO_INCREMENT PRIMARY KEY,
name varchar(255) NOT NULL,
courage smallint NOT NULL,
stupidity smallint NOT NULL
)ENGINE = INNODB;

CREATE TABLE Ship(
id int AUTO_INCREMENT PRIMARY KEY,
name varchar(255) NOT NULL,
seats int NOT NULL,
stages int NOT NULL,
lander boolean NOT NULL
)ENGINE = INNODB;

CREATE TABLE Planet(
id int AUTO_INCREMENT PRIMARY KEY,
name varchar(255) NOT NULL,
radius float NOT NULL,
inclination float NOT NULL,
gravity float NOT NULL,
atmosphere boolean NOT NULL
)ENGINE = INNODB;

CREATE TABLE Moon(
id int AUTO_INCREMENT PRIMARY KEY,
name varchar(255) NOT NULL,
radius float NOT NULL,
gravity float NOT NULL,
atmosphere boolean NOT NULL,
orbits int NOT NULL,
FOREIGN KEY (orbits) REFERENCES Planet(id) ON DELETE CASCADE
)ENGINE = INNODB;

CREATE TABLE Mission(
id int AUTO_INCREMENT PRIMARY KEY,
name varchar(255) NOT NULL,
first_launch_year int,
ship_id int,
FOREIGN KEY (ship_id) REFERENCES Ship(id) ON DELETE CASCADE
)ENGINE = INNODB;

CREATE TABLE Kerbal_Mission(
kerbal_id int NOT NULL,
mission_id int NOT NULL,
PRIMARY KEY (kerbal_id, mission_id),
FOREIGN KEY (kerbal_id) REFERENCES Kerbal(id) ON DELETE CASCADE,
FOREIGN KEY (mission_id) REFERENCES Mission(id) ON DELETE CASCADE
)ENGINE = INNODB;

```



```sql
CREATE TABLE Mission_Planet(
mission_id int NOT NULL,
planet_id int NOT NULL,
PRIMARY KEY (mission_id, planet_id),
FOREIGN KEY (mission_id) REFERENCES Mission(id) ON DELETE CASCADE,
FOREIGN KEY (planet_id) REFERENCES Planet(id) ON DELETE CASCADE
)ENGINE = INNODB;

CREATE TABLE Mission_Moon(
mission_id int NOT NULL,
moon_id int NOT NULL,
PRIMARY KEY (mission_id, moon_id),
FOREIGN KEY (mission_id) REFERENCES Mission(id) ON DELETE CASCADE,
FOREIGN KEY (moon_id) REFERENCES Moon(id) ON DELETE CASCADE
)ENGINE = INNODB;

-- Populated some initial data for viewing, can be deleted/changed via GUI
INSERT INTO Kerbal(name, courage, stupidity) VALUES
('Jebediah Kerman', 4, 9),
('Bob Kerman', 8, 2),
('Buzz Aldrin', 8, 4),
('Neil Armstrong', 9, 1);

INSERT INTO Ship(name, seats, stages, lander) VALUES
('Falcon', 2, 5, FALSE),
('Eagle', 2, 4, TRUE),
('Sparrow', 2, 2, TRUE);

INSERT INTO Mission(name, first_launch_year, ship_id) VALUES
('Kerbite I', 2002, 1),
('Apollo XI', 1969, 2),
('Unity II', 204, 3);

INSERT INTO Kerbal_Mission(kerbal_id, mission_id) VALUES
(1,1),
(2,1),
(2,3),
(3,2),
(4,2),
(4,3);

INSERT INTO Planet(name, radius, inclination, gravity, atmosphere) VALUES
('Kerbin', 600, 0, 9.81, TRUE),
('Eve', 700, 2.1, 16.7, TRUE),
('Jool', 6000, 1.3, 7.85, TRUE),
('Duna', 320, 0.06, 2.94, TRUE);

INSERT INTO Moon(name, radius, gravity, atmosphere, orbits) VALUES
('Mun', 200, 1.63, FALSE, 1),
('Gilly', 13, 0.049, FALSE, 2),
('Laythe', 500, 7.85, TRUE, 3),
('Tylo', 600, 7.85, FALSE, 3),
('Bop', 65, 0.589, FALSE, 3),
('Ike', 130, 1.1, FALSE, 4);

INSERT INTO Mission_Planet(mission_id, planet_id) VALUES
(1, 2),
(1, 3),
(3, 4);

INSERT INTO Mission_Moon(mission_id, moon_id) VALUES
(1, 4),
(1, 5),
(2, 1),
(3, 6);
```