People Table

This table contains a list of all people working with Imagimind Studios, as well as their basic descriptive and contact information. The two subtypes of this table are musicians and sound engineers, therefore a musician can be a sound engineer and vice versa. The primary key is pid (people id).

Functional Dependencies:

```
pid => first_name, last_name, gender, cell_phone, email
```

Create Table Statement:

```
create table People(
pid char(8) not null,
first_name char(25) not null,
last_name char(25) not null,
gender char(1) check(gender = 'M' or gender = 'F'),
cell_phone char(12) not null,
email char(50) not null,
primary key(pid)
);
```

Sample Data:

Musicians Table

This table contains a list of all Musicians who have worked on albums at Imagimind studios. Each row describes musical preferences and a short description of a musician. Note that this is an entity subtype of People, therefore inherits all attributes of that table. The primary key is pid (people id).

Functional Dependencies:

pid => favorite_instrument, description

Create Table Statement:

```
create table Musicians(
pid char(8) not null,
favorite_instrument char(20) default('vocals'),
description text not null,
primary key(pid),
foreign key(pid) references people(pid)
);
```

Sound_Engineer Table

The Sound_Engineer table contains a list of all engineers who record and mix music to produce the albums at Imagimind studios. It contains information about each engineer's years of experience in the field as well as whether or not they have obtained a degree in the field. Note that this is also an entity subtype of People, therefore inherits all attributes of that table. The primary key is pid (people identification)

Functional Dependencies:

    pid => years_experience, has_degree

Create Table Statement:

```
create table sound_engineers(
pid char(8) not null,
years_experience int default(0),
has_degree boolean default('F'),
primary key(pid),
foreign key(pid) references people(pid)
);
```

Bands Table

The Bands table contains a list of all the bands who have ever recorded an album here at Imagimind studios. Each row contains basic descriptive information about the band such as their name, year formed, and a musical description of the band. The primary key is band_id.

Functional Dependencies:

```
band_id =>  band_name, year_formed, description
```

Create Table Statement:

```
create table Bands(
band_id char(8) not null,
band_name char(50) not null,
year_formed int not null,
description text not null,
primary key(band_id)
);
```

Musical Roles Table

The Musical Roles Table contains a list of specific roles and descriptions a musician would play in a band such as Guitarist, Pianist, etc. This table allows for each Musician to take on multiple roles in a band, since it is not uncommon for one musician to play multiple instruments on a single track. The primary key is name.

Functional dependencies:

Name => description

Create Table Statement:

```
create table Musical_Roles(
name char(25) not null,
description text not null,
primary key(name)
);
```

Band Musicians Table

This table serves as the connection between bands and the musicians in those bands, as well as the role they play in said band

Create Table Statement:

```
create table Band_Musicians(
musician char(8) not null,
band_id char(8) not null,
role_name char(25) not null,
primary key (musician,band_id,role_name),
foreign key (musician) references Musicians(pid),
foreign key(band_id) references Bands(band_id),
foreign key (role_name) references Musical_Roles(name)
);
```

Genres Table

The Genres table contains a list of all musical genres which are currently being produced or have been produced at the studio, as well as a short description of that genre. The description is included since many new genres can be created with the fusion of one or more different genres. The possibilities are endless. If a new type of album is produced that falls under a different genre, a new record will be created. The primary key it the genre_name.

Functional Dependencies:

Genre_name => description

Create Table Statement:

```
create table Genres(
genre_name varchar(25) not null,
description text not null,
primary key(genre_name)
);
```

Albums Table

The Albums table stores all information about albums produced at Imagimind Studios. The duration column is measured in hours:minutes:seconds. The primary key is album_id.

Functional Dependencies:

album_id => title, description, duration, year_produced, genre

Create Table Statement:

```
create table Albums(
album_id char(8) not null,
title char(50) not null,
description text not null,
duration interval default('00:00:00'),
year_produced int not null,
genre varchar(25) not null,
primary key(album_id),
foreign key(genre) references Genres(genre_name)
);
```

Band_Albums Table

This table serves as the connection between the albums produced and the bands that were recorded to make those albums.

Create table statement:

```
create table Band_Albums(
band_id char(8) not null,
album_id char(8) not null,
primary key(band_id,album_id),
foreign key(band_id) references Bands(band_id),
foreign key(album_id) references Albums(album_id)
);
```

Album Engineer Table

This table serves as a connection between the albums produced and the engineers who produced them.

Create Table Statement:

```
create table Album_Engineers(
engineer char(8) not null,
album char(8) not null,
primary key(engineer,album),
foreign key(engineer) references sound_engineers(pid),
foreign key(album) references albums(album_id)
);
```

Album Songs Table

This table stores all descriptive attributes about the songs that are contained in each album. The primary key is song_id.

Create Table Statement

```
create table album_songs(
song_id char(8) not null,
title varchar(25) not null,
track_num int not null,
duration interval default('00:00:00'),
album char(8) not null,
primary key(song_id),
foreign key(album) references Albums(album_id)
);
```

Views

Jazz Guitarists

```
create view Jazz_Guitarists as
select first_name as Jazz_Guitarist_First, last_name as Jazz_Guitarist_Last
from People
where pid in(
        select pid
        from musicians
        where pid in (select musician
                        from Band_Musicians
                        where role_name = 'Guitarist'
                         and band_id in (
                                select band_id
                                from Band_Albums
                                where album_id in(
                                        select album_id
                                        from Albums
                                        where genre = (
                                                select genre_name
                                                from genres
                                                where genre_name = 'Jazz')))));
```

Jazz Pianists

```
create view Jazz_Pianists as
select first_name as Jazz_Pianist_First, last_name as Jazz_Pianist_Last
from People
where pid in(
        select pid
        from musicians
        where pid in (select musician
                        from Band_Musicians
                        where role_name = 'Pianist'
                         and band_id in (
                                select band_id
                                from Band_Albums
                                where album_id in(
                                        select album_id
                                        from Albums
                                        where genre = (
                                                select genre_name
                                                from genres
                                                where genre_name = 'Jazz')))));
```

Blues Songs

```
create view Blues_Songs
select distinct
                bands.band_name as "Artist",
                album_songs.title as "song_name" ,
                album_songs.duration,
                albums.title as "album_title",
                albums.year_produced
from
        bands inner join band_albums on bands.band_id = band_albums.band_id
        inner join albums on band_albums.album_id = albums.album_id
        inner join album_songs on albums.album_id = album_songs.album
        inner join genres on albums.genre = genres.genre_name
        where genres.genre_name = 'Blues';
```

Rock Songs

```
create view Rock_Songs as
select distinct
        bands.band_name as "Artist",
        album_songs.title as "song_name" ,
        album_songs.duration,
        albums.title as "album_title",
        albums.year_produced
from bands inner join band_albums on bands.band_id = band_albums.band_id
        inner join albums on band_albums.album_id = albums.album_id
        inner join genres on albums.genre = genres.genre_name
        where genres.genre_name = 'Rock'
```

<u>Stored Procedures:</u>

<u>Albums for a given band</u>

Function Definition:

```
create function bandsAlbums(band text)
returns TABLE(title char(50), year_produced int, genre varchar(25)) as $$
begin
    return query select albums.title, albums.year_produced, albums.genre
    from bands
        inner join band_albums on bands.band_id = band_albums.band_id
        inner join albums on band_albums.album_id = albums.album_id
        where bands.band_name = 'Flounder';
end;
$$ language PLPGSQL
```

Function Call:

```
select bandsAlbums('Flounder')
```

Musicians for a given band

Function definition:

```
	create function bandsMusicians(band text)
	returns TABLE (first_name char(25), last_name char(25)) as $$
	begin
		return query select people.first_name, people.last_name
		from people
			inner join musicians on people.pid = musicians.pid
			inner join band_musicians b on b.musician = musicians.pid
			inner join bands on bands.band_id = b.band_id
			where bands.band_name = band;
	end;
	$$ language PLPGSQL
```

Function Call:

```
select bandsMusicians('Flounder');
```

Engineers for a Given Album

Function Definition:

```
create function Albums_Engineers(albumname text)
returns TABLE (first_name char(25), last_name char(25)) as $$
begin
        return query select people.first_name, people.last_name
        from people
        inner join sound_engineers SE on SE.pid = People.pid
        inner join album_engineers AE on AE.engineer = SE.pid
        inner join albums on albums.album_id = AE.album
        where albums.title = albumname;
end;
$$ language PLPGSQL
```

Function Call:

```
select Albums_Engineers('Snow Blues')
```

Albums an engineer has worked on

Function Definition:

```
create function engineersWorks(engineer_name text)
returns table (title char(50), year_produced int, genre varchar(25)) as $$
begin
	return query select albums.title, albums.year_produced, albums.genre
	from albums
	inner join album_engineers AE on albums.album_id = AE.album
	inner join sound_engineers SE on AE.engineer = SE.pid
	inner join people on people.pid = SE.pid
	where people.last_name = 'Yoyo';
end;
$$ language PLPGSQL
```

Function Call:

```
select engineersWorks('Yoyo')
```

Albums a musician has worked on

Function Definition:

```
create function musiciansWork(musician_name text)
returns table (title char(50), year_produced int, genre varchar(25)) as $$
begin
	return query select distinct albums.title, albums.year_produced, albums.genre
	from albums
	inner join band_albums BA on BA.album_id = albums.album_id
	inner join bands on bands.band_id = BA.band_id
	inner join band_musicians BM on BM.band_id = bands.band_id
	inner join Musicians on musicians.pid = BM.musician
	inner join People on people.pid = Musicians.pid
	where People.last_name = musician_name;
end;
$$ language PLPGSQL
```

Function Call

```
select musiciansWork('Hendrix')
```