

Wires Database

Matt Spitulnik



May 7, 2022

IST 659

Deliverable 1

Table of Contents

[Part 1 2](#_Toc106720940)

[Summary 2](#_Toc106720941)

[Stakeholders 2](#_Toc106720942)

[Business Rules 2](#_Toc106720943)

[Data Questions 2](#_Toc106720944)

[Conceptual Model 3](#_Toc106720945)

[Logical Model 4](#_Toc106720946)

[Part 2 4](#_Toc106720947)

[Data Definition Language – Physical Database Design 4](#_Toc106720948)

[Data Manipulation Language – Data Creation 6](#_Toc106720949)

[Data Manipulation 13](#_Toc106720950)

[Answering Data Questions 15](#_Toc106720951)

[User Interface 24](#_Toc106720952)

[Reflection 29](#_Toc106720953)

# Part 1

## Summary

My wife and I purchased our home in 2019, and since then we have slowly been putting it in order. Throughout our house, on every floor, in every room, I probably have over 100 various cables, cords, and chargers for all different kinds of electronics. One of my next projects is to get all these wires categorized and organized once and for all. I will be using this opportunity to build a database that I can use to plan how I will store them going forward.

## Stakeholders

* Myself
  + The person that owns the wires and would like to be able to continue doing so.
* My Wife
  + The person that has been very patient so far but is bound to start asking questions someday.
* Our Children
  + Re-organizing and consolidating the wires will probably open more space to store additional toys.

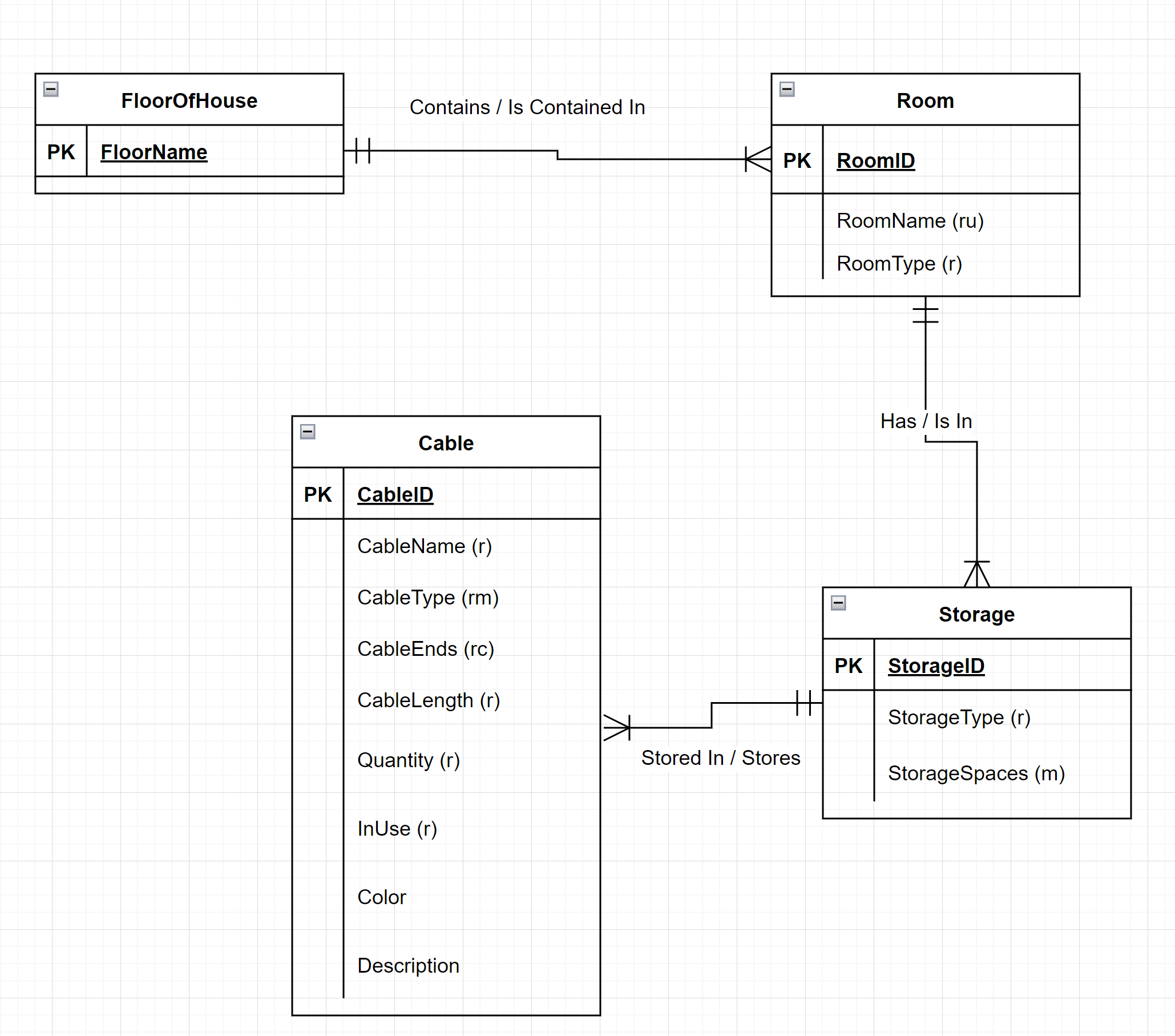
## Business Rules

* Any power cables that are permanently attached to electronics are not going to be counted, only standalone and detachable power cables will be included in the count.
* The CableName category will be in respect to its primary function, for example; extension cord, HDMI cable, USB-c cable, VGA cable, etc.
* The CableEnd1 and CableEnd2 categories will reflect specifically what comprises the cable. For example, a USB to USB-c cable would have CableEnd1=USB-c, and CableEnd2=USB.
* The CableType category will reflect which of these functions it is used for: power, data transfer, audio, video, audio/video, networking, and maintenance.
* Any cable that is currently in use will not be considered in any kind of storage, but will be connected to the “None” attribute in in the Storage table.
* Some storage devices, like backpacks and briefcases, store cables but are mobile and won’t have dedicated rooms or floors.
* The five floors in the house are the following: FirstFloor, SecondFloor, Attic, Basement, Garage. The Basement and Garage do not have individual rooms, so these floors will have rooms also called Basement and Garage so that items in the Storage table can connect back to the Floor table through the Room table.
* The cable lengths will all be reflected in feet.

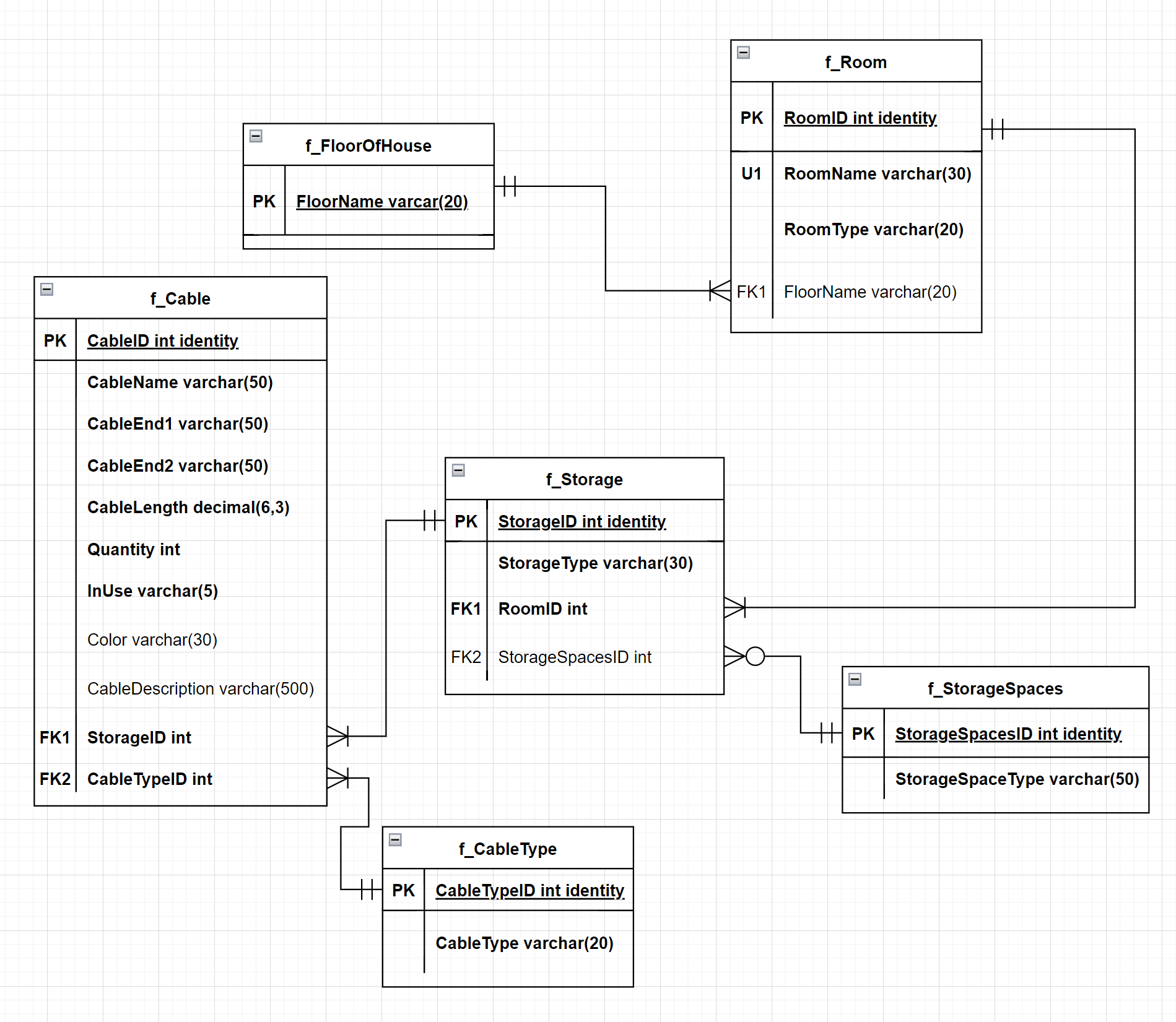
## Data Questions

* How can I better organize and store the different cables in our house?
* Which cables can be consolidated and spaced freed up?
* Which cables do I have excess of and can get rid of?
* Which cables are obsolete and therefor no longer needed?

## Conceptual Model



## Logical Model



# Part 2

## Data Definition Language – Physical Database Design

-----------------------Put in Drop Statements so Code Can be Continually Run

DROP TABLE IF EXISTS f\_Cable

DROP TABLE IF EXISTS f\_CableType

DROP TABLE IF EXISTS f\_Storage

DROP TABLE IF EXISTS f\_StorageSpaces

DROP TABLE IF EXISTS f\_Room

DROP TABLE IF EXISTS f\_FloorOfHouse

----------------------- Start of Table Creation

-- Creating the FloorOfHouse table

CREATE TABLE f\_FloorOfHouse (

-- Column for FloorOfHouse

FloorName varchar(20) not null

-- Contraint on FloorOfHouse Table

CONSTRAINT PK\_f\_FloorOfHouse PRIMARY KEY (FloorName)

)

-- End of FloorOfHouse Creation

-- Creating the Room table

CREATE TABLE f\_Room (

-- Columns for Room table

RoomID int identity,

RoomName varchar(30) not null,

RoomType varchar(20) not null,

FloorName varchar(20)

-- Constraints

CONSTRAINT PK\_f\_Room PRIMARY KEY (RoomID),

CONSTRAINT U1\_f\_Room UNIQUE(RoomName),

CONSTRAINT FK1\_f\_Room FOREIGN KEY (FloorName) REFERENCES f\_FloorOfHouse (FloorName)

)

-- End of Room Table creation

-- Creating StorageSpaces Table

CREATE TABLE f\_StorageSpaces (

-- Columns for StorageSpaces

StorageSpacesID int identity,

StorageSpaceType varchar(50) not null

-- Constraints for StorageSpaces

CONSTRAINT PK\_StorageSpaces PRIMARY KEY (StorageSpacesID)

)

-- End StorageSpaces Creation

-- Creating Storage table

CREATE TABLE f\_Storage (

-- Columns for Storage table

StorageID int identity,

StorageType varchar(30) not null,

RoomID int not null,

StorageSpacesID int

-- Constraints on Storage Table

CONSTRAINT PK\_Storage PRIMARY KEY (StorageID),

CONSTRAINT FK1\_Storage FOREIGN KEY (RoomID) REFERENCES f\_Room(RoomID),

CONSTRAINT FK2\_Storage FOREIGN KEY (StorageSpacesID) REFERENCES f\_StorageSpaces(StorageSpacesID)

)

-- End Storage table creation

-- Creating the CableType table

CREATE TABLE f\_CableType (

-- Columns for CableType

CableTypeID int identity,

CableType varchar(20) not null

-- Constraints on CableType table

CONSTRAINT PK\_f\_CableType PRIMARY KEY (CableTypeID)

)

-- End CableType Table Creation

-- Creating the Cable table

CREATE TABLE f\_Cable (

-- Columns for Cable table

CableID int identity,

CableName varchar(50) not null,

CableEnd1 varchar(50) not null,

CableEnd2 varchar(50) not null,

CableLength decimal(6,2) not null,

Quantity int not null,

InUse varchar(5),

Color varchar(30),

CableDescription varchar(500),

StorageID int not null,

CableTypeID int not null

-- Constraints for Cable table

CONSTRAINT PK\_f\_Cable PRIMARY KEY (CableID),

CONSTRAINT FK1\_f\_Cable FOREIGN KEY (StorageID) REFERENCES f\_Storage(StorageID),

CONSTRAINT FK2\_f\_Cable FOREIGN KEY (CableTypeID) REFERENCES f\_CableType(CableTypeID)

)

## Data Manipulation Language – Data Creation

-----------------------Start of Data Creation

-- Insert data for FloorOfHouse

INSERT INTO f\_FloorOfHouse(FloorName)

VALUES

('No Floor'),

('Garage'),

('Basement'),

('1st Floor'),

('2nd Floor'),

('3rd Floor'),

('Attic')

-- Insert data for existing Rooms

INSERT INTO f\_Room(RoomName, RoomType, FloorName)

VALUES

('Garage', 'Storage','Garage'),

('Basement', 'Storage', 'Basement'),

('Mud Room','Functional', '1st Floor'),

('Kitchen', 'Functional', '1st Floor'),

('Laundry', 'Functional', '1st Floor'),

('Bathroom 1', 'Functional', '1st Floor'),

('Den', 'Living', '1st Floor'),

('Living Room', 'Living', '1st Floor'),

('Dining Room', 'Living', '1st Floor'),

('Sun Porch', 'Living', '1st Floor'),

('Bathroom 2', 'Functional', '2nd Floor'),

('Kid''s Room', 'Bed', '2nd Floor'),

('Parent''s Room', 'Bed', '2nd Floor'),

('Guest''s Room', 'Bed', '2nd Floor'),

('Office', 'Functional', 'Attic'),

('Man Cave', 'Living', 'Attic'),

('Workout Area', 'Functional', 'Attic'),

('No Room', 'None', 'No Floor')

-- Insert data for StorageSpaces

INSERT INTO f\_StorageSpaces(StorageSpaceType)

VALUES

('Drawer'),

('Shelf'),

('Pocket'),

('Trunk')

-- Insert data for Storage

INSERT INTO f\_Storage(StorageType, RoomID, StorageSpacesID)

VALUES

('None',1,NULL),

('Car', 1, 4),

('Shelving', 1, 2),

('None',2,Null),

('Tool Chest', 2, 1),

('Tool Chest', 2, 2),

('Cabinets', 2, 2),

('None',3,NULL),

('Rolling Island', 3, 1),

('None',4,NULL),

('Green Buffet', 4, 1),

('Kitchen Cabinets', 4, 1),

('None',5,NULL),

('Cabinets', 5, 2),

('None', 6, NULL),

('Medicine Cabinet', 6, 2),

('None',7, NULL),

('TV Stand', 7, 1),

('TV stand', 7, 2),

('Storage Cabinet', 7, 1),

('Storage Cabinet', 7, 2),

('None', 8, NULL),

('TV Stand', 8, 1),

('TV Stand', 8, 2),

('Side Table', 8, 1),

('None', 9, NULL),

('Buffet', 9, 1),

('None', 10, NULL),

('Metal Cabinet', 10, 1),

('Desk Cabinet', 10, 1),

('Desk Cabinet', 10, 2),

('None', 11, NULL),

('Medicine Cabinet', 11, 2),

('None', 12, NULL),

('Dresser', 12, 1),

('None', 13, NULL),

('Left Bedside Table', 13, 1),

('Right Bedside Table', 13, 1),

('Closet', 13, 2),

('None', 14, NULL),

('Media Cabinet', 14, 2),

('Desk Cabinet', 14, 1),

('Desk Cabinet', 14, 2),

('Bedside Table', 14, 1),

('Closet', 14, 2),

('None', 15, NULL),

('File Cabinet', 15, 1),

('File Cabinet', 15, 2),

('None', 16, NULL),

('Storage Cabinet', 16, 1),

('Storage Cabinet', 16, 2),

('Bookshelf', 16, 2),

('None', 17, NULL),

('Storage Cabinet', 17, 1),

('Storage Cabinet', 17, 2),

('TV Stand', 17, 1),

('None', 18, NULL),

('Laptop Bag', 18, 3),

('Backpack', 18, 3)

--Insert data for CableType

INSERT INTO f\_CableType(CableType)

VALUES

('Power'),

('Data Transfer'),

('Audio'),

('Video'),

('Audio/Video'),

('Networking'),

('Maintenance')

-- Insert data in Cable

/\*

In order to get all of the cable data into the Cable table, I manually performed data entry in an excel sheet, then imported that data into a table named "ImportedCableData" using the "Import Flat File" option. I then copied the data from ImportedCableData to f\_Cable

\*/

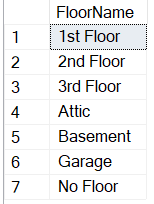
-- Copy the data from ImportedCableData to f\_Cable

INSERT INTO f\_cable SELECT \* FROM ImportedCableData

---------------------------------- Check data in tables.

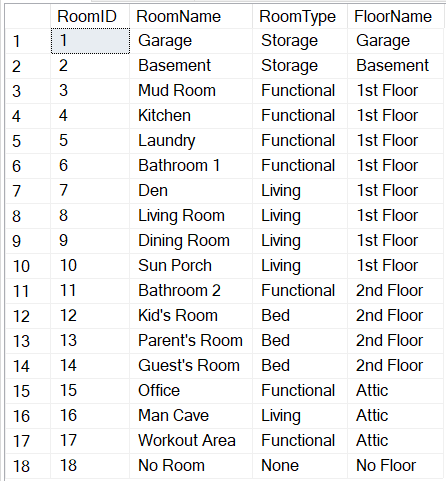
-- Check data in FloorOfHouse

SELECT \* FROM f\_FloorOfHouse



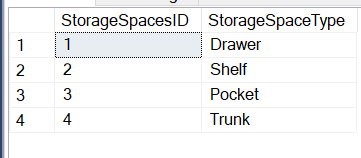
-- Check data in Rooms

SELECT \* FROM f\_Room



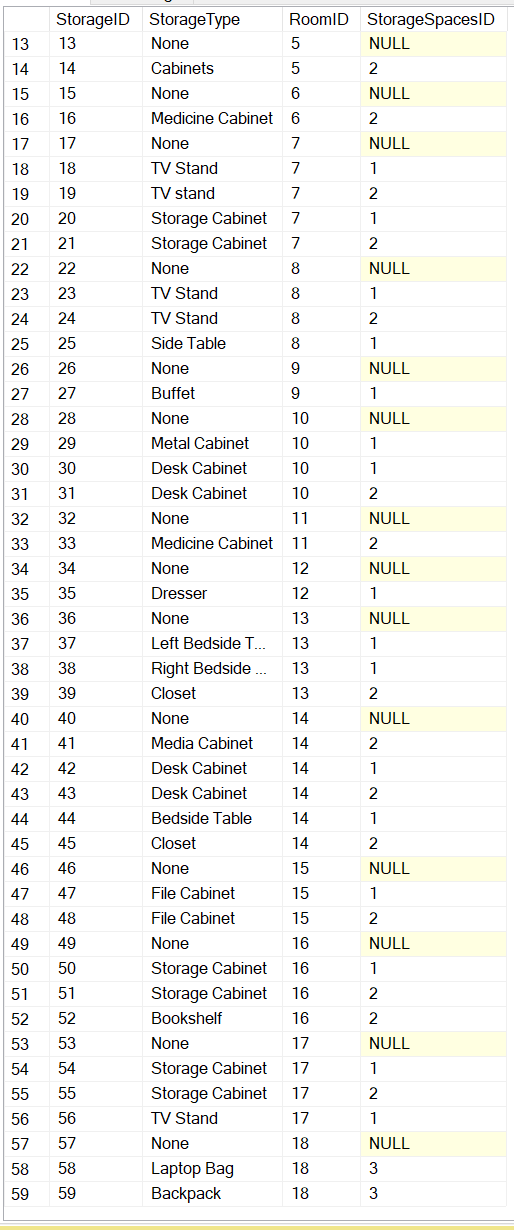
-- Check data in StorageSpace

SELECT \* FROM f\_StorageSpaces



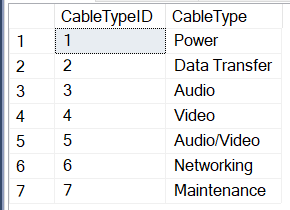
-- Check data in Storage

SELECT \* FROM f\_Storage



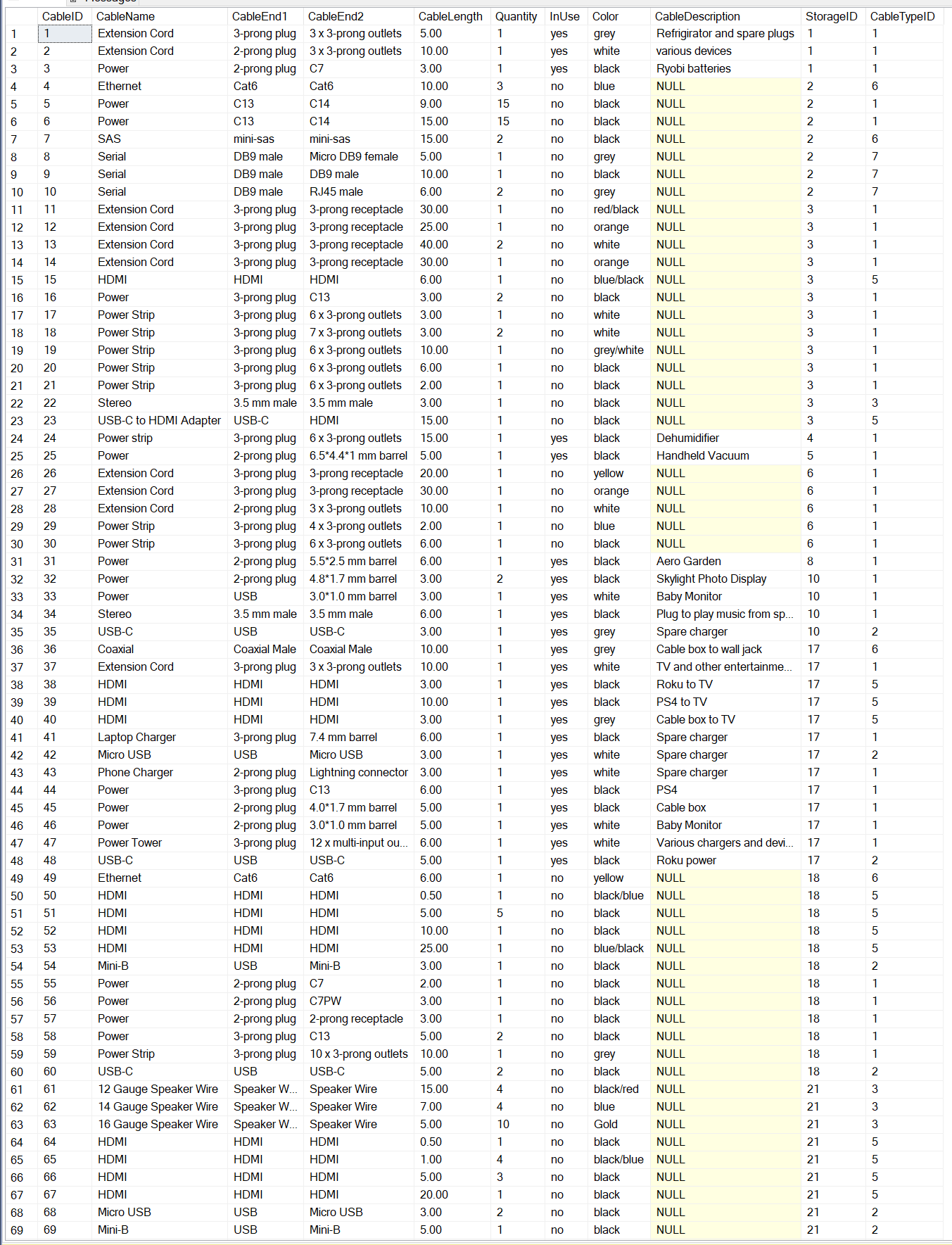
-- Check data in CableType

SELECT \* FROM f\_CableType



-- Check data in f\_Cable (not all entries shown)

SELECT \* FROM f\_Cable



## Data Manipulation

-------------------------------------Create Procedures

-- Create Procedure for Adding StorageSpaces if New Storage is Purchased and has New Types of Storage Space

GO

CREATE OR ALTER PROCEDURE f\_AddStorageSpace(@NewStorageSpaceType varchar(50)) AS

BEGIN

INSERT INTO f\_StorageSpaces(StorageSpaceType)

VALUES (@NewStorageSpaceType)

END

-- Test the f\_AddStorageSpace procedure.

EXEC f\_AddStorageSpace 'SecretPassage'

-- Confrim the add worked.

SELECT \* FROM f\_StorageSpaces

-- Create Procedure for Deleting a StorageSpace line item

GO

CREATE OR ALTER PROCEDURE f\_DelStorageSpace(@DelStorageSpaceID int) AS

BEGIN

DELETE FROM f\_StorageSpaces WHERE f\_StorageSpaces.StorageSpacesID = @DelStorageSpaceID

END

-- Test the delete storage space procedure.

EXEC f\_DelStorageSpace 5

-- Confirm the delete worked.

SELECT \* FROM f\_StorageSpaces

-- Create Procedure for Adding a new Type of Storage

GO

CREATE OR ALTER PROCEDURE f\_AddStorageType(@NewStorageSpaceType varchar(50), @InputRoomID int, @InputStorageSpacesID int) AS

BEGIN

INSERT INTO f\_Storage(StorageType, RoomID, StorageSpacesID)

VALUES (@NewStorageSpaceType, @InputRoomID, @InputStorageSpacesID)

END

-- Test the add storage type procedure.

EXEC f\_AddStorageType BottomLessChest,15, 1

-- Confirm it worked.

SELECT \* FROM f\_Storage

-- Create Procedure for Deleting a Storage type line item

GO

CREATE OR ALTER PROCEDURE f\_DelStorageType(@DelStorageID int) AS

BEGIN

DELETE FROM f\_Storage WHERE f\_Storage.StorageID = @DelStorageID

END

-- Test that the delete storage ID procedure works.

EXEC f\_DelStorageType 60

-- Confirm that it worked.

SELECT \* FROM f\_Storage

-- Create Procedure for Adding a New Cable Type

GO

CREATE OR ALTER PROCEDURE f\_AddCableType(@NewCableType varchar(50)) AS

BEGIN

INSERT INTO f\_CableType(CableType)

VALUES (@NewCableType)

END

-- Test that add cable type procedure works

EXEC f\_AddCableType VirtualRealityCable

-- Confirm that it worked.

SELECT \* FROM f\_CableType

-- Create Procedure for Deleting a Cable Type

GO

CREATE OR ALTER PROCEDURE f\_DelCableType(@DelCableTypeID int) AS

BEGIN

DELETE FROM f\_CableType WHERE CableTypeID = @DelCableTypeID

END

-- Test that delete cable type procedure works.

EXEC f\_DelCableType 8

-- Confirm that it worked.

SELECT \* FROM f\_CableType

-- Create Procedure for Adding a New Cable

GO

CREATE OR ALTER PROCEDURE f\_AddCable(

@NewCable varchar(50),

@NewCableEnd1 varchar(50),

@NewCableEnd2 varchar(50),

@NewCableLength decimal(6,3),

@NewCableQuantity int,

@NewCableInUse varchar(5),

@NewCableColor varchar(30),

@NewCableDescription varchar(500),

@NewStorageID int,

@NewCableTypeID int)

AS

BEGIN

INSERT INTO f\_Cable(CableName,

CableEnd1,

CableEnd2,

CableLength,

Quantity,

InUse,

Color,

CableDescription,

StorageID,

CableTypeID)

VALUES (@NewCable,

@NewCableEnd1,

@NewCableEnd2,

@NewCableLength,

@NewCableQuantity,

@NewCableInUse,

@NewCableColor,

@NewCableDescription,

@NewStorageID,

@NewCableTypeID)

END

-- Test add new cable procedure.

EXEC f\_AddCable 'Extension Cord', '3-prong plug', '3-prong receptacle', 30, 1, 'no', 'orange', null, 3, 1

-- Confirm it worked

SELECT \* FROM f\_Cable

-- Create Procedure for Deleting a Cable

GO

CREATE OR ALTER PROCEDURE f\_DelCable(@DelCableID int) AS

BEGIN

DELETE FROM f\_Cable WHERE CableID = @DelCableID

END

-- Test delete cable procedure.

EXEC f\_DelCable 234

-- Confirm it worked.

SELECT \* FROM f\_Cable

-- Create procedure for updating cable quantity in the cable table if cables are added or removed.

CREATE OR ALTER PROCEDURE f\_UpdateCableQuantity(@CableID int, @UpdateCableQuantity int) AS

BEGIN

UPDATE f\_Cable SET Quantity = @UpdateCableQuantity

WHERE f\_Cable.CableID = @CableID

END

--Test that updating the cable quantity works.

SELECT \* FROM f\_Cable

EXEC f\_UpdateCableQuantity 1,3

## Answering Data Questions

-------------------------------------Create Floor/Room Function

-- Create the FloorRoom function, which allows user to input if they want to look at a specific floor or room, then the specific floor or room, and then a cable type, and then returns how many of those types of cables in that floor or room they have.

CREATE OR ALTER FUNCTION dbo.f\_FloorRoom(@FloorOrRoom varchar(30), @SpecificFloorRoom varchar(30), @CableType varchar(30))

RETURNS int AS

BEGIN

DECLARE @CableTypeCount int

/\*

Create an IF statement within the function that determines if the FloorRoomDecided variable should be pulled from the Floor or Room table.

\*/

IF (@FloorOrRoom = 'Floor')

BEGIN

SELECT @CableTypeCount = SUM(Quantity) FROM f\_Cable

LEFT JOIN f\_Storage ON f\_Storage.StorageID = f\_Cable.StorageID

LEFT JOIN f\_Room ON f\_Room.RoomID = f\_Storage.RoomID

LEFT JOIN f\_FloorOfHouse ON f\_FloorOfHouse.FloorName = f\_Room.FloorName

LEFT JOIN f\_CableType ON f\_CableType.CableTypeID = f\_Cable.CableTypeID

WHERE f\_FloorOFHouse.FloorName = @SpecificFloorRoom AND CableType = @CableType

END

ELSE

BEGIN

SELECT @CableTypeCount = SUM(Quantity) FROM f\_Cable

LEFT JOIN f\_Storage ON f\_Storage.StorageID = f\_Cable.StorageID

LEFT JOIN f\_Room ON f\_Room.RoomID = f\_Storage.RoomID

LEFT JOIN f\_FloorOfHouse ON f\_FloorOfHouse.FloorName = f\_Room.FloorName

LEFT JOIN f\_CableType ON f\_CableType.CableTypeID = f\_Cable.CableTypeID

WHERE f\_Room.RoomName = @SpecificFloorRoom AND CableType = @CableType

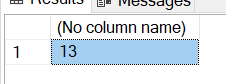
END

RETURN @CableTypeCount

END

-- Run the function.

SELECT dbo.f\_FloorRoom('Room', 'Den', 'Power')



-------------------------------------Create Views

-- Create a view to see the total count of cables in the house

CREATE OR ALTER VIEW TotalCableCount AS

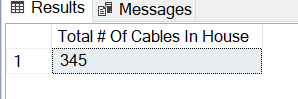
SELECT SUM(Quantity) AS "Total # Of Cables In House" FROM f\_Cable

GO

-- Test TotalCableCount

SELECT \* FROM TotalCableCount

-- I estimated at least 100 cables but was a bit off…



-- Create view to see total cable count per floor

GO

CREATE OR ALTER VIEW CablesPerFloor AS

SELECT

f\_FloorOfHouse.FloorName AS "Floor of House",

SUM(f\_Cable.Quantity) "# of Cables"

FROM f\_Cable

JOIN f\_Storage ON f\_Storage.StorageID = f\_Cable.StorageID

JOIN f\_Room ON f\_Room.RoomID = f\_Storage.RoomID

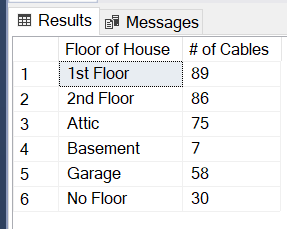
JOIN f\_FloorOfHouse ON f\_FloorOfHouse.FloorName = f\_Room.FloorName

GROUP BY f\_FloorOfHouse.FloorName

-- Test Cables Per Floor

SELECT \* FROM CablesPerFloor

-- Cables are pretty evenly distributed between the three main floors, need to get most of them up to the attic.



-- Create view to see total cable type per floor

GO

CREATE OR ALTER VIEW CableTypesPerFloor AS

SELECT

f\_FloorOfHouse.FloorName AS "Floor of House",

f\_CableType.CableType AS "Cable Type",

SUM(f\_Cable.Quantity) "# of Cables"

FROM f\_Cable

JOIN f\_Storage ON f\_Storage.StorageID = f\_Cable.StorageID

JOIN f\_Room ON f\_Room.RoomID = f\_Storage.RoomID

JOIN f\_FloorOfHouse ON f\_FloorOfHouse.FloorName = f\_Room.FloorName

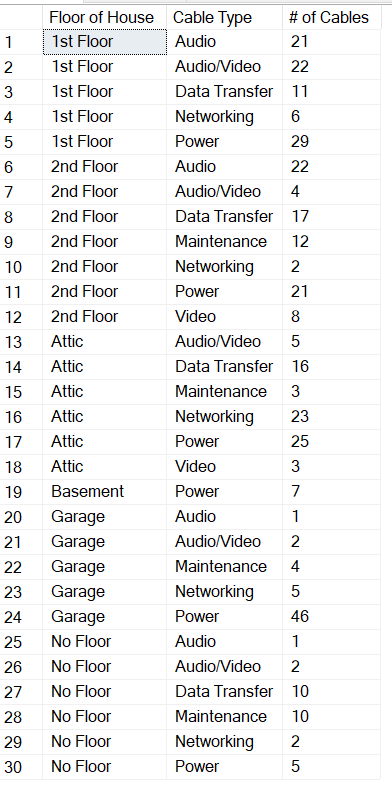
JOIN f\_CableType ON f\_CableType.CableTypeID = f\_Cable.CableTypeID

GROUP BY f\_FloorOfHouse.FloorName, f\_CableType.CableType

-- Test Cable Types Per Floor

SELECT \* FROM CableTypesPerFloor ORDER BY "Floor of House"

--Lots of power cables around the house.



-- View Total Number of Cables based on Cable Name and Cable Ends

GO

CREATE OR ALTER VIEW TotalCableName AS

SELECT

CableName,

CableEnd1,

CableEnd2,

SUM(Quantity) AS "# of Cables"

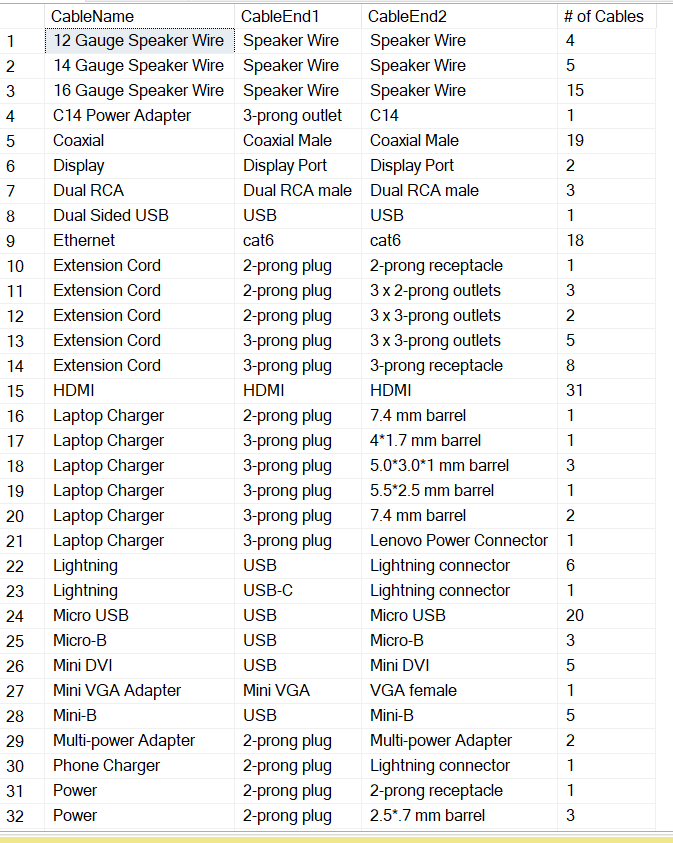
FROM f\_Cable

GROUP BY CableName, CableEnd1, CableEnd2

-- View total number of cables by cable name and ends (not all cables shown)

SELECT \* FROM TotalCableName

--Lots of HDMI and C13 to C14 Power Cables.



-- View Breakdown of Cables Being Used and Not Used

GO

CREATE OR ALTER VIEW CablesUsed AS

SELECT

CableName,

InUse,

SUM(Quantity) AS Quantity

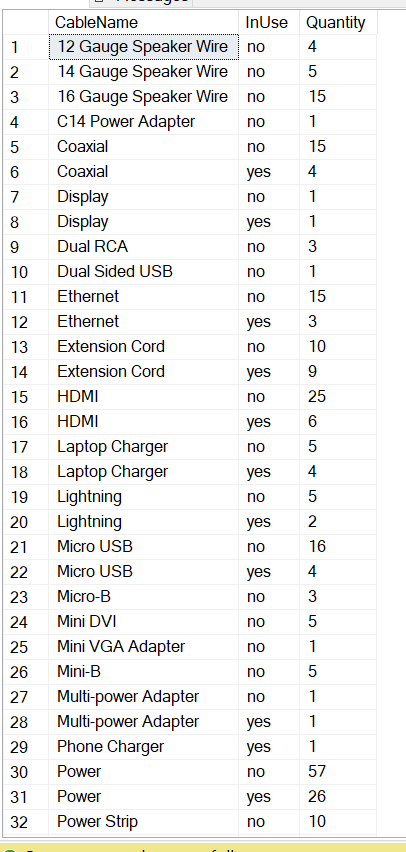
FROM f\_Cable

GROUP BY InUse, CableName

-- View Cables Used and not used (not all cables shown).

SELECT \* FROM CablesUsed ORDER BY CableName

--Not using a whole lot of the total HDMI cables I have.



-- Breakdown of cables being used versus not used with cable Ends Included

CREATE OR ALTER VIEW CablesUsed AS

SELECT

CableName,

InUse,

CONCAT(CableEnd1,(' to '), CableEnd2) AS "Full Cable",

SUM(Quantity) AS Quantity

FROM f\_Cable

GROUP BY InUse, CableName, CableEnd1, CableEnd2

-- View how many cables are used and not used with cable ends included.

SELECT \* FROM CablesUsedEnds

--Not using any of the C13 to C14 cables, can probably toss them.



-- Create view that shows cables per storage item per room

GO

CREATE OR ALTER VIEW CablesPerStoragePerRoom AS

SELECT

f\_Room.RoomName AS Room,

f\_Storage.StorageType AS Storage,

SUM(f\_Cable.Quantity) AS "# Of Cables"

FROM f\_Cable

JOIN f\_Storage ON f\_Storage.StorageID = f\_Cable.StorageID

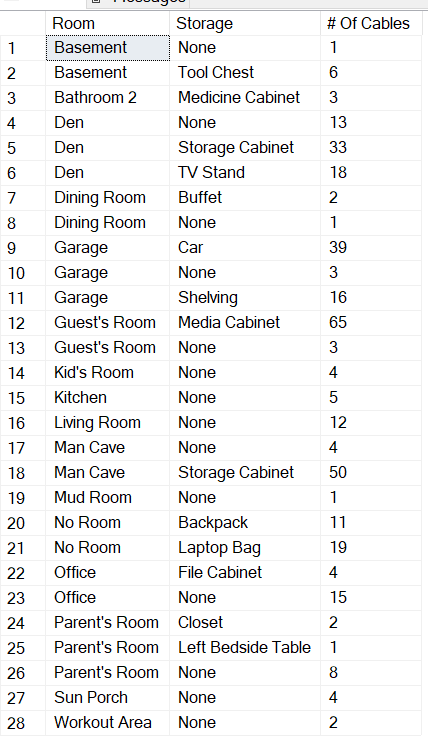
JOIN f\_Room ON f\_Room.RoomID = f\_Storage.RoomID

GROUP BY f\_Room.RoomName, f\_Storage.StorageType

-- View how many cables we have in each piece of storage in each room.

SELECT \* FROM CablesPerStoragePerRoom ORDER BY "Room"

--Media cabinet in the guest room has the most cables to move around, but storage cabinet in Attic has the end most, which makes moving cables to the Attic more complicated.



-- Create view to see Storage Spaces that are available

CREATE or ALTER VIEW AvailableStorageSpace AS

SELECT

f\_FloorOfHouse.FloorName AS "Floor",

f\_Room.RoomName AS Room,

f\_Storage.StorageType AS "Storage Space",

f\_StorageSpaces.StorageSpaceType AS "Available Space",

SUM(f\_Cable.Quantity) AS "Quantity of Cables"

FROM f\_Storage

FULL JOIN f\_Cable ON f\_Cable.StorageID = f\_Storage.StorageID

FULL JOIN f\_Room ON f\_Room.RoomID = f\_Storage.RoomID

FULL JOIN f\_FloorOfHouse ON f\_FloorOfHouse.FloorName = f\_Room.FloorName

FULL JOIN f\_StorageSpaces ON f\_Storage.StorageSpacesID = f\_StorageSpaces.StorageSpacesID

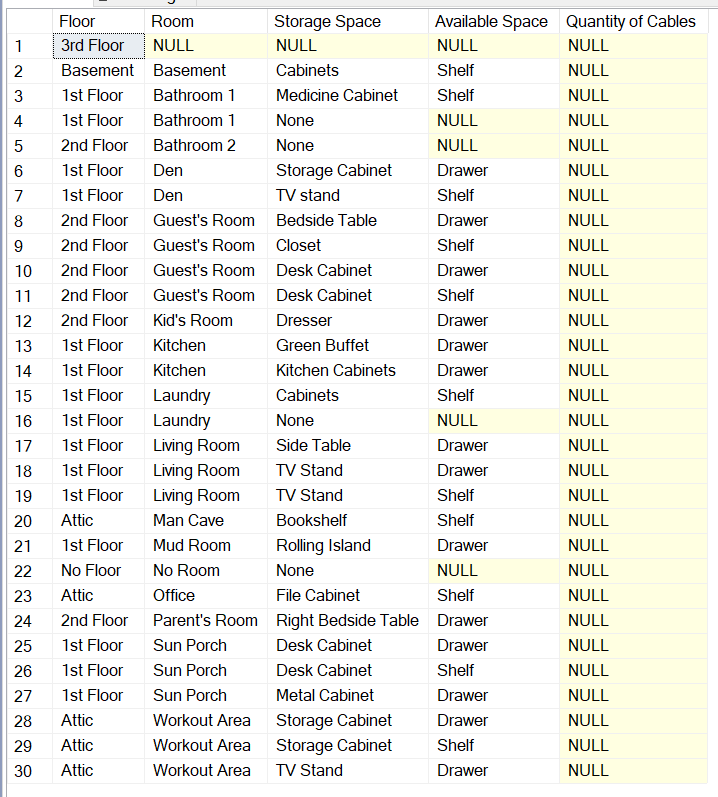
GROUP BY RoomName,f\_FloorOfHouse.FloorName, StorageType, StorageSpaceType

-- Find storage spaces that have no cables

SELECT \* FROM AvailableStorageSpace

SELECT \* FROM AvailableStorageSpace WHERE "Quantity of Cables" IS NULL

--Lots of spaces around the house still available to hold cables.



/\*

Create a view to see how many total feet of cable length I have just in case I ever need to create a makeshift rope to save someone stuck in quicksand or pull off a Mission Impossible-esque heist.

\*/

GO

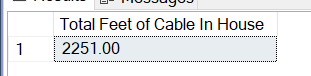
CREATE OR ALTER VIEW MakeshiftRope AS

SELECT SUM(Quantity\*CableLength) AS "Total Feet of Cable In House" FROM f\_Cable

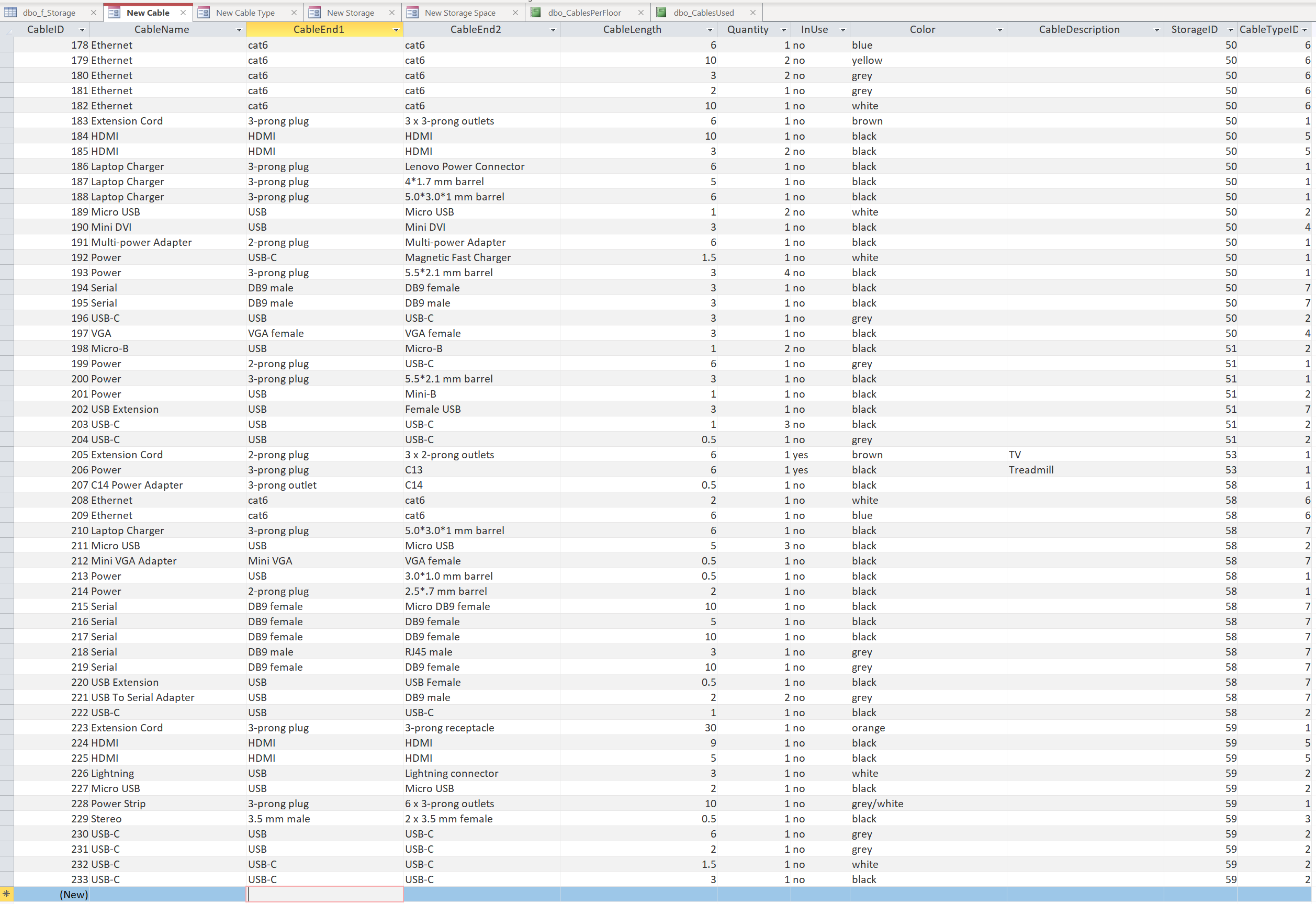
-- Test MakeshiftRope

SELECT \* FROM MakeshiftRope

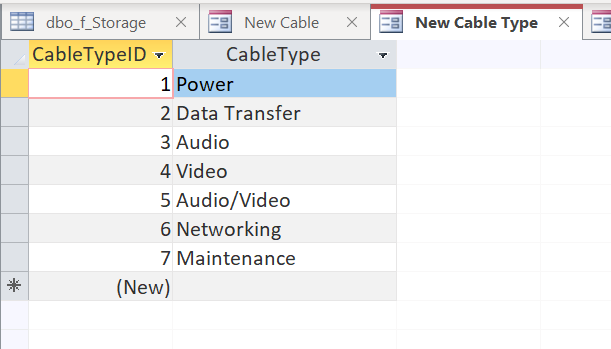
--It would be a very large rope.



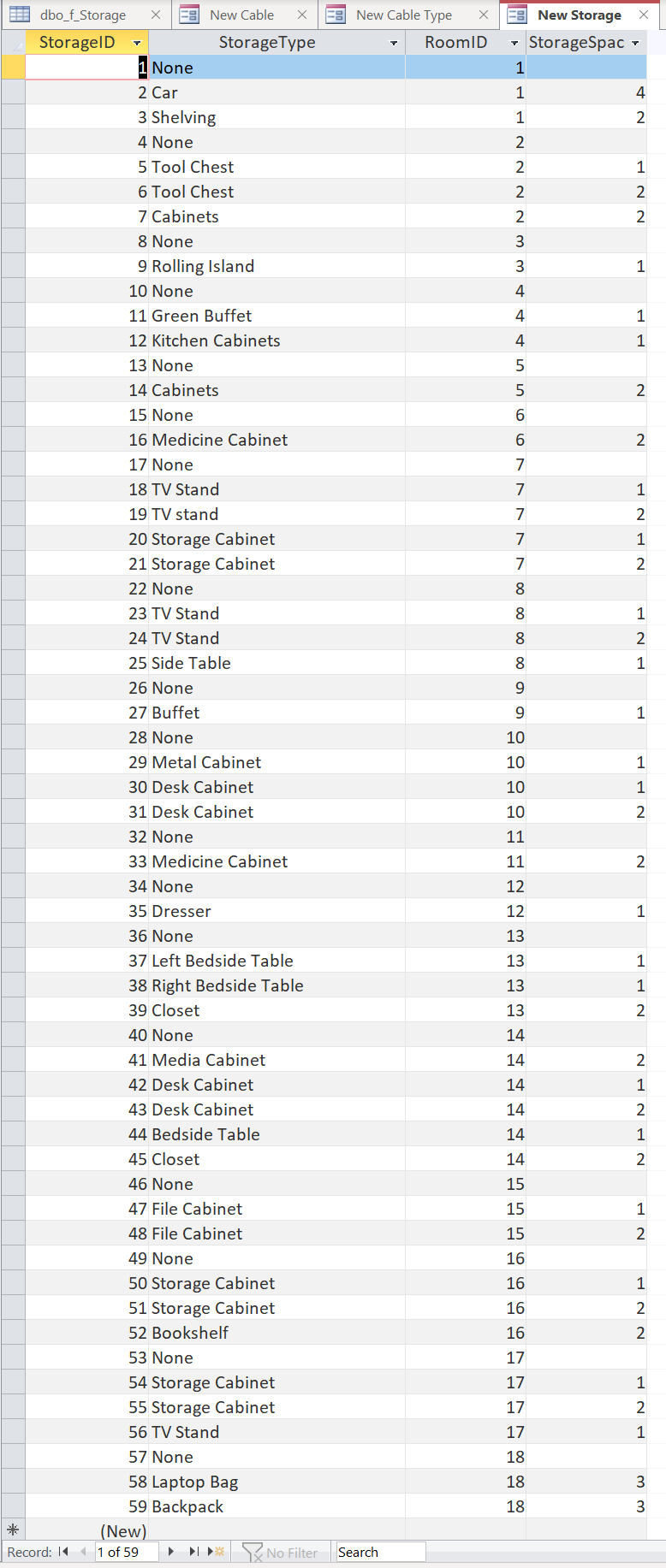
## User Interface

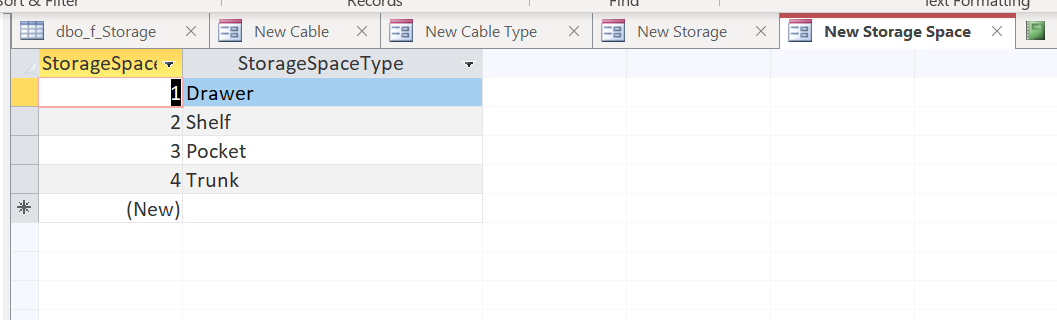
Form for adding new cables:  


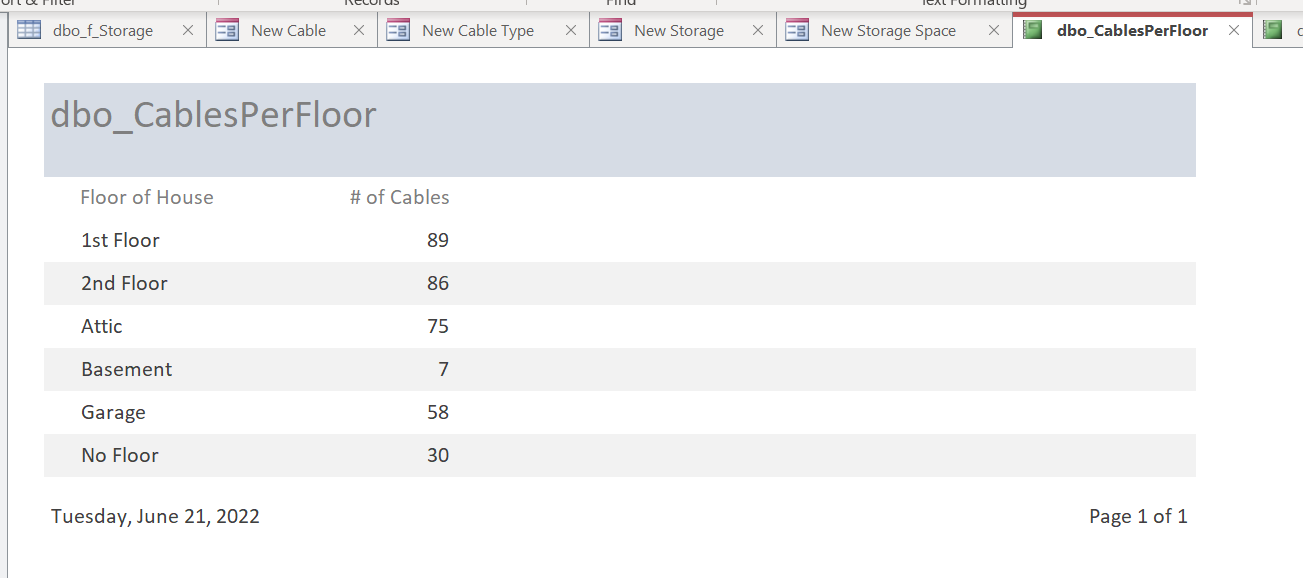
Form for adding new cable type:

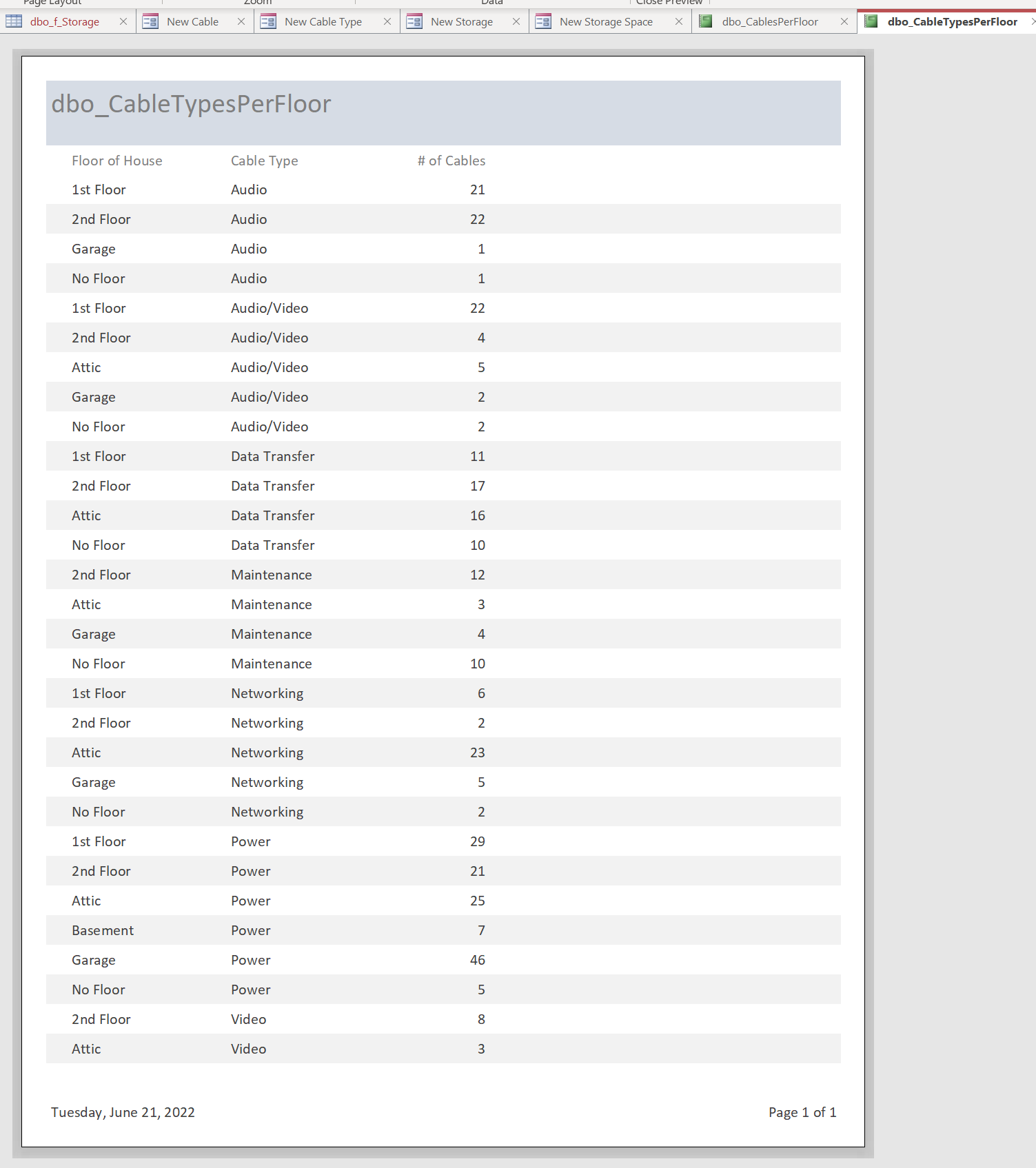


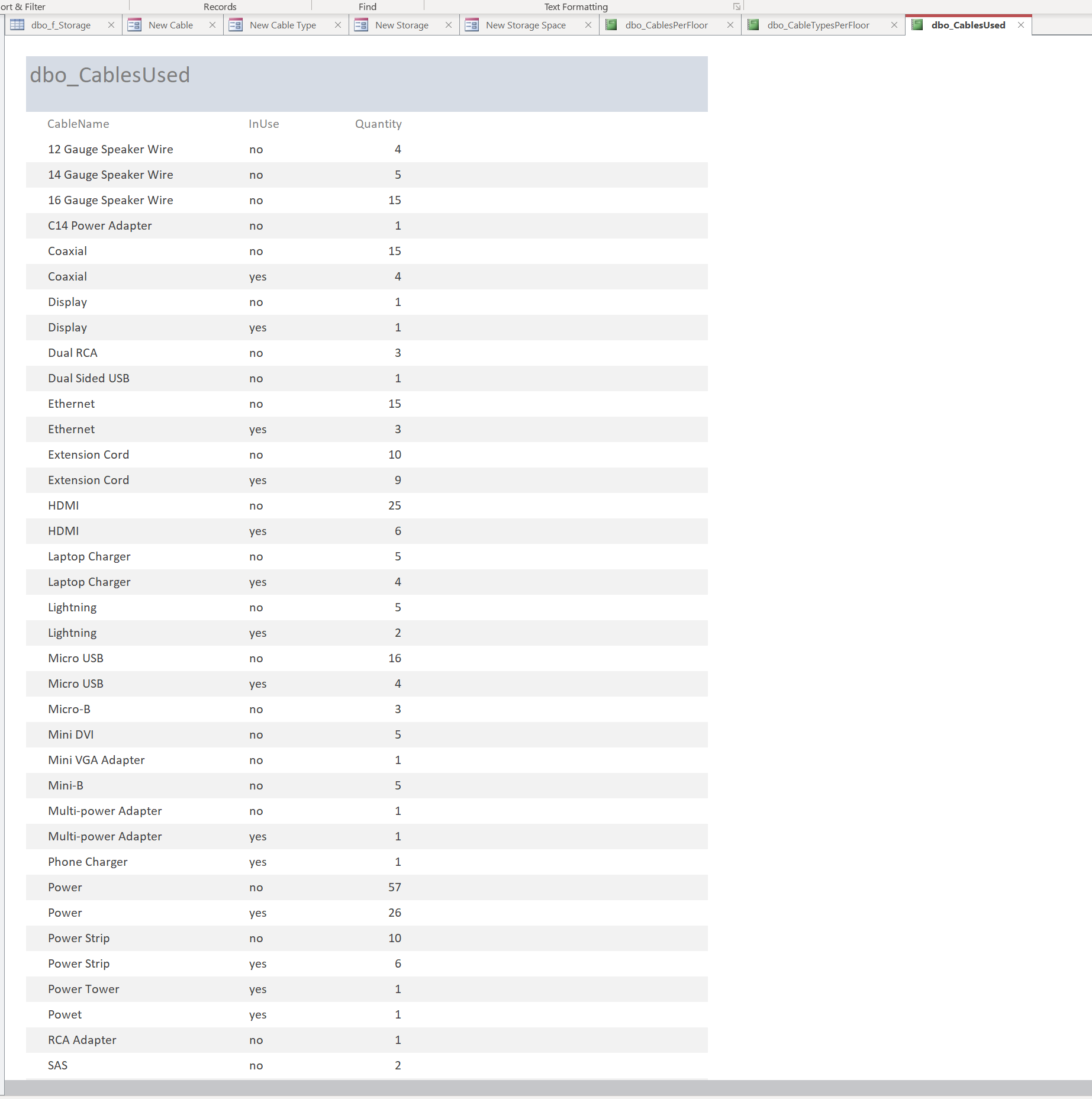
Form for adding new storage:



Form for adding new storage space within the different physical storage:  


Report showing number of cables per floor:  


Report showing number of cable types per floor:  


Report showing number of cables used and not used:  


## Reflection

Now that the project is complete, I think it would be really helpful if I had collected data in the beginning since I already knew what the project was going to be about. Having the data first I think would have led me to building a data base that better fit the data set. For example, there is such a large variation in the different cable names and cable ends I was working with that creating some kind of standardized naming convention could potentially have made the data entry process a lot easier. It would have made the data querying portion a lot simpler too, since it took some work to break the cables down into groups that made sense to analyze. Setting some additional constraints on the CableName, CableEnd1, and CableEnd2 columns also would of made things easier, since I continually forgot over the course of entering 233 line items whether it was supposed to be 3-prong, 3 prong, or 3prong, and I had to go back after the fact to do some manual cleanup. Finally, having some way to have conceptualized how many cables could actually be stored in each form of storage would have added some usability to the database. It obviously would have been rough estimates, but I could have really made some predictions on how I could move cables around from one storage space to another.