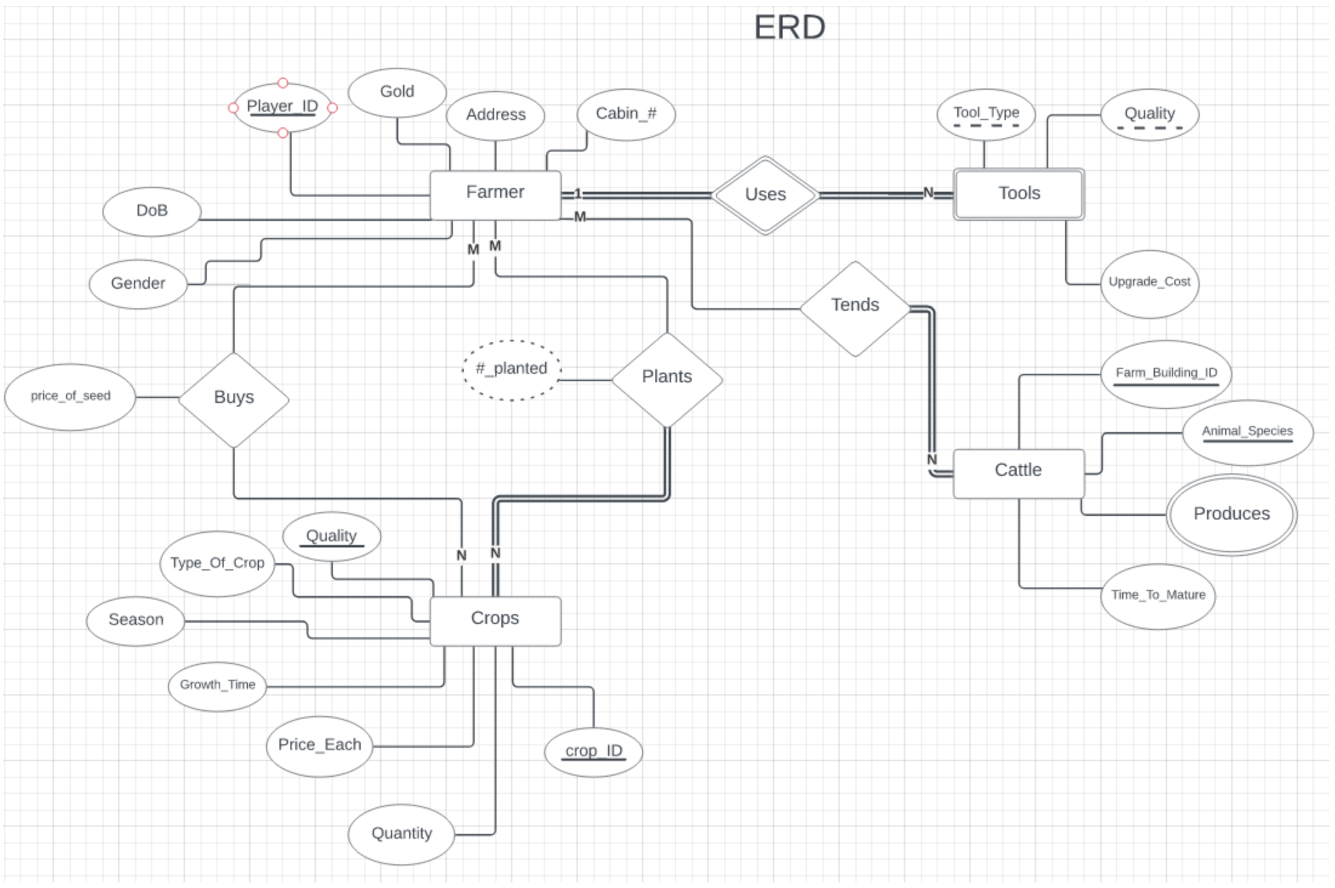
Stardew Valley Farm Database

The purpose of my database is to provide data to the user as if they are currently playing Stardew Valley themselves and need a representation of what is going on in their current playthrough.

The information contained inside this database includes what type of tools each farmer currently has, the crops that are currently present in their game, which farmer buys and plants crops, and what type of cattle are present and producing items on the farm. The user will be able to calculate profits depending on the quantity of their currently owned crops, and will be able to update the rows accordingly based on the transaction. Trying to sell crops that are non-existent or specifying a quantity less than the currently owned amount is not allowed.

ERD



Relational Schema

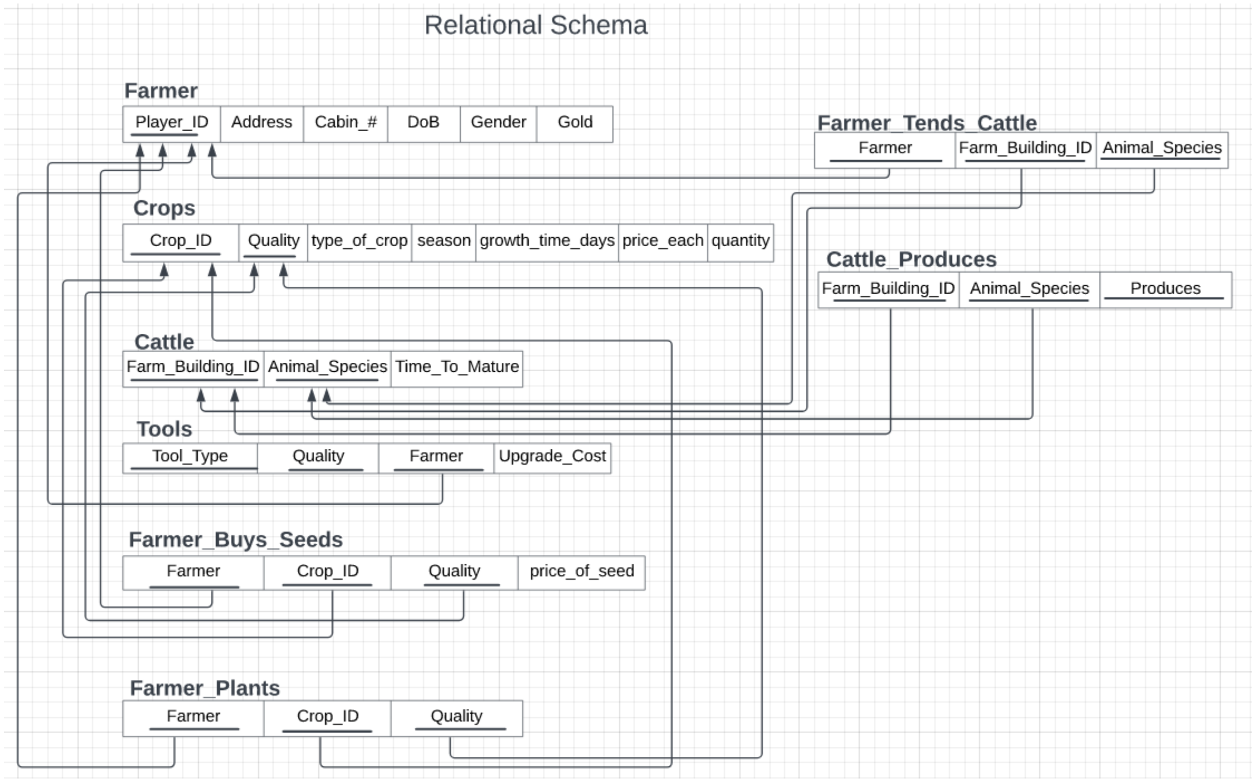


Table Descriptions

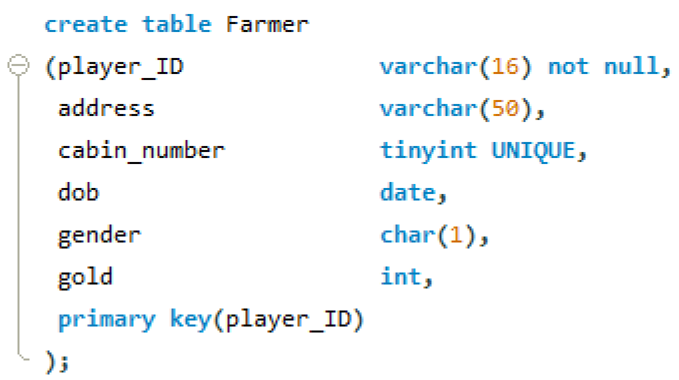
Table 1: Farmer

Purpose: To list which farmers (players) are currently part of this playthrough of the game.

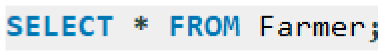
Attributes: player_ID, address, cabin_number, dob, gender, gold

Keys: **player_ID:** Unique to each player in the game

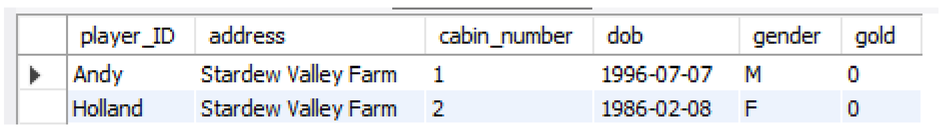
Table Creation:



```sql
create table Farmer
(player_ID               varchar(16) not null,
 address                 varchar(50),
 cabin_number            tinyint UNIQUE,
 dob                     date,
 gender                  char(1),
 gold                    int,
 primary key(player_ID)
);
```



```sql
SELECT * FROM Farmer;
```



player_ID	address	cabin_number	dob	gender	gold
Andy	Stardew Valley Farm	1	1996-07-07	M	0
Holland	Stardew Valley Farm	2	1986-02-08	F	0

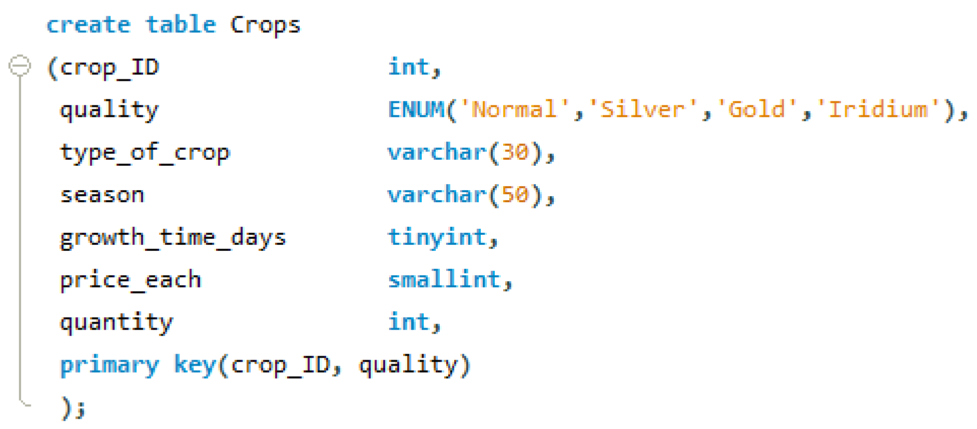
Table 2: Crops

Purpose: List all crops that exist inside the players' playthrough

Attributes: Crop_ID, Quality, type_of_crop, season, growth_time_days, price_each, quantity

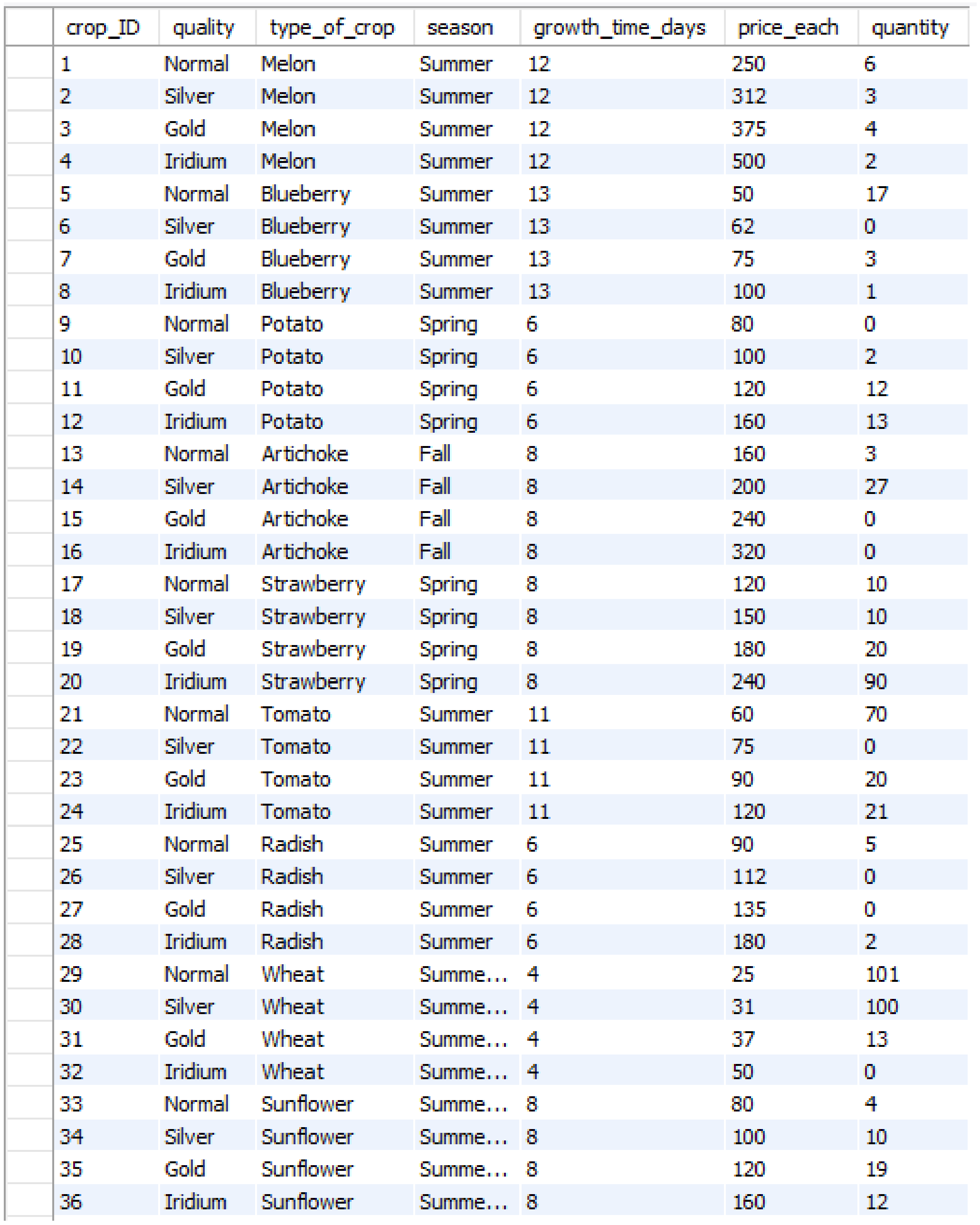
Keys: **Crop_ID:** Uniquely identifies each instance of a certain crop, **Quality:** Differentiates each crop with the same type_of_crop attribute.

Table Creation:

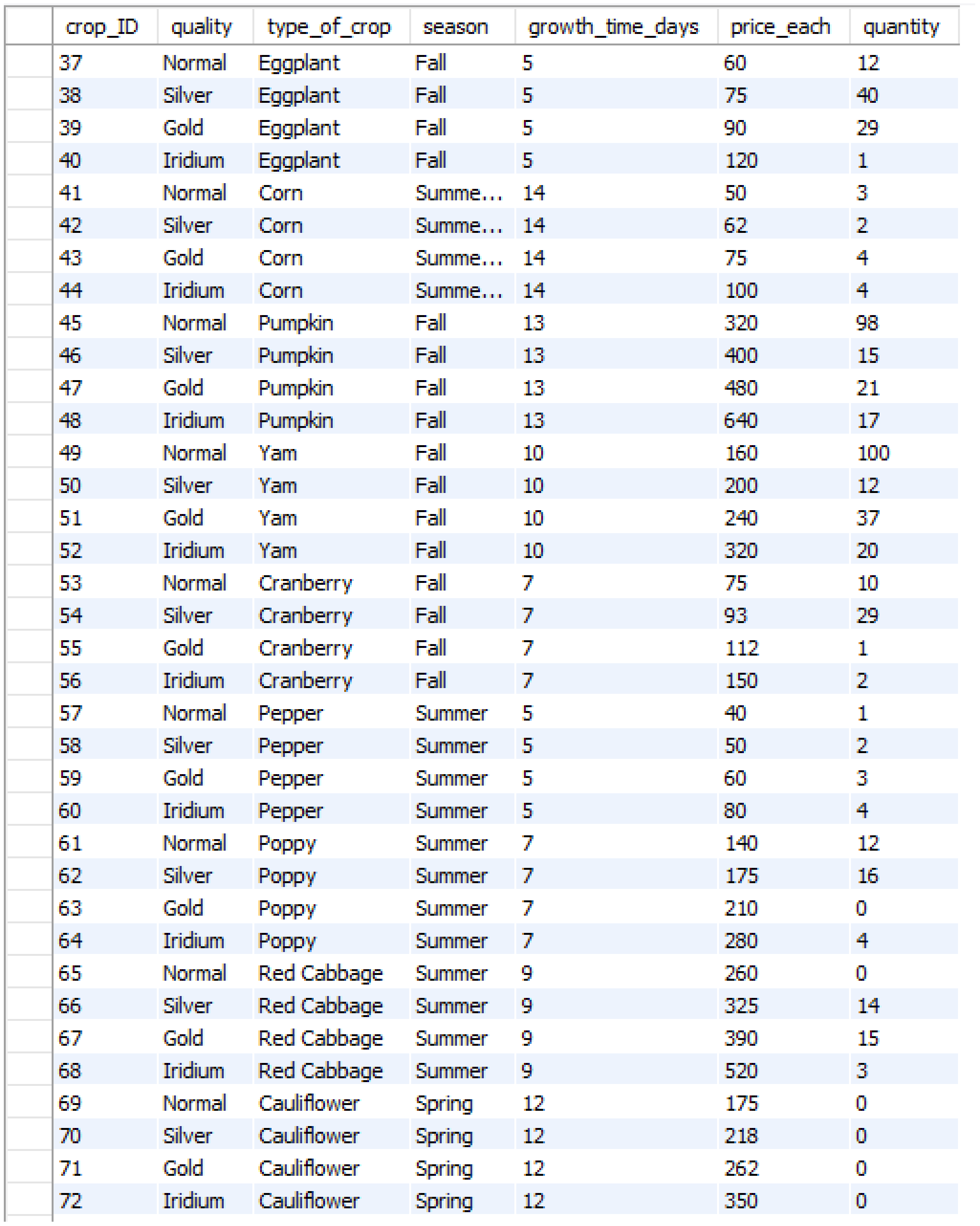


```
create table Crops
(crop_ID                int,
 quality                ENUM('Normal','Silver','Gold','Iridium'),
 type_of_crop           varchar(30),
 season                 varchar(50),
 growth_time_days       tinyint,
 price_each             smallint,
 quantity               int,
 primary key(crop_ID, quality)
 );
```

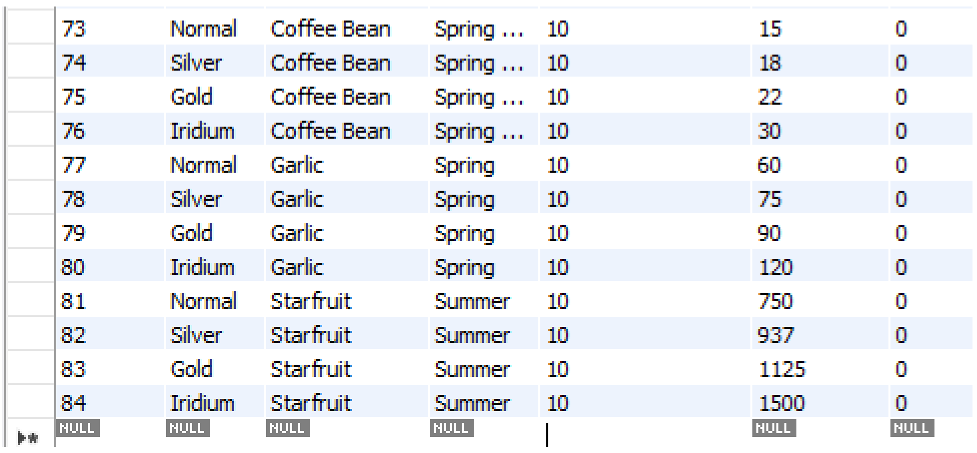
```sql
SELECT * FROM Crops;
```



crop_ID	quality	type_of_crop	season	growth_time_days	price_each	quantity
1	Normal	Melon	Summer	12	250	6
2	Silver	Melon	Summer	12	312	3
3	Gold	Melon	Summer	12	375	4
4	Iridium	Melon	Summer	12	500	2
5	Normal	Blueberry	Summer	13	50	17
6	Silver	Blueberry	Summer	13	62	0
7	Gold	Blueberry	Summer	13	75	3
8	Iridium	Blueberry	Summer	13	100	1
9	Normal	Potato	Spring	6	80	0
10	Silver	Potato	Spring	6	100	2
11	Gold	Potato	Spring	6	120	12
12	Iridium	Potato	Spring	6	160	13
13	Normal	Artichoke	Fall	8	160	3
14	Silver	Artichoke	Fall	8	200	27
15	Gold	Artichoke	Fall	8	240	0
16	Iridium	Artichoke	Fall	8	320	0
17	Normal	Strawberry	Spring	8	120	10
18	Silver	Strawberry	Spring	8	150	10
19	Gold	Strawberry	Spring	8	180	20
20	Iridium	Strawberry	Spring	8	240	90
21	Normal	Tomato	Summer	11	60	70
22	Silver	Tomato	Summer	11	75	0
23	Gold	Tomato	Summer	11	90	20
24	Iridium	Tomato	Summer	11	120	21
25	Normal	Radish	Summer	6	90	5
26	Silver	Radish	Summer	6	112	0
27	Gold	Radish	Summer	6	135	0
28	Iridium	Radish	Summer	6	180	2
29	Normal	Wheat	Summe…	4	25	101
30	Silver	Wheat	Summe…	4	31	100
31	Gold	Wheat	Summe…	4	37	13
32	Iridium	Wheat	Summe…	4	50	0
33	Normal	Sunflower	Summe…	8	80	4
34	Silver	Sunflower	Summe…	8	100	10
35	Gold	Sunflower	Summe…	8	120	19
36	Iridium	Sunflower	Summe…	8	160	12



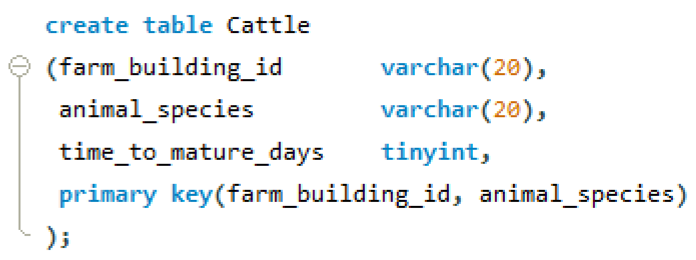
crop_ID	quality	type_of_crop	season	growth_time_days	price_each	quantity
37	Normal	Eggplant	Fall	5	60	12
38	Silver	Eggplant	Fall	5	75	40
39	Gold	Eggplant	Fall	5	90	29
40	Iridium	Eggplant	Fall	5	120	1
41	Normal	Corn	Summe...	14	50	3
42	Silver	Corn	Summe...	14	62	2
43	Gold	Corn	Summe...	14	75	4
44	Iridium	Corn	Summe...	14	100	4
45	Normal	Pumpkin	Fall	13	320	98
46	Silver	Pumpkin	Fall	13	400	15
47	Gold	Pumpkin	Fall	13	480	21
48	Iridium	Pumpkin	Fall	13	640	17
49	Normal	Yam	Fall	10	160	100
50	Silver	Yam	Fall	10	200	12
51	Gold	Yam	Fall	10	240	37
52	Iridium	Yam	Fall	10	320	20
53	Normal	Cranberry	Fall	7	75	10
54	Silver	Cranberry	Fall	7	93	29
55	Gold	Cranberry	Fall	7	112	1
56	Iridium	Cranberry	Fall	7	150	2
57	Normal	Pepper	Summer	5	40	1
58	Silver	Pepper	Summer	5	50	2
59	Gold	Pepper	Summer	5	60	3
60	Iridium	Pepper	Summer	5	80	4
61	Normal	Poppy	Summer	7	140	12
62	Silver	Poppy	Summer	7	175	16
63	Gold	Poppy	Summer	7	210	0
64	Iridium	Poppy	Summer	7	280	4
65	Normal	Red Cabbage	Summer	9	260	0
66	Silver	Red Cabbage	Summer	9	325	14
67	Gold	Red Cabbage	Summer	9	390	15
68	Iridium	Red Cabbage	Summer	9	520	3
69	Normal	Cauliflower	Spring	12	175	0
70	Silver	Cauliflower	Spring	12	218	0
71	Gold	Cauliflower	Spring	12	262	0
72	Iridium	Cauliflower	Spring	12	350	0



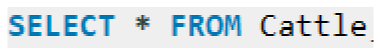
73	Normal	Coffee Bean	Spring ...	10	15	0
74	Silver	Coffee Bean	Spring ...	10	18	0
75	Gold	Coffee Bean	Spring ...	10	22	0
76	Iridium	Coffee Bean	Spring ...	10	30	0
77	Normal	Garlic	Spring	10	60	0
78	Silver	Garlic	Spring	10	75	0
79	Gold	Garlic	Spring	10	90	0
80	Iridium	Garlic	Spring	10	120	0
81	Normal	Starfruit	Summer	10	750	0
82	Silver	Starfruit	Summer	10	937	0
83	Gold	Starfruit	Summer	10	1125	0
84	Iridium	Starfruit	Summer	10	1500	0
NULL	NULL	NULL	NULL		NULL	NULL

Table 3: Cattle

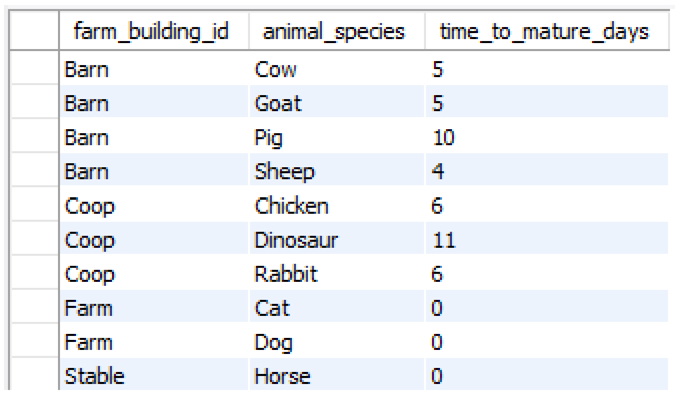
Purpose: List all cattle in the playthrough with their expected time to maturity
Attributes: farm_building_id, animal_species, time_to_mature_days
Keys: **farm_building_id:** Uniquely identify where the animal belongs on the farm.
animal_species: What type of animal is present in the playthrough
Table Creation:



```
create table Cattle
(farm_building_id       varchar(20),
 animal_species         varchar(20),
 time_to_mature_days    tinyint,
 primary key(farm_building_id, animal_species)
);
```



```
SELECT * FROM Cattle
```



farm_building_id	animal_species	time_to_mature_days
Barn	Cow	5
Barn	Goat	5
Barn	Pig	10
Barn	Sheep	4
Coop	Chicken	6
Coop	Dinosaur	11
Coop	Rabbit	6
Farm	Cat	0
Farm	Dog	0
Stable	Horse	0

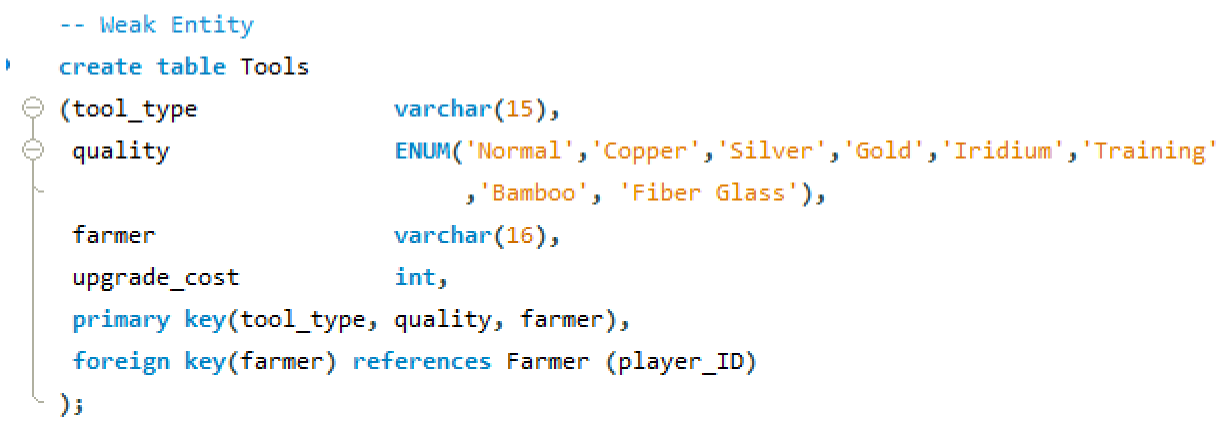
Table 4: Tools

Purpose: List all tools that are owned by each of the farmers. Weak entity with identifying entity being the Farmer table.

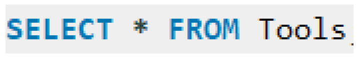
Attributes: tool_type, quality, farmer, upgrade_cost

Keys: **tool_type:** Name of tool, **quality:** one of four identifying qualities, **farmer:** needed to uniquely assign the existence of the tool to a player.

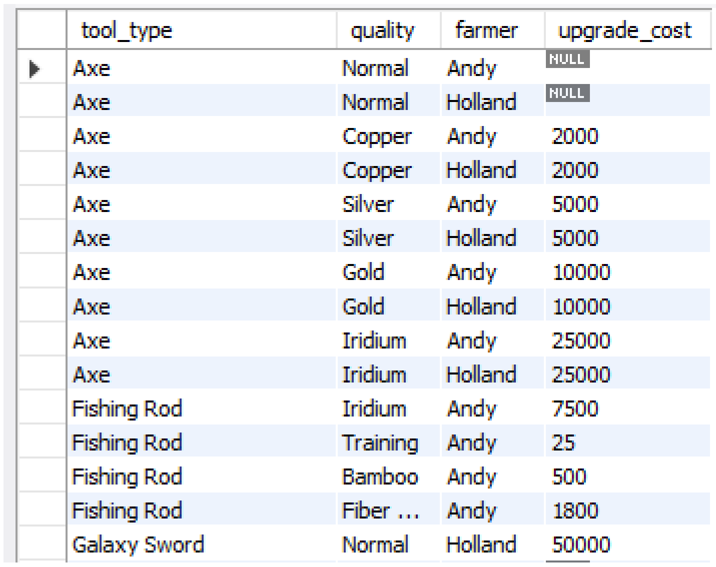
Table Creation:



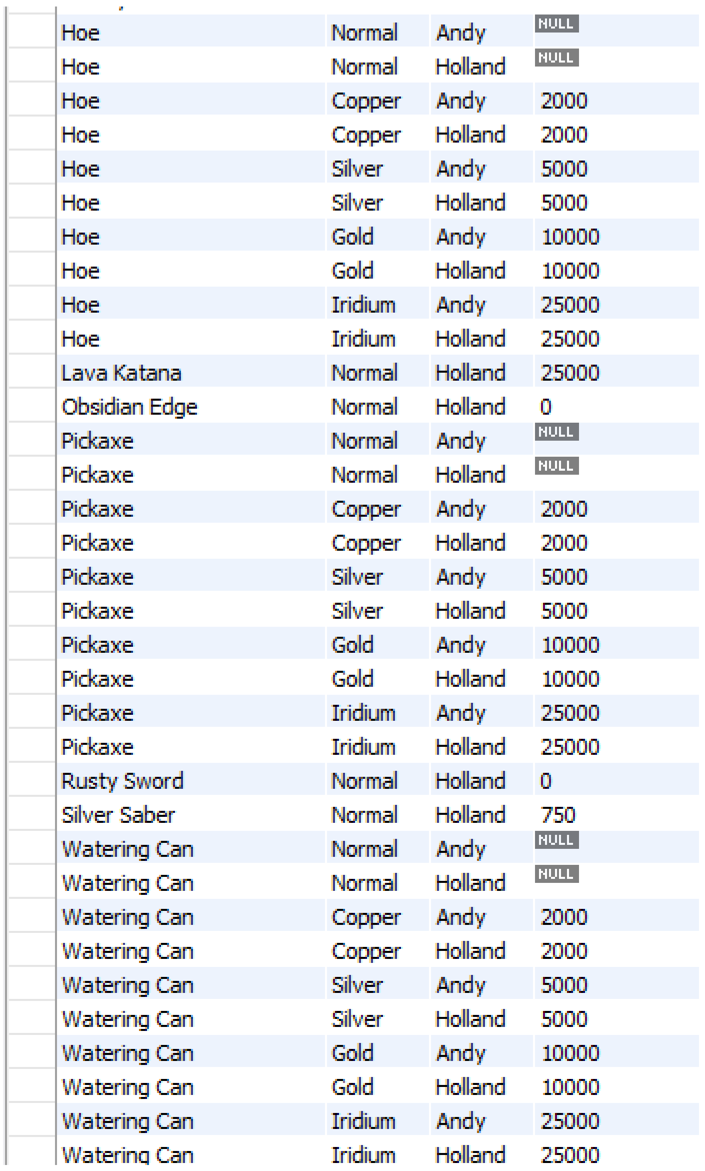
```
-- Weak Entity
create table Tools
(tool_type            varchar(15),
 quality              ENUM('Normal','Copper','Silver','Gold','Iridium','Training'
                          ,'Bamboo', 'Fiber Glass'),
 farmer               varchar(16),
 upgrade_cost         int,
 primary key(tool_type, quality, farmer),
 foreign key(farmer) references Farmer (player_ID)
);
```



```
SELECT * FROM Tools
```



tool_type	quality	farmer	upgrade_cost
Axe	Normal	Andy	NULL
Axe	Normal	Holland	NULL
Axe	Copper	Andy	2000
Axe	Copper	Holland	2000
Axe	Silver	Andy	5000
Axe	Silver	Holland	5000
Axe	Gold	Andy	10000
Axe	Gold	Holland	10000
Axe	Iridium	Andy	25000
Axe	Iridium	Holland	25000
Fishing Rod	Iridium	Andy	7500
Fishing Rod	Training	Andy	25
Fishing Rod	Bamboo	Andy	500
Fishing Rod	Fiber ...	Andy	1800
Galaxy Sword	Normal	Holland	50000



Hoe	Normal	Andy	NULL
Hoe	Normal	Holland	NULL
Hoe	Copper	Andy	2000
Hoe	Copper	Holland	2000
Hoe	Silver	Andy	5000
Hoe	Silver	Holland	5000
Hoe	Gold	Andy	10000
Hoe	Gold	Holland	10000
Hoe	Iridium	Andy	25000
Hoe	Iridium	Holland	25000
Lava Katana	Normal	Holland	25000
Obsidian Edge	Normal	Holland	0
Pickaxe	Normal	Andy	NULL
Pickaxe	Normal	Holland	NULL
Pickaxe	Copper	Andy	2000
Pickaxe	Copper	Holland	2000
Pickaxe	Silver	Andy	5000
Pickaxe	Silver	Holland	5000
Pickaxe	Gold	Andy	10000
Pickaxe	Gold	Holland	10000
Pickaxe	Iridium	Andy	25000
Pickaxe	Iridium	Holland	25000
Rusty Sword	Normal	Holland	0
Silver Saber	Normal	Holland	750
Watering Can	Normal	Andy	NULL
Watering Can	Normal	Holland	NULL
Watering Can	Copper	Andy	2000
Watering Can	Copper	Holland	2000
Watering Can	Silver	Andy	5000
Watering Can	Silver	Holland	5000
Watering Can	Gold	Andy	10000
Watering Can	Gold	Holland	10000
Watering Can	Iridium	Andy	25000
Watering Can	Iridium	Holland	25000

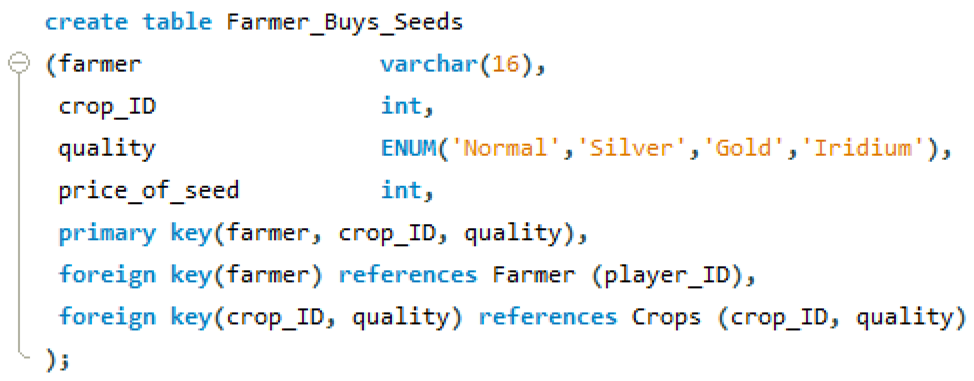
Table 5: Farmer_Buys_Seeds

Purpose: List which specific farmer is purchasing seeds of a unique type of crop from a store in game.

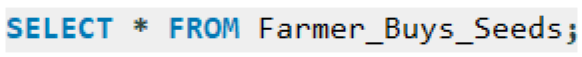
Attributes: farmer, crop_ID, quality, price_of_seed

Keys: **farmer:** unique instance of a player**, crop_ID:** unique instance of certain crop**, quality:** quality identifying the crop(seed) we are looking at

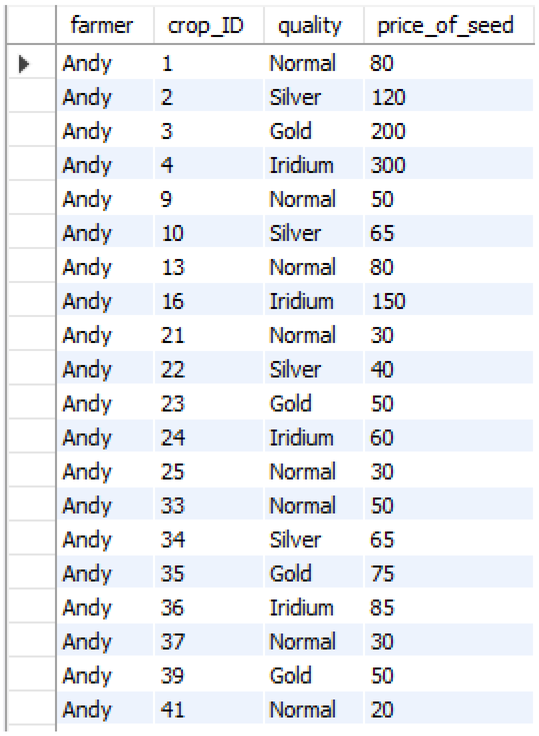
Table Creation:



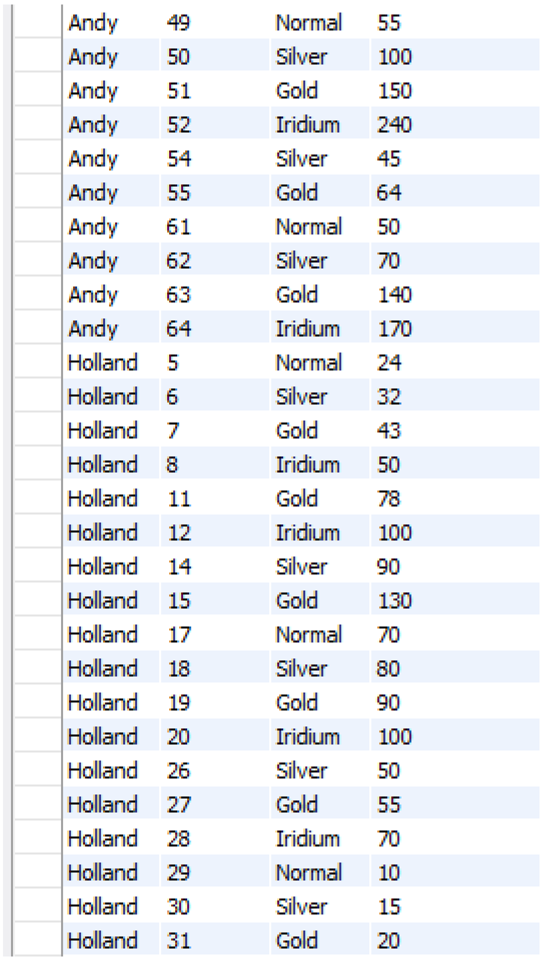
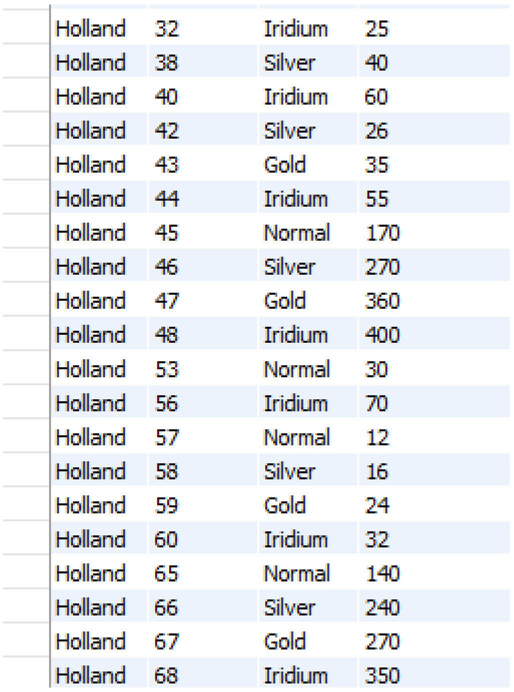
```
create table Farmer_Buys_Seeds
(farmer                 varchar(16),
 crop_ID                int,
 quality                ENUM('Normal','Silver','Gold','Iridium'),
 price_of_seed          int,
 primary key(farmer, crop_ID, quality),
 foreign key(farmer) references Farmer (player_ID),
 foreign key(crop_ID, quality) references Crops (crop_ID, quality)
);
```



```
SELECT * FROM Farmer_Buys_Seeds;
```



farmer	crop_ID	quality	price_of_seed
Andy	1	Normal	80
Andy	2	Silver	120
Andy	3	Gold	200
Andy	4	Iridium	300
Andy	9	Normal	50
Andy	10	Silver	65
Andy	13	Normal	80
Andy	16	Iridium	150
Andy	21	Normal	30
Andy	22	Silver	40
Andy	23	Gold	50
Andy	24	Iridium	60
Andy	25	Normal	30
Andy	33	Normal	50
Andy	34	Silver	65
Andy	35	Gold	75
Andy	36	Iridium	85
Andy	37	Normal	30
Andy	39	Gold	50
Andy	41	Normal	20

Andy	49	Normal	55
Andy	50	Silver	100
Andy	51	Gold	150
Andy	52	Iridium	240
Andy	54	Silver	45
Andy	55	Gold	64
Andy	61	Normal	50
Andy	62	Silver	70
Andy	63	Gold	140
Andy	64	Iridium	170
Holland	5	Normal	24
Holland	6	Silver	32
Holland	7	Gold	43
Holland	8	Iridium	50
Holland	11	Gold	78
Holland	12	Iridium	100
Holland	14	Silver	90
Holland	15	Gold	130
Holland	17	Normal	70
Holland	18	Silver	80
Holland	19	Gold	90
Holland	20	Iridium	100
Holland	26	Silver	50
Holland	27	Gold	55
Holland	28	Iridium	70
Holland	29	Normal	10
Holland	30	Silver	15
Holland	31	Gold	20
Holland	32	Iridium	25
Holland	38	Silver	40
Holland	40	Iridium	60
Holland	42	Silver	26
Holland	43	Gold	35
Holland	44	Iridium	55
Holland	45	Normal	170
Holland	46	Silver	270
Holland	47	Gold	360
Holland	48	Iridium	400
Holland	53	Normal	30
Holland	56	Iridium	70
Holland	57	Normal	12
Holland	58	Silver	16
Holland	59	Gold	24
Holland	60	Iridium	32
Holland	65	Normal	140
Holland	66	Silver	240
Holland	67	Gold	270
Holland	68	Iridium	350

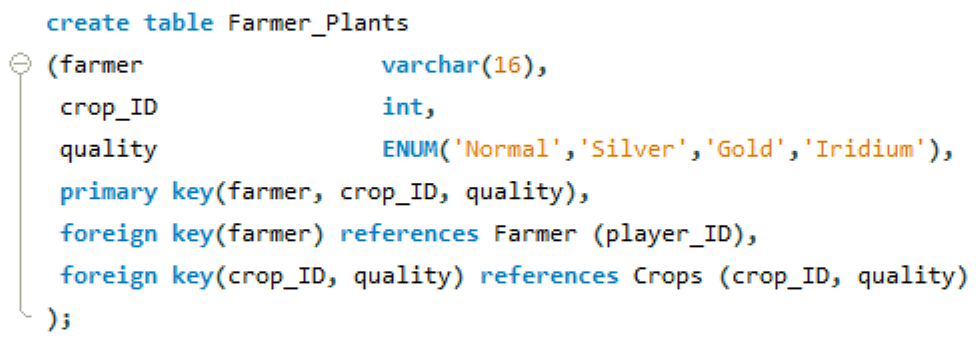
Table 6: Farmer_Plants

Purpose: List which farmer is responsible for planting which seeds

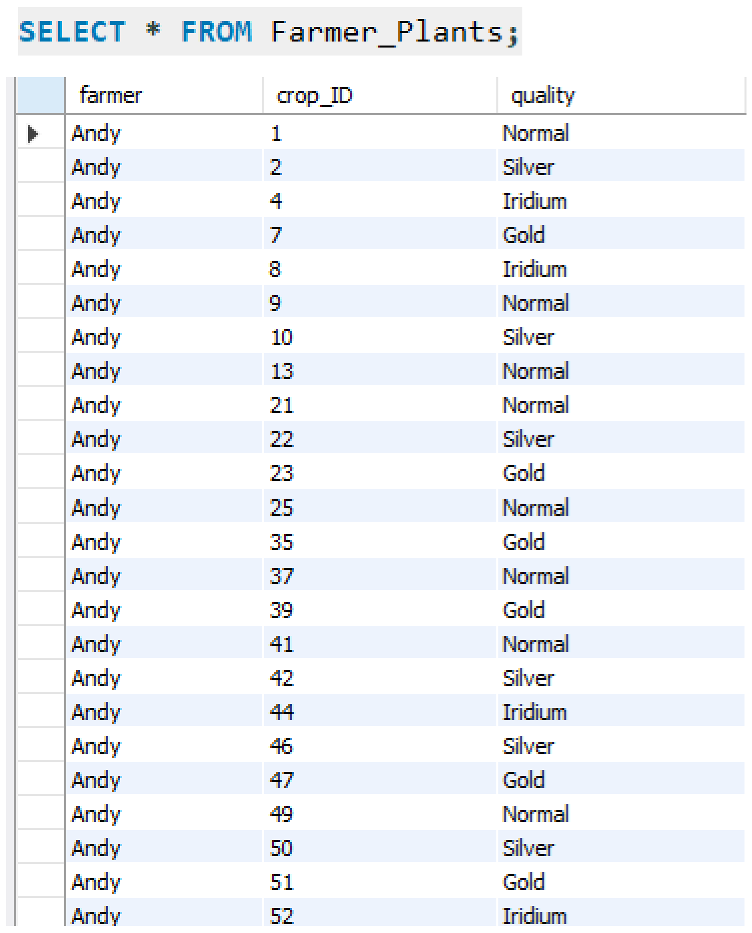
Attributes: farmer, crop_ID, quality

Keys: **farmer:** unique instance of a player**, crop_ID:** unique instance of certain crop, **quality:** quality identifying the crop we are looking at

Table Creation:

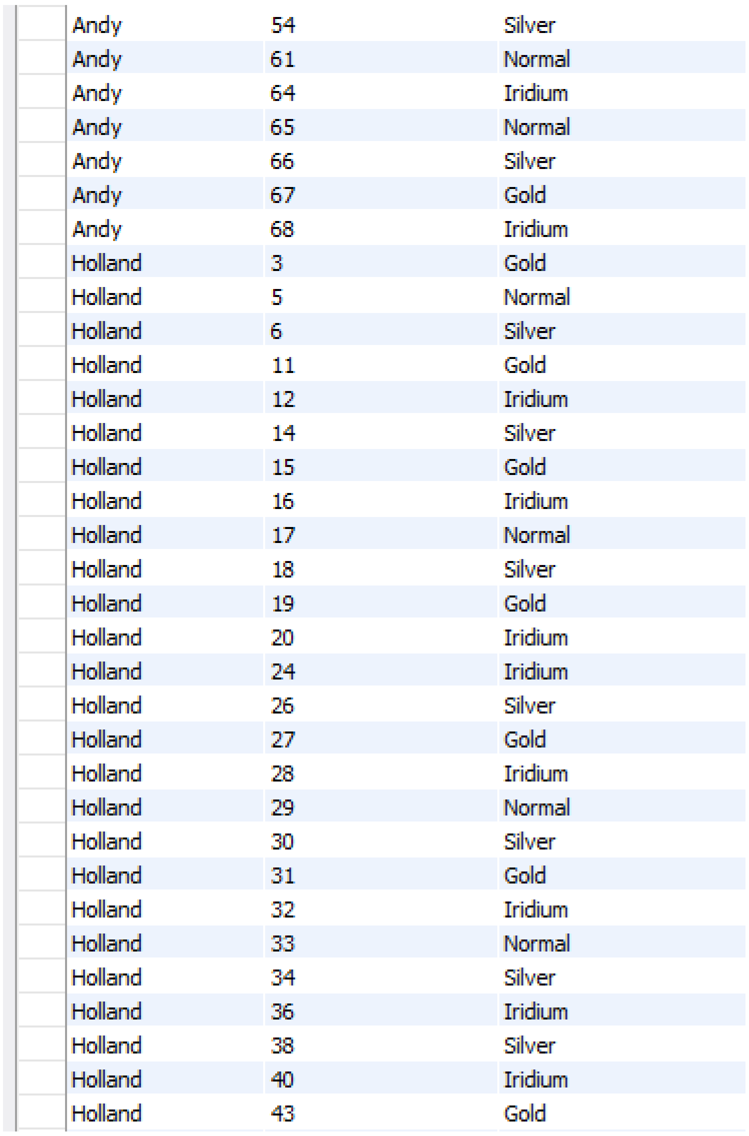
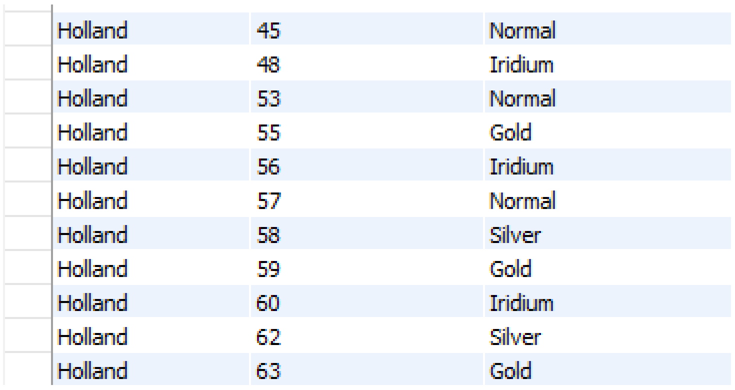


```
create table Farmer_Plants
(farmer                 varchar(16),
 crop_ID                int,
 quality                ENUM('Normal','Silver','Gold','Iridium'),
 primary key(farmer, crop_ID, quality),
 foreign key(farmer) references Farmer (player_ID),
 foreign key(crop_ID, quality) references Crops (crop_ID, quality)
);
```



```
SELECT * FROM Farmer_Plants;
```

farmer	crop_ID	quality
Andy	1	Normal
Andy	2	Silver
Andy	4	Iridium
Andy	7	Gold
Andy	8	Iridium
Andy	9	Normal
Andy	10	Silver
Andy	13	Normal
Andy	21	Normal
Andy	22	Silver
Andy	23	Gold
Andy	25	Normal
Andy	35	Gold
Andy	37	Normal
Andy	39	Gold
Andy	41	Normal
Andy	42	Silver
Andy	44	Iridium
Andy	46	Silver
Andy	47	Gold
Andy	49	Normal
Andy	50	Silver
Andy	51	Gold
Andy	52	Iridium

Andy	54	Silver
Andy	61	Normal
Andy	64	Iridium
Andy	65	Normal
Andy	66	Silver
Andy	67	Gold
Andy	68	Iridium
Holland	3	Gold
Holland	5	Normal
Holland	6	Silver
Holland	11	Gold
Holland	12	Iridium
Holland	14	Silver
Holland	15	Gold
Holland	16	Iridium
Holland	17	Normal
Holland	18	Silver
Holland	19	Gold
Holland	20	Iridium
Holland	24	Iridium
Holland	26	Silver
Holland	27	Gold
Holland	28	Iridium
Holland	29	Normal
Holland	30	Silver
Holland	31	Gold
Holland	32	Iridium
Holland	33	Normal
Holland	34	Silver
Holland	36	Iridium
Holland	38	Silver
Holland	40	Iridium
Holland	43	Gold
Holland	45	Normal
Holland	48	Iridium
Holland	53	Normal
Holland	55	Gold
Holland	56	Iridium
Holland	57	Normal
Holland	58	Silver
Holland	59	Gold
Holland	60	Iridium
Holland	62	Silver
Holland	63	Gold

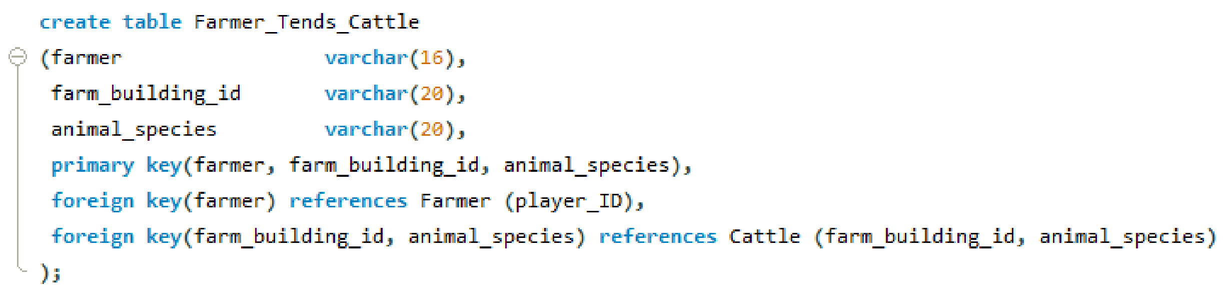
Table 7: Farmer_Tends_Cattle

Purpose: List which farmer is responsible for taking care of certain cattle.

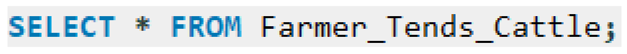
Attributes: farmer, farm_building_id, animal_species

Keys: **farmer:** unique instance of a player **, farm_building_id:** which building the animal belongs to**, animal_species:** what type of animal

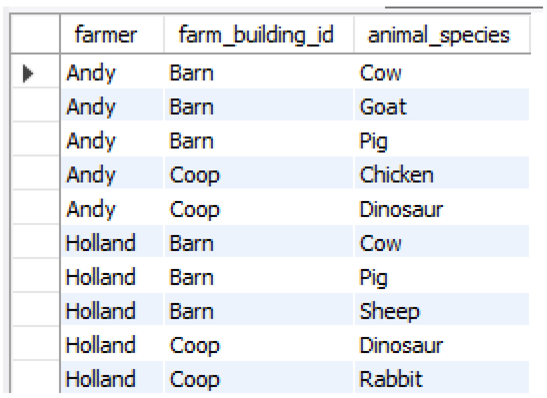
Table Creation:



```
create table Farmer_Tends_Cattle
(farmer                 varchar(16),
 farm_building_id       varchar(20),
 animal_species         varchar(20),
 primary key(farmer, farm_building_id, animal_species),
 foreign key(farmer) references Farmer (player_ID),
 foreign key(farm_building_id, animal_species) references Cattle (farm_building_id, animal_species)
);
```



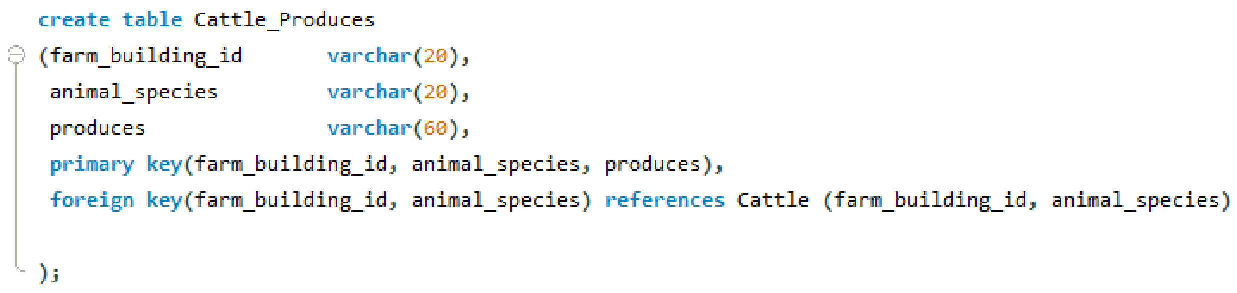
```
SELECT * FROM Farmer_Tends_Cattle;
```



farmer	farm_building_id	animal_species
Andy	Barn	Cow
Andy	Barn	Goat
Andy	Barn	Pig
Andy	Coop	Chicken
Andy	Coop	Dinosaur
Holland	Barn	Cow
Holland	Barn	Pig
Holland	Barn	Sheep
Holland	Coop	Dinosaur
Holland	Coop	Rabbit

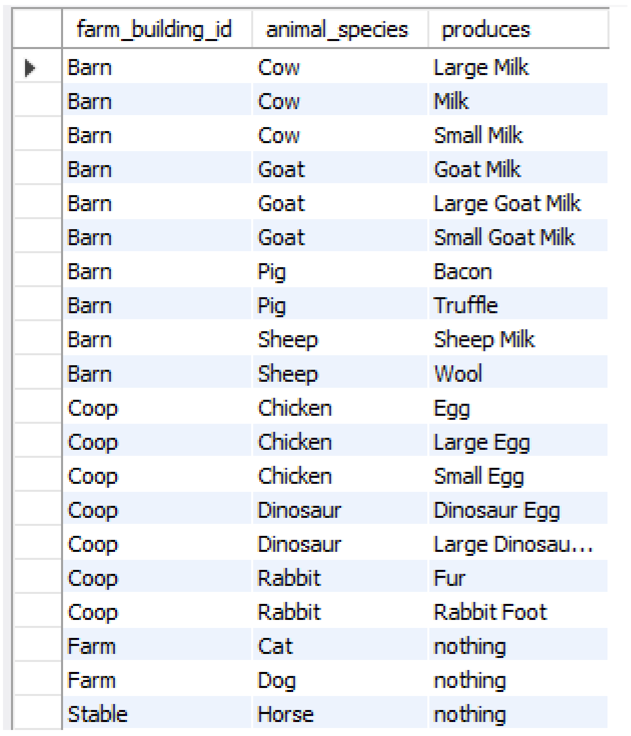
Table 8: Cattle_Produces

Purpose: List all of the animals and what they will be producing for the farmers.
Attributes: farm_building_id, animal_species, produces
Keys: **farm_building_id:** which building the animal belongs to**, animal_species:** what type of animal **, produces:** multi-attribute for what each animal is producing
Table Creation:



```sql
create table Cattle_Produces
(farm_building_id       varchar(20),
 animal_species         varchar(20),
 produces               varchar(60),
 primary key(farm_building_id, animal_species, produces),
 foreign key(farm_building_id, animal_species) references Cattle (farm_building_id, animal_species)

);
```

```sql
SELECT * FROM Cattle_Produces;
```

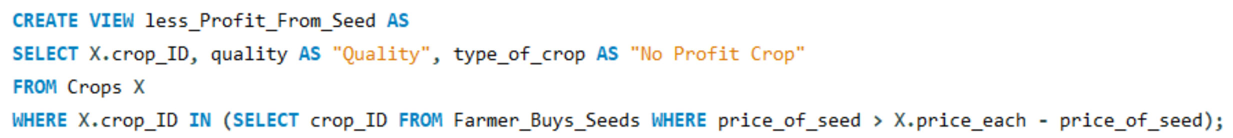


farm_building_id	animal_species	produces
Barn	Cow	Large Milk
Barn	Cow	Milk
Barn	Cow	Small Milk
Barn	Goat	Goat Milk
Barn	Goat	Large Goat Milk
Barn	Goat	Small Goat Milk
Barn	Pig	Bacon
Barn	Pig	Truffle
Barn	Sheep	Sheep Milk
Barn	Sheep	Wool
Coop	Chicken	Egg
Coop	Chicken	Large Egg
Coop	Chicken	Small Egg
Coop	Dinosaur	Dinosaur Egg
Coop	Dinosaur	Large Dinosau...
Coop	Rabbit	Fur
Coop	Rabbit	Rabbit Foot
Farm	Cat	nothing
Farm	Dog	nothing
Stable	Horse	nothing

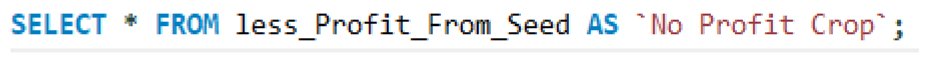
Views, Functions, Procedures, Queries

View: Shows a table of which seeds will produce a negative profit in correlation of which crop and quality they produce.

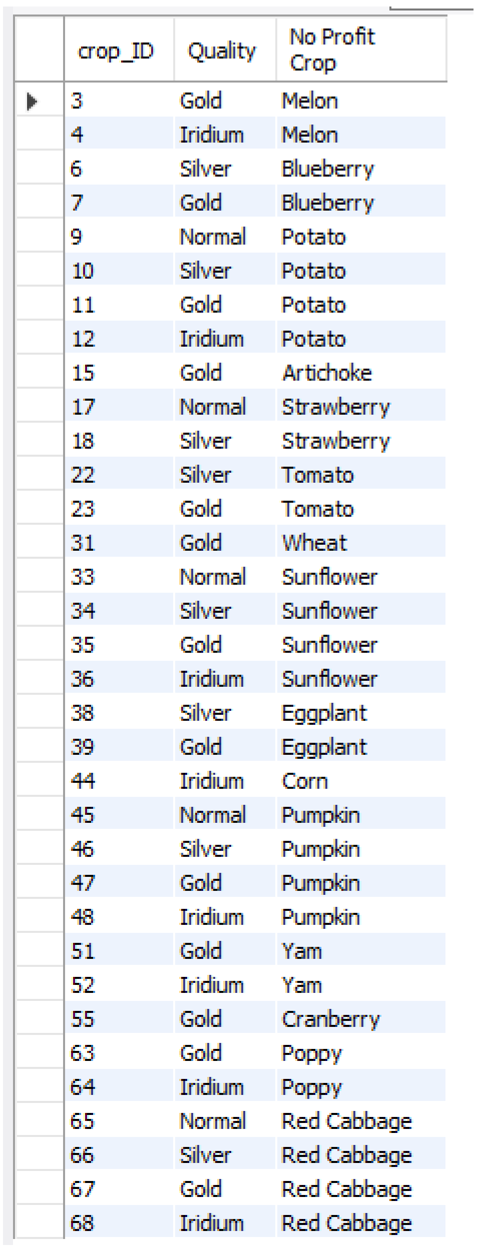
SQL Code:



```sql
CREATE VIEW less_Profit_From_Seed AS
SELECT X.crop_ID, quality AS "Quality", type_of_crop AS "No Profit Crop"
FROM Crops X
WHERE X.crop_ID IN (SELECT crop_ID FROM Farmer_Buys_Seeds WHERE price_of_seed > X.price_each - price_of_seed);
```



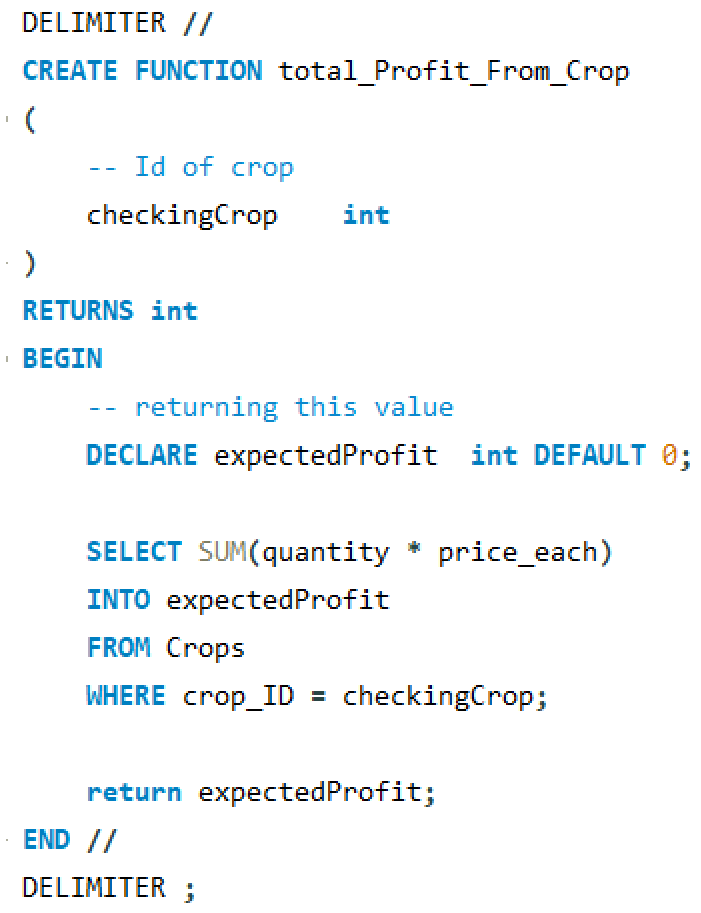
```sql
SELECT * FROM less_Profit_From_Seed AS `No Profit Crop`;
```



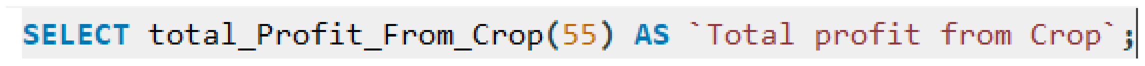
crop_ID	Quality	No Profit Crop
3	Gold	Melon
4	Iridium	Melon
6	Silver	Blueberry
7	Gold	Blueberry
9	Normal	Potato
10	Silver	Potato
11	Gold	Potato
12	Iridium	Potato
15	Gold	Artichoke
17	Normal	Strawberry
18	Silver	Strawberry
22	Silver	Tomato
23	Gold	Tomato
31	Gold	Wheat
33	Normal	Sunflower
34	Silver	Sunflower
35	Gold	Sunflower
36	Iridium	Sunflower
38	Silver	Eggplant
39	Gold	Eggplant
44	Iridium	Corn
45	Normal	Pumpkin
46	Silver	Pumpkin
47	Gold	Pumpkin
48	Iridium	Pumpkin
51	Gold	Yam
52	Iridium	Yam
55	Gold	Cranberry
63	Gold	Poppy
64	Iridium	Poppy
65	Normal	Red Cabbage
66	Silver	Red Cabbage
67	Gold	Red Cabbage
68	Iridium	Red Cabbage

Function: Finds the total amount of profit if all quantity of a certain crop is sold

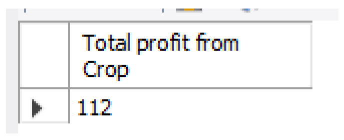
SQL Code:



```
DELIMITER //
CREATE FUNCTION total_Profit_From_Crop
(
    -- Id of crop
    checkingCrop    int
)
RETURNS int
BEGIN
    -- returning this value
    DECLARE expectedProfit  int DEFAULT 0;

    SELECT SUM(quantity * price_each)
    INTO expectedProfit
    FROM Crops
    WHERE crop_ID = checkingCrop;

    return expectedProfit;
END //
DELIMITER ;
```



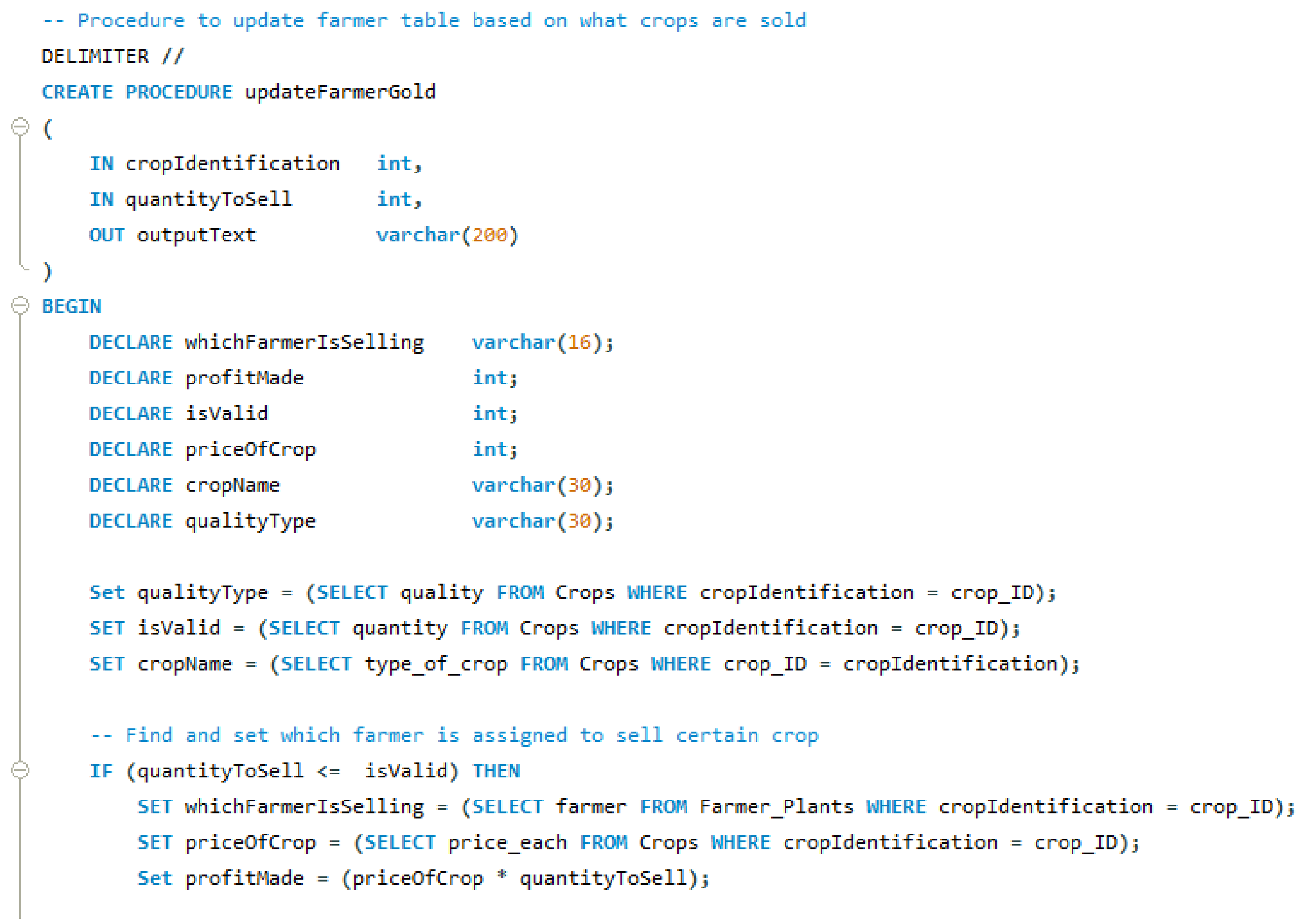
```
SELECT total_Profit_From_Crop(55) AS `Total profit from Crop`;
```



```
Total profit from Crop
112
```

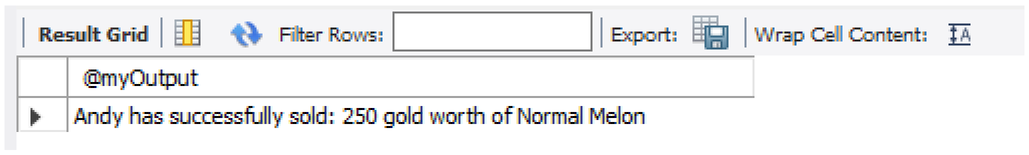
Procedure: Takes as input crop_ID and quantity to sell and updates the Farmer table and Crops table accordingly.

SQL Code:

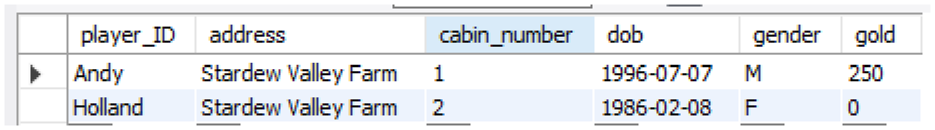


```sql
-- Procedure to update farmer table based on what crops are sold
DELIMITER //
CREATE PROCEDURE updateFarmerGold
(
    IN cropIdentification   int,
    IN quantityToSell       int,
    OUT outputText          varchar(200)
)
BEGIN
    DECLARE whichFarmerIsSelling    varchar(16);
    DECLARE profitMade              int;
    DECLARE isValid                 int;
    DECLARE priceOfCrop             int;
    DECLARE cropName                varchar(30);
    DECLARE qualityType             varchar(30);

    Set qualityType = (SELECT quality FROM Crops WHERE cropIdentification = crop_ID);
    SET isValid = (SELECT quantity FROM Crops WHERE cropIdentification = crop_ID);
    SET cropName = (SELECT type_of_crop FROM Crops WHERE crop_ID = cropIdentification);

    -- Find and set which farmer is assigned to sell certain crop
    IF (quantityToSell <=  isValid) THEN
        SET whichFarmerIsSelling = (SELECT farmer FROM Farmer_Plants WHERE cropIdentification = crop_ID);
        SET priceOfCrop = (SELECT price_each FROM Crops WHERE cropIdentification = crop_ID);
        Set profitMade = (priceOfCrop * quantityToSell);
        -- Update the Farmer's Profit
        UPDATE Farmer
        SET gold = profitMade
        WHERE player_ID = whichFarmerIsSelling;

        -- Update the crops quantity column
        UPDATE Crops
        SET quantity = (quantity - quantityToSell)
        WHERE crop_ID = cropIdentification;
        SET outputText = CONCAT(whichFarmerIsSelling , " has successfully sold: ", profitMade,
                                " gold worth of ", qualityType, " " ,cropName);

    ELSE
        SET outputText = CONCAT('You are trying to sell more quantity than you currently own...');
    END IF;

END //
DELIMITER ;
```

```
CALL updateFarmerGold(1, 1, @myOutput);
SELECT @myOutput;
```



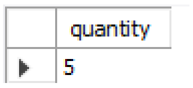
	@myOutput
▶	Andy has successfully sold: 250 gold worth of Normal Melon



	player_ID	address	cabin_number	dob	gender	gold
▶	Andy	Stardew Valley Farm	1	1996-07-07	M	250
	Holland	Stardew Valley Farm	2	1986-02-08	F	0

Note: Andy's Gold is now updated with the crop that was sold.

```
SELECT quantity FROM Crops WHERE crop_ID = 1;
```

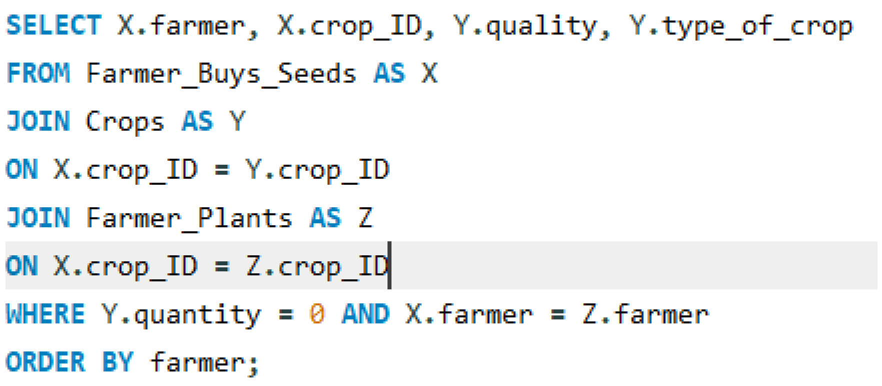


	quantity
▶	5

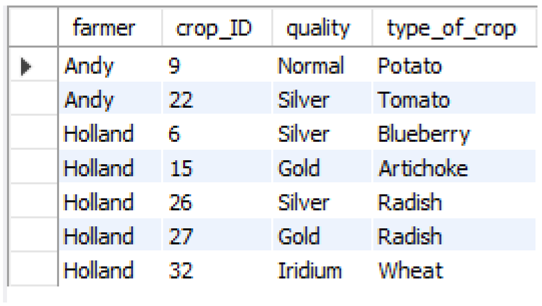
Note: Crop table quantity has been updated accordingly for that crop_ID.

Query 1: Uses the **JOINING** of 3 tables and **WHERE** clause with multiple conditions. Find out which farmer bought seeds and have not had them planted by their respective farmer (quantity 0 in Crops)

SQL Code:



```sql
SELECT X.farmer, X.crop_ID, Y.quality, Y.type_of_crop
FROM Farmer_Buys_Seeds AS X
JOIN Crops AS Y
ON X.crop_ID = Y.crop_ID
JOIN Farmer_Plants AS Z
ON X.crop_ID = Z.crop_ID
WHERE Y.quantity = 0 AND X.farmer = Z.farmer
ORDER BY farmer;
```

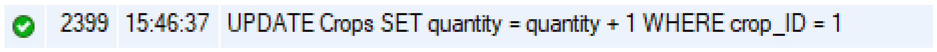


farmer	crop_ID	quality	type_of_crop
Andy	9	Normal	Potato
Andy	22	Silver	Tomato
Holland	6	Silver	Blueberry
Holland	15	Gold	Artichoke
Holland	26	Silver	Radish
Holland	27	Gold	Radish
Holland	32	Iridium	Wheat

Query 2: Uses the **UPDATE** clause. Updates the quantity attribute in the Crops table. This can be seen as an insert of a duplicate crop. In this example, we are adding another Normal quality Melon to our database.

SQL Code:

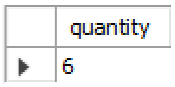
```
UPDATE Crops
SET quantity = quantity + 1
WHERE crop_ID = 1;
```



```
2399  15:46:37  UPDATE Crops SET quantity = quantity + 1 WHERE crop_ID = 1
```

Note: Checking to see if one melon has been added to the database. Observe that from the previous procedure call, we have lost one melon because we sold it and added the profit to Andy’s ‘gold’ attribute.

```
SELECT quantity FROM Crops WHERE crop_ID = 1;
```

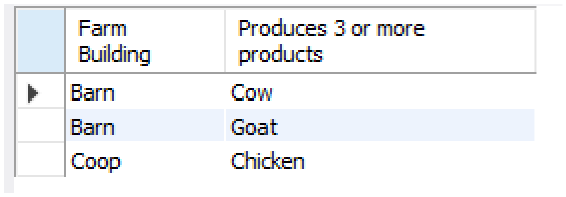


	quantity
▶	6

Query 3: Uses **GROUP BY** and **HAVING**. Shows a table of animal species that produce 3 or more products

SQL Code:

```
SELECT farm_building_id AS "Farm Building",  animal_species AS "Produces 3 or more products"
FROM Cattle_Produces
GROUP BY farm_building_id, animal_species
HAVING COUNT(produces) >= 3;
```

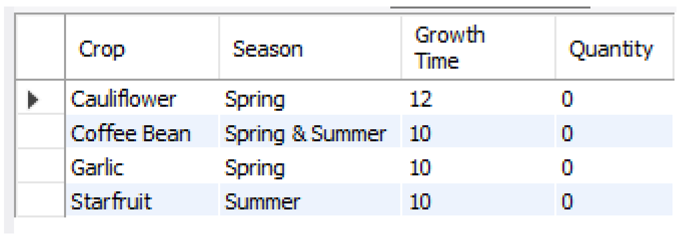


Farm Building	Produces 3 or more products
Barn	Cow
Barn	Goat
Coop	Chicken

Query 4: Uses **DISTINCT** and **WHERE NOT EXISTS**. Shows a table of crops that are stored in the database that have not yet had seeds bought for them yet

SQL Code:

```
SELECT DISTINCT type_of_crop AS "Crop",
season AS "Season", growth_time_days AS "Growth Time", quantity AS "Quantity"
FROM Crops AS X
WHERE NOT EXISTS
(SELECT crop_ID FROM Farmer_Buys_Seeds WHERE X.crop_ID = Farmer_Buys_Seeds.crop_ID);
```

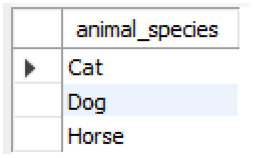


Crop	Season	Growth Time	Quantity
Cauliflower	Spring	12	0
Coffee Bean	Spring & Summer	10	0
Garlic	Spring	10	0
Starfruit	Summer	10	0

Query 5: Uses **WHERE IN**. Shows a table of which animals are somewhere on the farm that do not produce something of value

SQL Code:

```
SELECT animal_species FROM Cattle
WHERE animal_species IN
(SELECT animal_species FROM Cattle_Produces WHERE produces = 'nothing');
```

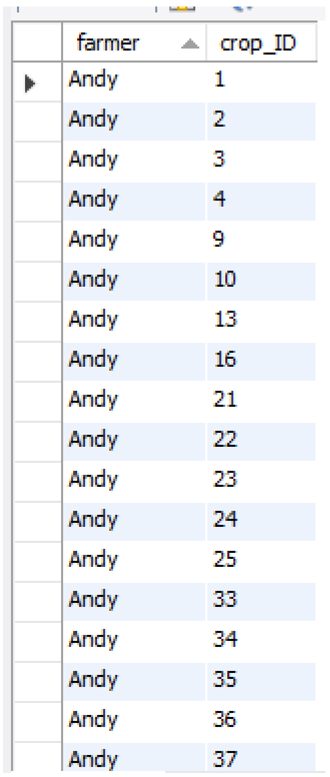
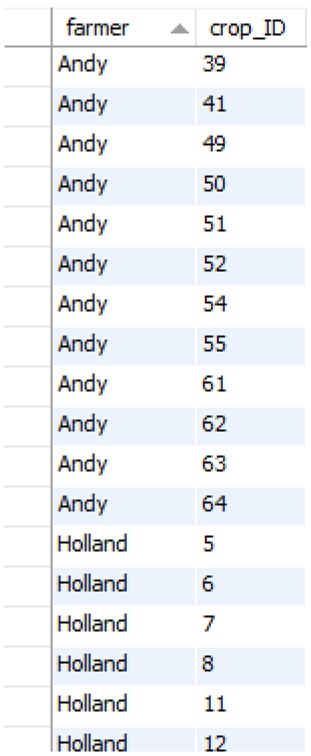
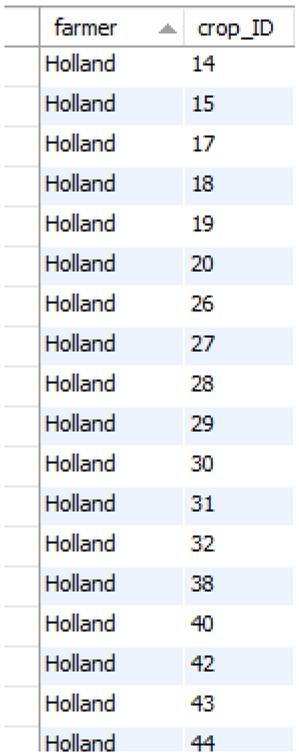
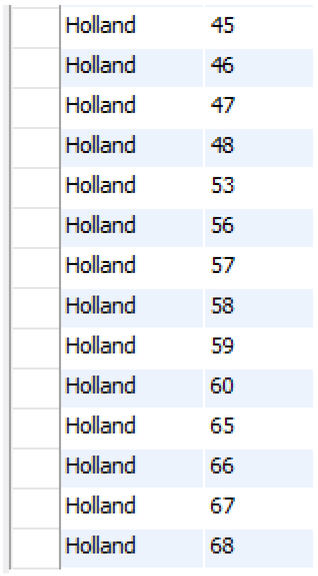


	animal_species
▶	Cat
	Dog
	Horse

Query 6: Uses **OUTER JOIN** and **UNION**. Prints a table of all crop_ID's and associated farmers that have bought seeds and planted seeds.

SQL Code:

```
SELECT X.farmer, X.crop_ID FROM Farmer_Buys_Seeds AS X
LEFT JOIN Farmer_Plants AS Y ON X.crop_ID = Y.crop_ID
UNION
SELECT Z.farmer, Z.crop_ID FROM Farmer_Buys_Seeds AS Z
RIGHT JOIN Farmer_Plants AS W ON Z.crop_ID = W.crop_ID;
```

farmer	crop_ID
Andy	1
Andy	2
Andy	3
Andy	4
Andy	9
Andy	10
Andy	13
Andy	16
Andy	21
Andy	22
Andy	23
Andy	24
Andy	25
Andy	33
Andy	34
Andy	35
Andy	36
Andy	37
Andy	39
Andy	41
Andy	49
Andy	50
Andy	51
Andy	52
Andy	54
Andy	55
Andy	61
Andy	62
Andy	63
Andy	64
Holland	5
Holland	6
Holland	7
Holland	8
Holland	11
Holland	12
Holland	14
Holland	15
Holland	17
Holland	18
Holland	19
Holland	20
Holland	26
Holland	27
Holland	28
Holland	29
Holland	30
Holland	31
Holland	32
Holland	38
Holland	40
Holland	42
Holland	43
Holland	44
Holland	45
Holland	46
Holland	47
Holland	48
Holland	53
Holland	56
Holland	57
Holland	58
Holland	59
Holland	60
Holland	65
Holland	66
Holland	67
Holland	68