Disgaea 2 Database

Michael Hercules Sirico

Table of contents

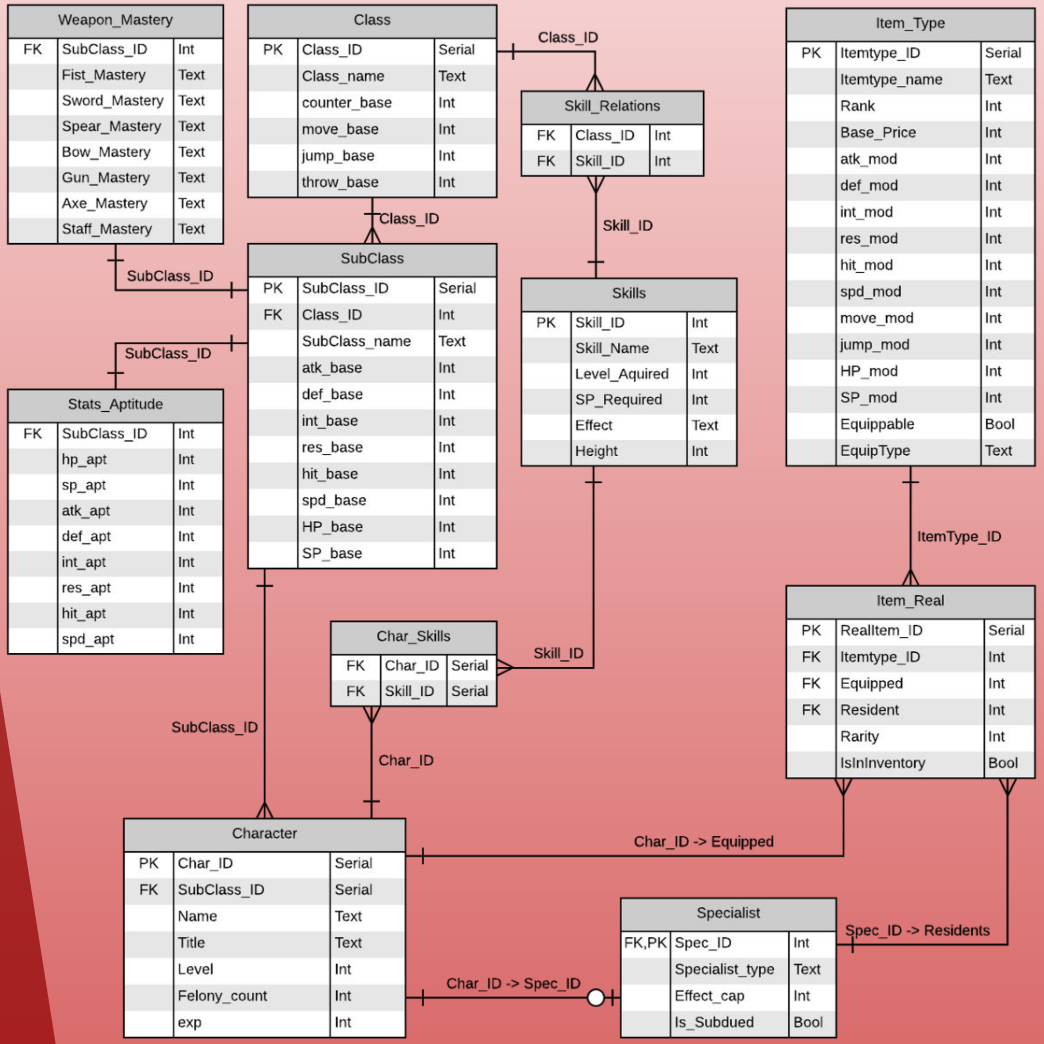
Executive Summary………………………………………………………………. 3
Entity Relationship Diagram…………………………………………………. 4
Tables………………………………………………………………………………. 5-26
View Definitions……………………………………………………………….. 27-32
Reports…………………………………………………………………………….. 33-34
Stored Procedures……………………………………………………………. 35-37
Triggers…………………………………………………………………………… 38
Security…………………………………………………………………………… 39-40
Implementation Notes……………………………………………............ 41
Known Problems……………………………………………………………. 41
Future Enhancements……………………………………………………. 41

Executive Summary

Disgaea 2 is a strategy role playing game that makes use of a lot of different game mechanics and a lot of data. The vast amount of information regarding Classes, Subclasses, base stats of these classes, and so on, may be overwhelming for players who are just beginning and starting out. For this reason, There should be an way to manage all the details and Data, so that nothing gets lost.

This database is a response to that, having been built to create and manage the vast majority of information and data that the game presents its players. By doing this, it will be far less likely that players will lose track of any sort of data, and they will have an easier time playing as a result of the data's organization and context.

ER-Diagram



Weapon_Mastery		
FK	SubClass_ID	Int
	Fist_Mastery	Text
	Sword_Mastery	Text
	Spear_Mastery	Text
	Bow_Mastery	Text
	Gun_Mastery	Text
	Axe_Mastery	Text
	Staff_Mastery	Text

Class		
PK	Class_ID	Serial
	Class_name	Text
	counter_base	Int
	move_base	Int
	jump_base	Int
	throw_base	Int

Skill_Relations		
FK	Class_ID	Int
FK	Skill_ID	Int

Item_Type		
PK	Itemtype_ID	Serial
	Itemtype_name	Text
	Rank	Int
	Base_Price	Int
	atk_mod	Int
	def_mod	Int
	int_mod	Int
	res_mod	Int
	hit_mod	Int
	spd_mod	Int
	move_mod	Int
	jump_mod	Int
	HP_mod	Int
	SP_mod	Int
	Equippable	Bool
	EquipType	Text

SubClass		
PK	SubClass_ID	Serial
FK	Class_ID	Int
	SubClass_name	Text
	atk_base	Int
	def_base	Int
	int_base	Int
	res_base	Int
	hit_base	Int
	spd_base	Int
	HP_base	Int
	SP_base	Int

Skills		
PK	Skill_ID	Int
	Skill_Name	Text
	Level_Aquired	Int
	SP_Required	Int
	Effect	Text
	Height	Int

Stats_Aptitude		
FK	SubClass_ID	Int
	hp_apt	Int
	sp_apt	Int
	atk_apt	Int
	def_apt	Int
	int_apt	Int
	res_apt	Int
	hit_apt	Int
	spd_apt	Int

Char_Skills		
FK	Char_ID	Serial
FK	Skill_ID	Serial

Item_Real		
PK	RealItem_ID	Serial
FK	Itemtype_ID	Int
FK	Equipped	Int
FK	Resident	Int
	Rarity	Int
	IsInInventory	Bool

Character		
PK	Char_ID	Serial
FK	SubClass_ID	Serial
	Name	Text
	Title	Text
	Level	Int
	Felony_count	Int
	exp	Int

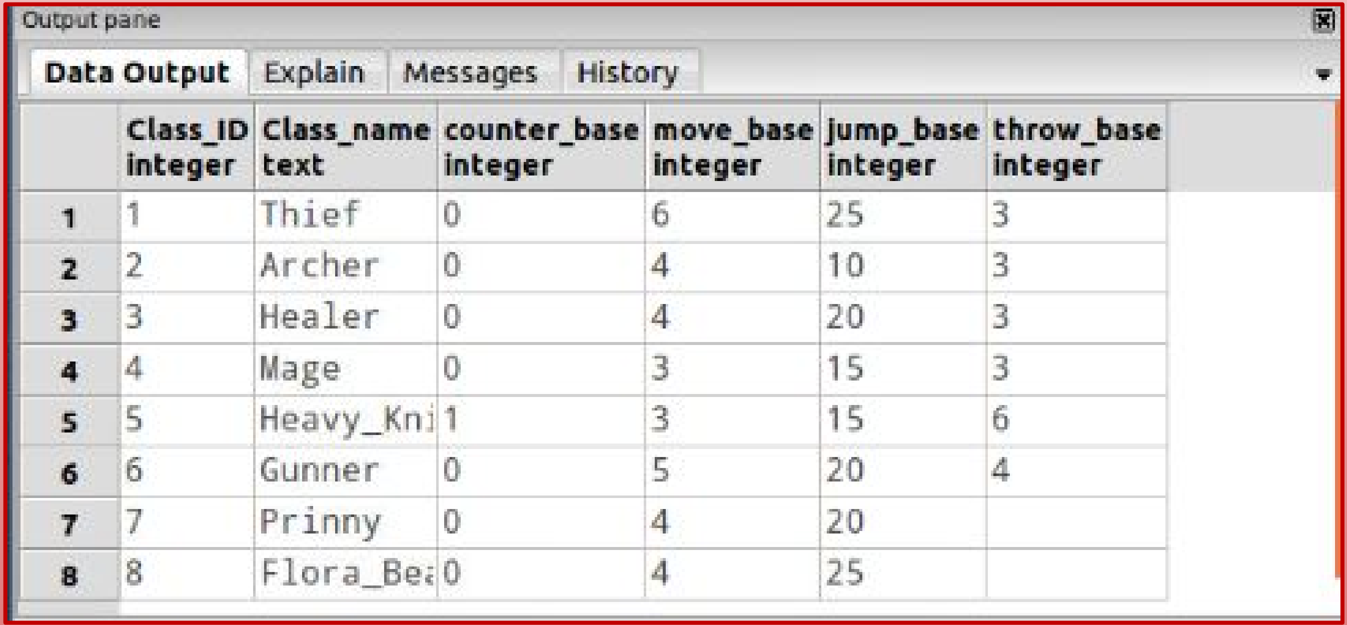
Specialist		
FK,PK	Spec_ID	Int
	Specialist_type	Text
	Effect_cap	Int
	Is_Subdued	Bool

Table: Class

This table holds the main information regarding Classes, including The Class name, and general base info about the Class

```
Create Table If Not Exists "Class" (
  "Class_ID" Serial Primary Key,
  "Class_name" Text Not Null,
  "counter_base" Int Not Null,
  "move_base" Int Not Null,
  "jump_base" Int Not Null,
  "throw_base" Int
);
```

Table: Class



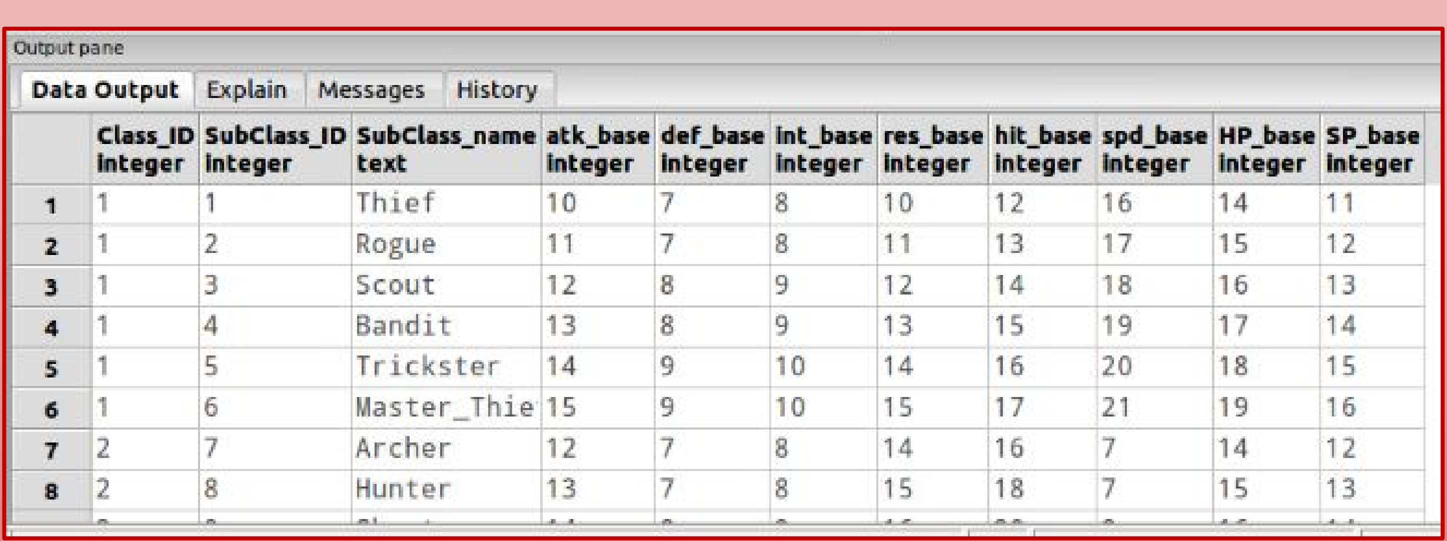
	Class_ID	Class_name	counter_base	move_base	jump_base	throw_base
	integer	text	integer	integer	integer	integer
1	1	Thief	0	6	25	3
2	2	Archer	0	4	10	3
3	3	Healer	0	4	20	3
4	4	Mage	0	3	15	3
5	5	Heavy_Kni	1	3	15	6
6	6	Gunner	0	5	20	4
7	7	Prinny	0	4	20	
8	8	Flora_Be	0	4	25	

Table: SubClass

This table holds the main information regarding Sub-Classes.

```
create table if not exists "SubClass" (
  "Class_ID" Int references "Class"("Class_ID"),
  "SubClass_ID" Serial primary key,
  "SubClass_name" Text Not Null,
  "atk_base" Int Not Null,
  "def_base" Int Not Null,
  "int_base" Int Not Null,
  "res_base" Int Not Null,
  "hit_base" Int Not Null,
  "spd_base" Int Not Null,
  "HP_base" Int Not Null,
  "SP_base" Int Not Null
);
```

Table: SubClass



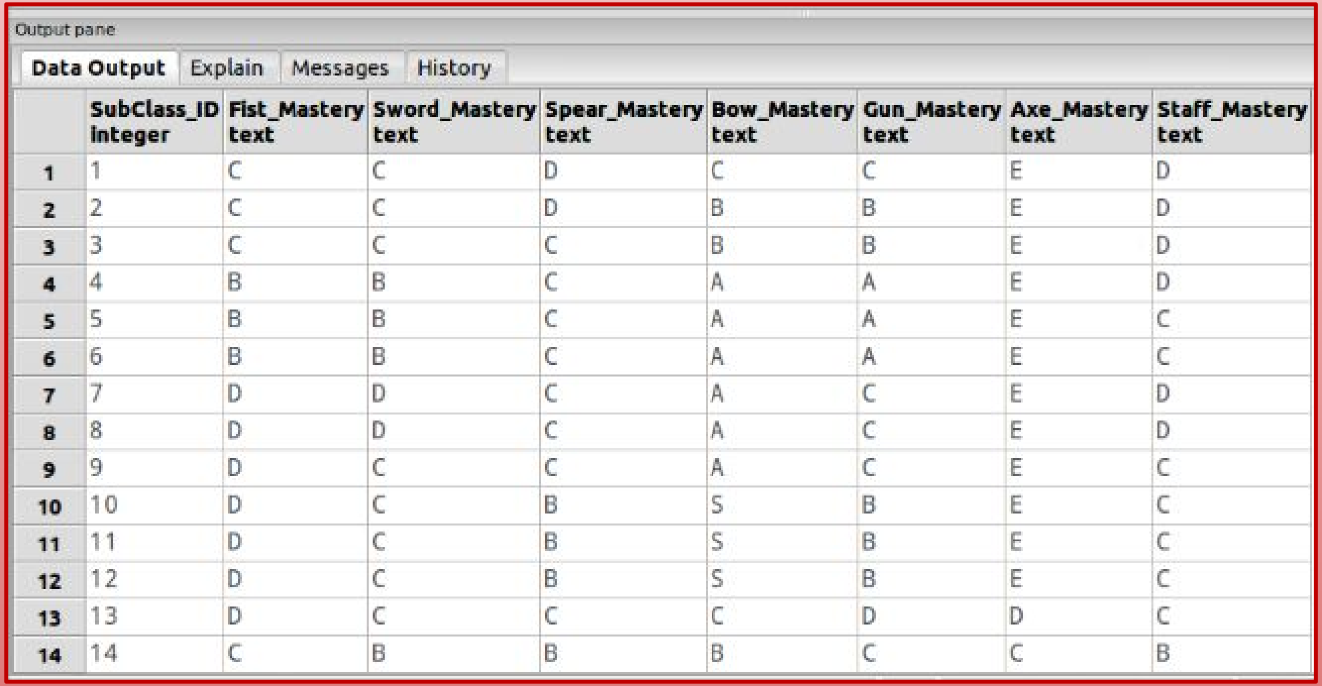
	Class_ID integer	SubClass_ID integer	SubClass_name text	atk_base integer	def_base integer	int_base integer	res_base integer	hit_base integer	spd_base integer	HP_base integer	SP_base integer
1	1	1	Thief	10	7	8	10	12	16	14	11
2	1	2	Rogue	11	7	8	11	13	17	15	12
3	1	3	Scout	12	8	9	12	14	18	16	13
4	1	4	Bandit	13	8	9	13	15	19	17	14
5	1	5	Trickster	14	9	10	14	16	20	18	15
6	1	6	Master_Thie	15	9	10	15	17	21	19	16
7	2	7	Archer	12	7	8	14	16	7	14	12
8	2	8	Hunter	13	7	8	15	18	7	15	13

Table: Weapon_Mastery

This table holds information regarding Mastery of Weapons, in Relation to specific SubClasses

```
create table if not exists "Weapon_Mastery" (
  "SubClass_ID" Int references "SubClass"("SubClass_ID"),
  "Fist_Mastery" Text,
  "Sword_Mastery" Text,
  "Spear_Mastery" Text,
  "Bow_Mastery" Text,
  "Gun_Mastery" Text,
  "Axe_Mastery" Text,
  "Staff_Mastery" Text
);
```

Table: Weapon_Mastery



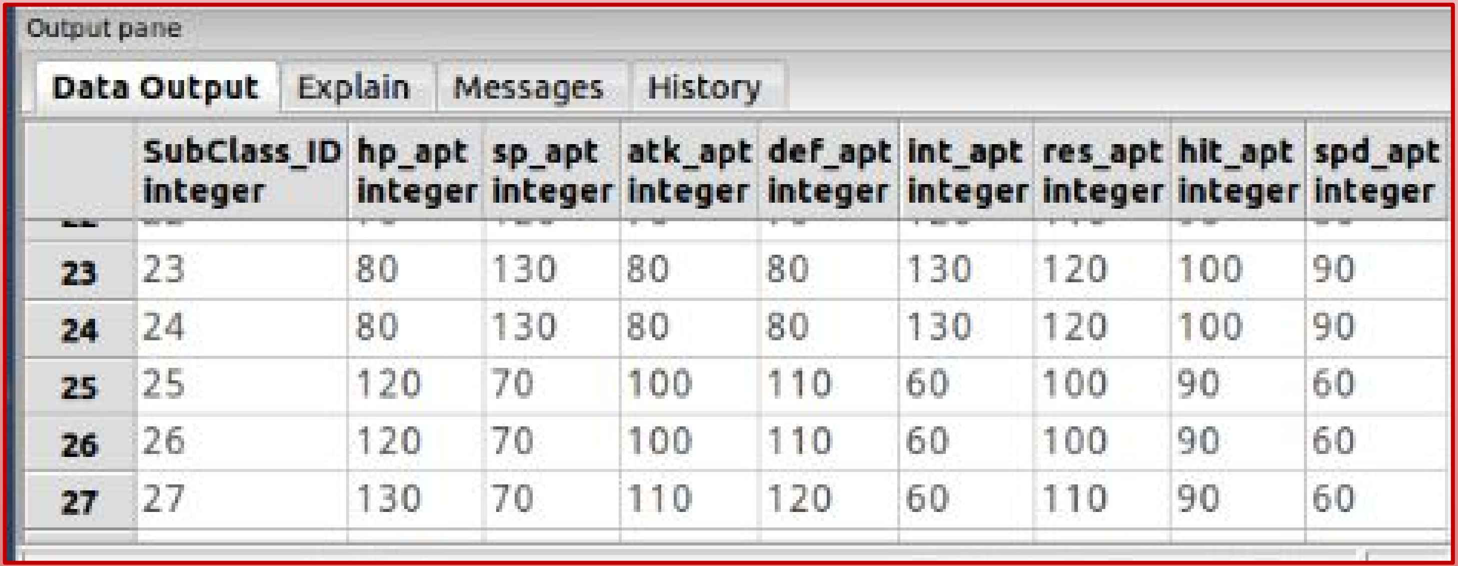
	SubClass_ID integer	Fist_Mastery text	Sword_Mastery text	Spear_Mastery text	Bow_Mastery text	Gun_Mastery text	Axe_Mastery text	Staff_Mastery text
1	1	C	C	D	C	C	E	D
2	2	C	C	D	B	B	E	D
3	3	C	C	C	B	B	E	D
4	4	B	B	C	A	A	E	D
5	5	B	B	C	A	A	E	C
6	6	B	B	C	A	A	E	C
7	7	D	D	C	A	C	E	D
8	8	D	D	C	A	C	E	D
9	9	D	C	C	A	C	E	C
10	10	D	C	B	S	B	E	C
11	11	D	C	B	S	B	E	C
12	12	D	C	B	S	B	E	C
13	13	D	C	C	C	D	D	C
14	14	C	B	B	B	C	C	B

Table: Stats_Aptitude

This table holds Stat Aptitude Details for Sub-Classes

```
create table if not exists "Stats_Aptitude" (
  "SubClass_ID" Int references "SubClass"("SubClass_ID"),
  "hp_apt" Int Not Null,
  "sp_apt" Int Not Null,
  "atk_apt" Int Not Null,
  "def_apt" Int Not Null,
  "int_apt" Int Not Null,
  "res_apt" Int Not Null,
  "hit_apt" Int Not Null,
  "spd_apt" Int Not Null
);
```

Table: Stats_Aptitude



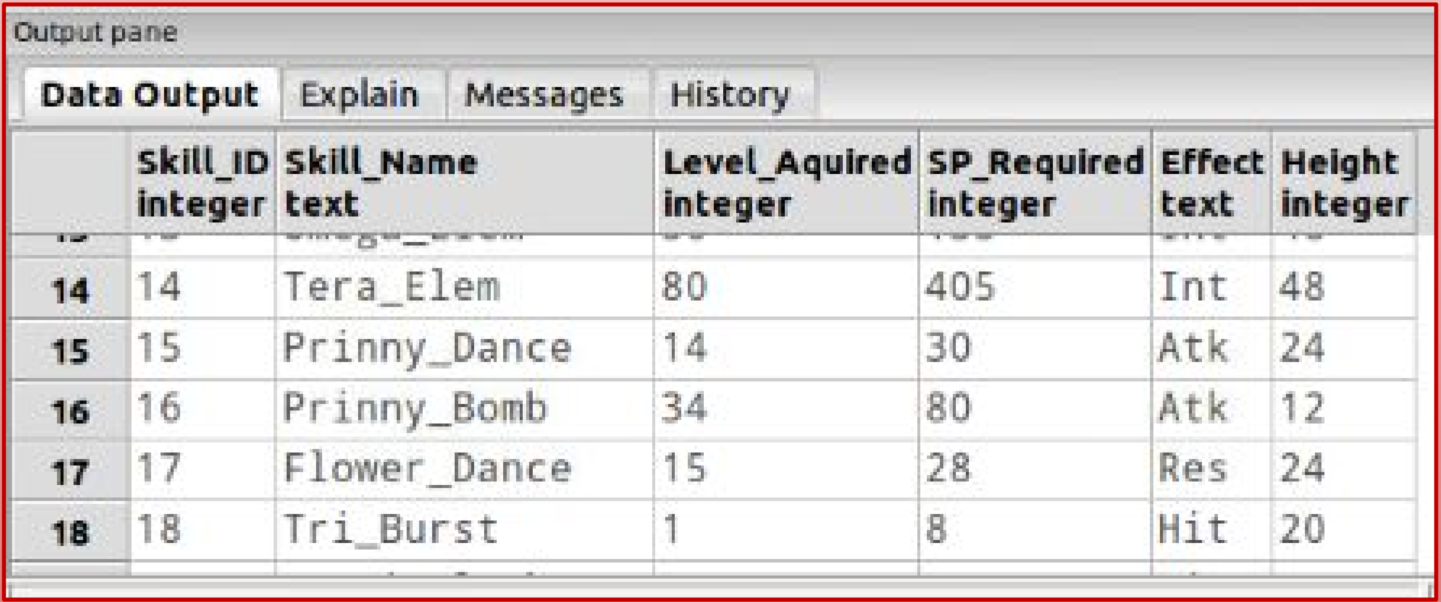
	SubClass_ID integer	hp_apt integer	sp_apt integer	atk_apt integer	def_apt integer	int_apt integer	res_apt integer	hit_apt integer	spd_apt integer
23	23	80	130	80	80	130	120	100	90
24	24	80	130	80	80	130	120	100	90
25	25	120	70	100	110	60	100	90	60
26	26	120	70	100	110	60	100	90	60
27	27	130	70	110	120	60	110	90	60

Table: Skills

This table holds Information Regarding all of the Skills

```
create table if not exists "Skills" (
  "Skill_ID" Serial primary key,
  "Skill_Name" Text Not Null,
  "Level_Aquired" Int Not Null,
  "SP_Required" Int Not Null,
  "Effect" Text Not Null,
  "Height" Int Not Null
);
```

Table: Skills



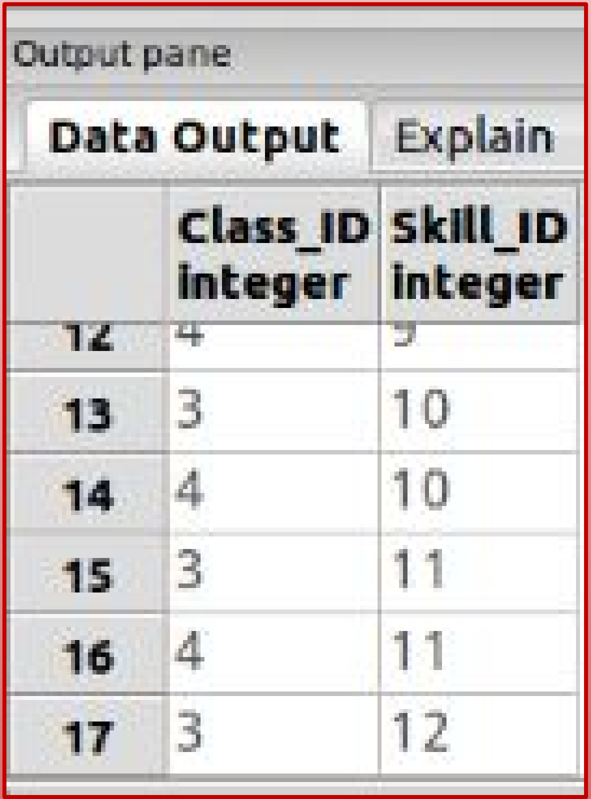
	Skill_ID	Skill_Name	Level_Aquired	SP_Required	Effect	Height
	integer	text	integer	integer	text	integer
14	14	Tera_Elem	80	405	Int	48
15	15	Prinny_Dance	14	30	Atk	24
16	16	Prinny_Bomb	34	80	Atk	12
17	17	Flower_Dance	15	28	Res	24
18	18	Tri_Burst	1	8	Hit	20

Table: Skill_Relations

This table Shows the Relations of Which Skills Are attainable by Which Classes

```
create table if not exists "Skill_Relations" (
  "Class_ID" Int references "Class"("Class_ID"),
  "Skill_ID" Int references "Skills"("Skill_ID")
);
```

Table: Skill_Relations



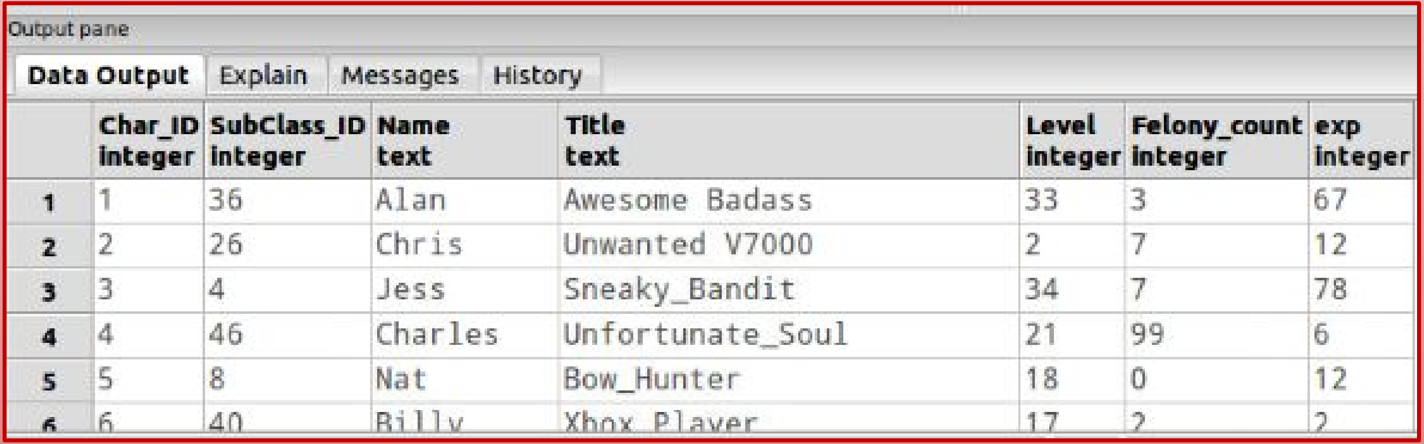
	Class_ID integer	Skill_ID integer
12	4	9
13	3	10
14	4	10
15	3	11
16	4	11
17	3	12

Table: Character

This table details of all Characters that Exist

```
create table if not exists "Character" (
  "Char_ID" Serial primary key,
  "SubClass_ID" Int references "SubClass"("SubClass_ID"),
  "Name" Text,
  "Title" Text,
  "Level" Int,
  "Felony_count" Int,
  "exp" Int
);
```

Table: Character



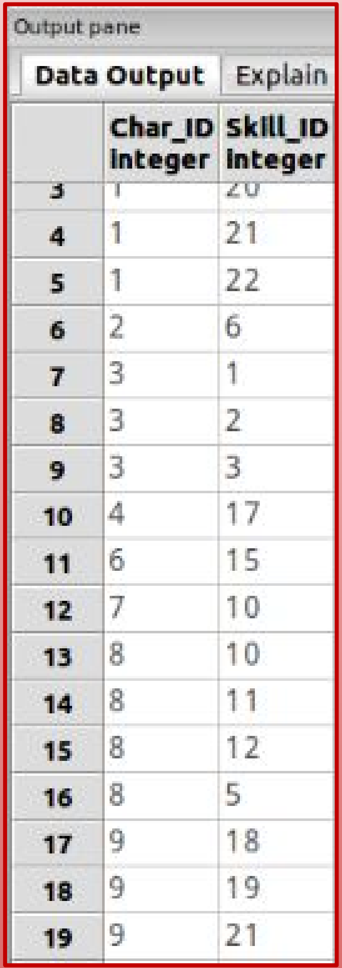
	Char_ID integer	SubClass_ID integer	Name text	Title text	Level integer	Felony_count integer	exp integer
1	1	36	Alan	Awesome Badass	33	3	67
2	2	26	Chris	Unwanted V7000	2	7	12
3	3	4	Jess	Sneaky_Bandit	34	7	78
4	4	46	Charles	Unfortunate_Soul	21	99	6
5	5	8	Nat	Bow_Hunter	18	0	12
6	6	40	Billy	Xbox Player	17	2	2

Table: Char_Skills

This table relates All existing Characters to Whatever Skills they have learned and achieved

```
create table if not exists "Char_Skills" (
  "Char_ID" Int references "Character"("Char_ID"),
  "Skill_ID" Int references "Skills"("Skill_ID")
);
```

Table: Char_Skills



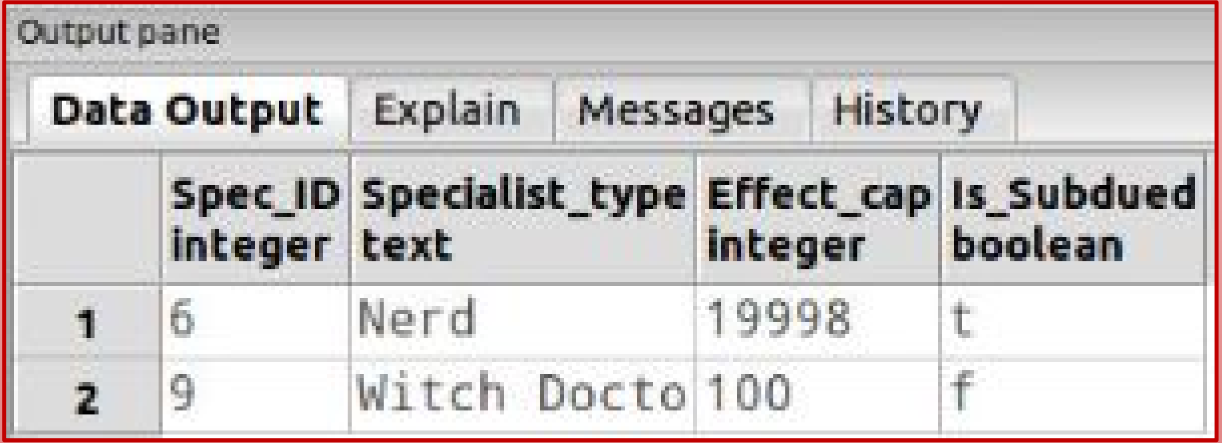
	Char_ID integer	Skill_ID integer
3	1	20
4	1	21
5	1	22
6	2	6
7	3	1
8	3	2
9	3	3
10	4	17
11	6	15
12	7	10
13	8	10
14	8	11
15	8	12
16	8	5
17	9	18
18	9	19
19	9	21

Table: Specialist

A SubType of Characters that can Reside within Items and affect them

```
create table if not exists "Specialist" (
  "Spec_ID" Int references "Character"("Char_ID") primary key,
  "Specialist_type" Text,
  "Effect_cap" Int,
  "Is_Subdued" Bool
);
```

Table: Specialist



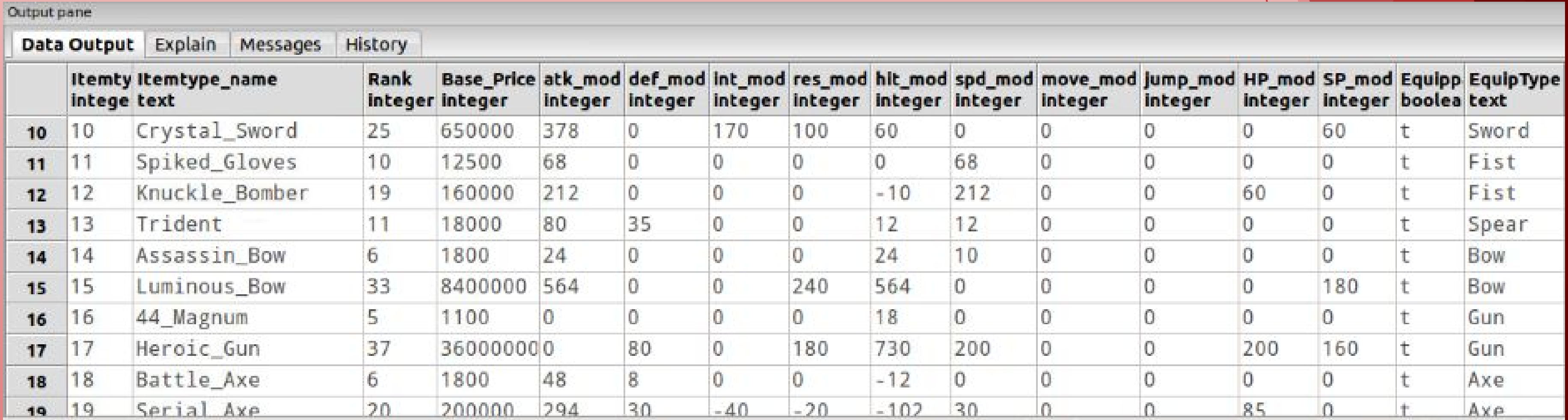
	Spec_ID integer	Specialist_type text	Effect_cap integer	Is_Subdued boolean
1	6	Nerd	19998	t
2	9	Witch Docto	100	f

Table: Item_Type

This Table Contains Base information for all item Types

```
create table if not exists "Item_Type" (
  "Itemtype_ID" Serial primary key,
  "Itemtype_name" Text,
  "Rank" Int,
  "Base_Price" Int Not Null,
  "atk_mod" Int Not Null,
  "def_mod" Int Not Null,
  "int_mod" Int Not Null,
  "res_mod" Int Not Null,
  "hit_mod" Int Not Null,
  "spd_mod" Int Not Null,
  "move_mod" Int Not Null,
  "jump_mod" Int Not Null,
  "HP_mod" Int Not Null,
  "SP_mod" Int Not Null,
  "Equippable" Bool Not Null,
  "EquipType" Text
);
```

Table: Item_Type



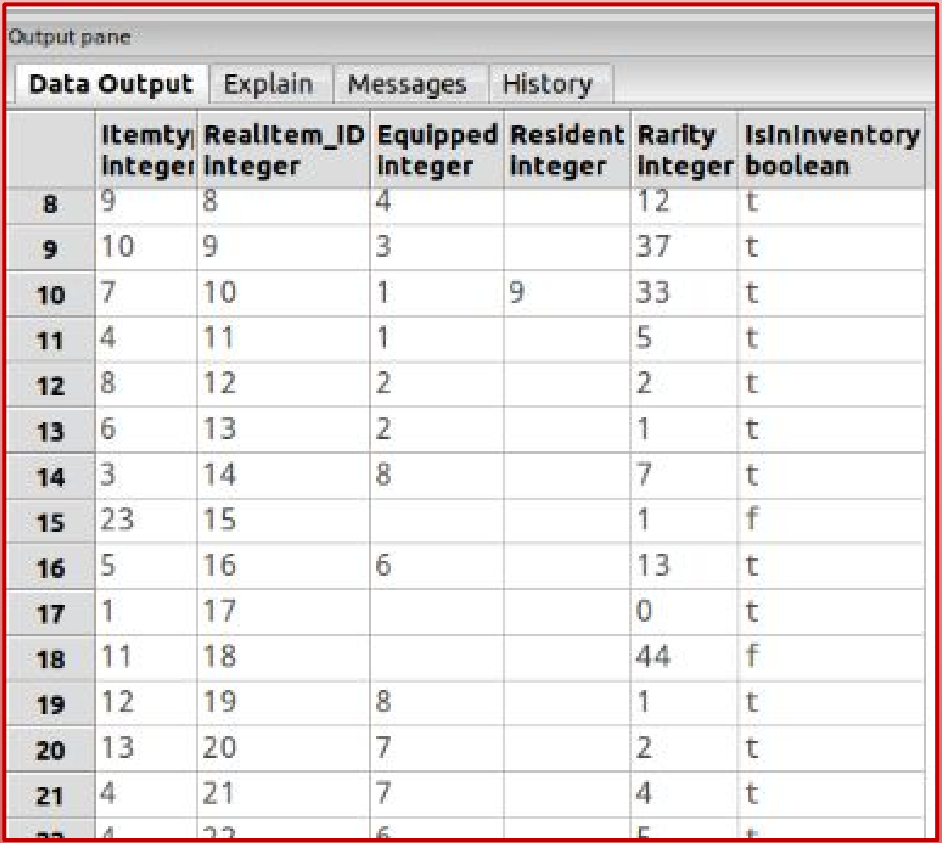
	Itemty intege	Itemtype_name text	Rank integer	Base_Price integer	atk_mod integer	def_mod integer	int_mod integer	res_mod integer	hit_mod integer	spd_mod integer	move_mod integer	jump_mod integer	HP_mod integer	SP_mod integer	Equipp boolea	EquipType text
10	10	Crystal_Sword	25	650000	378	0	170	100	60	0	0	0	0	60	t	Sword
11	11	Spiked_Gloves	10	12500	68	0	0	0	0	68	0	0	0	0	t	Fist
12	12	Knuckle_Bomber	19	160000	212	0	0	0	-10	212	0	0	60	0	t	Fist
13	13	Trident	11	18000	80	35	0	0	12	12	0	0	0	0	t	Spear
14	14	Assassin_Bow	6	1800	24	0	0	0	24	10	0	0	0	0	t	Bow
15	15	Luminous_Bow	33	8400000	564	0	0	240	564	0	0	0	0	180	t	Bow
16	16	44_Magnum	5	1100	0	0	0	0	18	0	0	0	0	0	t	Gun
17	17	Heroic_Gun	37	36000000	0	80	0	180	730	200	0	0	200	160	t	Gun
18	18	Battle_Axe	6	1800	48	8	0	0	-12	0	0	0	0	0	t	Axe
19	19	Serial_Axe	20	200000	294	30	-40	-20	-102	30	0	0	85	0	t	Axe

Table: Item_Real

This Table Contains Items that are Real and Exist in game

```
create table if not exists "Item_Real" (
  "Itemtype_ID" Int references "Item_Type"("Itemtype_ID"),
  "RealItem_ID" Serial primary key,
  "Equipped" Int references "Character"("Char_ID"),
  "Resident" Int references "Specialist"("Spec_ID"),
  "Rarity" Int,
  "IsInInventory" Bool,
  Constraint Resident_Equip check ("Resident"<>"Equipped")
);
```

Table: Item_Real



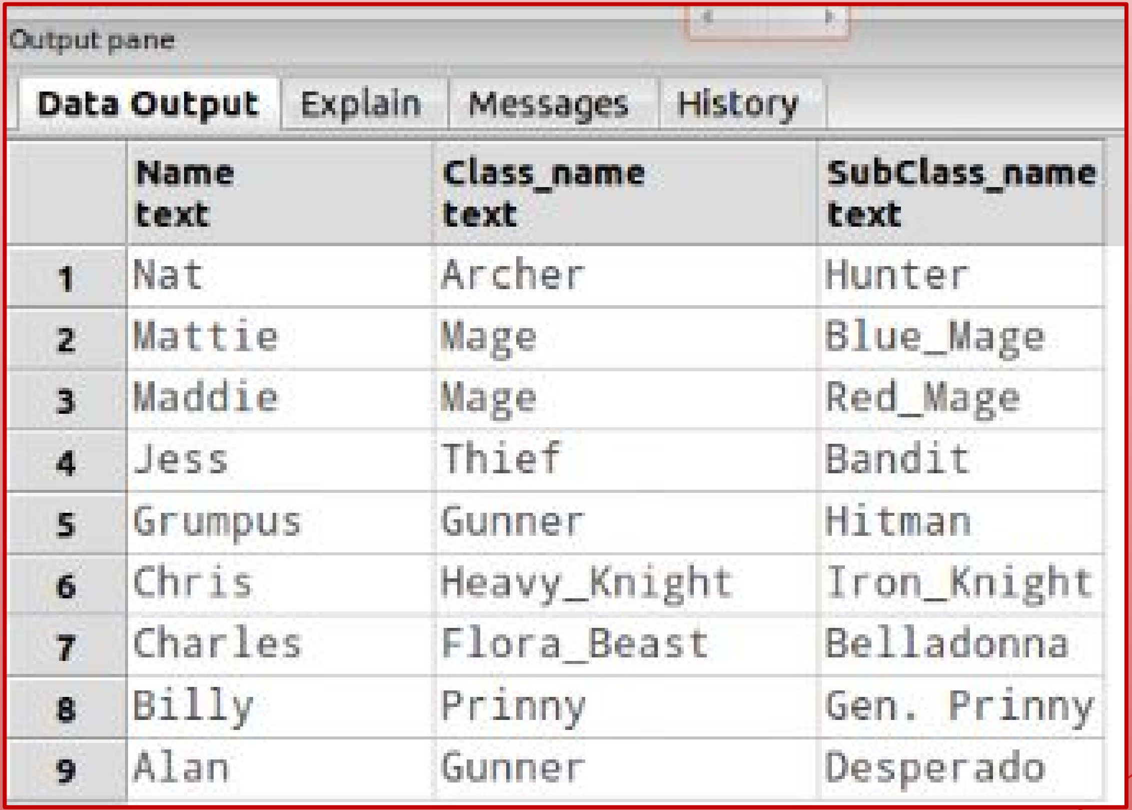
	Itemty integer	RealItem_ID integer	Equipped integer	Resident integer	Rarity integer	IsInInventory boolean
8	9	8	4		12	t
9	10	9	3		37	t
10	7	10	1	9	33	t
11	4	11	1		5	t
12	8	12	2		2	t
13	6	13	2		1	t
14	3	14	8		7	t
15	23	15			1	f
16	5	16	6		13	t
17	1	17			0	t
18	11	18			44	f
19	12	19	8		1	t
20	13	20	7		2	t
21	4	21	7		4	t
22	4	22	6		5	t

View: Character_Classes

This View will return all Characters, along with their Class Name, and SubClass Name, Ordered by Character Name In Reverse Alphabetical Order

```
Create view "Character_Classes" As
Select "Character"."Name", "Class"."Class_name", "SubClass"."SubClass_name"
from "Character"
Left outer join "SubClass" On
"Character"."SubClass_ID" = "SubClass"."SubClass_ID"
inner join "Class" on
"SubClass"."Class_ID" = "Class"."Class_ID"
Order By "Character"."Name" Desc;
```

View: Character_Classes



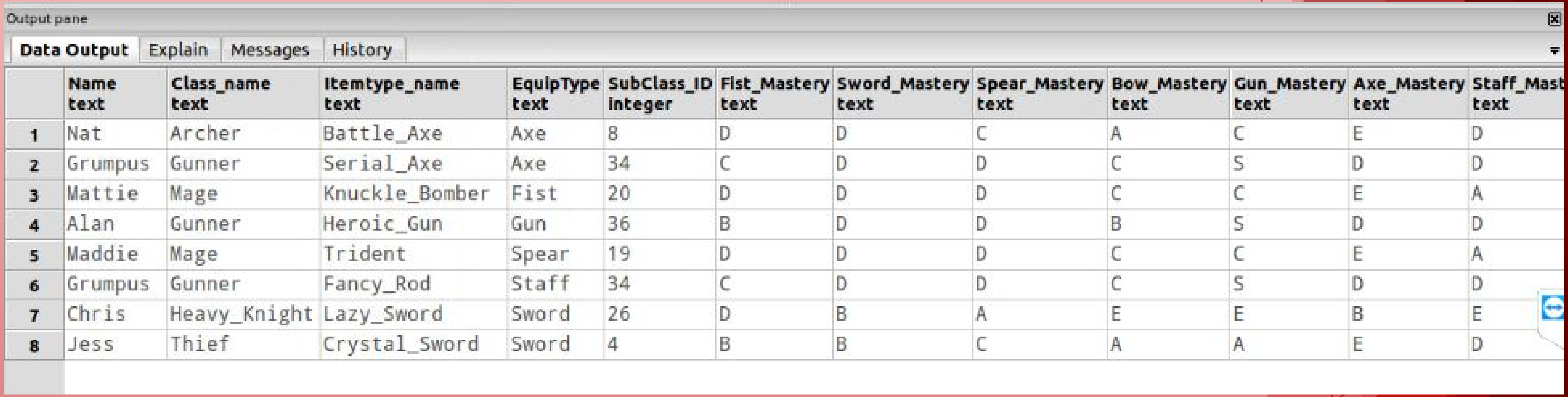
	Name text	Class_name text	SubClass_name text
1	Nat	Archer	Hunter
2	Mattie	Mage	Blue_Mage
3	Maddie	Mage	Red_Mage
4	Jess	Thief	Bandit
5	Grumpus	Gunner	Hitman
6	Chris	Heavy_Knight	Iron_Knight
7	Charles	Flora_Beast	Belladonna
8	Billy	Prinny	Gen. Prinny
9	Alan	Gunner	Desperado

View: Equipped_Mastery_Class

This view will return all Characters with Equipped Weapons (Not Armor),
The character's Subclass name, the name of their Equipped Weapon,
That Weapon's Type, and that subclass' Weapon Mastery, Ordered By Equip Type

```
Create View "Equipped_Mastery_Class" As
Select "Character"."Name", "Class"."Class_name", "Item_Type"."Itemtype_name",
"Item_Type"."EquipType", "Weapon_Mastery".*
From "Item_Real"
inner Join "Item_Type"
On "Item_Real"."Itemtype_ID" = "Item_Type"."Itemtype_ID"
inner Join "Character"
On "Item_Real"."Equipped" = "Character"."Char_ID"
Inner Join "Weapon_Mastery"
On "Character"."SubClass_ID" = "Weapon_Mastery"."SubClass_ID"
Inner Join "SubClass"
on "Character"."SubClass_ID" = "SubClass"."SubClass_ID"
Inner Join "Class"
on "SubClass"."Class_ID" = "Class"."Class_ID"
Where ("Item_Type"."Equippable" <> false)
And ("Item_Real"."Equipped" is not null)
And ("Item_Type"."EquipType" <> 'Armor')
Order By "Item_Type"."EquipType" ASC;
```

View: Equipped_Mastery_Class



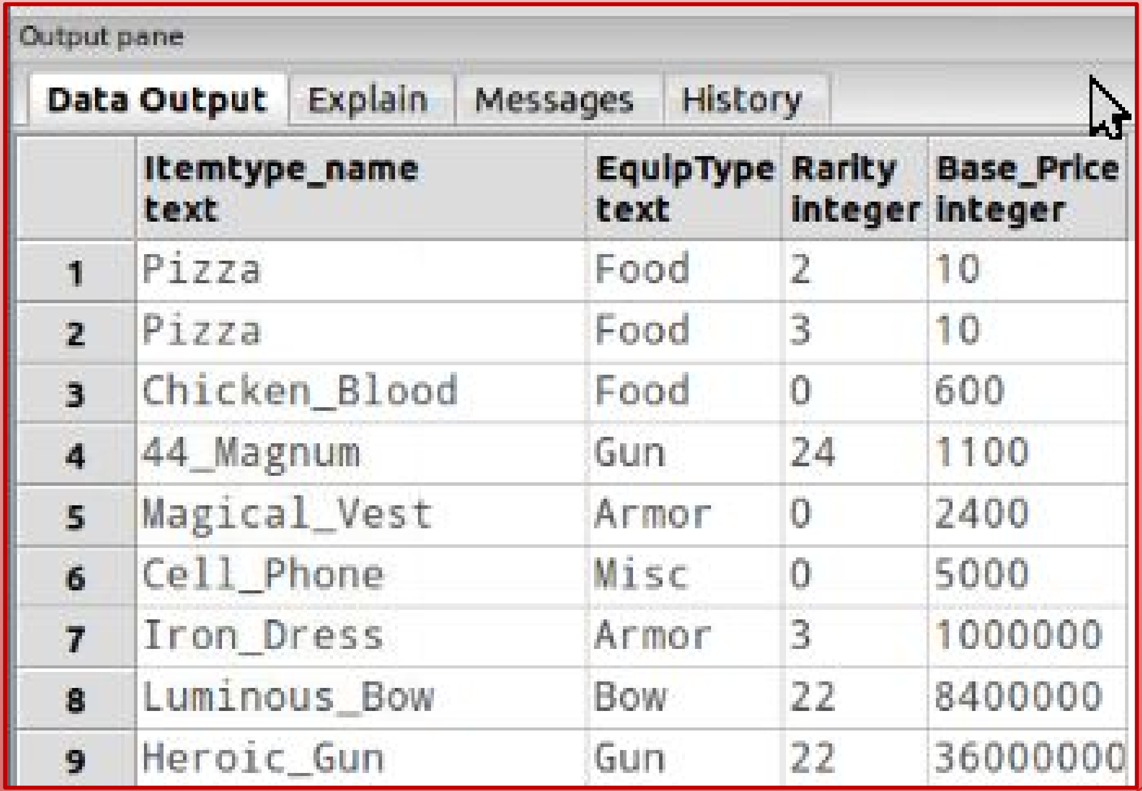
	Name	Class_name	Itemtype_name	EquipType	SubClass_ID	Fist_Mastery	Sword_Mastery	Spear_Mastery	Bow_Mastery	Gun_Mastery	Axe_Mastery	Staff_Mast
	text	text	text	text	integer	text	text	text	text	text	text	text
1	Nat	Archer	Battle_Axe	Axe	8	D	D	C	A	C	E	D
2	Grumpus	Gunner	Serial_Axe	Axe	34	C	D	D	C	S	D	D
3	Mattie	Mage	Knuckle_Bomber	Fist	20	D	D	D	C	C	E	A
4	Alan	Gunner	Heroic_Gun	Gun	36	B	D	D	B	S	D	D
5	Maddie	Mage	Trident	Spear	19	D	D	D	C	C	E	A
6	Grumpus	Gunner	Fancy_Rod	Staff	34	C	D	D	C	S	D	D
7	Chris	Heavy_Knight	Lazy_Sword	Sword	26	D	B	A	E	E	B	E
8	Jess	Thief	Crystal_Sword	Sword	4	B	B	C	A	A	E	D

View: emptyItems

This view will return the name, rarity, and price of all
Items in your inventory that are not equipped to any character,
And that have no residents

```
Create view "emptyItems" As
Select "Item_Type"."Itemtype_name", "Item_Type"."EquipType", "Item_Real"."Rarity",
"Item_Type"."Base_Price"
from "Item_Type", "Item_Real"
where "Item_Real"."Resident" is null
AND "Item_Real"."Equipped" is null
AND "Item_Real"."IsInInventory" = true
AND "Item_Real"."Itemtype_ID" = "Item_Type"."Itemtype_ID"
ORDER BY "Item_Type"."Base_Price" Asc;
```

View: emptyItems

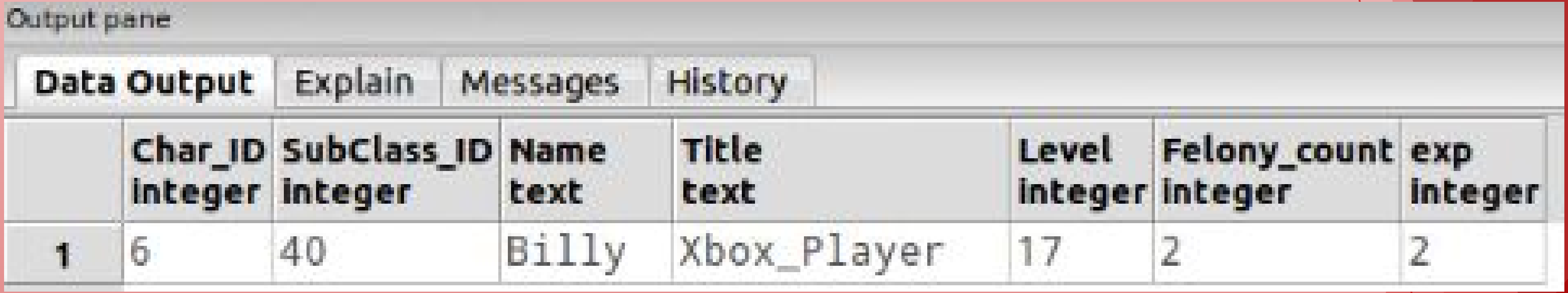


	Itemtype_name text	EquipType text	Rarity integer	Base_Price integer
1	Pizza	Food	2	10
2	Pizza	Food	3	10
3	Chicken_Blood	Food	0	600
4	44_Magnum	Gun	24	1100
5	Magical_Vest	Armor	0	2400
6	Cell_Phone	Misc	0	5000
7	Iron_Dress	Armor	3	1000000
8	Luminous_Bow	Bow	22	8400000
9	Heroic_Gun	Gun	22	36000000

Report: All Characters with No Weapons Equipped

```
Select *
 from "Character"
where "Character"."Char_ID" Not in(
Select "Character"."Char_ID"
From "Item_Real"
inner Join "Item_Type"
On "Item_Real"."Itemtype_ID" = "Item_Type"."Itemtype_ID"
inner Join "Character"
On "Item_Real"."Equipped" = "Character"."Char_ID"
Where ("Item_Type"."Equippable" <> false)
And ("Item_Real"."Equipped" is not null)
And ("Item_Type"."EquipType" <> 'Armor')
Order By "Item_Type"."EquipType" ASC);
```

Report: All Characters with No Weapons Equipped



	Char_ID integer	SubClass_ID integer	Name text	Title text	Level integer	Felony_count integer	exp integer
1	6	40	Billy	Xbox_Player	17	2	2

StoredProcedure

```
Create or Replace Function CalculateAtk("Input_Char_ID" Int) as
$$
Declare
atk_base Int :=
(Select "SubClass"."atk_base"
from "Character",
inner join "SubClass"
on "Character"."SubClass_ID" = "SubClass"."SubClass_ID" Input_Char_ID
Where "Character"."Char_ID" = "Input_Char_ID");
equip_mod Int :=
(Select "Item_Type"."atk_mod"
/*Continued on the next page */
```

StoredProcedure

```
/*Continued*/
from "Character"
Inner Join "Item_Real"
on "Item_Real"."Equipped" = "Input_Char_ID"
Inner Join "Item_Type”
on "Item_Real"."Itemtype_ID" = "Item_Type"."Itemtype_ID"
atk_final Int :=
(Select sum(atk_base, equip_mod));
Begin
return atk_final;
end;
$$ Language plpgsql;
```

StoredProcedure

```
Create or Replace Function LevelUp("Input_Char_ID" Int) AS $$
Declare
level := (Select "Character"."Level"
from "Character");
exp := (Select "Character"."Level"
from "Character");
Begin
if exp >= 100
then Level = Level + 1
Update "Character"."Level"
where "Character"."Char_ID" = "Input_Char_ID";
```

Triggers

```
Create Trigger expLevel
After Insert Or Update on "Character"."exp"
For "Character"."exp"
Execute Procedure Levelup('Char_ID");
```

Security

```
Create Role Admin;
Grant All on All Tables
In Schema Public
To Admin;
```

Security

```
Create Role Player;

Grant Select, Insert, Update On "Character", "Specialist", "Char_Skills"

to Player;

Grant Select, Update On "Item_Real"

to Player;

Grant Select On "Class", "SubClass", "Weapon_Mastery", "Stats_Aptitude",
  "Skills", "Skill_Relations", "Item_Type"

To Player;
```

Implementation notes

Aptitudes should be Percentages, but are stored as Decimals, with 100 being 100%

Known Problems

Stored Procedure Issues, commented out in Sql;

Future enhancements

Fix all Known Problems

Configure Limits for Equips, Specialists

Create More StoredProcedures for all stats, not just ATK