CS:GO Steam Marketplace

Database Management Final Project

Professor: Alan Labouseur

CMPT 308 Database Management

Marist College

Jesse Opitz

Table of Contents

Executive Summary 3
ER Diagram 4
Tables ..... 5 – 17
- Skins/Weapons 5
- ItemsInInventory 6, 7
- SkinsOnWeapons 8
- Keys 9
- Stickers 9, 10
- OpCases 10, 11
- Users 11, 12
- UserItems 12, 13
- CSGOMarketplace 14, 15
- OnMarketplace 16, 17

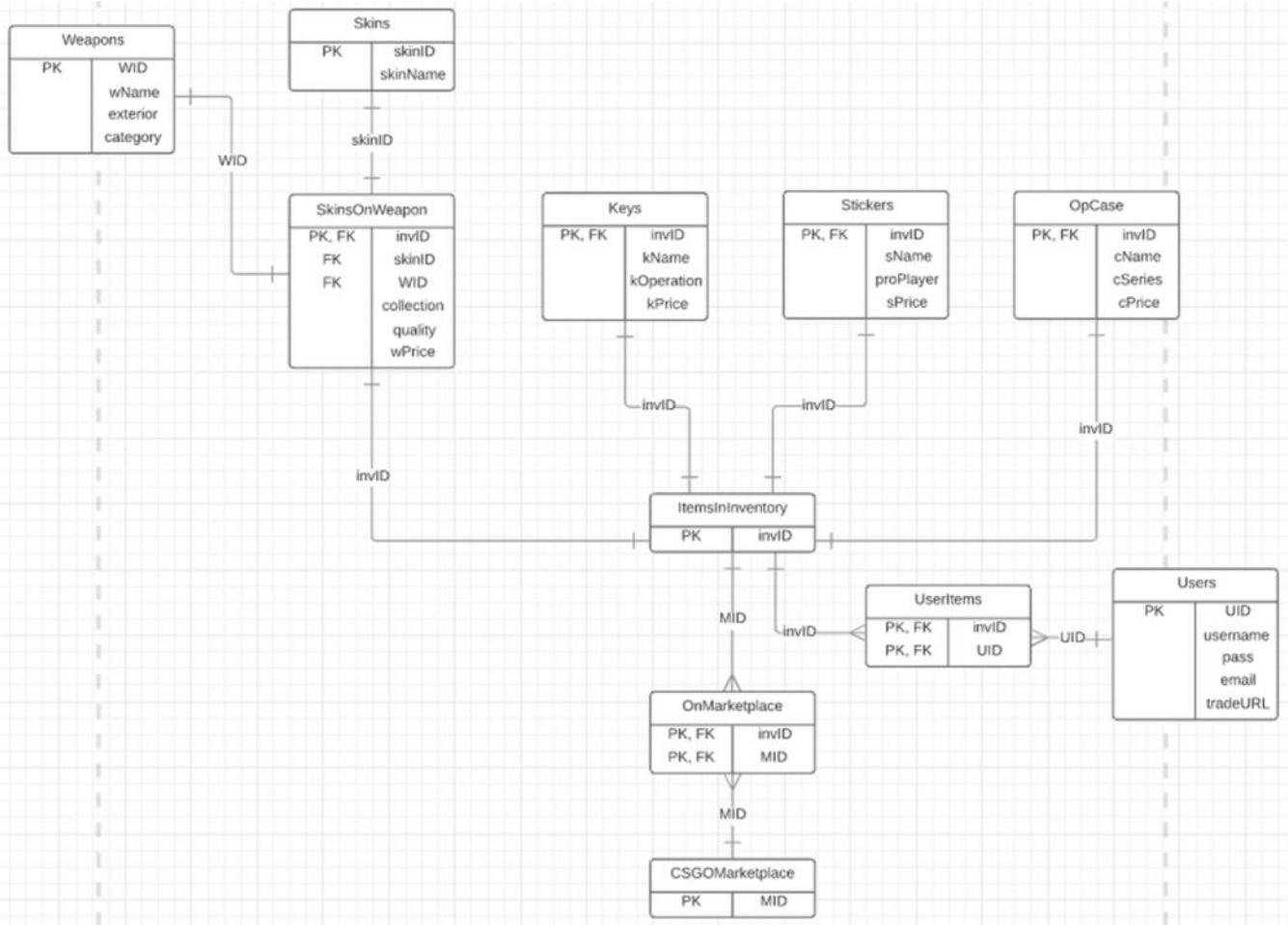
Check Constraints 18
Views 19-22
- userWeaponsOnMarketplace 19
- userKeysOnMarketplace 20
- userStickersOnMarketplace 21
- userCasesOnMarketplace 22

Reports and Queries 23-24
Stored Procedures/Triggers 25
Security 26
Important Notes/ Known Problems / Future Enhancements 27

Executive Summary

In order to have a successful marketplace, your database must be flawless. I have created a well replicated, if not, better database design for the Steam Marketplace Counter Strike: Global Offensive(CS:GO) section. The goal of this database is for users to sell the weapons in their inventory, for money, through steam. Potential users include people looking to create a marketplace to sell CS:GO items, PC gamers, and anyone interested in selling weapons earned in CS:GO.

ER Diagram


Weapons
PK
WID
wName
exterior
category
Skins
PK
skinID
skinName
WID
skinID
SkinsOnWeapon
PK, FK
FK
FK
invID
skinID
WID
collection
quality
wPrice
Keys
PK, FK
invID
kName
kOperation
kPrice
Stickers
PK, FK
invID
sName
proPlayer
sPrice
OpCase
PK, FK
invID
cName
cSeries
cPrice
invID
invID
invID
invID
ItemsInInventory
PK
invID
MID
invID
UserItems
PK, FK
PK, FK
invID
UID
UID
Users
PK
UID
username
pass
email
tradeURL
OnMarketplace
PK, FK
PK, FK
invID
MID
MID
CSGOMarketplace
PK
MID

Tables

Skins Table:

The Skins table is to store information about different skins.

```
CREATE TABLE Skins (
    skinID VARCHAR(10) PRIMARY KEY UNIQUE NOT NULL,
    skinName VARCHAR(100) NOT NULL
);
```

Functional Dependencies: skinID → skinName

	skinid character varying(10)	skinname character varying(100)
1	s000000001	Predator
2	s000000002	Safari Mesh
3	s000000003	Case Hardened
4	s000000004	Griffin
5	s000000005	Knight
6	s000000006	Ricochet
7	s000000007	Man-o-war
8	s000000008	Asiimov
9	s000000009	Red FragCam
10	s000000010	Scorpion
11	s000000011	Stained
12	s000000012	Crimson Web

Weapons Table:

The Weapons table holds information about different weapons.

```
CREATE TABLE Weapons(
    WID VARCHAR(10) PRIMARY KEY  NOT NULL,
    wName VARCHAR(100) NOT NULL,
    exterior VARCHAR(100) NOT NULL,
    category VARCHAR(20) NOT NULL
);
```

Functional Dependencies: WID → wName, exterior, category

	wid character varying(10)	wname character varying(100)	exterior character varying(100)	category character varying(20)
1	w000000001	AK-47	Field-Tested	Normal
2	w000000002	AK-47	Factory New	Somvenir
3	w000000003	AK-47	Minimal Wear	StatTrak
4	w000000004	M4A4	Factory New	StatTrak
5	w000000005	M4A1-S	Factory New	Normal
6	w000000006	AmG	Well-Worn	Normal
7	w000000007	AmG	Factory New	StatTrak
8	w000000008	AWP	Feild-Tested	Normal
9	w000000009	AWP	Battle-Scarred	StatTrak
10	w000000010	AWP	Field-Tested	StatTrak
11	w000000011	p2000	Battle-Scarred	StatTrak
12	w000000012	p2000	Minimal Wear	Normal
13	w000000013	Bowie Knife	Field-Tested	*

ItemsInInventory Table:

The ItemsInInventory table holds all items that can be in the inventory and the user's tradeURL. The tradeURL is unique to every user and allows users to interact with each other's inventory.

```
CREATE TABLE ItemsInInventory(
   invID VARCHAR(10) PRIMARY KEY NOT NULL
);
```

Functional Dependencies: invID → N/A

Sample data on next page:

	invid character varying(10)
1	i000000001
2	i000000002
3	i000000003
4	i000000004
5	i000000005
6	i000000006
7	i000000007
8	i000000008
9	i000000009
10	i000000010
11	i000000011
12	i000000012
13	i000000013
14	i000000014
15	i000000015
16	i000000016
17	i000000017
18	i000000018
19	i000000019
20	i000000020
21	i000000021
22	i000000022
23	i000000023
24	i000000024
25	i000000025
26	i000000026
27	i000000027
28	i000000028
29	i000000029
30	i000000030
31	i000000031
32	i000000032
33	i000000033
34	i000000034
35	i000000035
35	i000000035
36	i000000036
37	i000000037
38	i000000038
39	i000000039
40	i000000040
41	i000000041
42	i000000042
43	i000000043
44	i000000044
45	i000000045
46	i000000046
47	i000000047
48	i000000048
49	i000000049
50	i000000050

SkinsOnWeapon Table:

The SkinsOnWeapon table is used to store the different types of weapons with skins that an inventory has.

```
CREATE TABLE SkinsOnWeapon(
   invID VARCHAR(10) PRIMARY KEY NOT NULL,
   skinID VARCHAR(10) NOT NULL,
   WID VARCHAR(10) NOT NULL,
   collection VARCHAR(100),
   quality VARCHAR(100) NOT NULL,
   wPriceUSD DECIMAL(5,2) NOT NULL,
   FOREIGN KEY (invID) REFERENCES ItemsInInventory(invID),
   FOREIGN KEY (skinID) REFERENCES Skins(skinID),
   FOREIGN KEY (WID) REFERENCES Weapons(WID)
);
```

Functional Dependencies: invID → skinID, WID, collection, quality, wPriceUSD

	invid character varying(10)	skinid character varying(10)	wid character varying(10)	collection character varying(100)	quality character varying(100)	wprice numeric(5,2)
1	i000000001	s000000001	w000000001	The Dmst	Indmstrial Grade Rifle	0.71
2	i000000002	s000000002	w000000002	The Dmst 2	Indmstrial Grade Rifle	138.00
3	i000000003	s000000003	w000000003	The Arms Deal	Classified Rifle	190.00
4	i000000004	s000000004	w000000004	The Vangamrd	Restricted Rifle	23.45
5	i000000005	s000000005	w000000005	The Vangamrd	Restricted Rifle	274.44
6	i000000006	s000000006	w000000006	The Revolver Case	Mil-Spec Grade Rifle	0.20
7	i000000007	s000000006	w000000007	The Revolver Case	Mil-Spec Grade Rifle	4.14
8	i000000008	s000000007	w000000008	The Chroma	Covert Sniper Rifle	11.75
9	i000000009	s000000008	w000000008	The Phoenix	Covert Sniper Rifle	40.82
10	i000000010	s000000008	w000000008	The Phoenix	Covert Sniper Rifle	40.82
11	i000000011	s000000007	w000000008	The Chroma	Covert Sniper Rifle	11.75
12	i000000012	s000000008	w000000009	The Phoenix	Covert Sniper Rifle	70.50
13	i000000013	s000000009	w000000011	The Arms Deal 3	Mil-Spec Grade Pistol	1.74
14	i000000014	s000000010	w000000012	The Dmst	Restricted Pistol	6.15
15	i000000015	s000000012	w000000013		Covert Knife	124.20

Keys Table:

The Keys table holds all the different types of keys available in the game CS:GO. This is not talking about anything to do with database or encryption keys. In CS:GO, keys are used to open cases and usually cost around USD$2.80.

```
CREATE TABLE Keys(
   invID VARCHAR(10) PRIMARY KEY NOT NULL,
   kName VARCHAR(100) NOT NULL,
   kOperation VARCHAR(100) NOT NULL,
   kPriceUSD DECIMAL(5,2) NOT NULL,
   FOREIGN KEY (invID) REFERENCES ItemsInInventory(invID)
);
```

Functional Dependencies: invID → kName, kOperation, kPriceUSD

	invid character varying(10)	kname character varying(100)	koperation character varying(100)	kprice numeric(5,2)
1	i000000016	Chroma 2 Case Key	Chroma 2	2.84
2	i000000017	Operation Vangamrd Case Key	Operation Vangamrd	2.81
3	i000000018	Revolver Case Key	Revolver	2.83
4	i000000019	eSports Key	eSports	2.84
5	i000000020	Chroma Case Key	Chroma	2.83
6	i000000021	Chroma Case Key	Chroma	2.83
7	i000000022	Hmntsman Case Key	Hmntsman	2.84
8	i000000023	Operation Wildfire Case Key	Operation Wildfire	2.84
9	i000000024	Chroma 2 Case Key	Chroma 2	2.84
10	i000000025	eSports Key	eSports	2.84

Stickers Table:

The stickers table holds all the available stickers that may be placed on weapons after purchasing.

```
CREATE TABLE Stickers(
   invID VARCHAR(10) PRIMARY KEY NOT NULL,
   sName VARCHAR(100) NOT NULL,
   proPlayer VARCHAR(100),
   sPriceUSD DECIMAL(5,2) NOT NULL,
```

```
   FOREIGN KEY (invID) REFERENCES ItemsInInventory(invID)
);
```

Functional Dependencies: invID → sName, sType, sCollection, proPlayer, sPriceUSD

	invid character varying(10)	sname character varying(100)	proplayer character varying(100)	sprice numeric(5,2)
1	i000000026	Terrorized		12.76
2	i000000027	PENTA Sports DreamHack 2014		2.43
3	i000000028	mnicorn		1.78
4	i000000029	The Sammrai		0.77
5	i000000030	flamie Cologne 2015	flamie	0.38

OpCase Table:

The OpCase tables stores different cases that are used in the game to obtain new skins when joined with a key. The Op before Case stands for operation because there is a new case for every new operation in CS:GO.

```
CREATE TABLE OpCase(
   invID VARCHAR(10) PRIMARY KEY NOT NULL,
   cName VARCHAR(100) NOT NULL,
   cSeries INT NOT NULL,
   cPriceUSD VARCHAR(100) NOT NULL,
   FOREIGN KEY (invID) REFERENCES ItemsInInventory(invID)
);
```

Functional Dependencies: invID → cName, cSeries, cPriceUSD

	invid character varying(10)	cname character varying(100)	cseries integer	cprice character varying(100)
1	i000000031	Shadow Case	80	0.05
2	i000000032	Revolver Case	111	0.05
3	i000000033	Chroma Case	38	0.12
4	i000000034	Falchion Case	50	0.05
5	i000000035	Shadow Case	80	0.05
6	i000000036	Operation Vangamrd Weapon Case	29	0.19
7	i000000037	Operation Bravo Case	3	11.07
8	i000000038	Shadow Case	80	0.05
9	i000000039	Falchion Case	50	0.05
10	i000000040	Chroma Case	38	0.12
11	i000000041	Chroma 2 Case	48	0.18
12	i000000042	Chroma 2 Case	48	0.18
13	i000000043	Operation Phoenix Weapon Case	11	0.12
14	i000000044	Falchion Case	50	0.05
15	i000000045	Falchion Case	50	0.05
16	i000000046	Operation Vangamrd Weapon Case	29	0.19
17	i000000047	eSports 2013 Winter Case	5	0.28
18	i000000048	Hmntsman Weapon Case	17	1.00
19	i000000049	Hmntsman Weapon Case	17	1.00
20	i000000050	Operation Vangamrd Weapon Case	29	0.19

Users Table:

The Users table holds the users information. There is a check constraint on pass (which stands for password) in order to keep users from inputting an insecure password using a length comparison statement and a simple regular expression. The check constraint forces users to input a password that is at least 8 characters and alphanumeric.

```
CREATE TABLE Users(
   UID VARCHAR(10) PRIMARY KEY NOT NULL,
   username VARCHAR(100) NOT NULL,
```

```
   pass VARCHAR(100) NOT NULL CHECK (char_length(pass) >= 8 AND pass ~ '[[:alpha:]]'
AND pass ~ '\d'),
   email VARCHAR(100) NOT NULL,
   tradeURL VARCHAR(100) NOT NULL
);
```

Functional Dependencies: UID → username, pass, email

	uid character varying(10)	username character varying(100)	pass character varying(100)	email character varying(100)	tradeurl character varying(100)
1	m000000001	wakefm1	craigSmcks6	wakefml1@gmail.com	https://steamcommmnity.com/tradeoffer/new/?partner=31319061&token=0sVoLE2z
2	m000000002	glassw4r3	bondJ4m3S12	glasswar@gmail.com	https://steamcommmnity.com/tradeoffer/new/?partner=31319062&token=0sVoLE2z
3	m000000003	co1n4ge	d0zerDZ035	coin4ge@gmail.com	https://steamcommmnity.com/tradeoffer/new/?partner=31319063&token=0sVoLE2z
4	m000000004	fr0nted	rHym1Ngq01	front@gmail.com	https://steamcommmnity.com/tradeoffer/new/?partner=31319064&token=0sVoLE2z
5	m000000005	arr3v41	iv0ry16m90	arreeval@gmail.com	https://steamcommmnity.com/tradeoffer/new/?partner=31319065&token=0sVoLE2z

UserItems Table:

The UserItems table is a weak entity used to combine the Users and ItemsInInventory tables. In this table, one will be able to find any items that a specific user owns. Also, the user's trade URL is used to go into a direct trade with that specific user, where the two users in the trade can swap anything in their inventory.

```
CREATE TABLE UserItems(
   UID VARCHAR(10) NOT NULL,
   invID VARCHAR(10) NOT NULL,
   FOREIGN KEY (UID) REFERENCES Users(UID),
   FOREIGN KEY (invID) REFERENCES ItemsInInventory(invID),
   PRIMARY KEY (invID, UID)
);
```

Functional Dependencies: UID, invID → tradeURL

Sample data on next page:

	uid character varying(10)	invid character varying(10)
1	m000000001	i000000001
2	m000000001	i000000002
3	m000000001	i000000003
4	m000000001	i000000004
5	m000000001	i000000005
6	m000000001	i000000006
7	m000000001	i000000007
8	m000000001	i000000008
9	m000000001	i000000009
10	m000000001	i000000010
11	m000000002	i000000011
12	m000000002	i000000012
13	m000000002	i000000013
14	m000000002	i000000014
15	m000000002	i000000015
16	m000000002	i000000016
17	m000000002	i000000017
18	m000000002	i000000018
19	m000000002	i000000019
20	m000000002	i000000020
21	m000000003	i000000021
22	m000000003	i000000022
23	m000000003	i000000023
24	m000000003	i000000024
25	m000000003	i000000025
26	m000000003	i000000026
27	m000000003	i000000027
28	m000000003	i000000028
29	m000000003	i000000029
30	m000000003	i000000030
31	m000000004	i000000031
32	m000000004	i000000032
33	m000000004	i000000033
34	m000000004	i000000034
35	m000000004	i000000035
36	m000000004	i000000036
37	m000000004	i000000037
38	m000000004	i000000038
39	m000000004	i000000039
40	m000000004	i000000040
41	m000000005	i000000041
42	m000000005	i000000042
43	m000000005	i000000043
44	m000000005	i000000044
45	m000000005	i000000045
46	m000000005	i000000046
47	m000000005	i000000047
48	m000000005	i000000048
49	m000000005	i000000049
50	m000000005	i000000050

CSGOMarketplace Table:

The CSGOMarketplace table stores each items Market ID and the quantity of that item on the CS:GO Marketplace.

```
CREATE TABLE CSGOMarketplace(
   MID VARCHAR(10) PRIMARY KEY NOT NULL
);
```

Functional Dependencies: MID → quantity

Sample data on next page:

	mid character varying(10)
1	m000000001
2	m000000002
3	m000000003
4	m000000004
5	m000000005
6	m000000006
7	m000000007
8	m000000008
9	m000000009
10	m000000010
11	m000000011
12	m000000012
13	m000000013
14	m000000014
15	m000000015
16	m000000016
17	m000000017
18	m000000018
19	m000000019
20	m000000020
21	m000000021
22	m000000022
23	m000000023
24	m000000024
25	m000000025
26	m000000026
27	m000000027
28	m000000028
29	m000000029
30	m000000030
31	m000000031
32	m000000032
33	m000000033
34	m000000034
35	m000000035
36	m000000036
37	m000000037
38	m000000038
39	m000000039
40	m000000040
41	m000000041
42	m000000042
43	m000000043
44	m000000044
45	m000000045
46	m000000046
47	m000000047
48	m000000048
49	m000000049
50	m000000050

OnMarketplace Table:

The OnMarketplace table is a weak entity between ItemsInInventory and CSGOMarketplace. It stores information of which user's item is on the CS:GO Marketplace.

```
CREATE TABLE OnMarketplace(
   invID VARCHAR(10),
   MID VARCHAR(10),
   FOREIGN KEY (invID) REFERENCES ItemsInInventory(invID),
   FOREIGN KEY (MID) REFERENCES CSGOMarketplace(MID),
   PRIMARY KEY (invID, MID)
);
```

Functional Dependencies: invID, MID → N/A

Sample data on next page:

	invid character varying(10)	mid character varying(10)
1	i000000001	m000000001
2	i000000002	m000000002
3	i000000003	m000000003
4	i000000004	m000000004
5	i000000005	m000000005
6	i000000006	m000000006
7	i000000007	m000000007
8	i000000008	m000000008
9	i000000009	m000000009
10	i000000010	m000000010
11	i000000011	m000000011
12	i000000012	m000000012
13	i000000013	m000000013
14	i000000014	m000000014
15	i000000015	m000000015
16	i000000016	m000000016
17	i000000017	m000000017
18	i000000018	m000000018
19	i000000019	m000000019
20	i000000020	m000000020
21	i000000021	m000000021
22	i000000022	m000000022
23	i000000023	m000000023
24	i000000024	m000000024
25	i000000025	m000000025
26	i000000026	m000000026
27	i000000027	m000000027
28	i000000028	m000000028
29	i000000029	m000000029
30	i000000030	m000000030
31	i000000031	m000000031
32	i000000032	m000000032
33	i000000033	m000000033
34	i000000040	m000000034

Check Constraints

I placed a check constraint on passwords in the user table to make sure the user's password was 8 or more characters and alphanumeric to provide password security.

```
CREATE TABLE Users(
   UID VARCHAR(10) PRIMARY KEY UNIQUE NOT NULL,
   username VARCHAR(100) NOT NULL,
   pass VARCHAR(100) NOT NULL CHECK (char_length(pass) >= 8 AND pass ~ '[[:alpha:]]'
AND pass ~ '\d'),
   email VARCHAR(100) NOT NULL
);
```

Views

userWeaponsOnMarketplace View:

Displays important information on weapons inside the marketplace.

```
CREATE VIEW userWeaponsOnMarketplace
   AS
   SELECT u.username, w.wName, w.exterior, w.category, s.skinName, sow.wPriceUSD
   FROM Users u,
       UserItems ui,
       ItemsInInventory iIni,
       OnMarketplace omp,
       CSGOMarketplace mp,
       SkinsOnWeapon sow,
       Weapons w,
       Skins s
   WHERE u.UID = ui.UID
        AND ui.invID = iIni.invID
        AND iIni.invID = omp.invID
        AND omp.MID = mp.MID
        AND iIni.invID = sow.invID
        AND sow.WID = w.WID
        AND sow.skinID = s.skinID
        ORDER BY iIni.invID;
```

	username character varying(100)	wname character varying(100)	exterior character varying(100)	category character varying(20)	skinname character varying(100)	wprice numeric(5,2)
1	wakefm1	AK-47	Field-Tested	Normal	Predator	0.71
2	wakefm1	AK-47	Factory New	Somvenir	Safari Mesh	138.00
3	wakefm1	AK-47	Minimal Wear	StatTrak	Case Hardened	190.00
4	wakefm1	M4A4	Factory New	StatTrak	Griffin	23.45
5	wakefm1	M4A1-S	Factory New	Normal	Knight	274.44
6	wakefm1	Aug	Well-Worn	Normal	Ricochet	0.20
7	wakefm1	Aug	Factory New	StatTrak	Ricochet	4.14
8	wakefm1	AWP	Feild-Tested	Normal	Man-o-war	11.75
9	wakefm1	AWP	Feild-Tested	Normal	Asiimov	40.82
10	wakefm1	AWP	Feild-Tested	Normal	Asiimov	40.82
11	glassw4r3	AWP	Feild-Tested	Normal	Man-o-war	11.75
12	glassw4r3	AWP	Battle-Scarred	StatTrak	Asiimov	70.50
13	glassw4r3	p2000	Battle-Scarred	StatTrak	Red FragCam	1.74
14	glassw4r3	p2000	Minimal Wear	Normal	Scorpion	6.15
15	glassw4r3	Bowie Knife	Field-Tested	*	Crimson Web	124.20

userKeysOnMarketplace View:
Displays all keys on the marketplace and who they belong too.

```
CREATE VIEW userKeysOnMarketplace
   AS
   SELECT u.username, kName, kPriceUSD
   FROM Users u,
      UserItems ui,
      ItemsInInventory iIni,
      OnMarketplace omp,
      CSGOMarketplace mp,
      Keys k
   WHERE u.UID = ui.UID
        AND ui.invID = iIni.invID
        AND iIni.invID = omp.invID
        AND omp.MID = mp.MID
        AND iIni.invID = k.invID
        ORDER BY iIni.invID;
```

	username character varying(100)	kname character varying(100)	kprice character varying(100)
1	glassw4r3	Chroma 2 Case Key	Chroma 2
2	glassw4r3	Operation Vangamrd Case Key	Operation Vangamrd
3	glassw4r3	Revolver Case Key	Revolver
4	glassw4r3	eSports Key	eSports
5	glassw4r3	Chroma Case Key	Chroma
6	co1n4ge	Chroma Case Key	Chroma
7	co1n4ge	Hmntsman Case Key	Hmntsman
8	co1n4ge	Operation Wildfire Case Key	Operation Wildfire
9	co1n4ge	Chroma 2 Case Key	Chroma 2
10	co1n4ge	eSports Key	eSports

userStickersOnMarketplace View:

Displays all stickers on the marketplace and who they belong too.

```
CREATE VIEW userStickersOnMarketplace
   AS
   SELECT u.username, sName, proPlayer, sPriceUSD
   FROM Users u,
       UserItems ui,
       ItemsInInventory iIni,
       OnMarketplace omp,
       CSGOMarketplace mp,
       Stickers s
   WHERE u.UID = ui.UID
        AND ui.invID = iIni.invID
        AND iIni.invID = omp.invID
        AND omp.MID = mp.MID
        AND iIni.invID = s.invID
        ORDER BY iIni.invID;
```

	username character varying(100)	sname character varying(100)	proplayer character varying(100)	sprice numeric(5,2)
1	co1n4ge	Terrorized		12.76
2	co1n4ge	PENTA Sports DreamHack 2014		2.43
3	co1n4ge	mnicorn		1.78
4	co1n4ge	The Sammrai		0.77
5	co1n4ge	flamie Cologne 2015	flamie	0.38

userCasesOnMarketplace View:

Displays all cases on the marketplace and who they belong too.

```
CREATE VIEW userCasesOnMarketplace
   AS
   SELECT u.username, cName, cSeries, cPriceUSD
   FROM Users u,
       UserItems ui,
       ItemsInInventory iIni,
       OnMarketplace omp,
       CSGOMarketplace mp,
       OpCase c
   WHERE u.UID = ui.UID
        AND ui.invID = iIni.invID
        AND iIni.invID = omp.invID
        AND omp.MID = mp.MID
        AND iIni.invID = c.invID
        ORDER BY iIni.invID;
```

	username character varying(100)	cname character varying(100)	cseries integer	cprice character varying(100)
1	fr0nted	Shadow Case	80	0.05
2	fr0nted	Revolver Case	111	0.05
3	fr0nted	Chroma Case	38	0.12
4	fr0nted	Chroma Case	38	0.12

Reports and their Queries

Query to select only AK-47 and what user owns it:

```
SELECT iIni.invID, u.username, w.wName, s.skinName
FROM Users u,
    UserItems ui,
    ItemsInInventory iIni,
    SkinsOnWeapon sow,
    Weapons w,
    Skins s
WHERE u.UID = ui.UID
     AND ui.invID = iIni.invID
     AND iIni.invID = sow.invID
     AND sow.WID = w.WID
     AND sow.skinID = s.skinID
     AND w.wName = 'AK-47'
```

	invid character varying(10)	username character varying(100)	wname character varying(100)	skinname character varying(100)
1	i000000001	wakefm1	AK-47	Predator
2	i000000002	wakefm1	AK-47	Safari Mesh
3	i000000003	wakefm1	AK-47	Case Hardened

Query to select items not on the marketplace:

```
SELECT iIni.invID
FROM ItemsInInventory iIni
WHERE iIni.invID not in ( SELECT invID
                          FROM OnMarketplace);
```

	invid character varying(10)
1	i000000034
2	i000000035
3	i000000036
4	i000000037
5	i000000038
6	i000000039
7	i000000041
8	i000000042
9	i000000043
10	i000000044
11	i000000045
12	i000000046
13	i000000047
14	i000000048
15	i000000049
16	i000000050

Stored Procedures/Triggers

deletedWeapon Function/Trigger:

The logDeletedWeapon trigger is supposed to execute the deletedWeapon function when an item is deleted from OnMarketplace. The deletedWeapon function is supposed to print that something was deleted and what was deleted.

```
CREATE OR REPLACE FUNCTION deletedWeapon(VARCHAR) RETURNS integer AS $$
<< outerblock >>
DECLARE
   invID VARCHAR := $1;
BEGIN
   RAISE NOTICE '--- Deleted --- %', invID;
END;
$$ LANGUAGE plpgsql;

EXECUTE deletedWeapon('u000000002');

Create Trigger logDeletedWeapon before delete on OnMarketplace
   EXEC FUNCTION deletedWeapon(UID);
```

Security

To implement security, I created two separate schemas:

CREATE SCHEMA allTables AUTHORIZATION admin, public, users;

With those schemas, I allowed different access to three different types of users:

Admin: Can modify, update and maintain database.

```
CREATE ROLE admin
REVOKE ALL ON ALL TABLES IN SCHEMA allTables
GRANT SELECT, INSERT, UPDATE, ALTER
ON ALL TABLES IN SCHEMA allTables
TO admin;
```

Public: Can see database and perform queries.

```
CREATE ROLE public
REVOKE ALL ON ALL TABLES IN SCHEMA allTables
GRANT SELECT
ON ALL TABLES IN SCHEMA allTables
TO public;
```

Users: Can perform queries on everying, but can only insert into things inside the accessMarket schema.

```
CREATE ROLE accountOwners
REVOKE ALL ON ALL TABLES IN SCHEMA allTables
GRANT SELECT
ON ALL TABLES IN SCHEMA allTables
TO accountOwners;

GRANT SELECT, INSERT, UPDATE ON OnMarketplace To
accountOwners;
```

Important Notes

The following are suggestion and/or requirements for implementation:

- Always create the non-weak entities first, as there will be errors if you apply them in another order. The order of the tables inside this documentation will suffice.
- When entering data be careful not to forget any pieces of information because there is very few spots that can be null.

Known Problems

While create the database, there are a few known problems with some of its traits. For example, neither the stored procedure, nor, trigger actually function properly. They will actually return an error and end abruptly. Other than that, the rest of the SQL commands will execute correctly.

Future Enhancements

In the future, we would like to increase the amount of games inside our marketplace because CS:GO is not the only game with an economy running on steam. Furthermore, there could be a quantity value in market to show how many of a specific item there is available on the market. This would allow easy access to useful information that could be implemented by adding a new column or using a view. In the future, there may need to be a Boolean column showing if an item is sold or not. After about two days of the item being sold, the item should be moved to another table where the item can be used as information to show how the prices of items change overtime.