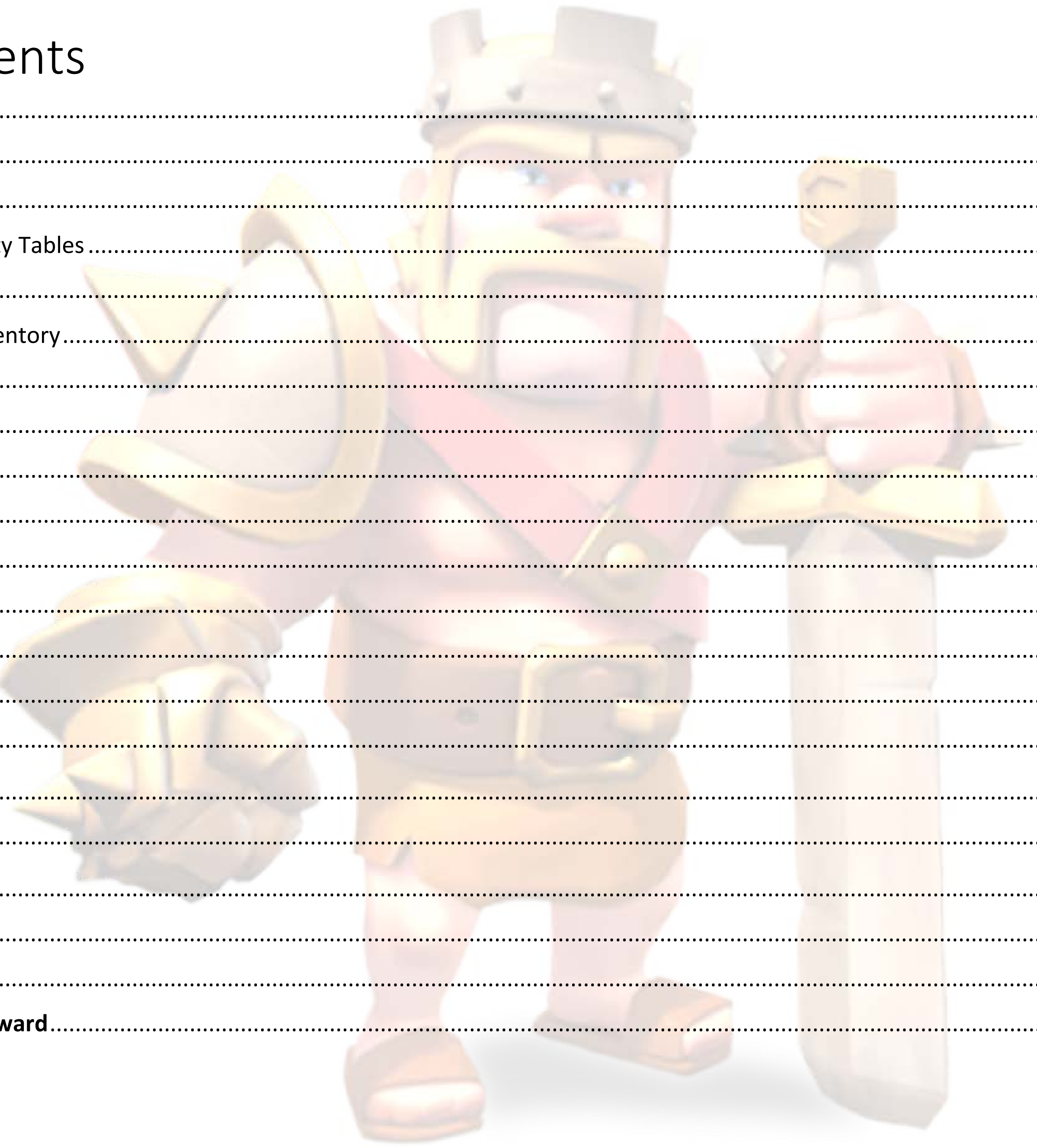
DATA BASE DESIGN PROJECT – SPRING 2015

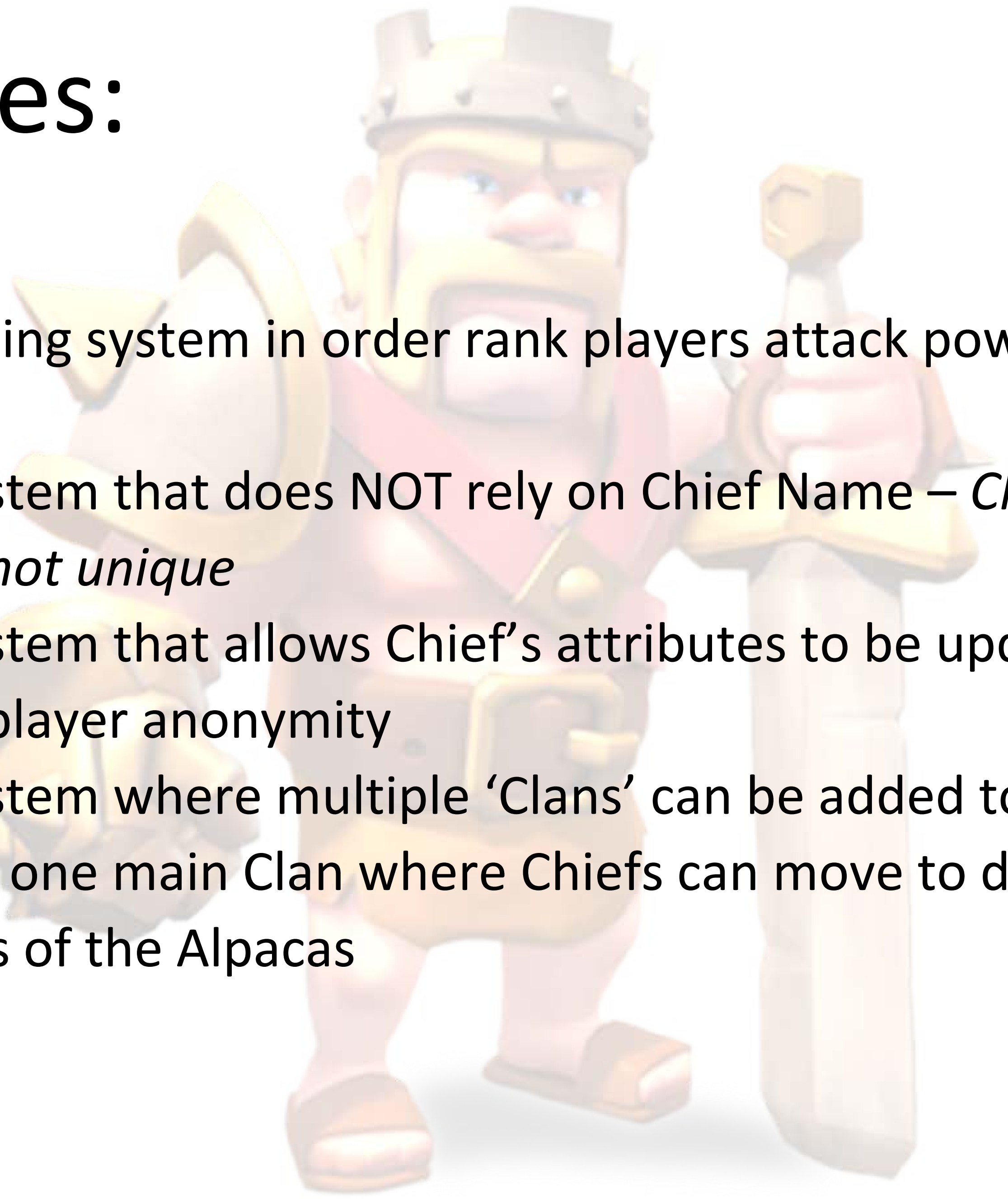
Clash of Clans: The Database

By Joe Strauss

Table of Contents

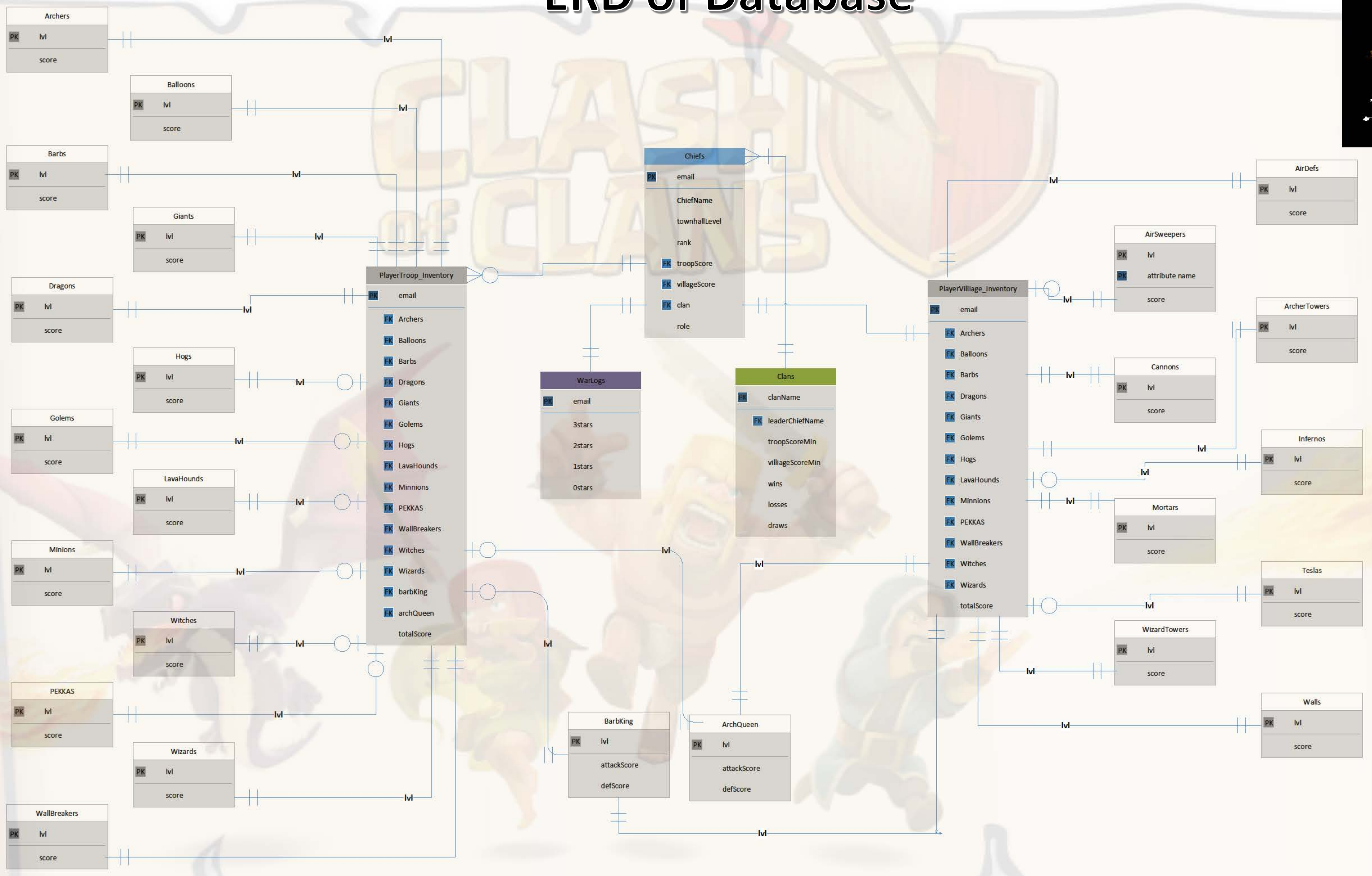


Objectives:



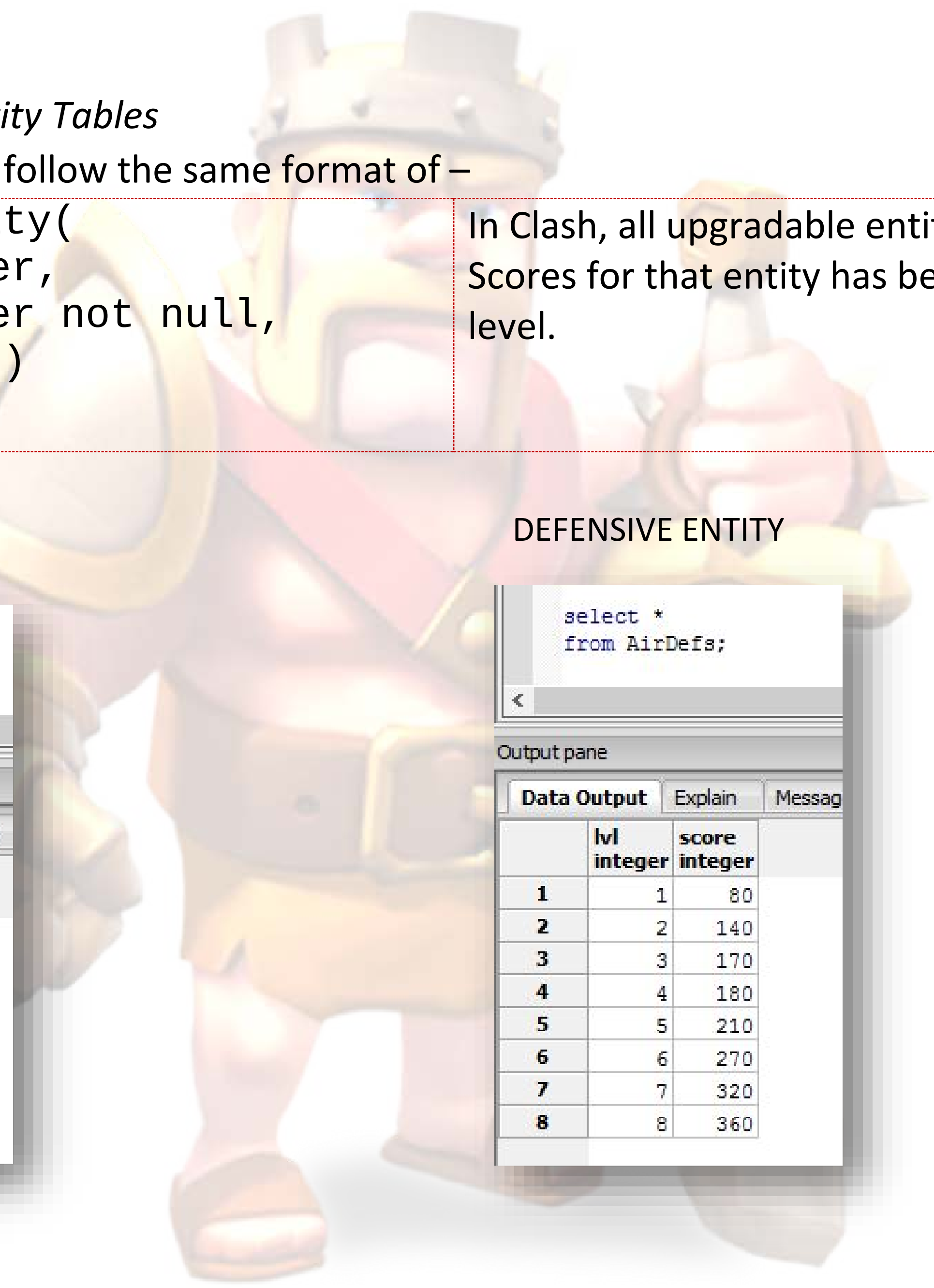
- Create a rating system in order rank players attack power & base strength
- Create a system that does NOT rely on Chief Name – *Clash Chief names are not unique*
- Create a system that allows Chief’s attributes to be updated while protecting player anonymity
- Create a system where multiple ‘Clans’ can be added to the umbrella of one main Clan where Chiefs can move to different subset clans of the Alpacas

ERD of Database



Note: Although there are multiple defensive structures per villiage, a Chief's rating will be based on the lowest level for that structure. For example, if a chief as two lvl-3 Wiz Towers and one lvl-4, then their rating will be lvl-3

Note: Walls will be based on majority. For example, if a chief has seventy lvl-7 walls and one hundred lvl-8 walls, then their wall rating will be lvl-9



Tables:

Offensive & Defensive Entity Tables

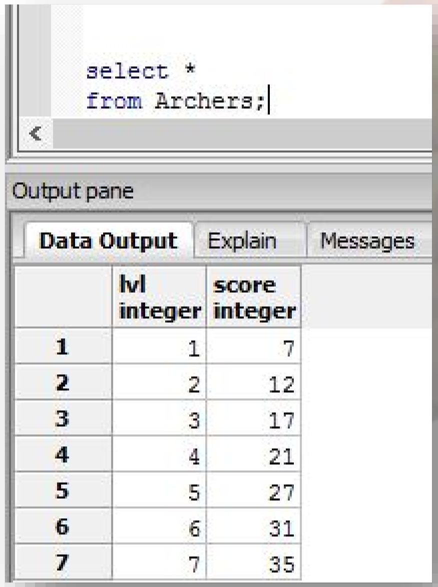
All of these tables will follow the same format of –

```
CREATE TABLE Entity(
  lvl      integer,
  score    integer not null,
  primary key(lvl)
);
```

In Clash, all upgradable entities start at level 1. Scores for that entity has been created for each level.

EXAMPLE:

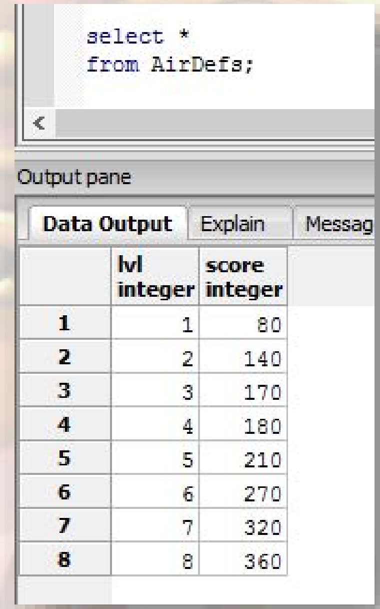
OFFENSIVE ENTITY



```
select *
from Archers;
```

	lvl integer	score integer
1	1	7
2	2	12
3	3	17
4	4	21
5	5	27
6	6	31
7	7	35

DEFENSIVE ENTITY



```
select *
from AirDefs;
```

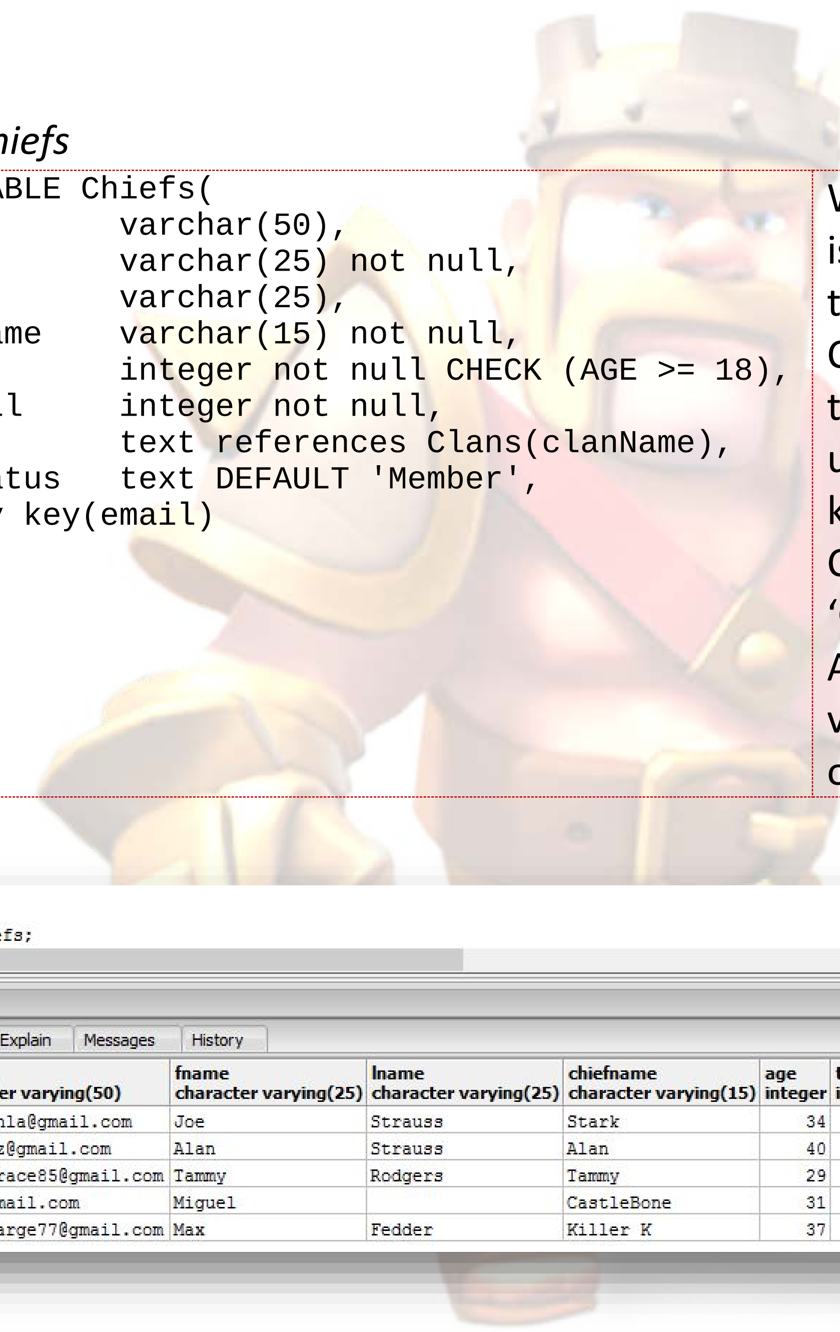
	lvl integer	score integer
1	1	80
2	2	140
3	3	170
4	4	180
5	5	210
6	6	270
7	7	320
8	8	360

Tables:

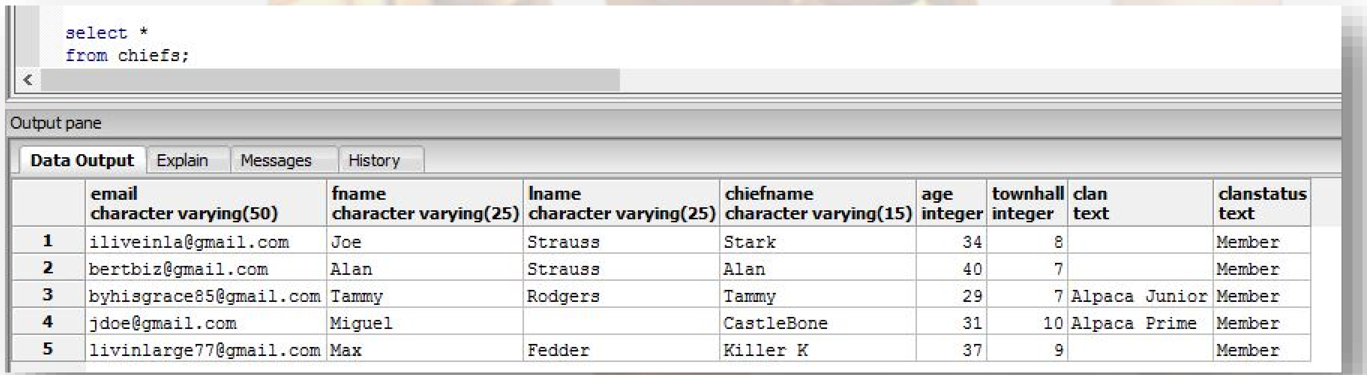
Alapaca Chiefs

```
CREATE TABLE Chiefs(
  email         varchar(50),
  fName         varchar(25) not null,
  lName         varchar(25),
  chiefName     varchar(15) not null,
  age           integer not null CHECK (AGE >= 18),
  townHall      integer not null,
  clan          text references Clans(clanName),
  clanStatus    text DEFAULT 'Member',
  primary key(email)
);
```

We can see that 'lName' may be NULL. This is because some who play Clash choose not to have their full identity known. Also, Clash of Clans does not require Chief names to be unique. This is ok because we are using a valid email address as the primary key. This works well because every 'Saved' Clash account must be linked to a valid 'Gmail Account' through Google+ or a valid Apple ID. We also have a constraint to verify that all players are at least 18 years of age.



Example:

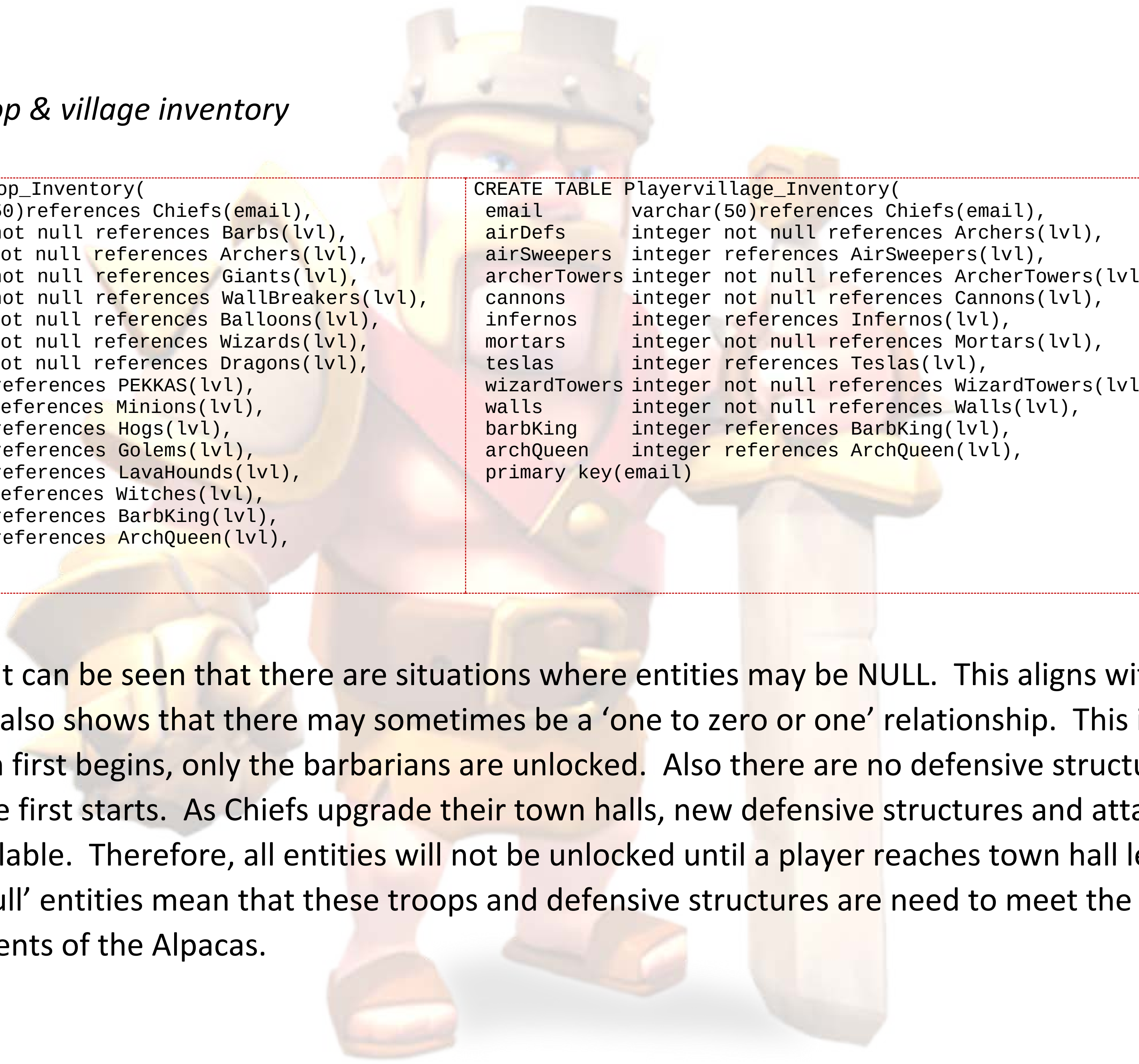


```
select *
from chiefs;
```

```
   email                   fname                   lname                   chiefname               age      townhall  clan           clanstatus
   character varying(50)   character varying(25)   character varying(25)   character varying(15)   integer  integer   text           text
1  iliveinla@gmail.com     Joe                     Strauss                 Stark                   34       8                        Member
2  bertbiz@gmail.com       Alan                    Strauss                 Alan                    40       7                        Member
3  byhisgrace85@gmail.com  Tammy                   Rodgers                 Tammy                   29       7         Alpaca Junior  Member
4  jdoe@gmail.com          Miguel                                          CastleBone              31       10        Alpaca Prime   Member
5  livinlarge77@gmail.com  Max                     Fedder                  Killer K                37       9                        Member
```

Tables:

Player's troop & village inventory



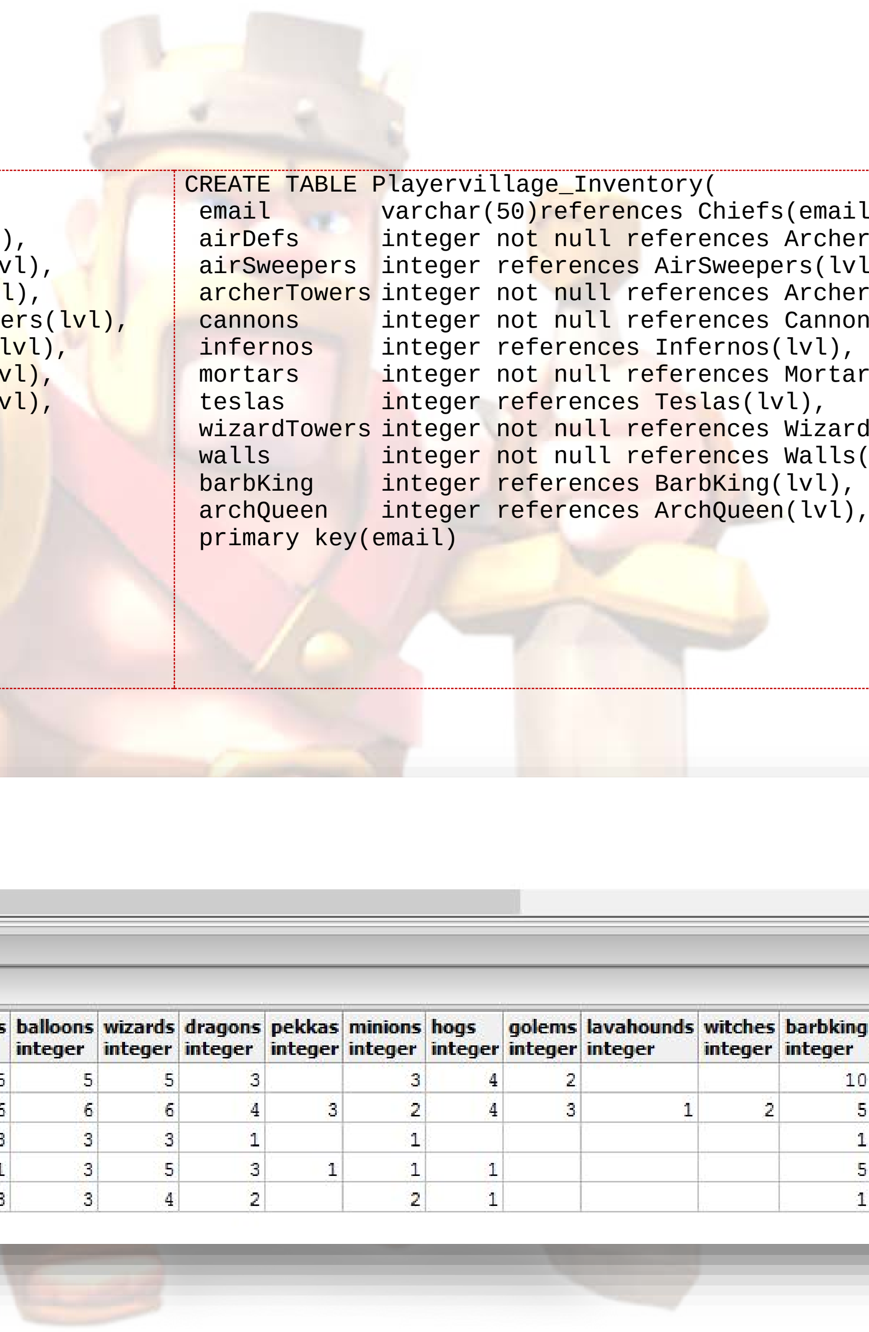
```
CREATE TABLE PlayerTroop_Inventory(
 email        varchar(50)references Chiefs(email),
 barbs        integer not null references Barbs(lvl),
 archers      integer not null references Archers(lvl),
 giants       integer not null references Giants(lvl),
 wallBreakers integer not null references WallBreakers(lvl),
 balloons     integer not null references Balloons(lvl),
 wizards      integer not null references Wizards(lvl),
 dragons      integer not null references Dragons(lvl),
 pekkas       integer references PEKKAS(lvl),
 minions      integer references Minions(lvl),
 hogs         integer references Hogs(lvl),
 golems       integer references Golems(lvl),
 lavaHounds   integer references LavaHounds(lvl),
 witches      integer references Witches(lvl),
 barbKing     integer references BarbKing(lvl),
 archQueen    integer references ArchQueen(lvl),
 primary key(email)
);
```

```
CREATE TABLE Playervillage_Inventory(
 email        varchar(50)references Chiefs(email),
 airDefs      integer not null references Archers(lvl),
 airSweepers  integer references AirSweepers(lvl),
 archerTowers integer not null references ArcherTowers(lvl),
 cannons      integer not null references Cannons(lvl),
 infernos     integer references Infernos(lvl),
 mortars      integer not null references Mortars(lvl),
 teslas       integer references Teslas(lvl),
 wizardTowers integer not null references WizardTowers(lvl),
 walls        integer not null references Walls(lvl),
 barbKing     integer references BarbKing(lvl),
 archQueen    integer references ArchQueen(lvl),
 primary key(email)
```

In both inventories it can be seen that there are situations where entities may be NULL. This aligns with the ERD diagram which also shows that there may sometimes be a 'one to zero or one' relationship. This is because when Clash first begins, only the barbarians are unlocked. Also there are no defensive structures built when the game first starts. As Chiefs upgrade their town halls, new defensive structures and attack troops become available. Therefore, all entities will not be unlocked until a player reaches town hall level 10. However, the 'no null' entities mean that these troops and defensive structures are need to meet the minimum requirements of the Alpacas.

Tables:

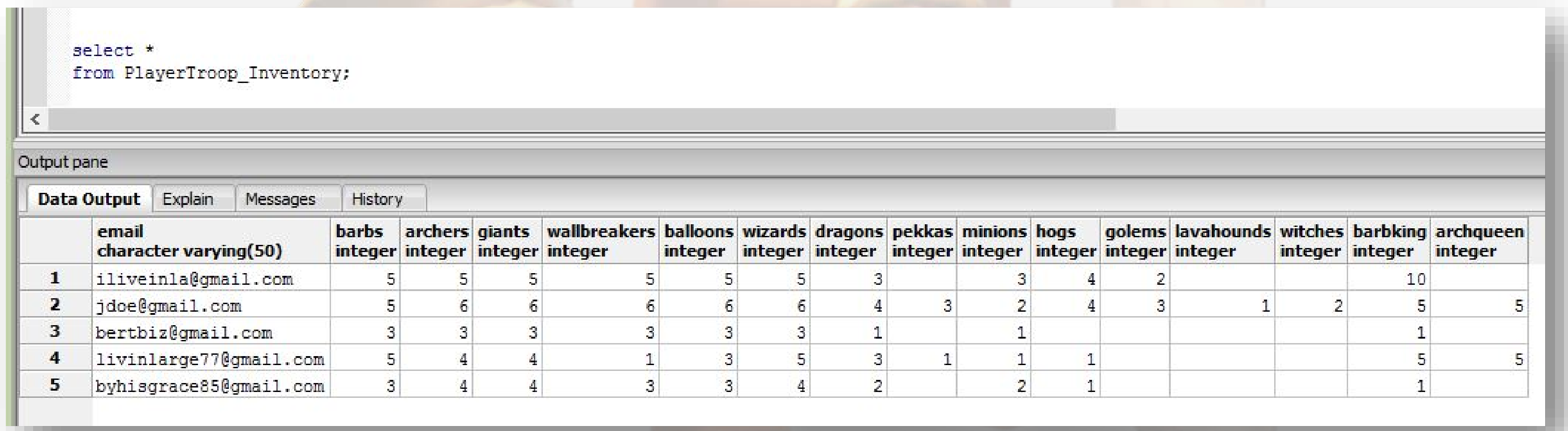
Player's troop & village inventory



```
CREATE TABLE PlayerTroop_Inventory(
 email        varchar(50)references Chiefs(email),
 barbs        integer not null references Barbs(lvl),
 archers      integer not null references Archers(lvl),
 giants       integer not null references Giants(lvl),
 wallBreakers integer not null references WallBreakers(lvl),
 balloons     integer not null references Balloons(lvl),
 wizards      integer not null references Wizards(lvl),
 dragons      integer not null references Dragons(lvl),
 pekkas       integer references PEKKAS(lvl),
 minions      integer references Minions(lvl),
 hogs         integer references Hogs(lvl),
 golems       integer references Golems(lvl),
 lavaHounds   integer references LavaHounds(lvl),
 witches      integer references Witches(lvl),
 barbKing     integer references BarbKing(lvl),
 archQueen    integer references ArchQueen(lvl),
 primary key(email)
);
```

```
CREATE TABLE Playervillage_Inventory(
 email        varchar(50)references Chiefs(email),
 airDefs      integer not null references Archers(lvl),
 airSweepers  integer references AirSweepers(lvl),
 archerTowers integer not null references ArcherTowers(lvl),
 cannons      integer not null references Cannons(lvl),
 infernos     integer references Infernos(lvl),
 mortars      integer not null references Mortars(lvl),
 teslas       integer references Teslas(lvl),
 wizardTowers integer not null references WizardTowers(lvl),
 walls        integer not null references Walls(lvl),
 barbKing     integer references BarbKing(lvl),
 archQueen    integer references ArchQueen(lvl),
 primary key(email)
```

Example:



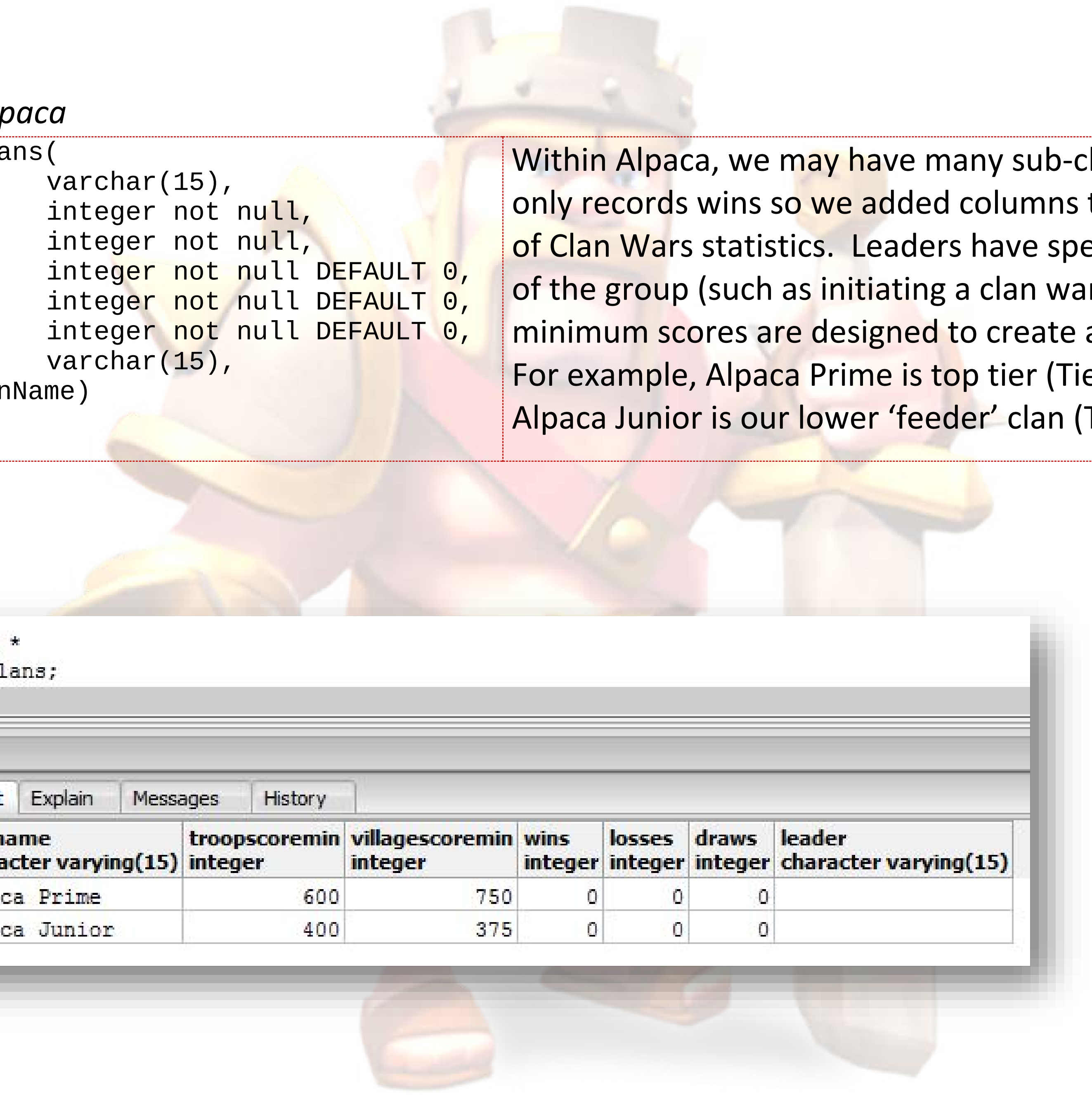
```
select *
from PlayerTroop_Inventory;
```

Output pane

	email character varying(50)	barbs integer	archers integer	giants integer	wallbreakers integer	balloons integer	wizards integer	dragons integer	pekkas integer	minions integer	hogs integer	golems integer	lavahounds integer	witches integer	barbking integer	archqueen integer
1	iliveinla@gmail.com	5	5	5	5	5	5	3		3	4	2			10	
2	jdoe@gmail.com	5	6	6	6	6	6	4	3	2	4	3	1	2	5	5
3	bertbiz@gmail.com	3	3	3	3	3	3	1		1					1	
4	livinlarge77@gmail.com	5	4	4	1	3	5	3	1	1	1				5	5
5	byhisgrace85@gmail.com	3	4	4	3	3	4	2		2	1				1	

Tables:

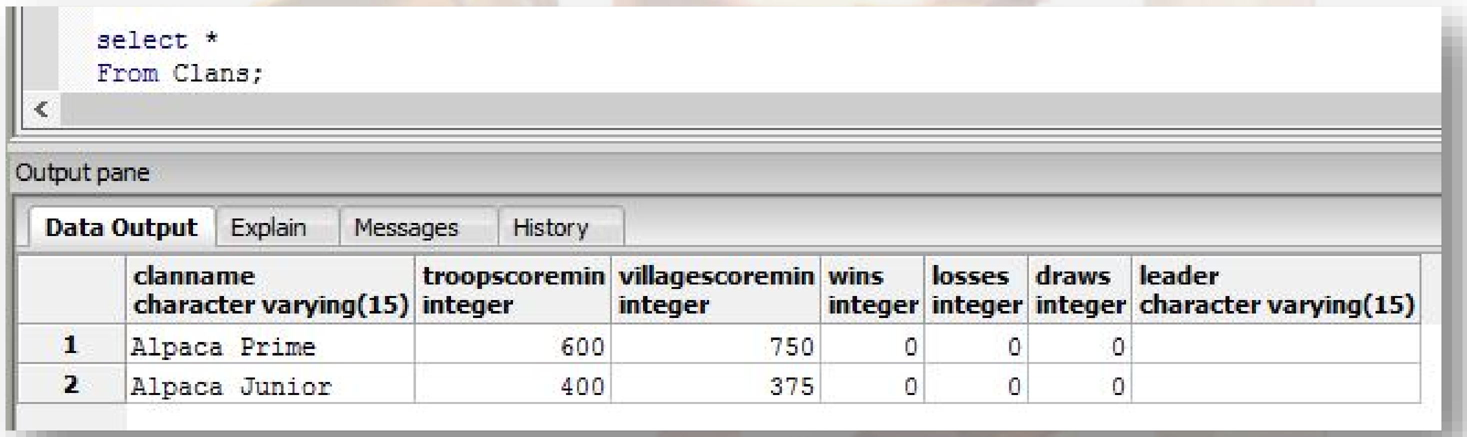
Clans of Alpaca



```
CREATE TABLE Clans(
clanName          varchar(15),
troopScoreMin     integer not null,
villageScoreMin   integer not null,
wins              integer not null DEFAULT 0,
losses            integer not null DEFAULT 0,
draws             integer not null DEFAULT 0,
leader            varchar(15),
primary key(clanName)
);
```

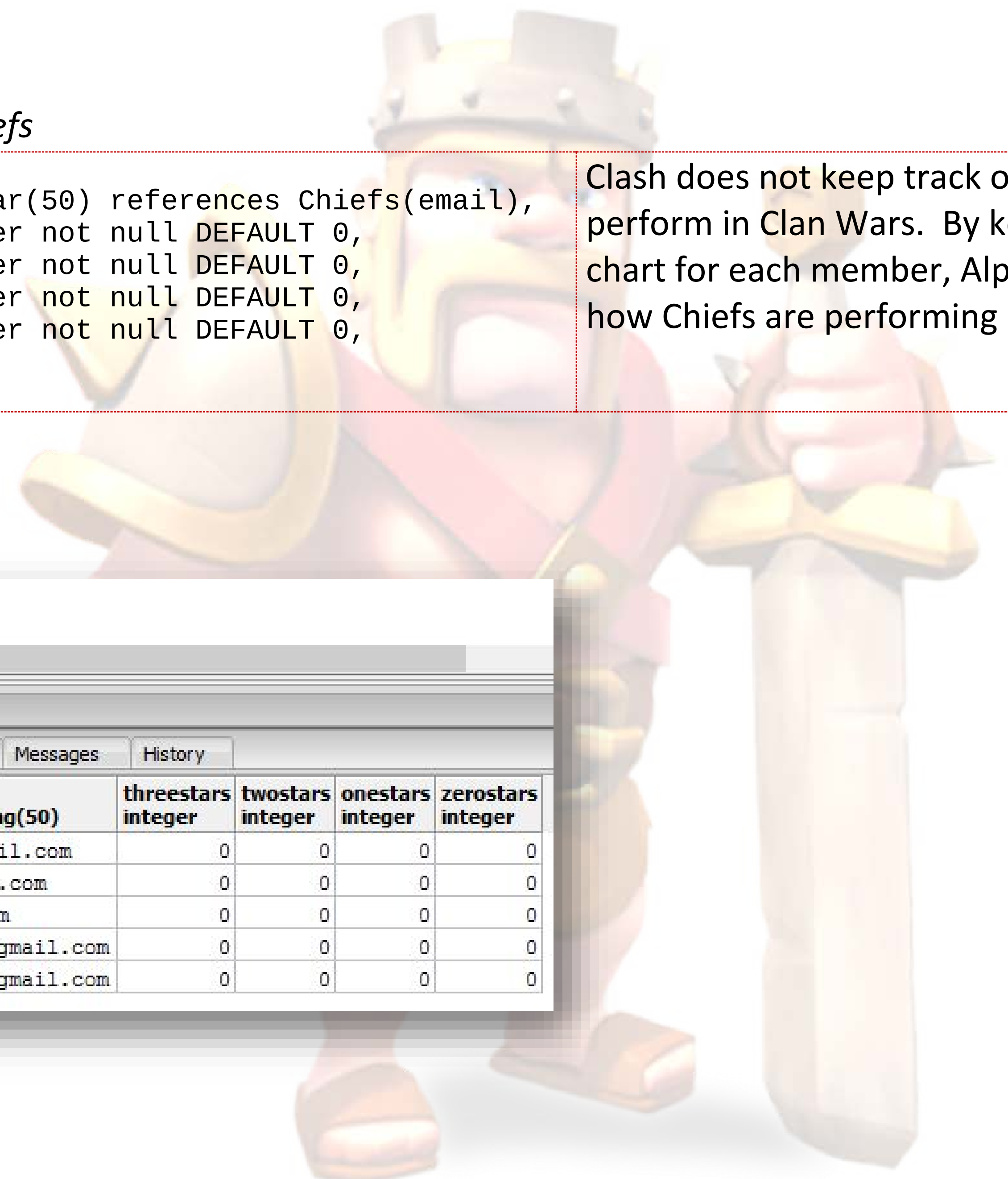
Within Alpaca, we may have many sub-clans. Clash only records wins so we added columns to keep track of Clan Wars statistics. Leaders have special functions of the group (such as initiating a clan war). The minimum scores are designed to create a Tier of clans. For example, Alpaca Prime is top tier (Tier 1) whereas Alpaca Junior is our lower 'feeder' clan (Tier 2).

Example:



```
select *
From Clans;
```

	clanname character varying(15)	troopscoremin integer	villagescoremin integer	wins integer	losses integer	draws integer	leader character varying(15)
1	Alpaca Prime	600	750	0	0	0	
2	Alpaca Junior	400	375	0	0	0	



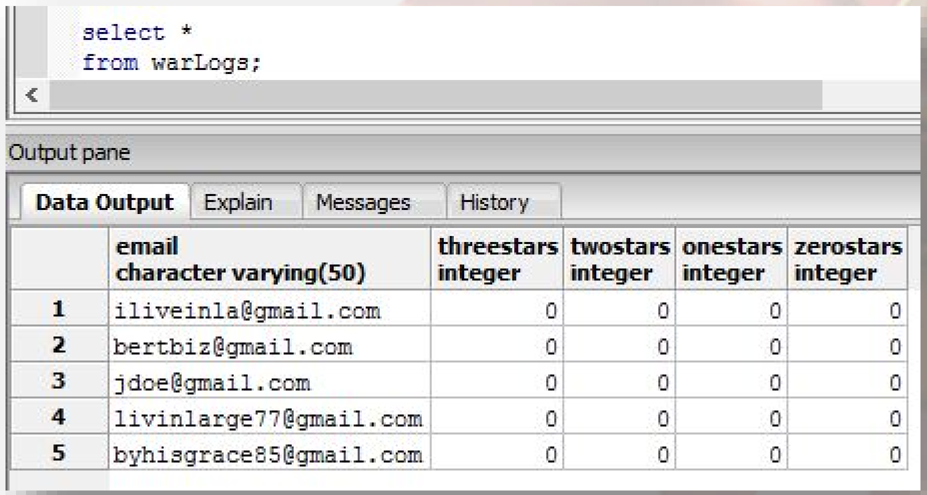
Tables:

War logs for Chiefs

```
CREATE TABLE WarLogs(
 email          varchar(50) references Chiefs(email),
 threeStars     integer not null DEFAULT 0,
 twoStars       integer not null DEFAULT 0,
 oneStars       integer not null DEFAULT 0,
 zeroStars      integer not null DEFAULT 0,
 primary key(email)
);
```

Clash does not keep track of how players perform in Clan Wars. By keeping a war log chart for each member, Alpacas can easily track how Chiefs are performing in wars.

Example:



```
select *
from warLogs;
```

	email character varying(50)	threestars integer	twostars integer	onestars integer	zerostars integer
1	iliveinla@gmail.com	0	0	0	0
2	bertbiz@gmail.com	0	0	0	0
3	jdoe@gmail.com	0	0	0	0
4	livinlarge77@gmail.com	0	0	0	0
5	byhisgrace85@gmail.com	0	0	0	0

Tables:

Functional Dependencies

Archers: lvl --> score
Balloons: lvl --> score
Barbs: lvl --> score
Giants: lvl --> score
Dragons: lvl --> score
Hogs: lvl --> score
Golems: lvl --> score
LavaHounds: lvl --> score
Minions: lvl --> score
WallBreakers: lvl --> score
Witches: lvl --> score
Wizards: lvl --> score
PEKKAS: lvl --> score
BarbKing: lvl --> score
ArchQueen: lvl --> score

AirDefs: lvl --> score
AirSweepers: lvl --> score
ArcherTowers: lvl --> score
Cannons: lvl --> score
Infernos: lvl --> score
Mortars: lvl --> score
Teslas: lvl --> score
WizardTowers: lvl --> score
Walls: lvl --> score

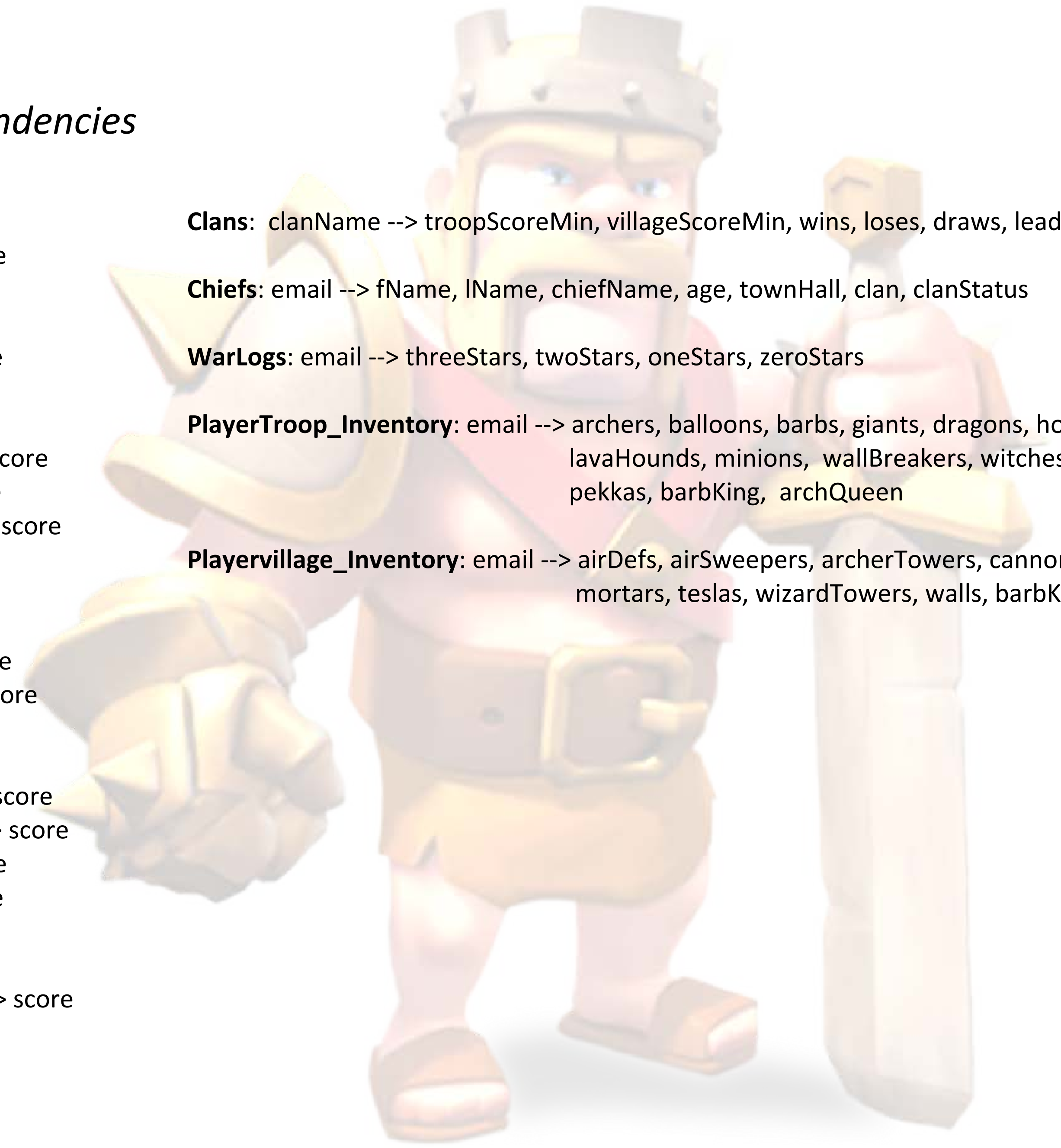
Clans: clanName --> troopScoreMin, villageScoreMin, wins, loses, draws, leader

Chiefs: email --> fName, lName, chiefName, age, townHall, clan, clanStatus

WarLogs: email --> threeStars, twoStars, oneStars, zeroStars

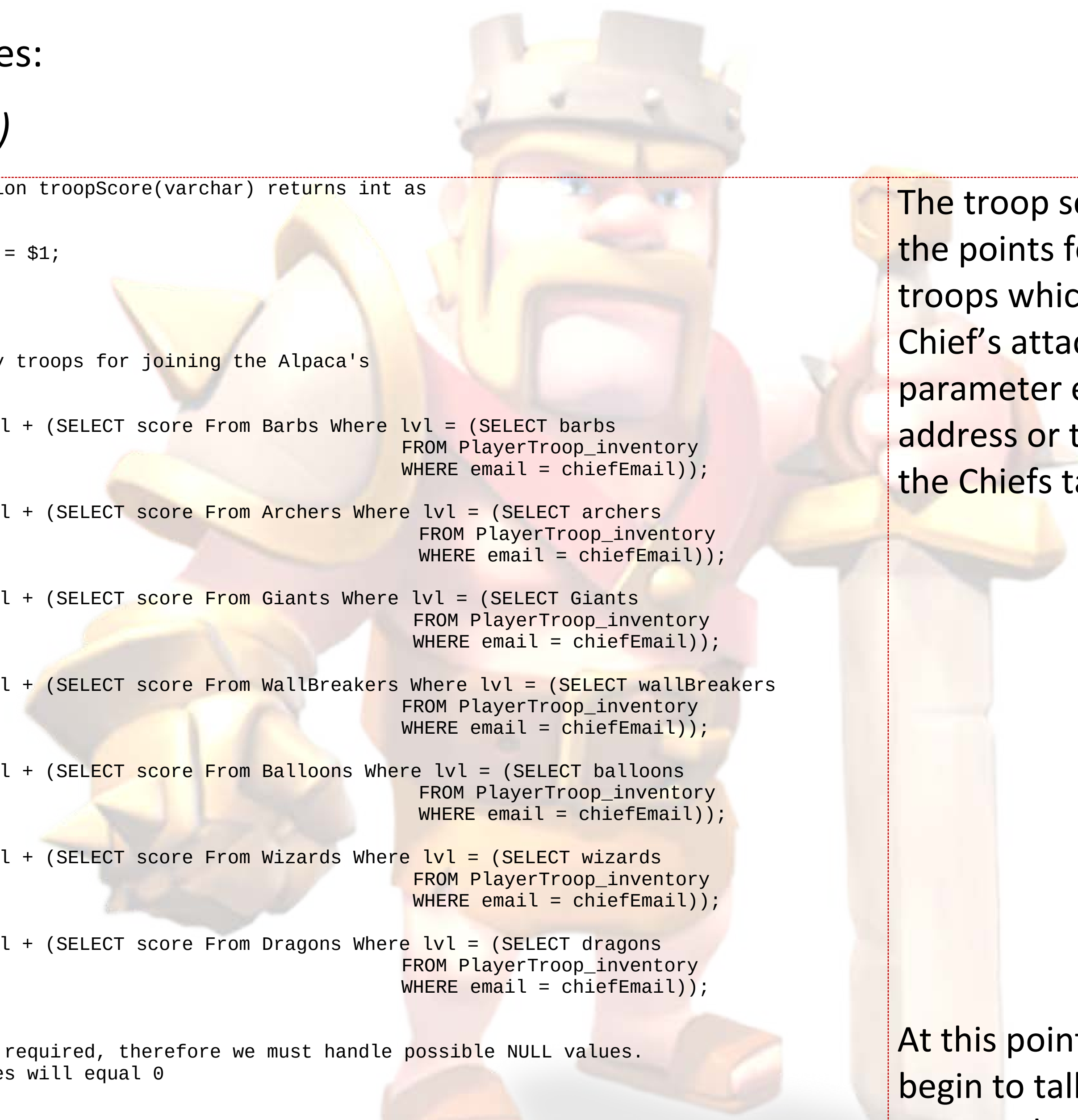
PlayerTroop_Inventory: email --> archers, balloons, barbs, giants, dragons, hogs, golems, lavaHounds, minions, wallBreakers, witches, wizards pekkas, barbKing, archQueen

Playervillage_Inventory: email --> airDefs, airSweepers, archerTowers, cannons, infernos, mortars, teslas, wizardTowers, walls, barbKing, archQueen



Stored Procedures:

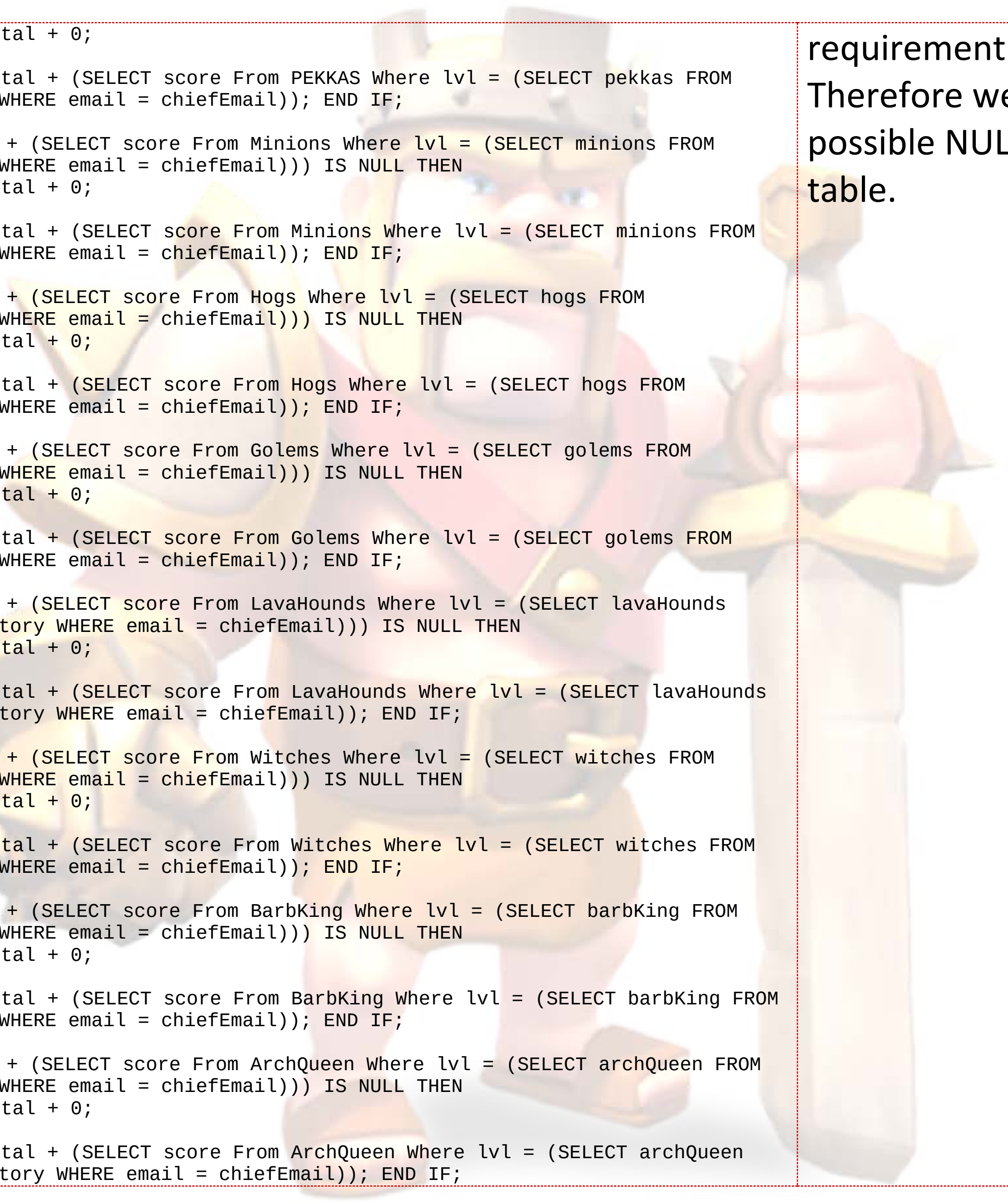
troopScore()



```
create or replace function troopScore(varchar) returns int as
$$
declare
    chiefEmail varchar := $1;
    total int = 0;

begin

  -- These are manditory troops for joining the Alpaca's


            total = total + (SELECT score From Barbs Where lvl = (SELECT barbs
                                                           FROM PlayerTroop_inventory
                                                           WHERE email = chiefEmail));

            total = total + (SELECT score From Archers Where lvl = (SELECT archers
                                                             FROM PlayerTroop_inventory
                                                             WHERE email = chiefEmail));

            total = total + (SELECT score From Giants Where lvl = (SELECT Giants
                                                            FROM PlayerTroop_inventory
                                                            WHERE email = chiefEmail));

            total = total + (SELECT score From WallBreakers Where lvl = (SELECT wallBreakers
                                                           FROM PlayerTroop_inventory
                                                           WHERE email = chiefEmail));

            total = total + (SELECT score From Balloons Where lvl = (SELECT balloons
                                                             FROM PlayerTroop_inventory
                                                             WHERE email = chiefEmail));

            total = total + (SELECT score From Wizards Where lvl = (SELECT wizards
                                                            FROM PlayerTroop_inventory
                                                            WHERE email = chiefEmail));

            total = total + (SELECT score From Dragons Where lvl = (SELECT dragons
                                                           FROM PlayerTroop_inventory
                                                           WHERE email = chiefEmail));


       -- These are not required, therefore we must handle possible NULL values.
       -- All NULL values will equal 0


               IF (total = total + (SELECT score From PEKKAS Where lvl = (SELECT pekkas FROM
           PlayerTroop_inventory WHERE email = chiefEmail))) IS NULL THEN
```

The troop score function tallies the points for all the levels of troops which are within the Chief's attack inventory. The parameter entered is the email address or the primary key of the Chiefs table.

At this point in the query, we begin to tally troops that are greater than the minimum



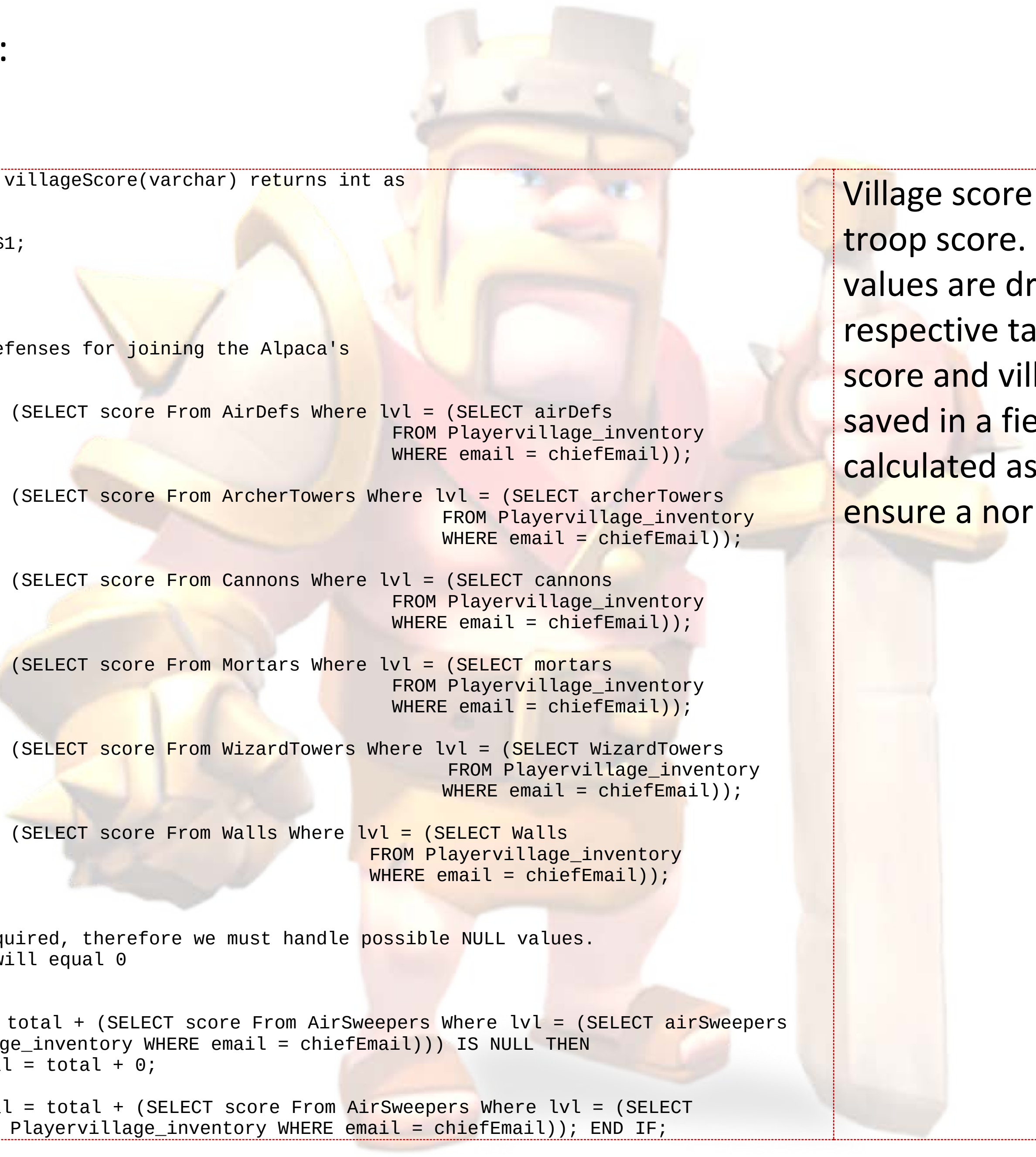
```
            total = total + 0;
    ELSE
            total = total + (SELECT score From PEKKAS Where lvl = (SELECT pekkas FROM
PlayerTroop_inventory WHERE email = chiefEmail)); END IF;

    IF (total = total + (SELECT score From Minions Where lvl = (SELECT minions FROM
PlayerTroop_inventory WHERE email = chiefEmail))) IS NULL THEN
            total = total + 0;
    ELSE
            total = total + (SELECT score From Minions Where lvl = (SELECT minions FROM
PlayerTroop_inventory WHERE email = chiefEmail)); END IF;

    IF (total = total + (SELECT score From Hogs Where lvl = (SELECT hogs FROM
PlayerTroop_inventory WHERE email = chiefEmail))) IS NULL THEN
            total = total + 0;
    ELSE
            total = total + (SELECT score From Hogs Where lvl = (SELECT hogs FROM
PlayerTroop_inventory WHERE email = chiefEmail)); END IF;

    IF (total = total + (SELECT score From Golems Where lvl = (SELECT golems FROM
PlayerTroop_inventory WHERE email = chiefEmail))) IS NULL THEN
            total = total + 0;
    ELSE
            total = total + (SELECT score From Golems Where lvl = (SELECT golems FROM
PlayerTroop_inventory WHERE email = chiefEmail)); END IF;

    IF (total = total + (SELECT score From LavaHounds Where lvl = (SELECT lavaHounds
FROM PlayerTroop_inventory WHERE email = chiefEmail))) IS NULL THEN
            total = total + 0;
    ELSE
            total = total + (SELECT score From LavaHounds Where lvl = (SELECT lavaHounds
FROM PlayerTroop_inventory WHERE email = chiefEmail)); END IF;

    IF (total = total + (SELECT score From Witches Where lvl = (SELECT witches FROM
PlayerTroop_inventory WHERE email = chiefEmail))) IS NULL THEN
            total = total + 0;
    ELSE
            total = total + (SELECT score From Witches Where lvl = (SELECT witches FROM
PlayerTroop_inventory WHERE email = chiefEmail)); END IF;

    IF (total = total + (SELECT score From BarbKing Where lvl = (SELECT barbKing FROM
PlayerTroop_inventory WHERE email = chiefEmail))) IS NULL THEN
            total = total + 0;
    ELSE
            total = total + (SELECT score From BarbKing Where lvl = (SELECT barbKing FROM
PlayerTroop_inventory WHERE email = chiefEmail)); END IF;

    IF (total = total + (SELECT score From ArchQueen Where lvl = (SELECT archQueen FROM
PlayerTroop_inventory WHERE email = chiefEmail))) IS NULL THEN
            total = total + 0;
    ELSE
            total = total + (SELECT score From ArchQueen Where lvl = (SELECT archQueen
FROM PlayerTroop_inventory WHERE email = chiefEmail)); END IF;
```

requirement to join the clan. Therefore we must account for possible NULL values within the table.

```
          RETURN total;


end;
$$
LANGUAGE plpgsql;
```

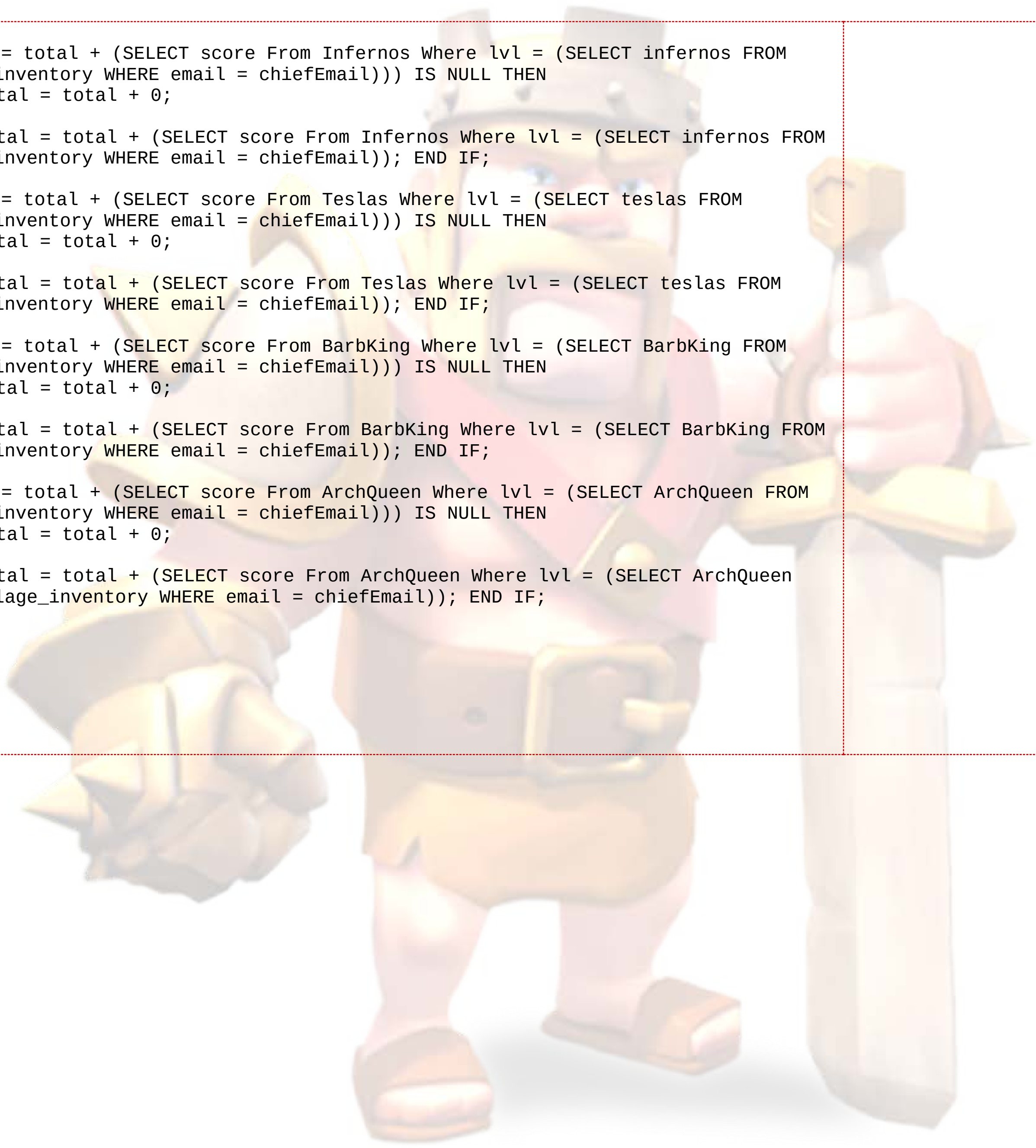
Stored Procedures:

villageScore()



```
create or replace function villageScore(varchar) returns int as
$$
declare
    chiefEmail varchar := $1;
    total int = 0;

begin

  -- These are manditory Defenses for joining the Alpaca's


           total = total + (SELECT score From AirDefs Where lvl = (SELECT airDefs
                                                              FROM Playervillage_inventory
                                                              WHERE email = chiefEmail));

           total = total + (SELECT score From ArcherTowers Where lvl = (SELECT archerTowers
                                                                   FROM Playervillage_inventory
                                                                   WHERE email = chiefEmail));

           total = total + (SELECT score From Cannons Where lvl = (SELECT cannons
                                                              FROM Playervillage_inventory
                                                              WHERE email = chiefEmail));

           total = total + (SELECT score From Mortars Where lvl = (SELECT mortars
                                                              FROM Playervillage_inventory
                                                              WHERE email = chiefEmail));

           total = total + (SELECT score From WizardTowers Where lvl = (SELECT WizardTowers
                                                                   FROM Playervillage_inventory
                                                                   WHERE email = chiefEmail));

           total = total + (SELECT score From Walls Where lvl = (SELECT Walls
                                                            FROM Playervillage_inventory
                                                            WHERE email = chiefEmail));


       -- These are not required, therefore we must handle possible NULL values.
       -- All NULL values will equal 0


               IF (total = total + (SELECT score From AirSweepers Where lvl = (SELECT airSweepers
           FROM Playervillage_inventory WHERE email = chiefEmail))) IS NULL THEN
                       total = total + 0;
               ELSE
                       total = total + (SELECT score From AirSweepers Where lvl = (SELECT
           airSweepers FROM Playervillage_inventory WHERE email = chiefEmail)); END IF;
```

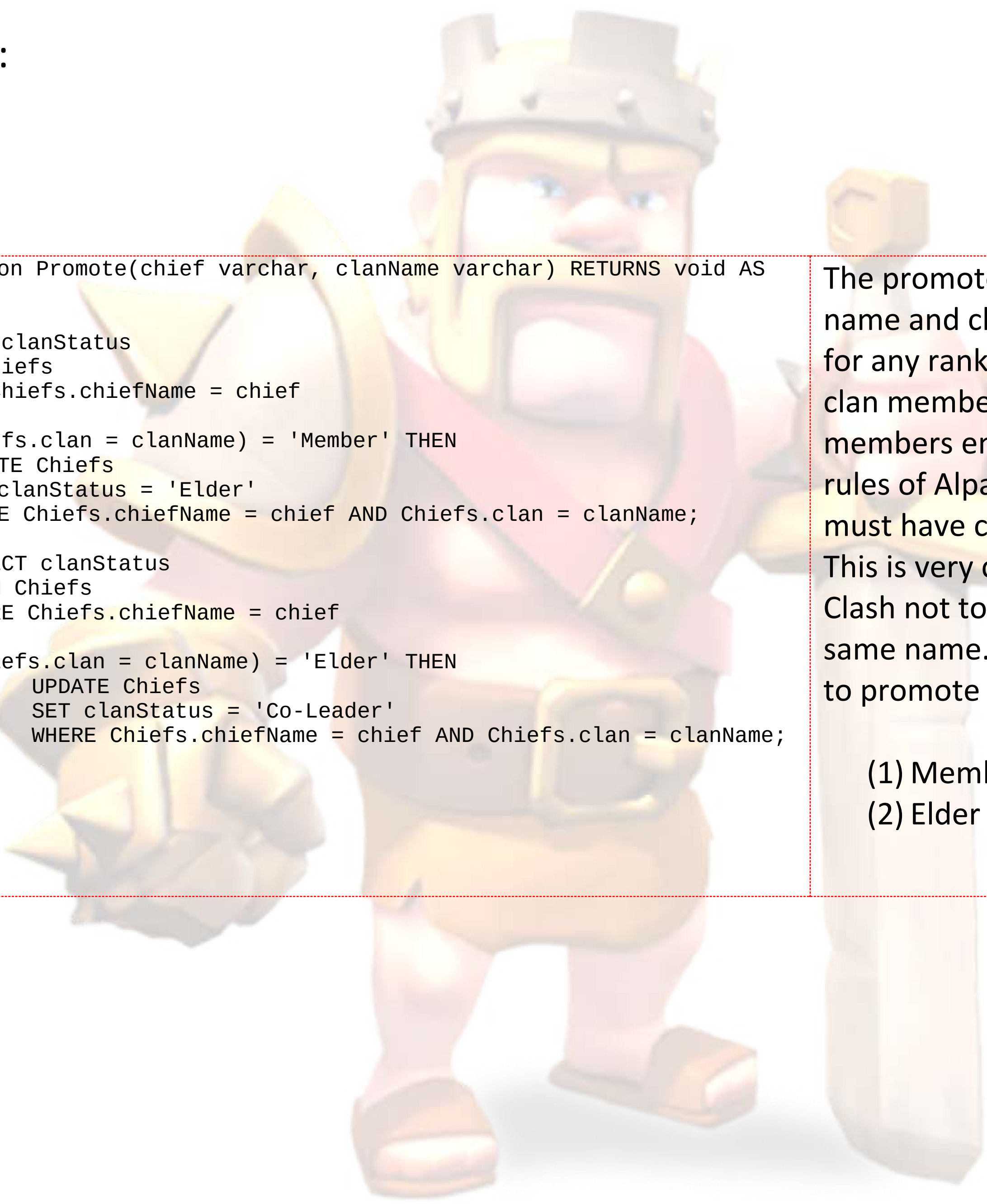
Village score is much like the troop score. All of the score values are drawn from their respective tables. Both troop score and village score are not saved in a field but rather are calculated as needed. This is to ensure a normalized database



```
            IF (total = total + (SELECT score From Infernos Where lvl = (SELECT infernos FROM
        Playervillage_inventory WHERE email = chiefEmail))) IS NULL THEN
                    total = total + 0;
            ELSE
                    total = total + (SELECT score From Infernos Where lvl = (SELECT infernos FROM
        Playervillage_inventory WHERE email = chiefEmail)); END IF;

            IF (total = total + (SELECT score From Teslas Where lvl = (SELECT teslas FROM
        Playervillage_inventory WHERE email = chiefEmail))) IS NULL THEN
                    total = total + 0;
            ELSE
                    total = total + (SELECT score From Teslas Where lvl = (SELECT teslas FROM
        Playervillage_inventory WHERE email = chiefEmail)); END IF;

            IF (total = total + (SELECT score From BarbKing Where lvl = (SELECT BarbKing FROM
        Playervillage_inventory WHERE email = chiefEmail))) IS NULL THEN
                    total = total + 0;
            ELSE
                    total = total + (SELECT score From BarbKing Where lvl = (SELECT BarbKing FROM
        Playervillage_inventory WHERE email = chiefEmail)); END IF;

            IF (total = total + (SELECT score From ArchQueen Where lvl = (SELECT ArchQueen FROM
        Playervillage_inventory WHERE email = chiefEmail))) IS NULL THEN
                    total = total + 0;
            ELSE
                    total = total + (SELECT score From ArchQueen Where lvl = (SELECT ArchQueen
        FROM Playervillage_inventory WHERE email = chiefEmail)); END IF;

        RETURN total;


end;
$$
LANGUAGE plpgsql;
```

Stored Procedures:

Promote()



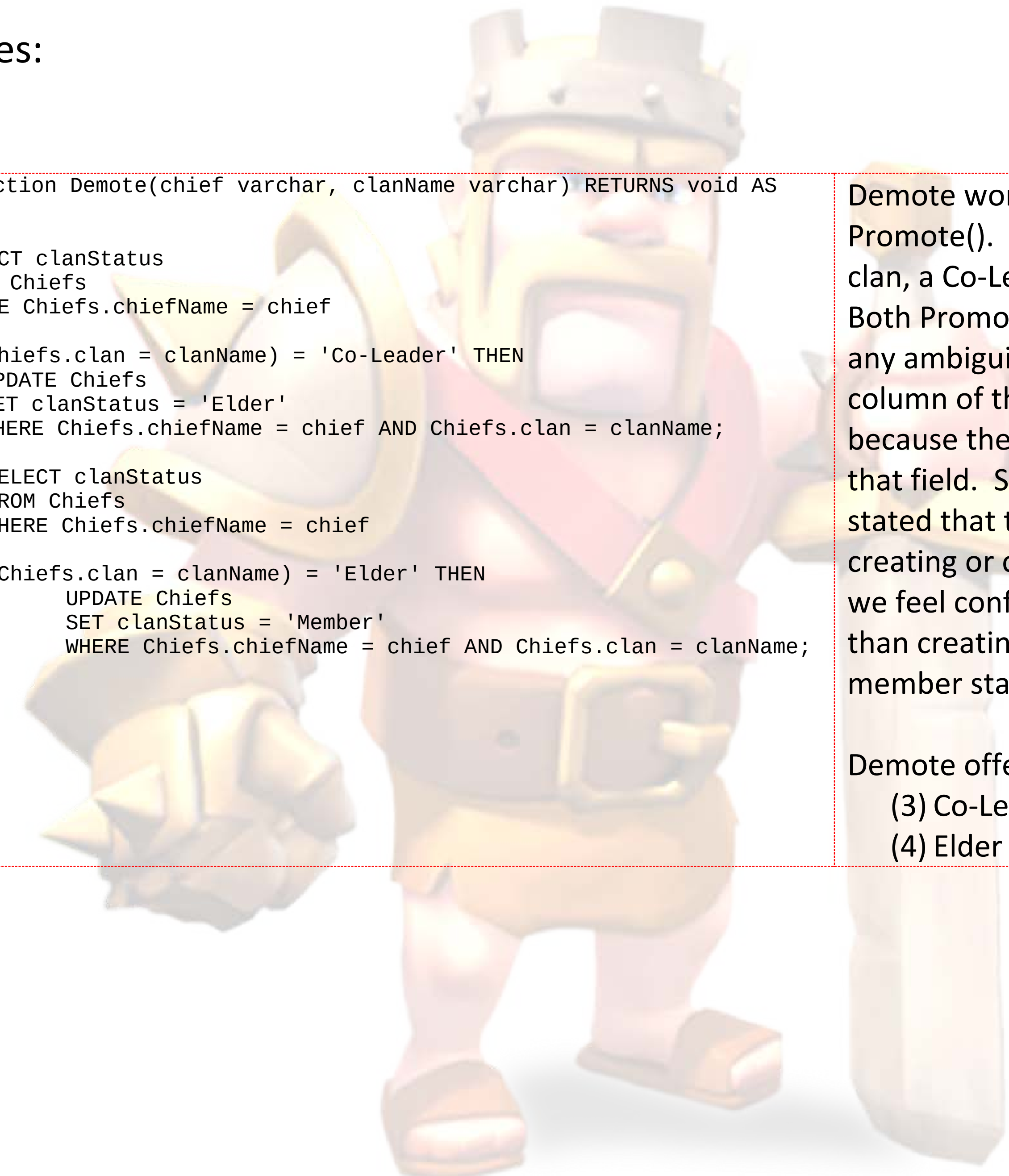
```
create or replace function Promote(chief varchar, clanName varchar) RETURNS void AS
$$
begin
             IF (SELECT clanStatus
                 FROM Chiefs
                 WHERE Chiefs.chiefName = chief
                 AND
                    Chiefs.clan = clanName) = 'Member' THEN
                    UPDATE Chiefs
                    SET clanStatus = 'Elder'
                    WHERE Chiefs.chiefName = chief AND Chiefs.clan = clanName;

             ELSIF (SELECT clanStatus
                    FROM Chiefs
                    WHERE Chiefs.chiefName = chief
                 AND
                     Chiefs.clan = clanName) = 'Elder' THEN
                           UPDATE Chiefs
                           SET clanStatus = 'Co-Leader'
                           WHERE Chiefs.chiefName = chief AND Chiefs.clan = clanName;
             ELSE

             END IF;
end;
$$
LANGUAGE plpgsql;
```

The promote function receives the chief name and clan subset. This this allows for any ranking Co-leader to promote clan members while still protecting each members email address. One of the rules of Alpaca is that each clan sub-set must have chiefs with unique names. This is very common among clans in Clash not to accept other chiefs with the same name. There are only two options to promote members:

(1) Member → Elder
(2) Elder → Co-Leader

Stored Procedures:

Demote()



```
create or replace function Demote(chief varchar, clanName varchar) RETURNS void AS
$$
begin
             IF (SELECT clanStatus
                 FROM Chiefs
                 WHERE Chiefs.chiefName = chief
                 AND
                    Chiefs.clan = clanName) = 'Co-Leader' THEN
                    UPDATE Chiefs
                    SET clanStatus = 'Elder'
                    WHERE Chiefs.chiefName = chief AND Chiefs.clan = clanName;

             ELSIF (SELECT clanStatus
                    FROM Chiefs
                    WHERE Chiefs.chiefName = chief
                 AND
                     Chiefs.clan = clanName) = 'Elder' THEN
                          UPDATE Chiefs
                          SET clanStatus = 'Member'
                          WHERE Chiefs.chiefName = chief AND Chiefs.clan = clanName;
             ELSE

             END IF;
end;
$$
LANGUAGE plpgsql;
```

Demote works on the same premise as Promote(). Using the Chief name and clan, a Co-Leader can demote chiefs. Both Promote and Demote help remove any ambiguity with the clanStatus column of the Chiefs table. This is because there is no typing involved in that field. SuperCell (Clash devs) has stated that they have no plans of creating or changing clan status titles so we feel confident in this choice rather than creating a dynamic table for clan member status.

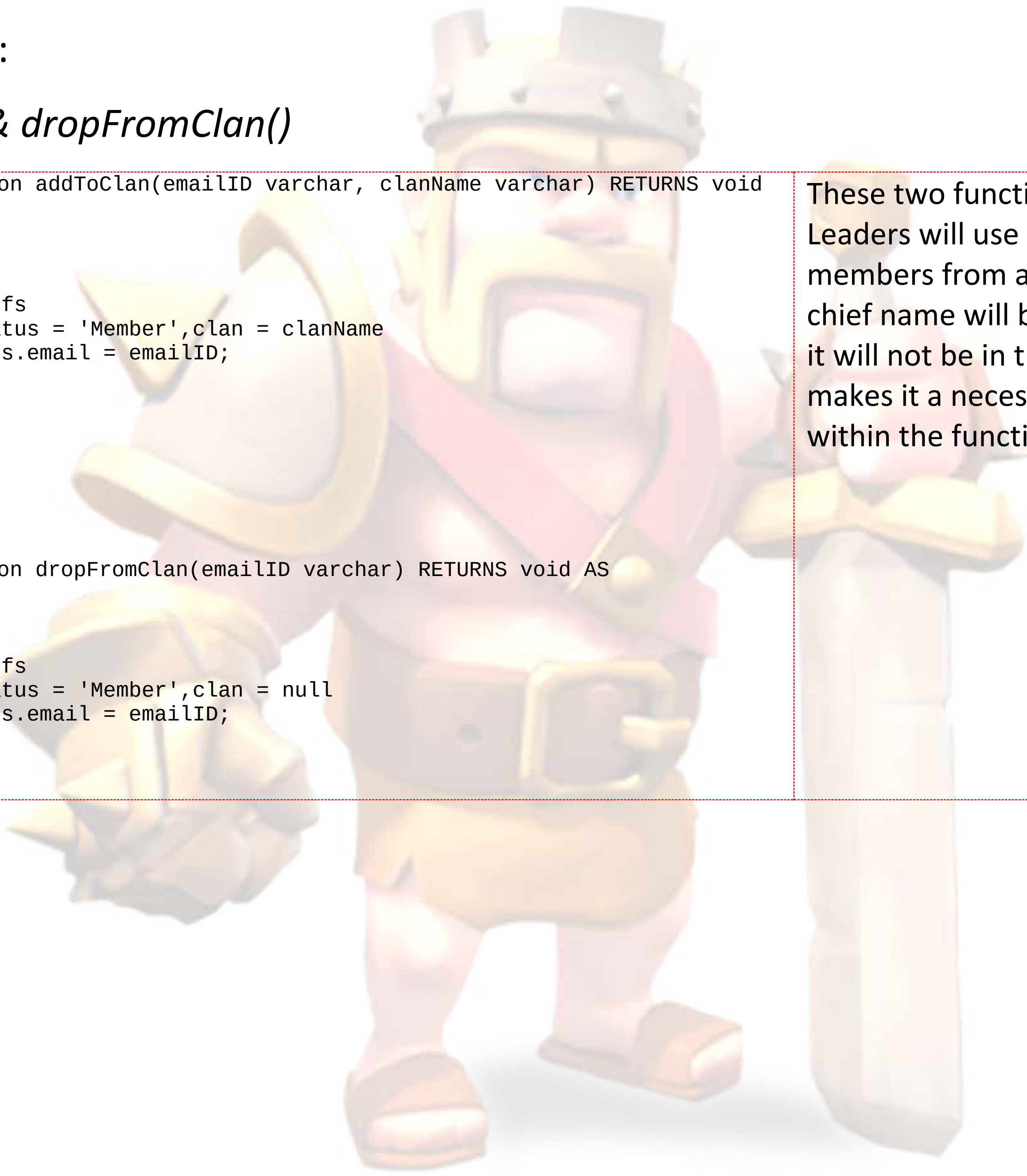
Demote offers two change options:

(3) Co-Leader → Elder

(4) Elder → Member

Stored Procedures:

addToClan() & dropFromClan()

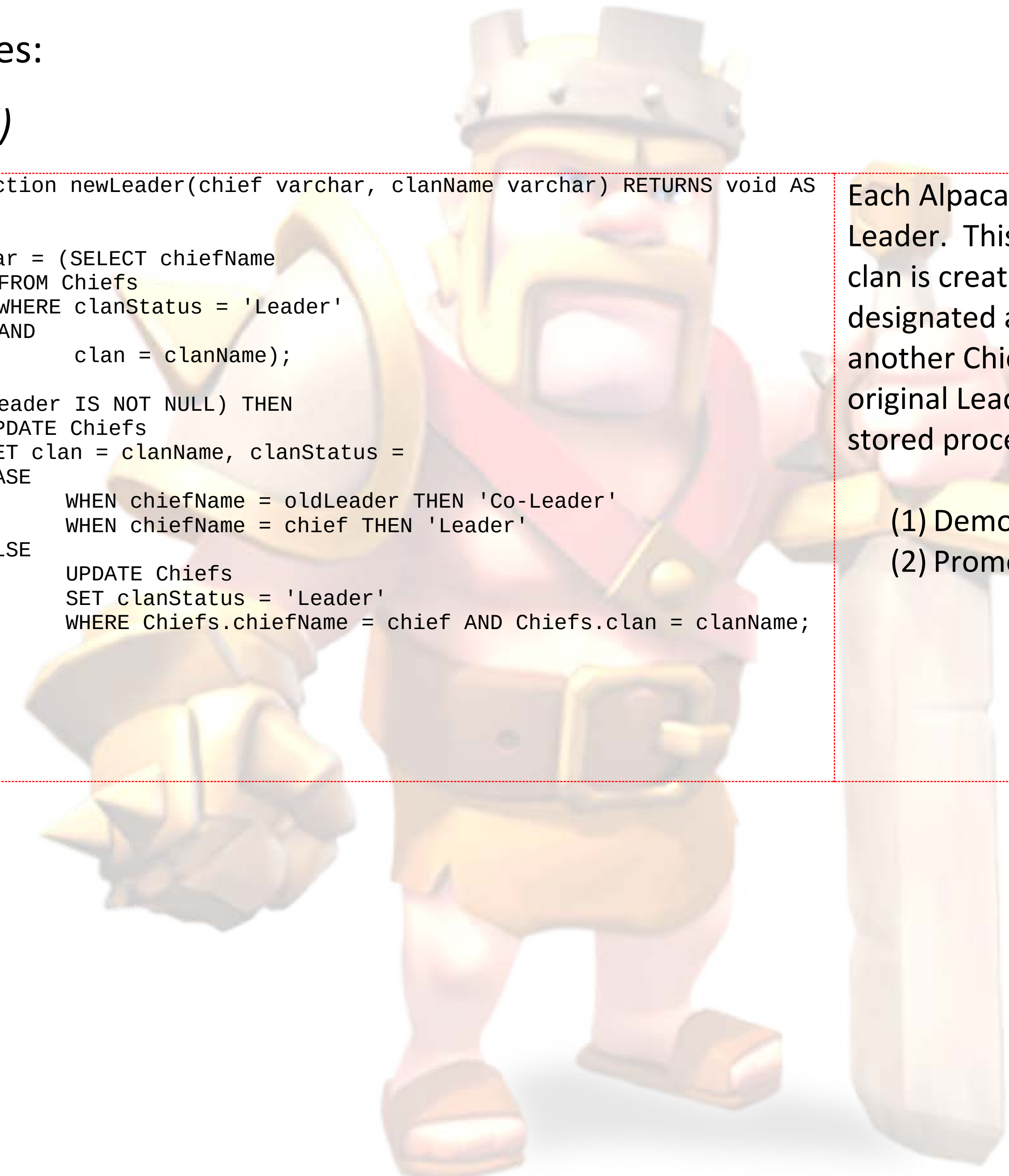


```
create or replace function addToClan(emailID varchar, clanName varchar) RETURNS void
AS
$$

begin
              UPDATE Chiefs
              SET clanStatus = 'Member',clan = clanName
              WHERE Chiefs.email = emailID;
end;
$$
LANGUAGE plpgsql;




create or replace function dropFromClan(emailID varchar) RETURNS void AS
$$

begin
              UPDATE Chiefs
              SET clanStatus = 'Member',clan = null
              WHERE Chiefs.email = emailID;
end;
$$
LANGUAGE plpgsql;
```

These two functions are what Alpaca Co-Leaders will use to add and remove members from a clan. Although, the chief name will be unique within the clan it will not be in the Chiefs table. This makes it a necessity to use the emailID within the function's arguments.

Stored Procedures:

newLeader()



```
create or replace function newLeader(chief varchar, clanName varchar) RETURNS void AS
-- $$
--   declare
--    oldLeader varchar = (SELECT chiefName
--                   FROM Chiefs
--                   WHERE clanStatus = 'Leader'
--                   AND
--                         clan = clanName);
-- begin
--           IF (oldLeader IS NOT NULL) THEN
--                  UPDATE Chiefs
--                  SET clan = clanName, clanStatus =
--                  CASE
--                         WHEN chiefName = oldLeader THEN 'Co-Leader'
--                         WHEN chiefName = chief THEN 'Leader'
--                  ELSE
--                         UPDATE Chiefs
--                         SET clanStatus = 'Leader'
--                         WHERE Chiefs.chiefName = chief AND Chiefs.clan = clanName;
--           END IF;
--
--
-- end;
-- $$
-- LANGUAGE plpgsql;
```

Each Alpaca sub-clan may have only one Leader. This is set-up by SuperCell. A clan is created by one Chief, that Chief is designated as the Leader. In order for another Chief to become a leader, the original Leader must be demoted. This stored procedure will do two processes:

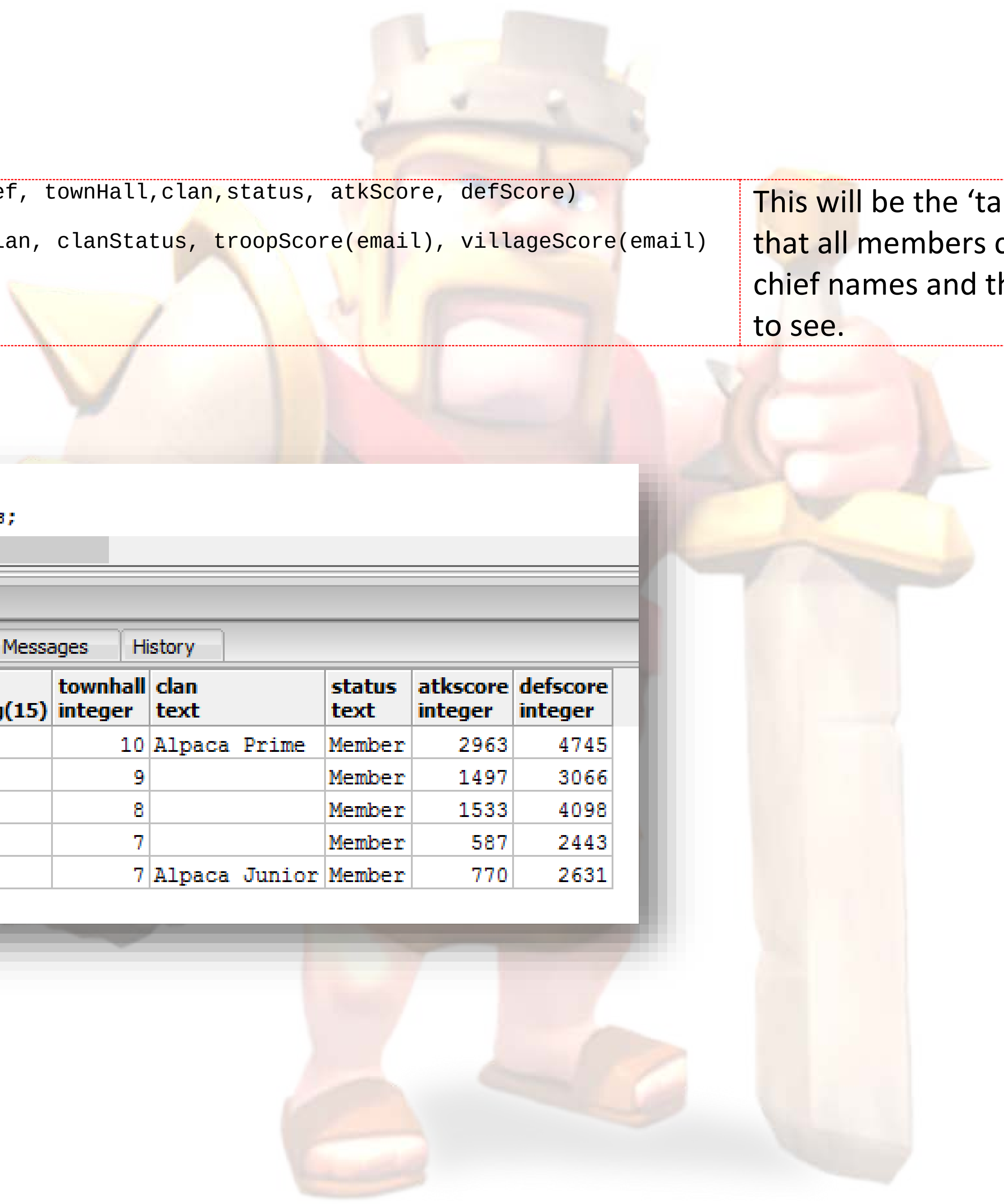
(1) Demote the former Leader
(2) Promote the new Leader

Views:

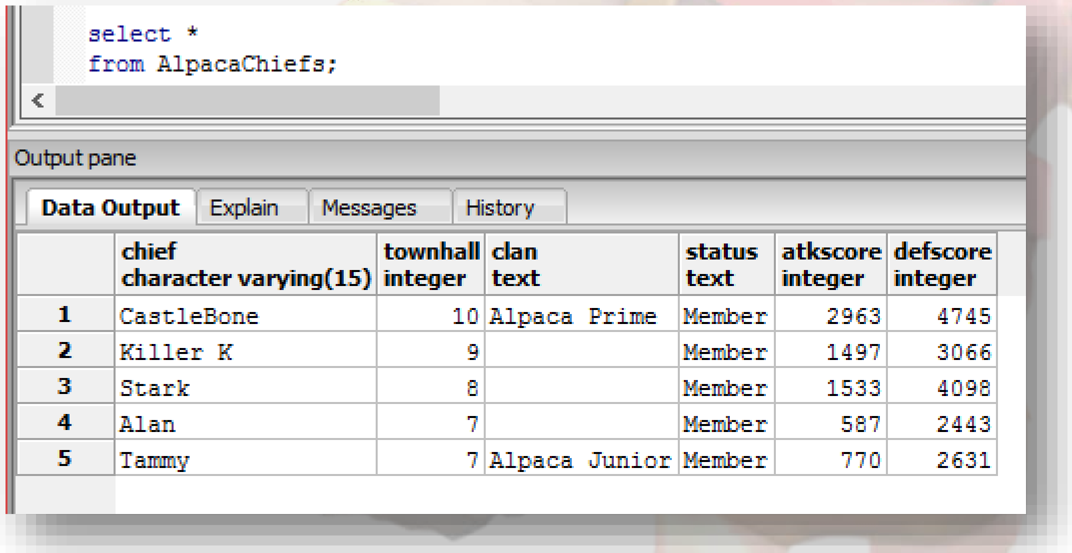
AlpacaChiefs

```
CREATE VIEW AlpacaChiefs (chief, townHall,clan,status, atkScore, defScore)
AS
SELECT chiefName, townHall, clan, clanStatus, troopScore(email), villageScore(email)
FROM Chiefs
ORDER BY townHall DESC;
```

This will be the 'table' of the clan chiefs that all members can see. We list the chief names and their base scores for all to see.



Example:

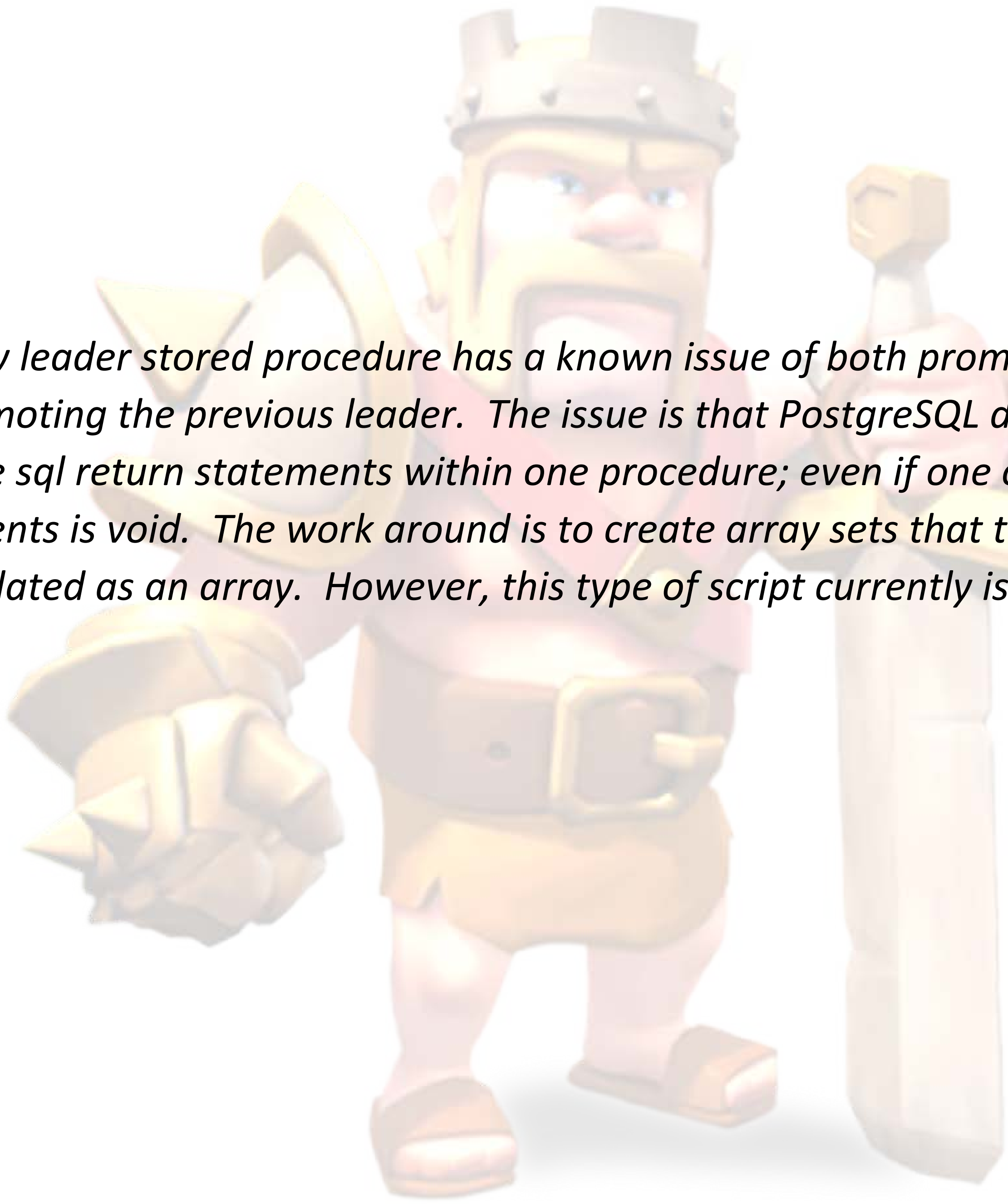


```
select *
from AlpacaChiefs;
```

	chief character varying(15)	townhall integer	clan text	status text	atkscore integer	defscore integer
1	CastleBone	10	Alpaca Prime	Member	2963	4745
2	Killer K	9		Member	1497	3066
3	Stark	8		Member	1533	4098
4	Alan	7		Member	587	2443
5	Tammy	7	Alpaca Junior	Member	770	2631

BUGS:

newLeader()



- *The new leader stored procedure has a known issue of both promoting a clan member and demoting the previous leader. The issue is that PostgreSQL does not allow for multiple sql return statements within one procedure; even if one of the return statements is void. The work around is to create array sets that then may be manipulated as an array. However, this type of script currently is out of scope for this project.*