Legend of the Galactic Heroes Database

Brandon Litwin

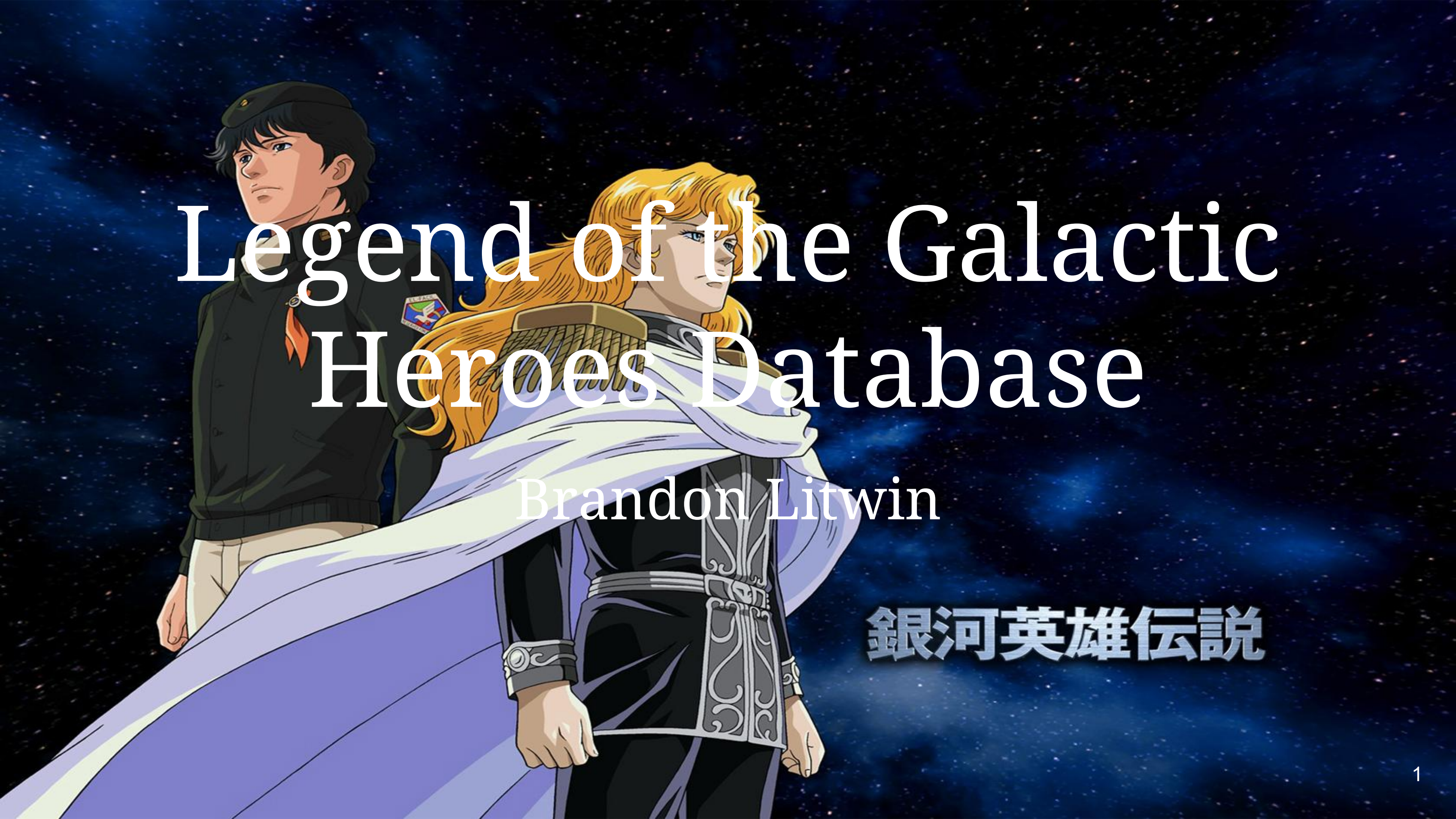


Table of Contents

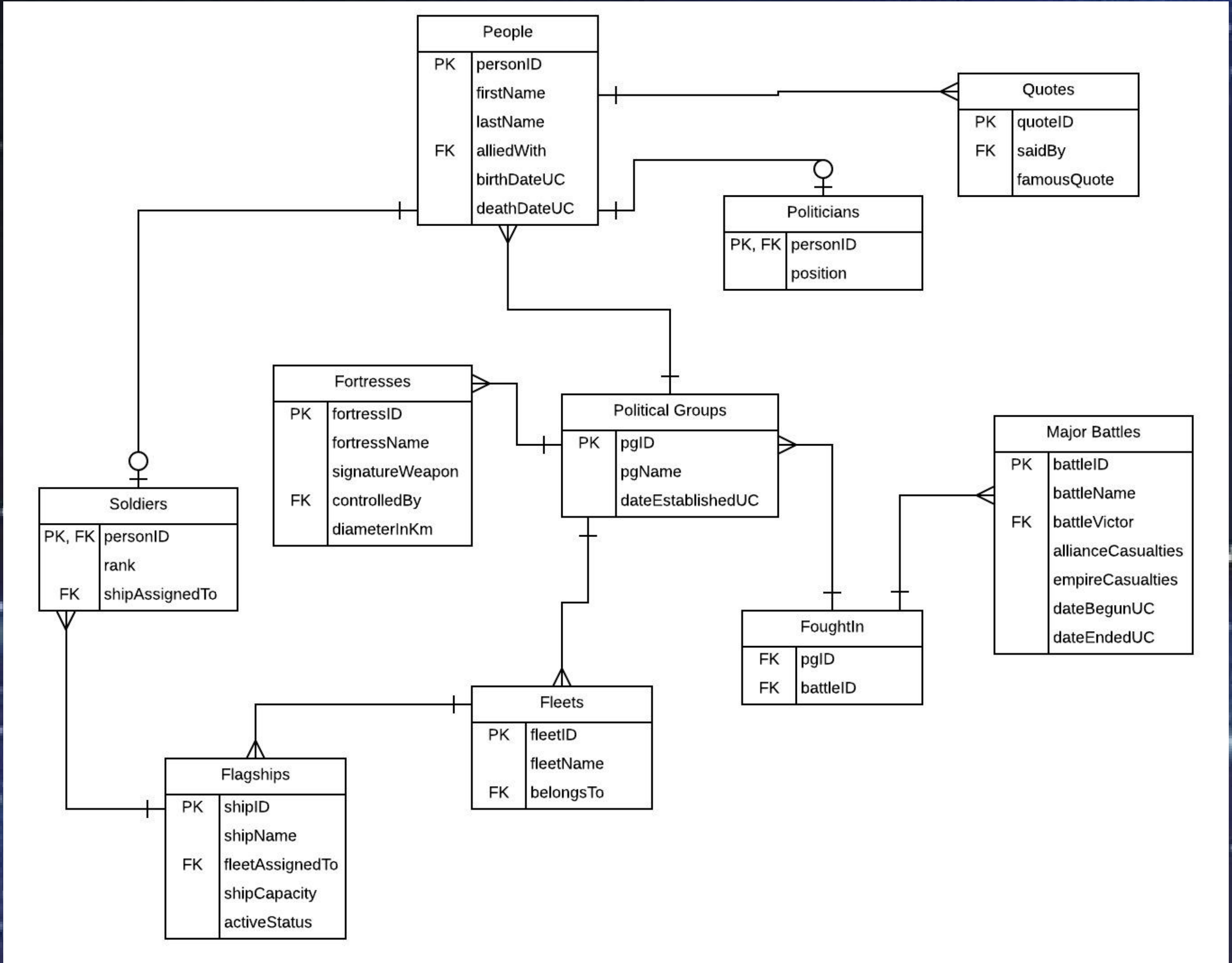
Executive Summary

“In every age, in every place, the deeds of men remain the same”.

In the fictional futuristic world of *Legend of the Galactic Heroes*, two political groups with vastly different beliefs are fighting over control of the universe. The following documentation details the design and implementation of a database dedicated to all of the key components of the Alliance-Imperial War. This includes an ER diagram of the database, followed by a description of every table with their SQL create statements, their functional dependencies, and sample data. Next, some useful views of the table, reports, and interesting queries will be shown. Then, stored procedures, triggers, and security will be discussed. Finally, some implementation notes, known problems, and future enhancements will be documented.

This database is intended to be used by anyone who is interested in watching *Legend of the Galactic Heroes*, but has found the vast number of characters, ship names, and other data to be confusing. The database includes the names and ranks of every key character in the show, as well as who they are allied with and which battles they have fought in.

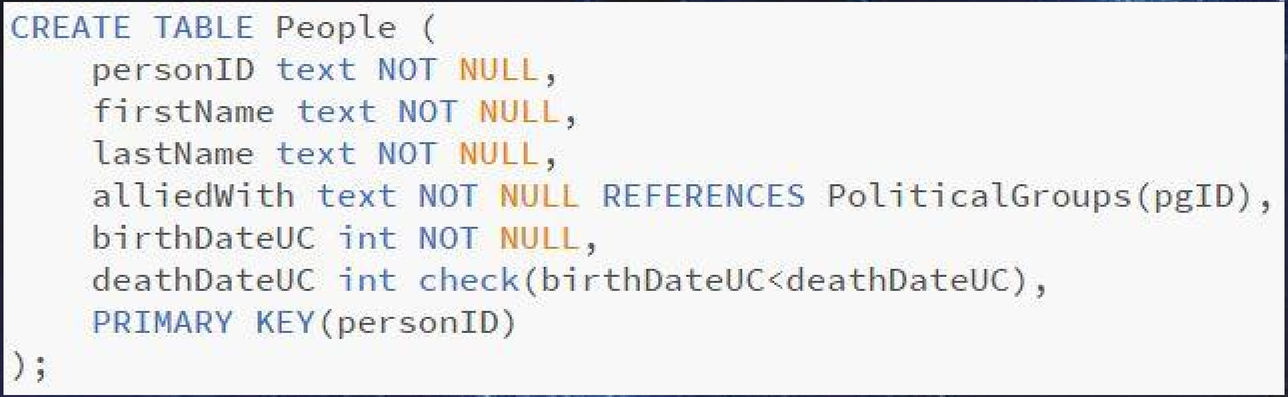
ERD


People
PK personID
firstName
lastName
FK alliedWith
birthDateUC
deathDateUC
Quotes
PK quoteID
FK saidBy
famousQuote
Politicians
PK, FK personID
position
Fortresses
PK fortressID
fortressName
signatureWeapon
FK controlledBy
diameterInKm
Political Groups
PK pgID
pgName
dateEstablishedUC
Major Battles
PK battleID
battleName
FK battleVictor
allianceCasualties
empireCasualties
dateBegunUC
dateEndedUC
Soldiers
PK, FK personID
rank
FK shipAssignedTo
FoughtIn
FK pgID
FK battleID
Fleets
PK fleetID
fleetName
FK belongsTo
Flagships
PK shipID
shipName
FK fleetAssignedTo
shipCapacity
activeStatus

Tables

People

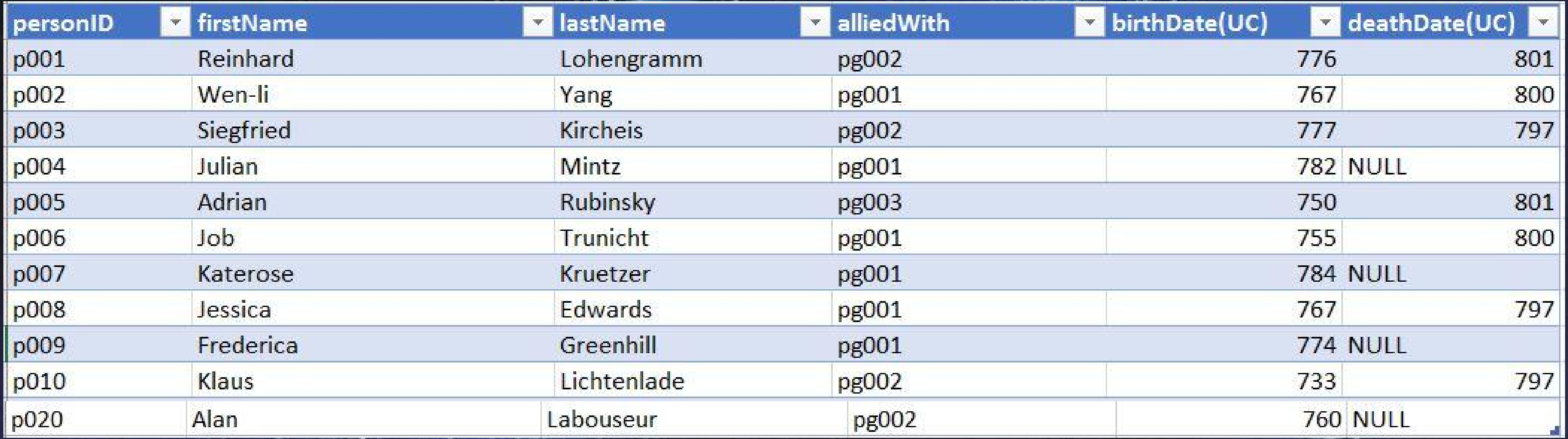
The People table keeps track of all the people that are in the database. These attributes are shared with the Soldiers and Politicians tables.



```
CREATE TABLE People (
    personID text NOT NULL,
    firstName text NOT NULL,
    lastName text NOT NULL,
    alliedWith text NOT NULL REFERENCES PoliticalGroups(pgID),
    birthDateUC int NOT NULL,
    deathDateUC int check(birthDateUC<deathDateUC),
    PRIMARY KEY(personID)
);
```

Functional Dependencies: personID → firstName, lastName, alliedWith, birthDateUC, deathDateUC

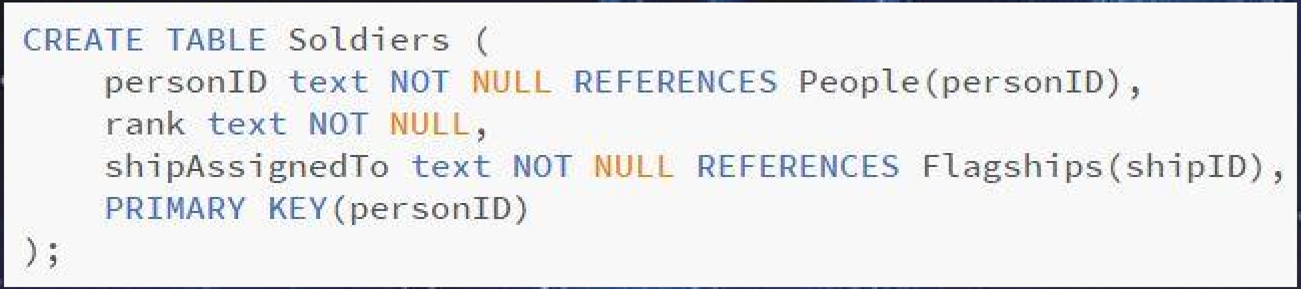
People Sample Data



personID	firstName	lastName	alliedWith	birthDate(UC)	deathDate(UC)
p001	Reinhard	Lohengramm	pg002	776	801
p002	Wen-li	Yang	pg001	767	800
p003	Siegfried	Kircheis	pg002	777	797
p004	Julian	Mintz	pg001	782	NULL
p005	Adrian	Rubinsky	pg003	750	801
p006	Job	Trunicht	pg001	755	800
p007	Katerose	Kruetzer	pg001	784	NULL
p008	Jessica	Edwards	pg001	767	797
p009	Frederica	Greenhill	pg001	774	NULL
p010	Klaus	Lichtenlade	pg002	733	797
p020	Alan	Labouseur	pg002	760	NULL

Soldiers

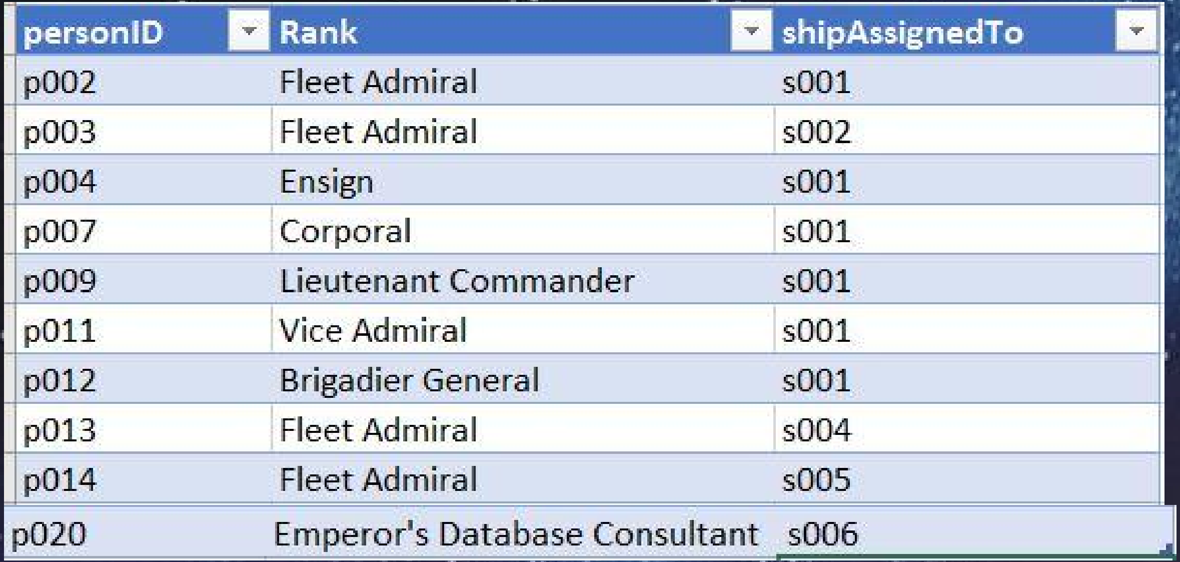
The Soldiers table keeps track of all of the soldiers who have fought for either the Galactic Empire or the Free Planets Alliance, and what ship they are assigned to. Soldiers are a subtype of People.



```sql
CREATE TABLE Soldiers (
    personID text NOT NULL REFERENCES People(personID),
    rank text NOT NULL,
    shipAssignedTo text NOT NULL REFERENCES Flagships(shipID),
    PRIMARY KEY(personID)
);
```

Functional Dependencies: personID → rank, shipAssignedTo

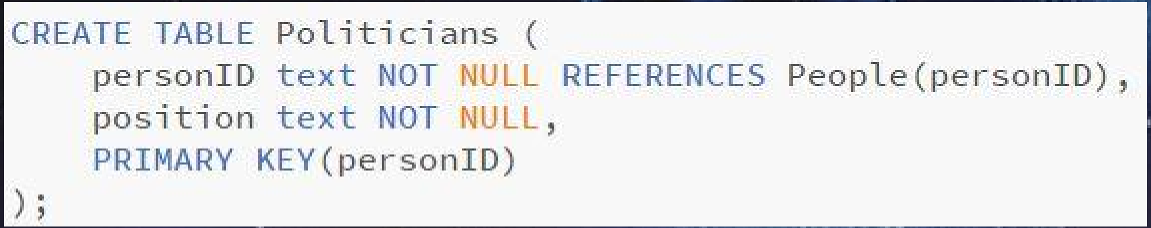
Soldiers Sample Data



personID	Rank	shipAssignedTo
p002	Fleet Admiral	s001
p003	Fleet Admiral	s002
p004	Ensign	s001
p007	Corporal	s001
p009	Lieutenant Commander	s001
p011	Vice Admiral	s001
p012	Brigadier General	s001
p013	Fleet Admiral	s004
p014	Fleet Admiral	s005
p020	Emperor's Database Consultant	s006

Politicians

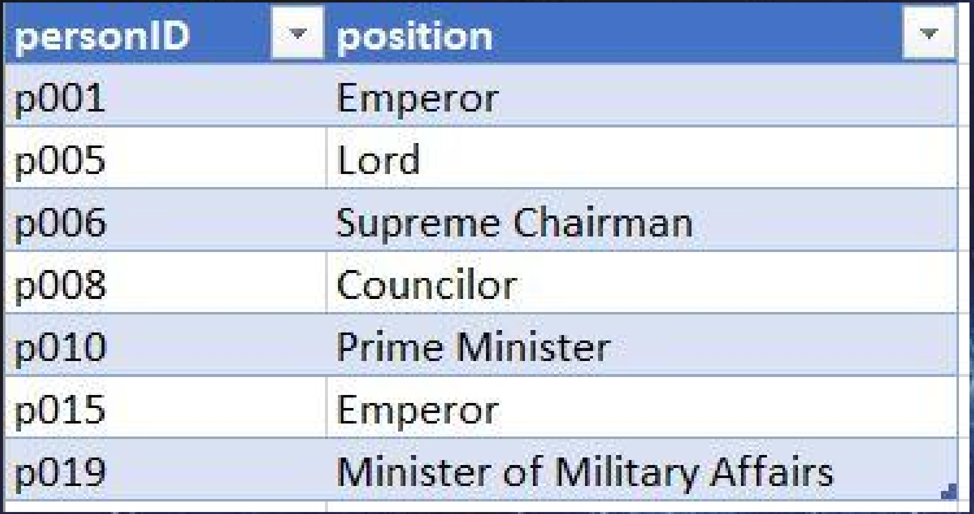
The Politicians table keeps track of all the key politicians involved with the Alliance-Imperial War. Politicians are a subtype of People.



```
CREATE TABLE Politicians (
    personID text NOT NULL REFERENCES People(personID),
    position text NOT NULL,
    PRIMARY KEY(personID)
);
```

Functional Dependencies: personID → position

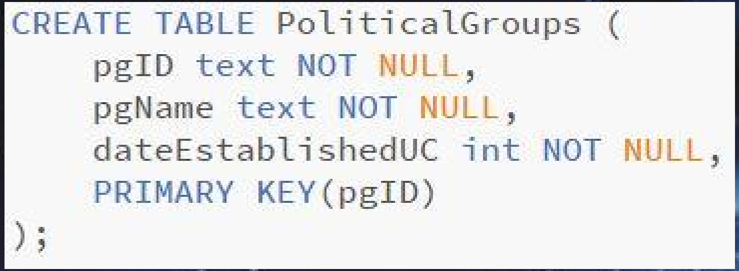
Politicians Sample Data



personID	position
p001	Emperor
p005	Lord
p006	Supreme Chairman
p008	Councilor
p010	Prime Minister
p015	Emperor
p019	Minister of Military Affairs

PoliticalGroups

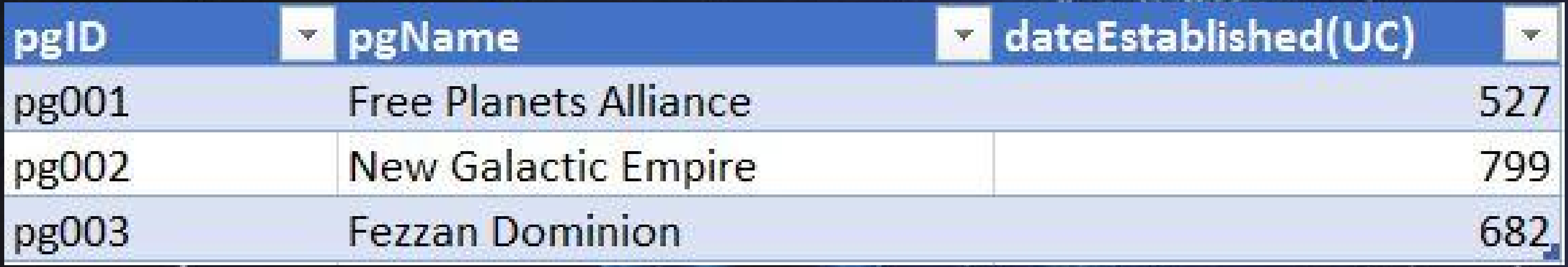
The PoliticalGroups table keeps track of the key political groups who participated in the Alliance-Imperial War, and when they were established.



```
CREATE TABLE PoliticalGroups (
    pgID text NOT NULL,
    pgName text NOT NULL,
    dateEstablishedUC int NOT NULL,
    PRIMARY KEY(pgID)
);
```

Functional Dependencies: pgID → pgName, dateEstablishedUC

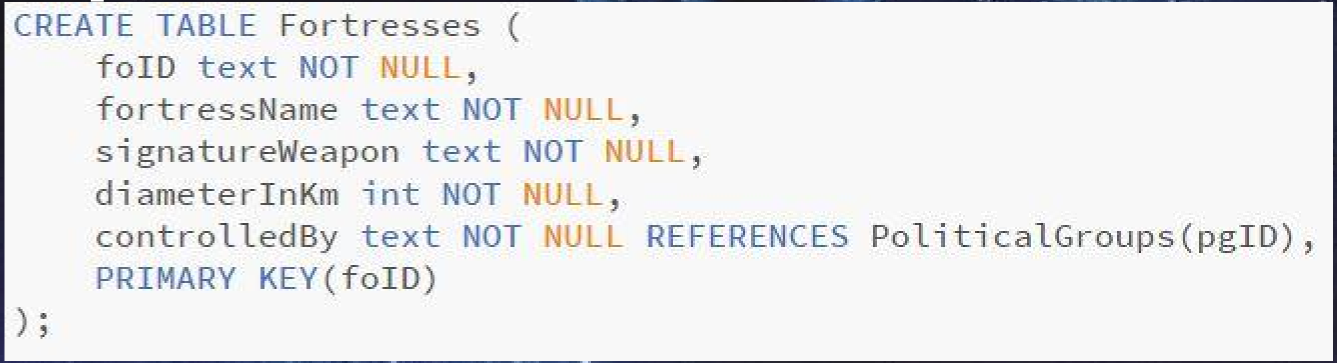
PoliticalGroups Sample Data



pgID	pgName	dateEstablished(UC)
pg001	Free Planets Alliance	527
pg002	New Galactic Empire	799
pg003	Fezzan Dominion	682

Fortresses

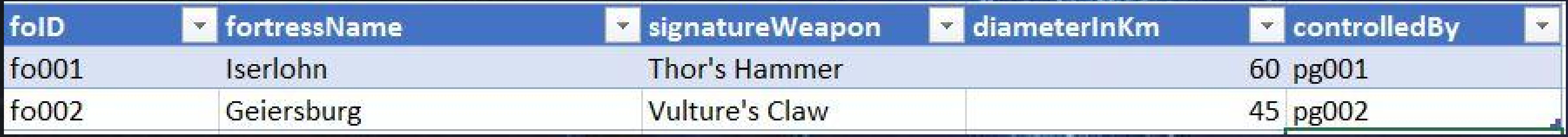
The Fortresses table keeps track of the main fortresses used during the Alliance-Imperial War, and who is currently in control. The fortresses have changed hands many times, which is significant because they have powerful weapons.



```
CREATE TABLE Fortresses (
    foID text NOT NULL,
    fortressName text NOT NULL,
    signatureWeapon text NOT NULL,
    diameterInKm int NOT NULL,
    controlledBy text NOT NULL REFERENCES PoliticalGroups(pgID),
    PRIMARY KEY(foID)
);
```

Functional Dependencies: fortressID → fortressName, signatureWeapon, controlledBy, and diameterInKm

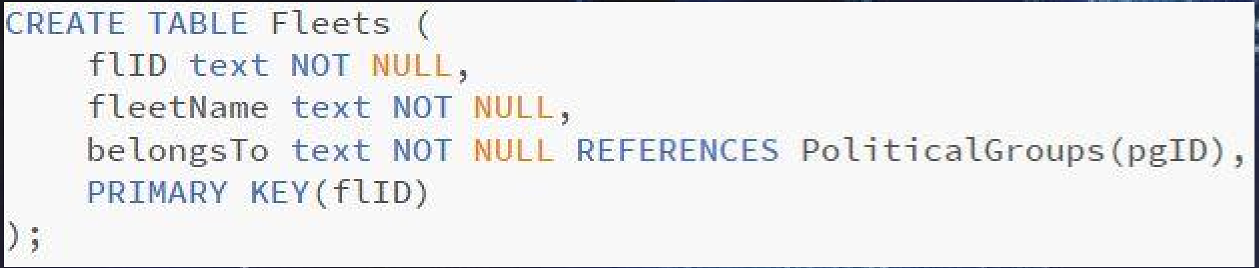
Fortresses Sample Data



foID	fortressName	signatureWeapon	diameterInKm	controlledBy
fo001	Iserlohn	Thor's Hammer	60	pg001
fo002	Geiersburg	Vulture's Claw	45	pg002

Fleets

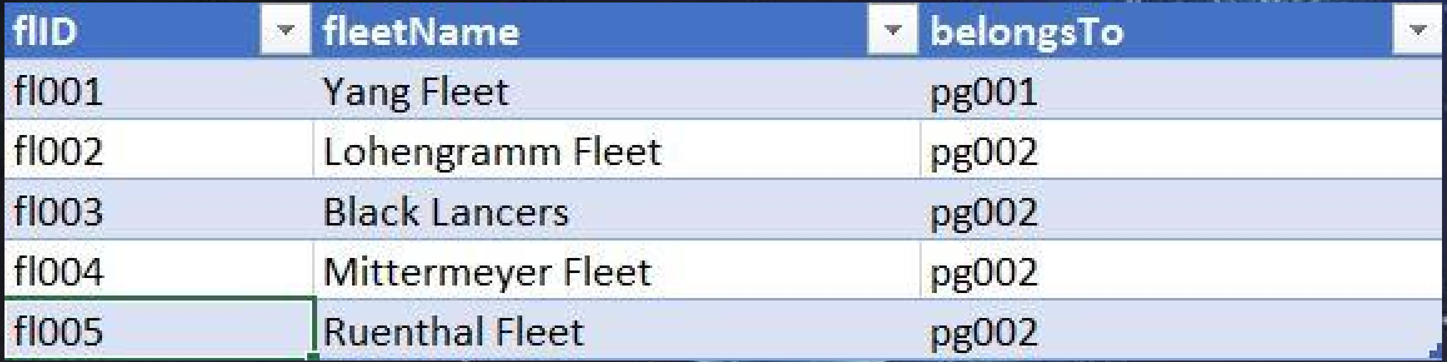
The Fleets table keeps track of all of the fleets used in the Alliance-Imperial War and which political group they belong to.



```
CREATE TABLE Fleets (
    flID text NOT NULL,
    fleetName text NOT NULL,
    belongsTo text NOT NULL REFERENCES PoliticalGroups(pgID),
    PRIMARY KEY(flID)
);
```

Functional Dependencies: fleetID → fleetName, belongsTo

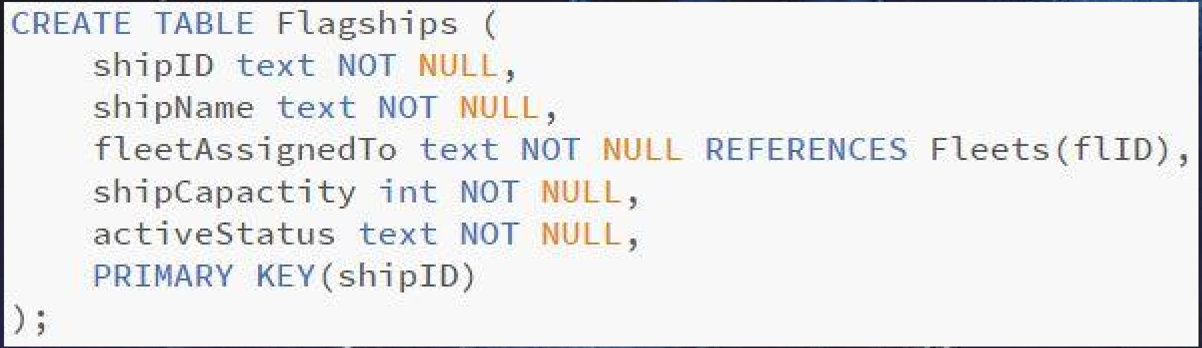
Fleets Sample Data



flID	fleetName	belongsTo
fl001	Yang Fleet	pg001
fl002	Lohengramm Fleet	pg002
fl003	Black Lancers	pg002
fl004	Mittermeyer Fleet	pg002
fl005	Ruenthal Fleet	pg002

Flagships

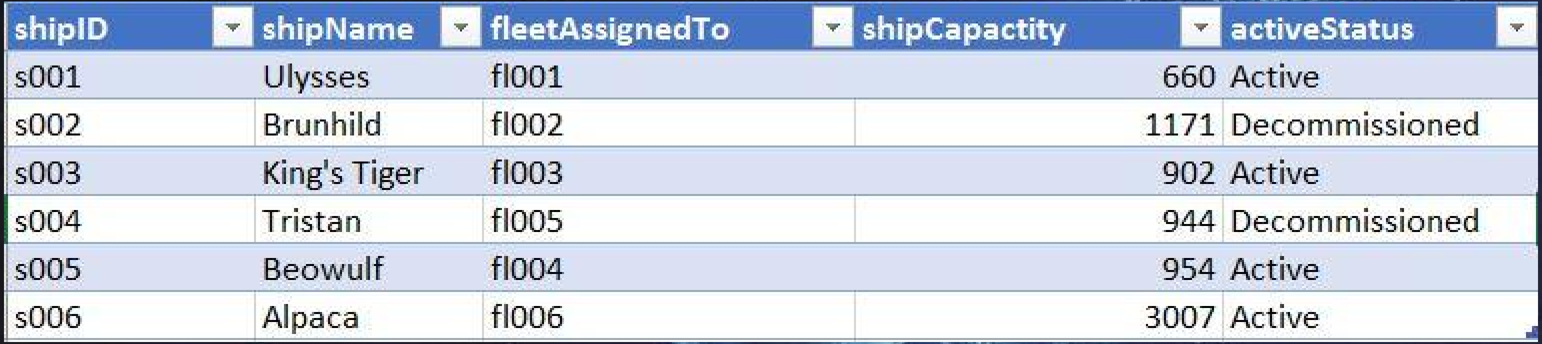
The Flagships table keeps track of the most significant ships used by both sides of the Alliance-Imperial War, including which fleet they belong to and whether they are currently active or decommissioned.



```
CREATE TABLE Flagships (
    shipID text NOT NULL,
    shipName text NOT NULL,
    fleetAssignedTo text NOT NULL REFERENCES Fleets(flID),
    shipCapactity int NOT NULL,
    activeStatus text NOT NULL,
    PRIMARY KEY(shipID)
);
```

Functional Dependencies: shipID → shipName, fleetAssignedTo, shipCapacity, activeStatus

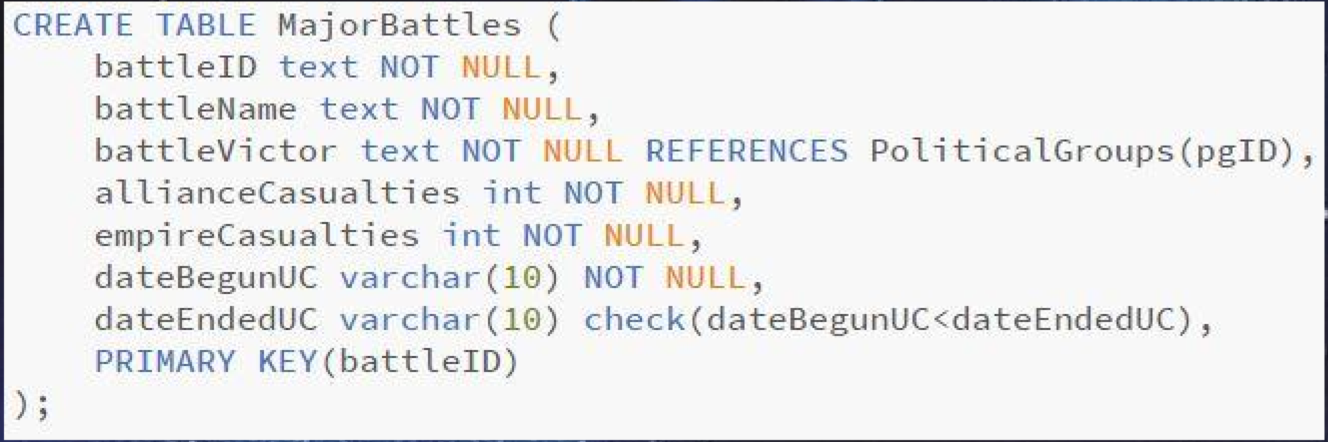
Flagships Sample Data



shipID	shipName	fleetAssignedTo	shipCapactity	activeStatus
s001	Ulysses	fl001	660	Active
s002	Brunhild	fl002	1171	Decommissioned
s003	King's Tiger	fl003	902	Active
s004	Tristan	fl005	944	Decommissioned
s005	Beowulf	fl004	954	Active
s006	Alpaca	fl006	3007	Active

MajorBattles

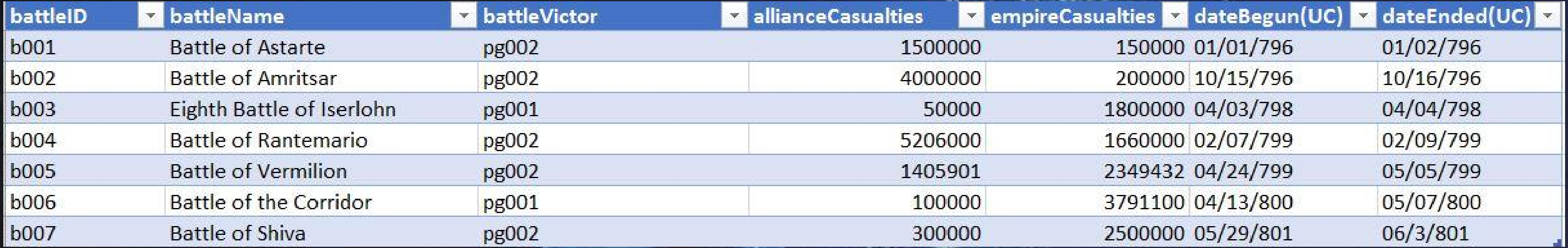
The MajorBattles table keeps track of all the key battles of the Alliance-Imperial War, including how long they lasted, total casualties, and the victor.



```
CREATE TABLE MajorBattles (
    battleID text NOT NULL,
    battleName text NOT NULL,
    battleVictor text NOT NULL REFERENCES PoliticalGroups(pgID),
    allianceCasualties int NOT NULL,
    empireCasualties int NOT NULL,
    dateBegunUC varchar(10) NOT NULL,
    dateEndedUC varchar(10) check(dateBegunUC<dateEndedUC),
    PRIMARY KEY(battleID)
);
```

Functional Dependencies: battleID → battleName, battleVictor, allianceCasualties, empireCasualties, dateBegunUC, and dateEndedUC

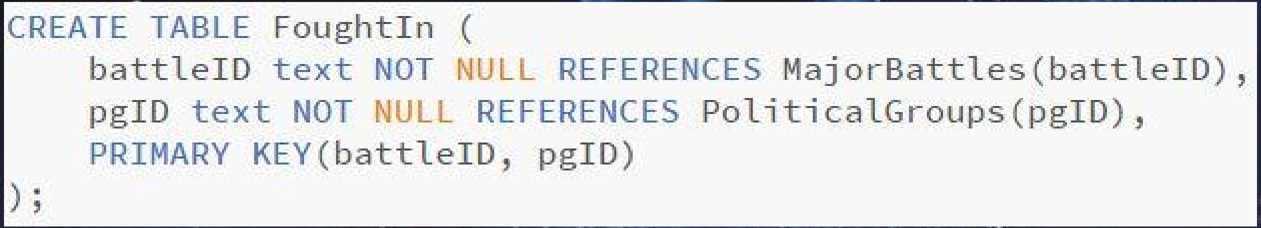
MajorBattles Sample Data



battleID	battleName	battleVictor	allianceCasualties	empireCasualties	dateBegun(UC)	dateEnded(UC)
b001	Battle of Astarte	pg002	1500000	150000	01/01/796	01/02/796
b002	Battle of Amritsar	pg002	4000000	200000	10/15/796	10/16/796
b003	Eighth Battle of Iserlohn	pg001	50000	1800000	04/03/798	04/04/798
b004	Battle of Rantemario	pg002	5206000	1660000	02/07/799	02/09/799
b005	Battle of Vermilion	pg002	1405901	2349432	04/24/799	05/05/799
b006	Battle of the Corridor	pg001	100000	3791100	04/13/800	05/07/800
b007	Battle of Shiva	pg002	300000	2500000	05/29/801	06/3/801

FoughtIn

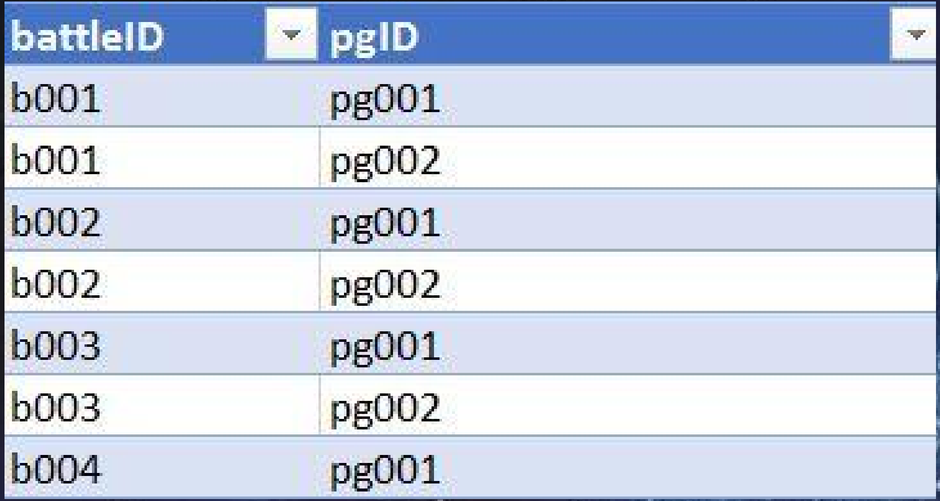
The FoughtIn table keeps track of which political groups fought in which battles, because PoliticalGroups to MajorBattles is a Many to Many relationship.



```sql
CREATE TABLE FoughtIn (
    battleID text NOT NULL REFERENCES MajorBattles(battleID),
    pgID text NOT NULL REFERENCES PoliticalGroups(pgID),
    PRIMARY KEY(battleID, pgID)
);
```

Functional Dependencies: battleID → pgID

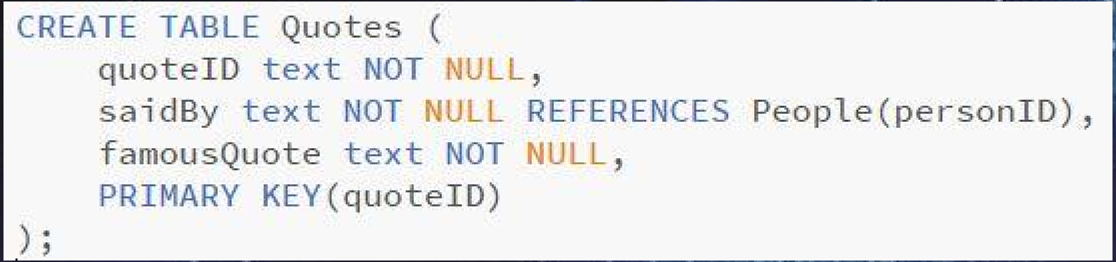
FoughtIn Sample Data



battleID	pgID
b001	pg001
b001	pg002
b002	pg001
b002	pg002
b003	pg001
b003	pg002
b004	pg001

Quotes

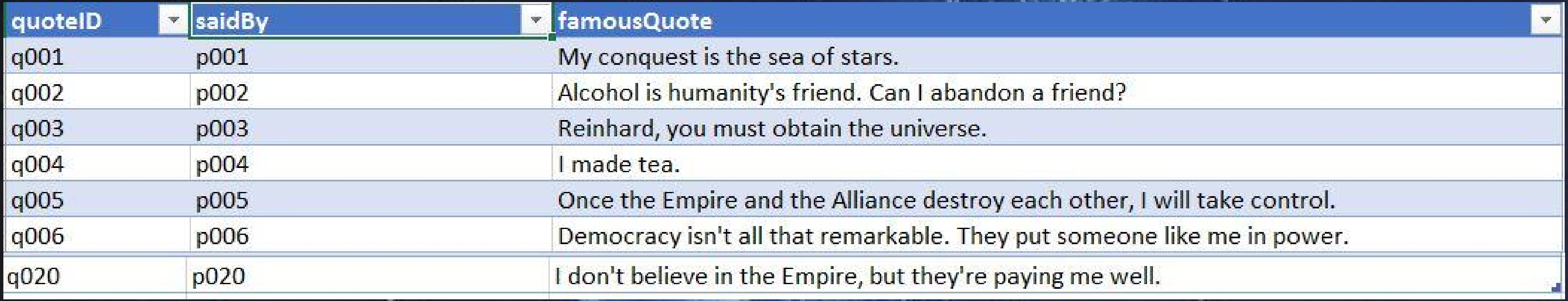
The Quotes table keeps track of the best quotes uttered by all the people in the database.



```
CREATE TABLE Quotes (
    quoteID text NOT NULL,
    saidBy text NOT NULL REFERENCES People(personID),
    famousQuote text NOT NULL,
    PRIMARY KEY(quoteID)
);
```

Functional Dependencies: quoteID → saidBy, famousQuote

Quotes Sample Data

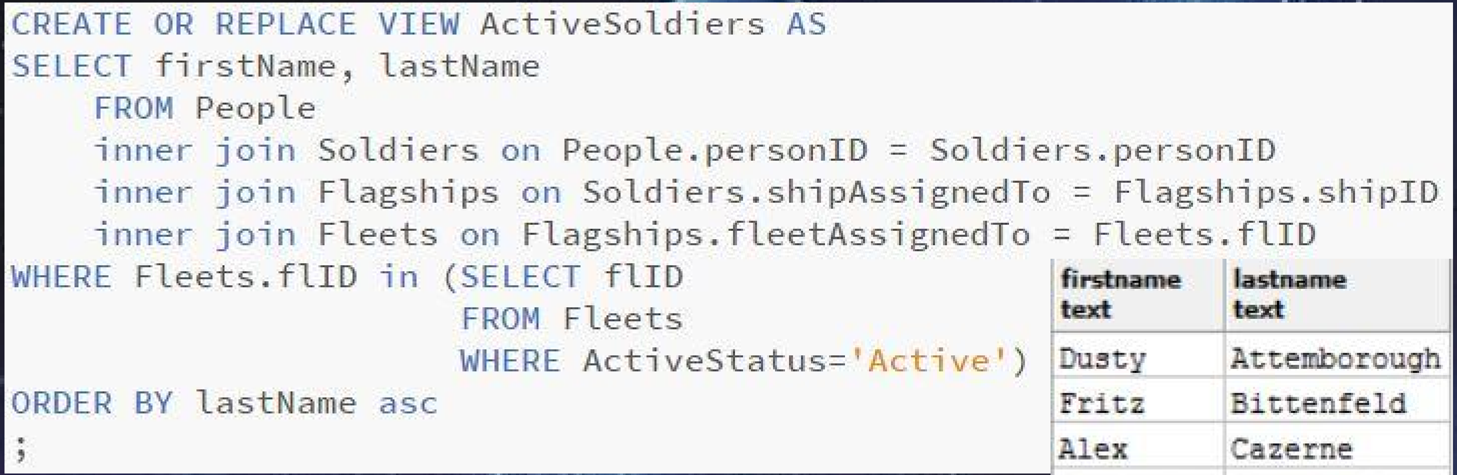


quoteID	saidBy	famousQuote
q001	p001	My conquest is the sea of stars.
q002	p002	Alcohol is humanity's friend. Can I abandon a friend?
q003	p003	Reinhard, you must obtain the universe.
q004	p004	I made tea.
q005	p005	Once the Empire and the Alliance destroy each other, I will take control.
q006	p006	Democracy isn't all that remarkable. They put someone like me in power.
q020	p020	I don't believe in the Empire, but they're paying me well.

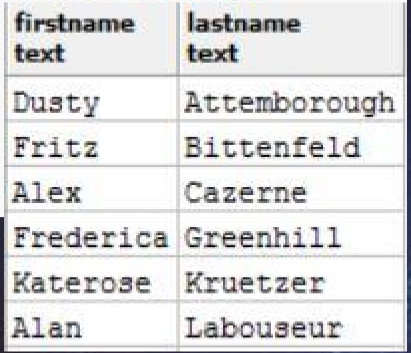
Views

ActiveSoldiers

Shows soldiers who are assigned to ships that are in active fleets.



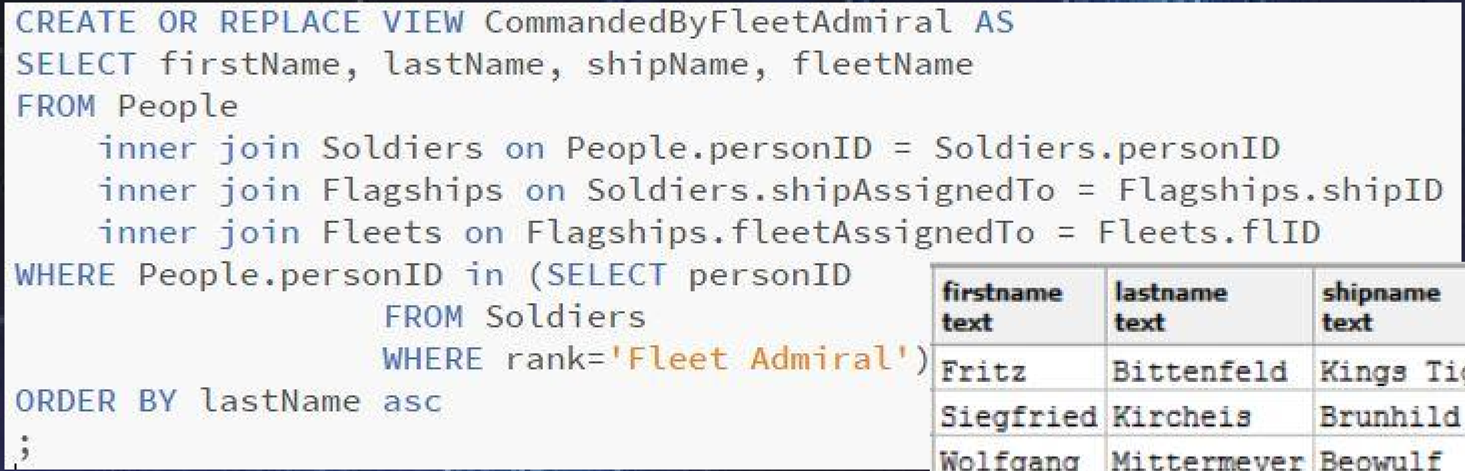
```
CREATE OR REPLACE VIEW ActiveSoldiers AS
SELECT firstName, lastName
    FROM People
    inner join Soldiers on People.personID = Soldiers.personID
    inner join Flagships on Soldiers.shipAssignedTo = Flagships.shipID
    inner join Fleets on Flagships.fleetAssignedTo = Fleets.flID
WHERE Fleets.flID in (SELECT flID
                      FROM Fleets
                      WHERE ActiveStatus='Active')
ORDER BY lastName asc
;
```



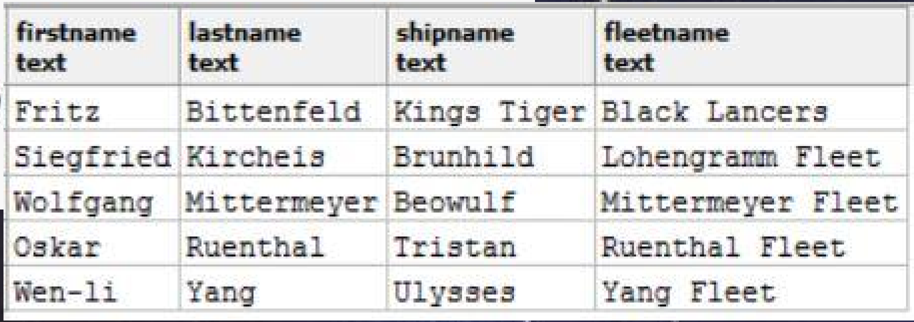
firstname text	lastname text
Dusty	Attemborough
Fritz	Bittenfeld
Alex	Cazerne
Frederica	Greenhill
Katerose	Kruetzer
Alan	Labouseur

CommandedByFleetAdmiral

Shows the name of the fleet that each Fleet Admiral commands, as well as the flagship within the fleet from which the Fleet Admiral commands from.



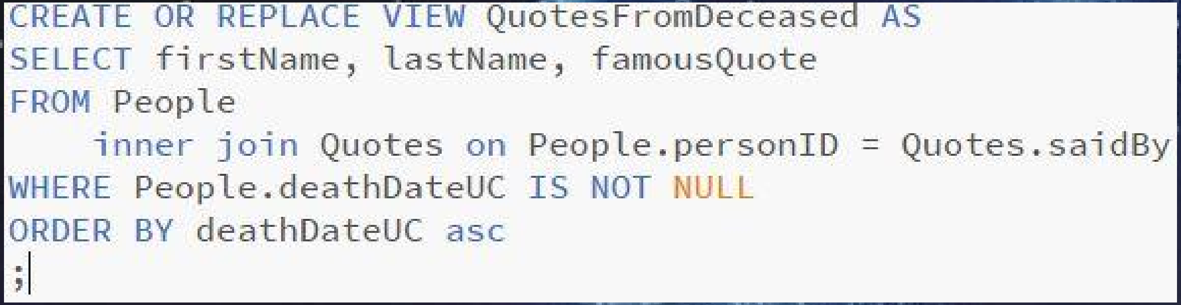
```
CREATE OR REPLACE VIEW CommandedByFleetAdmiral AS
SELECT firstName, lastName, shipName, fleetName
FROM People
    inner join Soldiers on People.personID = Soldiers.personID
    inner join Flagships on Soldiers.shipAssignedTo = Flagships.shipID
    inner join Fleets on Flagships.fleetAssignedTo = Fleets.flID
WHERE People.personID in (SELECT personID
                  FROM Soldiers
                  WHERE rank='Fleet Admiral')
ORDER BY lastName asc
;
```



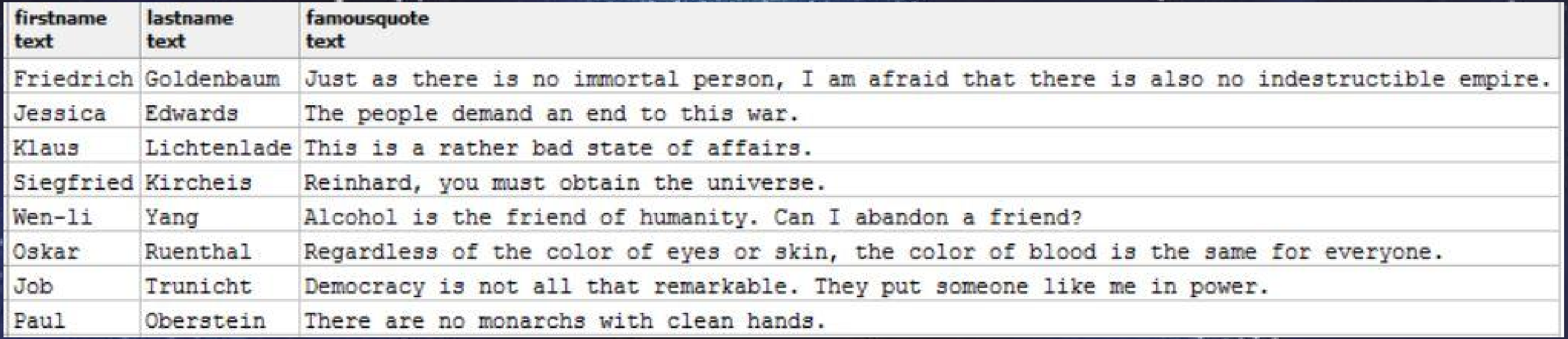
firstname text	lastname text	shipname text	fleetname text
Fritz	Bittenfeld	Kings Tiger	Black Lancers
Siegfried	Kircheis	Brunhild	Lohengramm Fleet
Wolfgang	Mittermeyer	Beowulf	Mittermeyer Fleet
Oskar	Ruenthal	Tristan	Ruenthal Fleet
Wen-li	Yang	Ulysses	Yang Fleet

QuotesFromDeceasedPeople

Shows quotes from people who have died, ordered by earliest to latest death date.



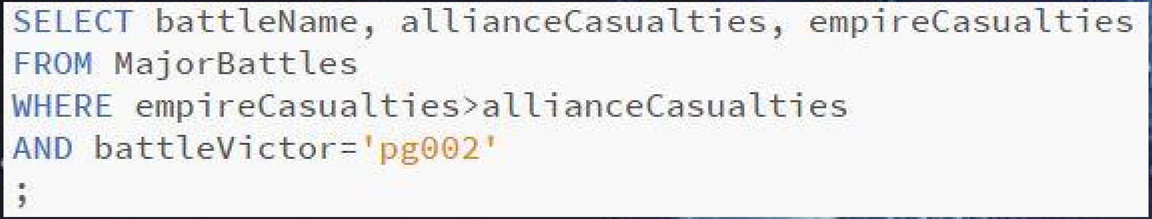
```
CREATE OR REPLACE VIEW QuotesFromDeceased AS
SELECT firstName, lastName, famousQuote
FROM People
    inner join Quotes on People.personID = Quotes.saidBy
WHERE People.deathDateUC IS NOT NULL
ORDER BY deathDateUC asc
;
```



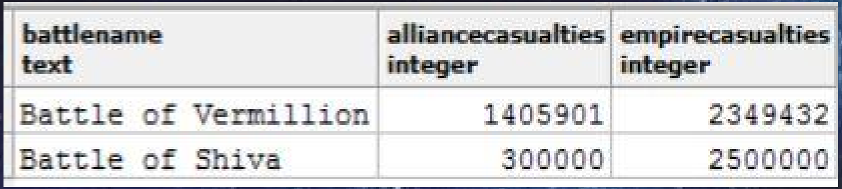
firstname text	lastname text	famousquote text
Friedrich	Goldenbaum	Just as there is no immortal person, I am afraid that there is also no indestructible empire.
Jessica	Edwards	The people demand an end to this war.
Klaus	Lichtenlade	This is a rather bad state of affairs.
Siegfried	Kircheis	Reinhard, you must obtain the universe.
Wen-li	Yang	Alcohol is the friend of humanity. Can I abandon a friend?
Oskar	Ruenthal	Regardless of the color of eyes or skin, the color of blood is the same for everyone.
Job	Trunicht	Democracy is not all that remarkable. They put someone like me in power.
Paul	Oberstein	There are no monarchs with clean hands.

Reports and Interesting Queries

Query that returns the names and casualties of battles in which the Empire suffered more casualties, but still won.

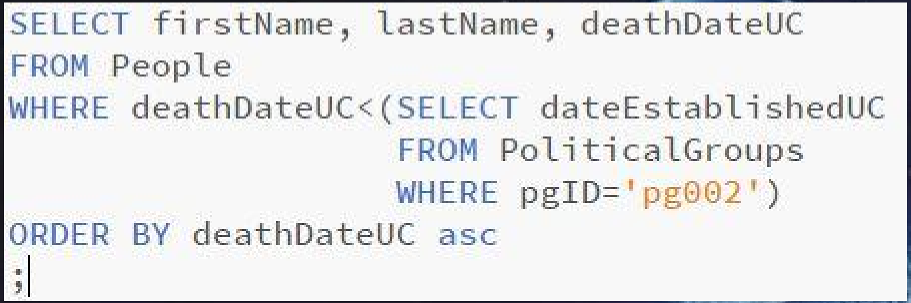


```sql
SELECT battleName, allianceCasualties, empireCasualties
FROM MajorBattles
WHERE empireCasualties>allianceCasualties
AND battleVictor='pg002'
;
```

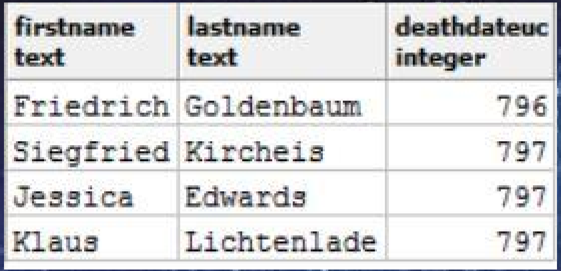


battlename text	alliancecasualties integer	empirecasualties integer
Battle of Vermillion	1405901	2349432
Battle of Shiva	300000	2500000

Query that returns the names of people who died before the formation of the New Galactic Empire, ordered by earliest to latest death date.



```
SELECT firstName, lastName, deathDateUC
FROM People
WHERE deathDateUC<(SELECT dateEstablishedUC
                   FROM PoliticalGroups
                   WHERE pgID='pg002')
ORDER BY deathDateUC asc
;
```

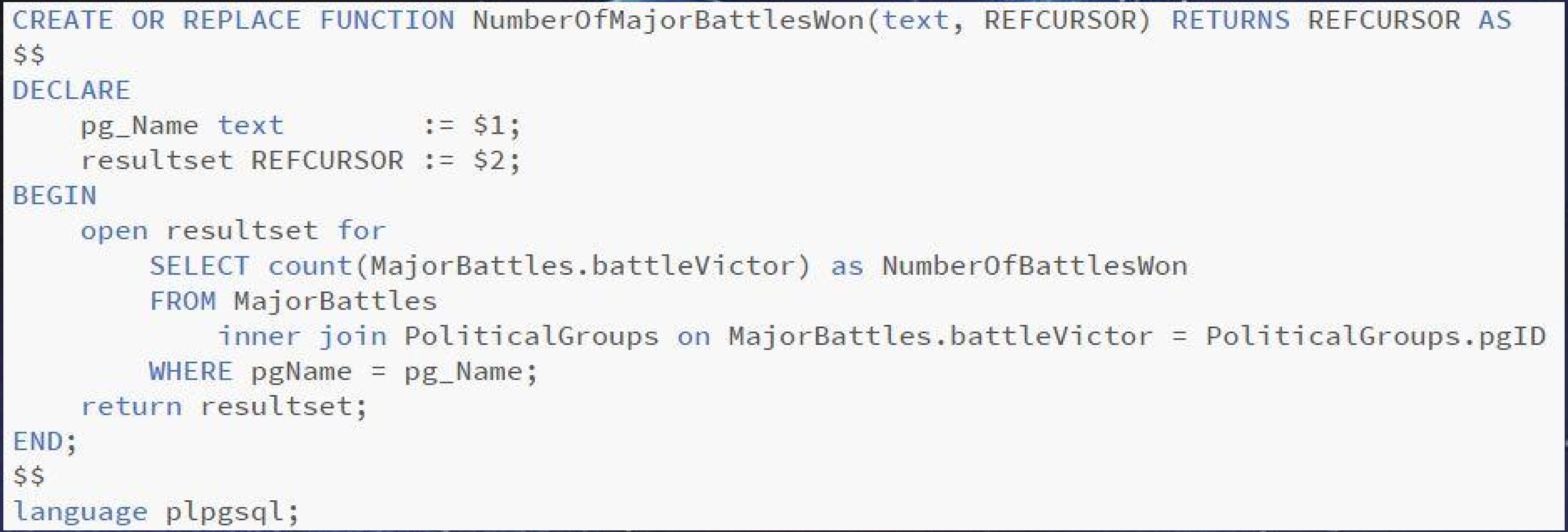


firstname text	lastname text	deathdateuc integer
Friedrich	Goldenbaum	796
Siegfried	Kircheis	797
Jessica	Edwards	797
Klaus	Lichtenlade	797

Stored Procedures

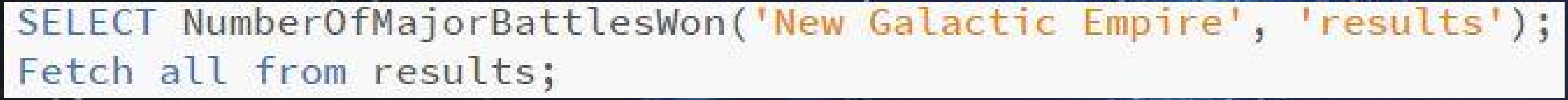
NumberOfMajorBattlesWon

Stored procedure that returns the number of battle a group has won.

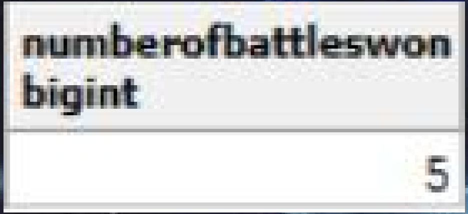


```
CREATE OR REPLACE FUNCTION NumberOfMajorBattlesWon(text, REFCURSOR) RETURNS REFCURSOR AS
$$
DECLARE
    pg_Name text        := $1;
    resultset REFCURSOR := $2;
BEGIN
    open resultset for
        SELECT count(MajorBattles.battleVictor) as NumberOfBattlesWon
        FROM MajorBattles
            inner join PoliticalGroups on MajorBattles.battleVictor = PoliticalGroups.pgID
        WHERE pgName = pg_Name;
    return resultset;
END;
$$
language plpgsql;
```

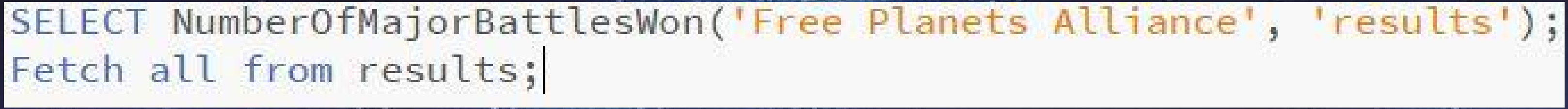
NumberOfMajorBattlesWon Samples



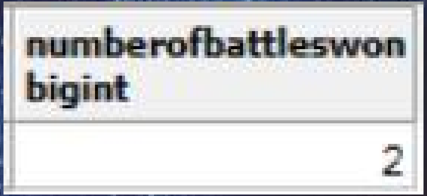
```
SELECT NumberOfMajorBattlesWon('New Galactic Empire', 'results');
Fetch all from results;
```



numberofbattleswon bigint
5

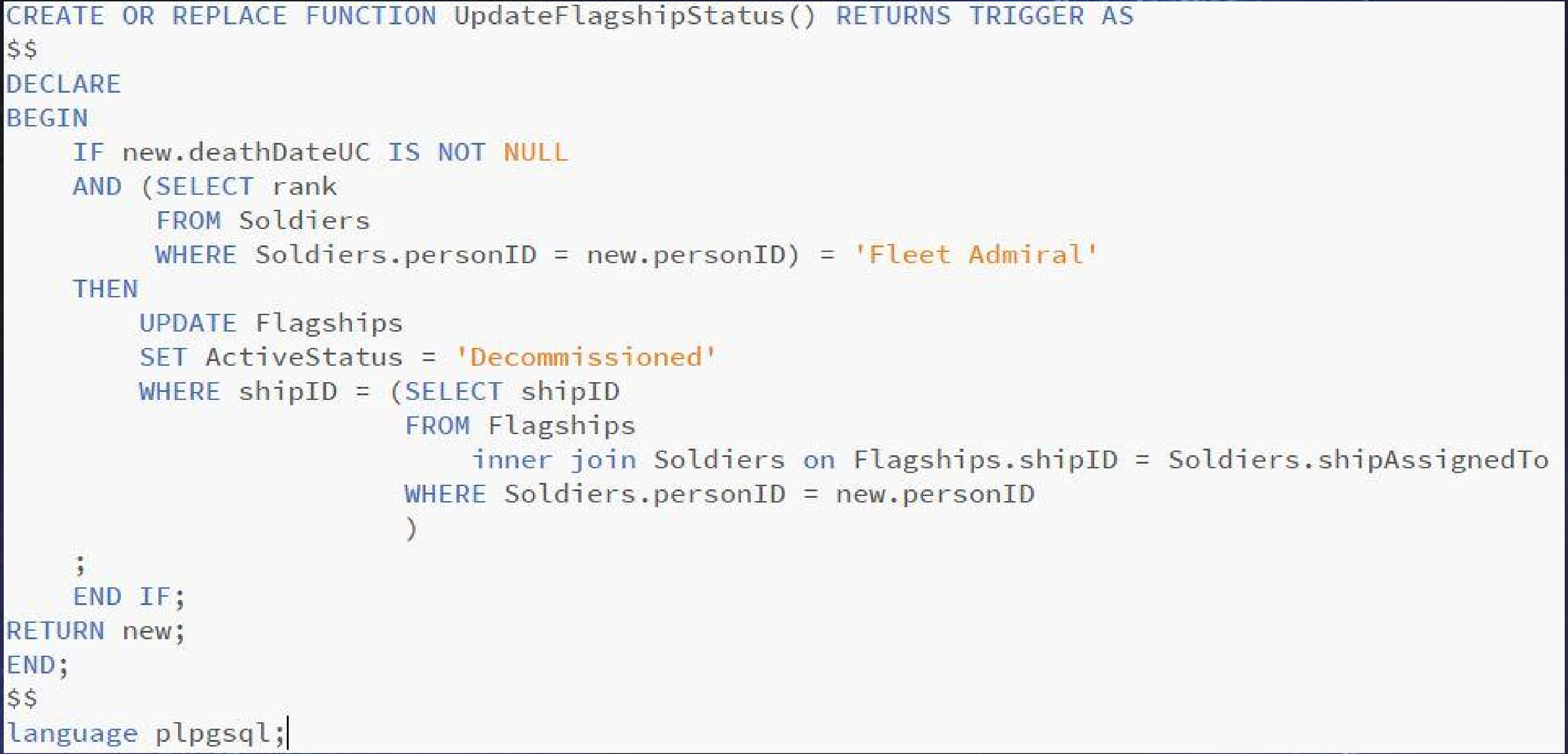


```
SELECT NumberOfMajorBattlesWon('Free Planets Alliance', 'results');
Fetch all from results;
```



numberofbattleswon bigint
2

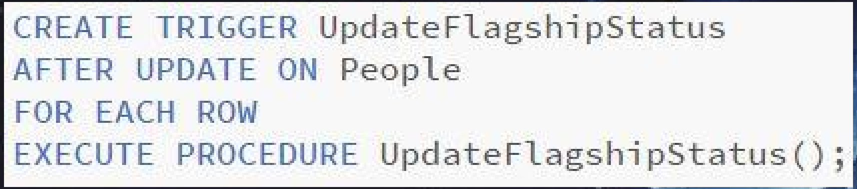
UpdateFlagshipStatus - Activated by a Trigger



```
CREATE OR REPLACE FUNCTION UpdateFlagshipStatus() RETURNS TRIGGER AS
$$
DECLARE
BEGIN
    IF new.deathDateUC IS NOT NULL
    AND (SELECT rank
         FROM Soldiers
         WHERE Soldiers.personID = new.personID) = 'Fleet Admiral'
    THEN
        UPDATE Flagships
        SET ActiveStatus = 'Decommissioned'
        WHERE shipID = (SELECT shipID
                        FROM Flagships
                            inner join Soldiers on Flagships.shipID = Soldiers.shipAssignedTo
                        WHERE Soldiers.personID = new.personID
                        )
    ;
    END IF;
RETURN new;
END;
$$
language plpgsql;
```

Trigger: UpdateFlagshipStatus

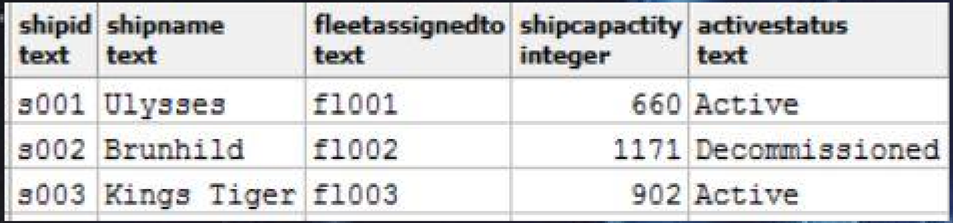
This trigger activates the stored procedure UpdateFlagshipStatus if a Fleet Admiral dies, because an Admiral always goes down with his ship.



```
CREATE TRIGGER UpdateFlagshipStatus
AFTER UPDATE ON People
FOR EACH ROW
EXECUTE PROCEDURE UpdateFlagshipStatus();
```

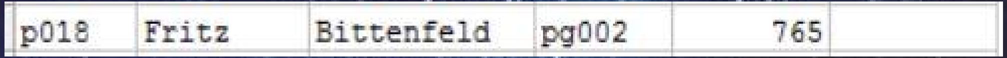
Before Trigger UpdateFlagshipStatus

Flagships



shipid text	shipname text	fleetassignedto text	shipcapactity integer	activestatus text
s001	Ulysses	f1001	660	Active
s002	Brunhild	f1002	1171	Decommissioned
s003	Kings Tiger	f1003	902	Active

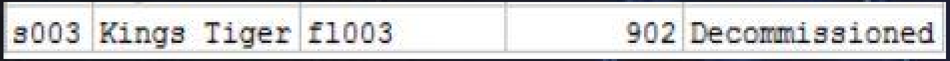
People



p018	Fritz	Bittenfeld	pg002	765	

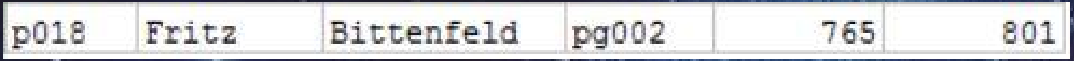
After Trigger UpdateFlagshipStatus

Flagships



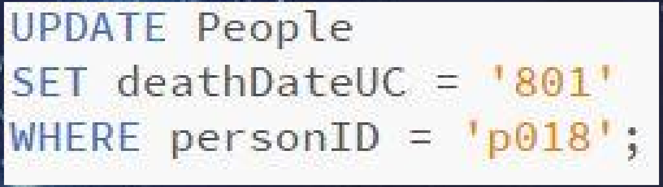
s003	Kings Tiger	f1003	902	Decommissioned

People



p018	Fritz	Bittenfeld	pg002	765	801

Triggering query



```
UPDATE People
SET deathDateUC = '801'
WHERE personID = 'p018';
```

Security

Admins are allowed all permissions on the database.

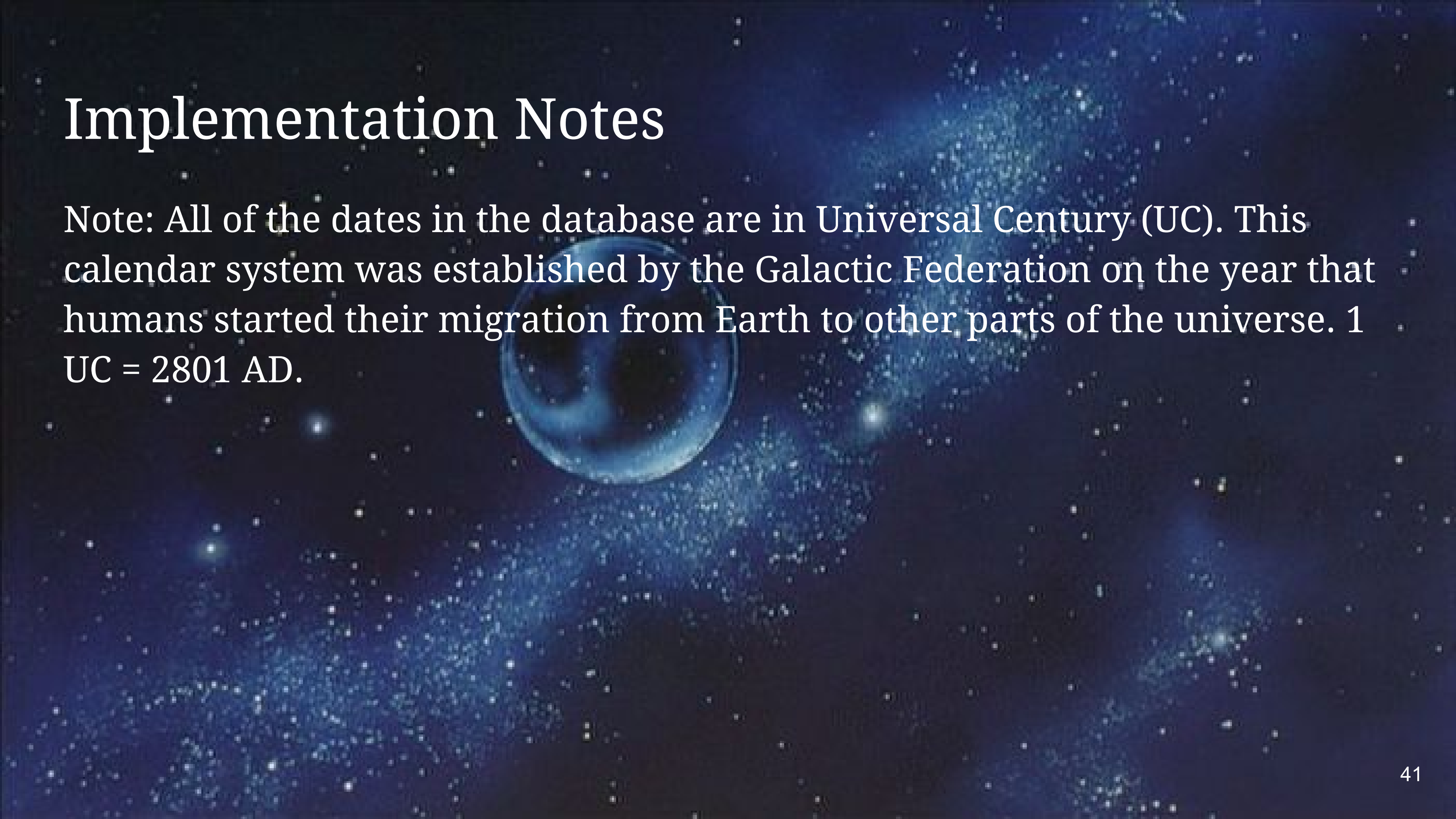
```
CREATE ROLE Admin;
GRANT ALL ON ALL TABLES IN SCHEMA PUBLIC TO Admin;
```

Historians are allowed to update, select, and insert data on all tables to keep track of new events.

```
CREATE ROLE Historian;
REVOKE ALL ON ALL TABLES IN SCHEMA PUBLIC FROM Historian;
GRANT SELECT, INSERT, UPDATE ON ALL TABLES IN SCHEMA PUBLIC TO Historian;
```

Viewers are only allowed to select data on all tables.

```
CREATE ROLE Viewer;
REVOKE ALL ON ALL TABLES IN SCHEMA PUBLIC FROM Viewer;
GRANT SELECT ON ALL TABLES IN SCHEMA PUBLIC TO Viewer;
```



Implementation Notes

Note: All of the dates in the database are in Universal Century (UC). This calendar system was established by the Galactic Federation on the year that humans started their migration from Earth to other parts of the universe. 1 UC = 2801 AD.

Known Problems

The ActiveSoldiers View does not account for dead soldiers; only those who are currently deployed on an active ship. So the View may actually show someone who has died while their ship is still intact. This problem would be most apparent during a live battle, where historians are updating the database to try to account for each lost person and ship.

Future Enhancements

- A way to see which ships participated in which battles
- Improve stored procedure for number of battles won to also convert to a percentage of the total battles participated in
- An improvement on the UpdateFlagshipStatus trigger that can be used to update a person’s death if the ship they are assigned to is destroyed in battle