Expedition 2

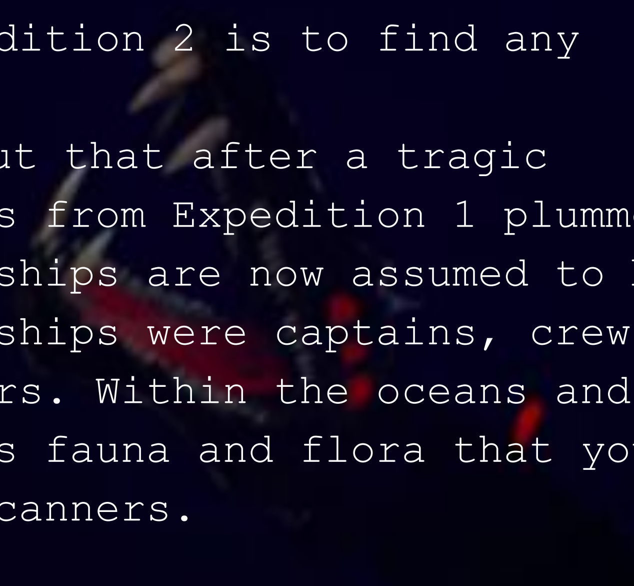
Database Design Proposal
Kevin Allegretti

TABLE OF CONTENTS

Overview

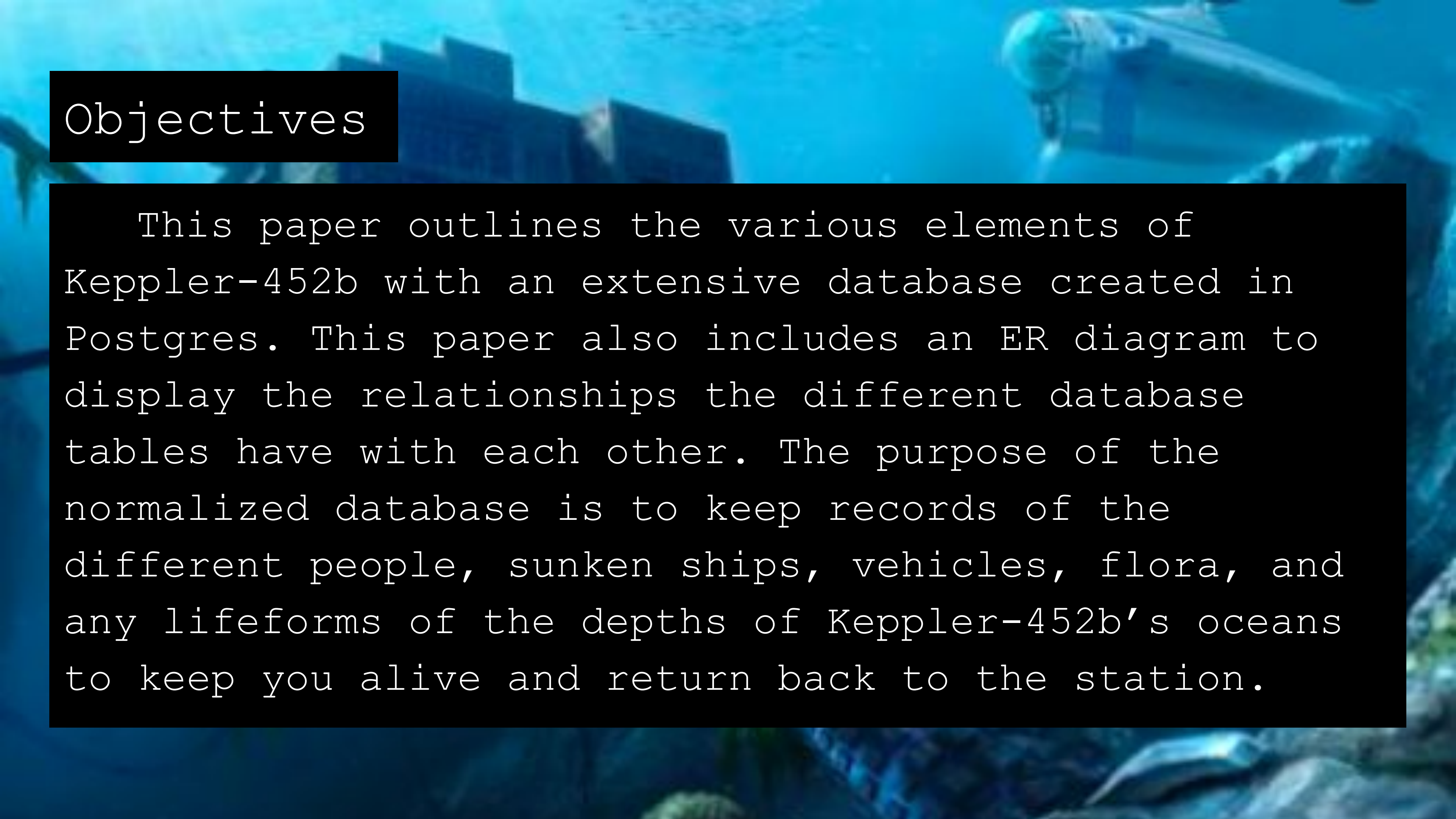
In the year 2022, 1400 thousand lightyears away from earth. You are the only survivor left on planet Keppler-452b, covered almost entirely in water. A group of space/sea ships was sent to this planet 15 years prior on “Expedition 1”. They were deployed as an exploration job with 3 different ships. However, after their supposed landing, no transmissions have been sent back to the space station that you were initially located on, outside Keppler's orbit. Your job on Expedition 2 is to find any survivors from Expedition 1.

You land safely in the water and figure out that after a tragic atmospheric breach had gone wrong, all 3 ships from Expedition 1 plummeted towards the planet at terminal velocity. The ships are now assumed to be at different depths of the ocean. Upon these ships were captains, crew workers, medical staff, and civilian passengers. Within the oceans and small islands of the planet, there are various fauna and flora that you find with different mechanical vehicles and scanners.

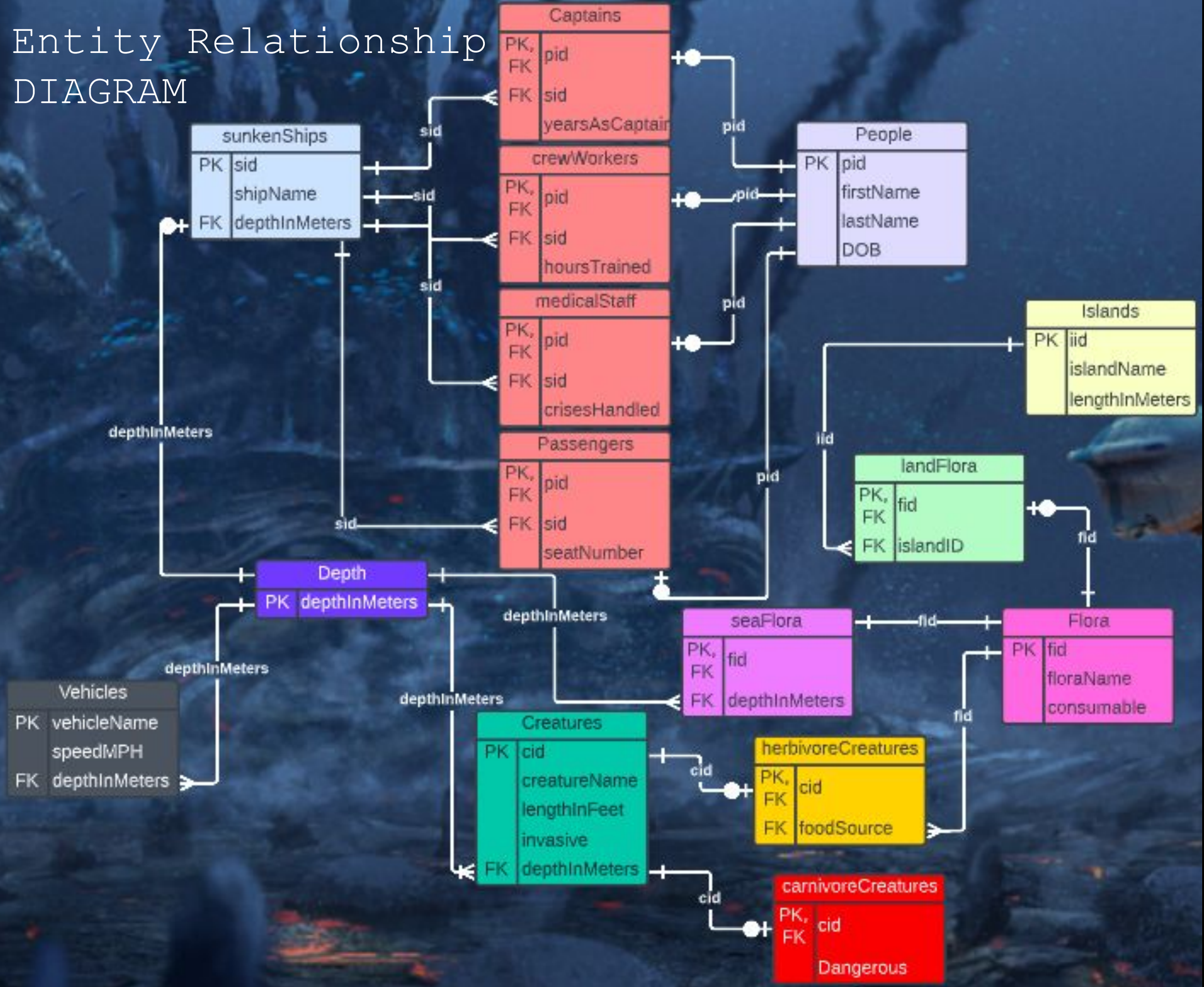


Objectives

This paper outlines the various elements of Keppler-452b with an extensive database created in Postgres. This paper also includes an ER diagram to display the relationships the different database tables have with each other. The purpose of the normalized database is to keep records of the different people, sunken ships, vehicles, flora, and any lifeforms of the depths of Keppler-452b’s oceans to keep you alive and return back to the station.



Entity Relationship DIAGRAM



Captains	
PK, FK	pid
FK	sid
	yearsAsCaptain

sunkenShips	
PK	sid
	shipName
FK	depthInMeters

crewWorkers	
PK, FK	pid
FK	sid
	hoursTrained

People	
PK	pid
	firstName
	lastName
	DOB

medicalStaff	
PK, FK	pid
FK	sid
	crisesHandled

Islands	
PK	iid
	islandName
	lengthInMeters

Passengers	
PK, FK	pid
FK	sid
	seatNumber

landFlora	
PK, FK	fid
FK	islandID

Depth	
PK	depthInMeters

seaFlora	
PK, FK	fid
FK	depthInMeters

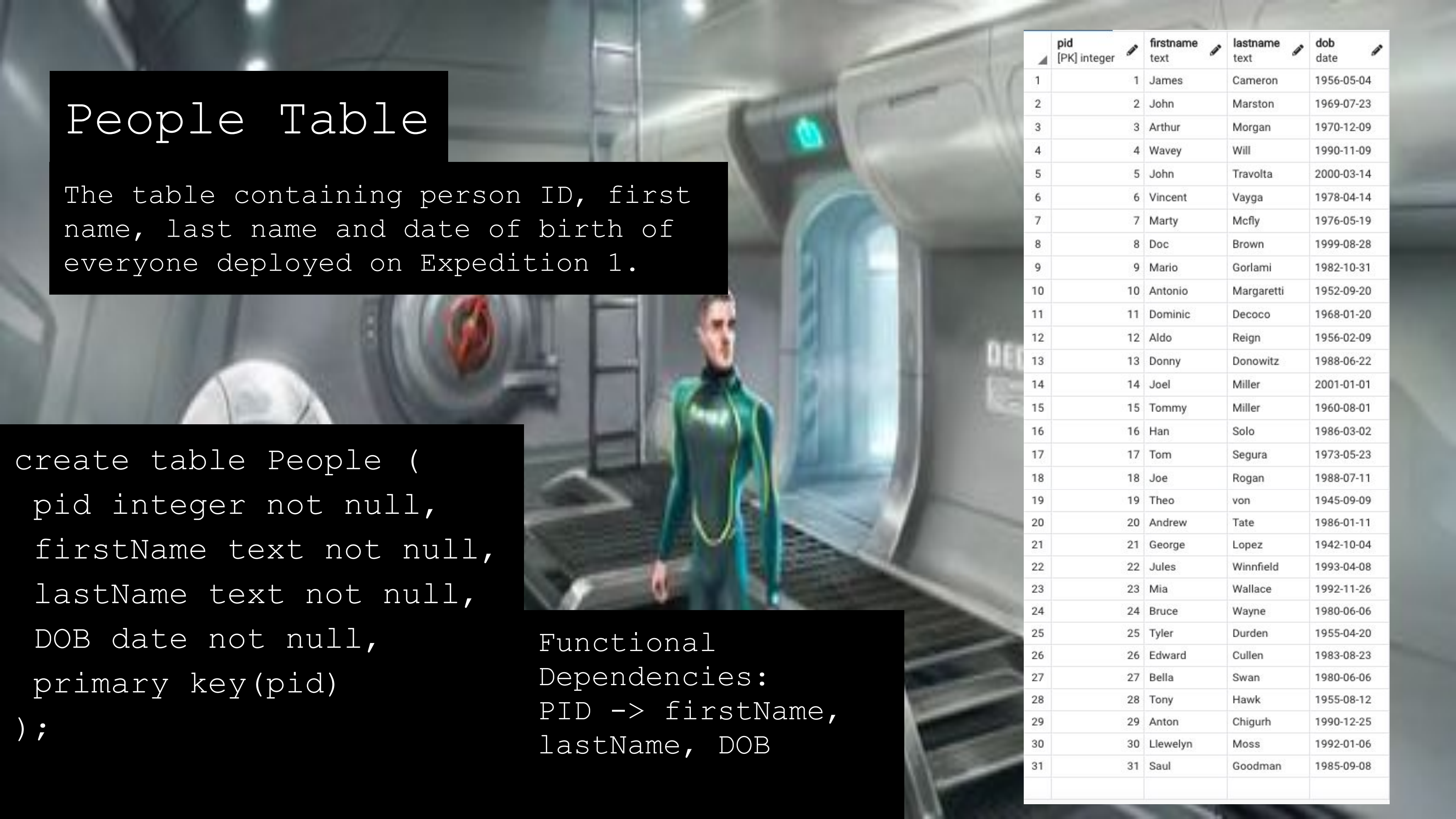
Flora	
PK	fid
	floraName
	consumable

Vehicles	
PK	vehicleName
	speedMPH
FK	depthInMeters

Creatures	
PK	cid
	creatureName
	lengthInFeet
	invasive
FK	depthInMeters

herbivoreCreatures	
PK, FK	cid
FK	foodSource

carnivoreCreatures	
PK, FK	cid
	Dangerous

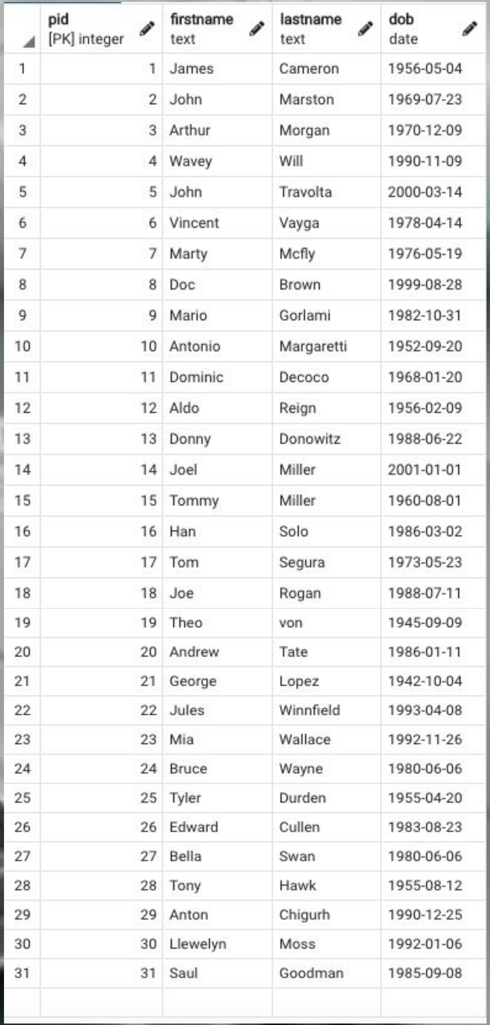


People Table

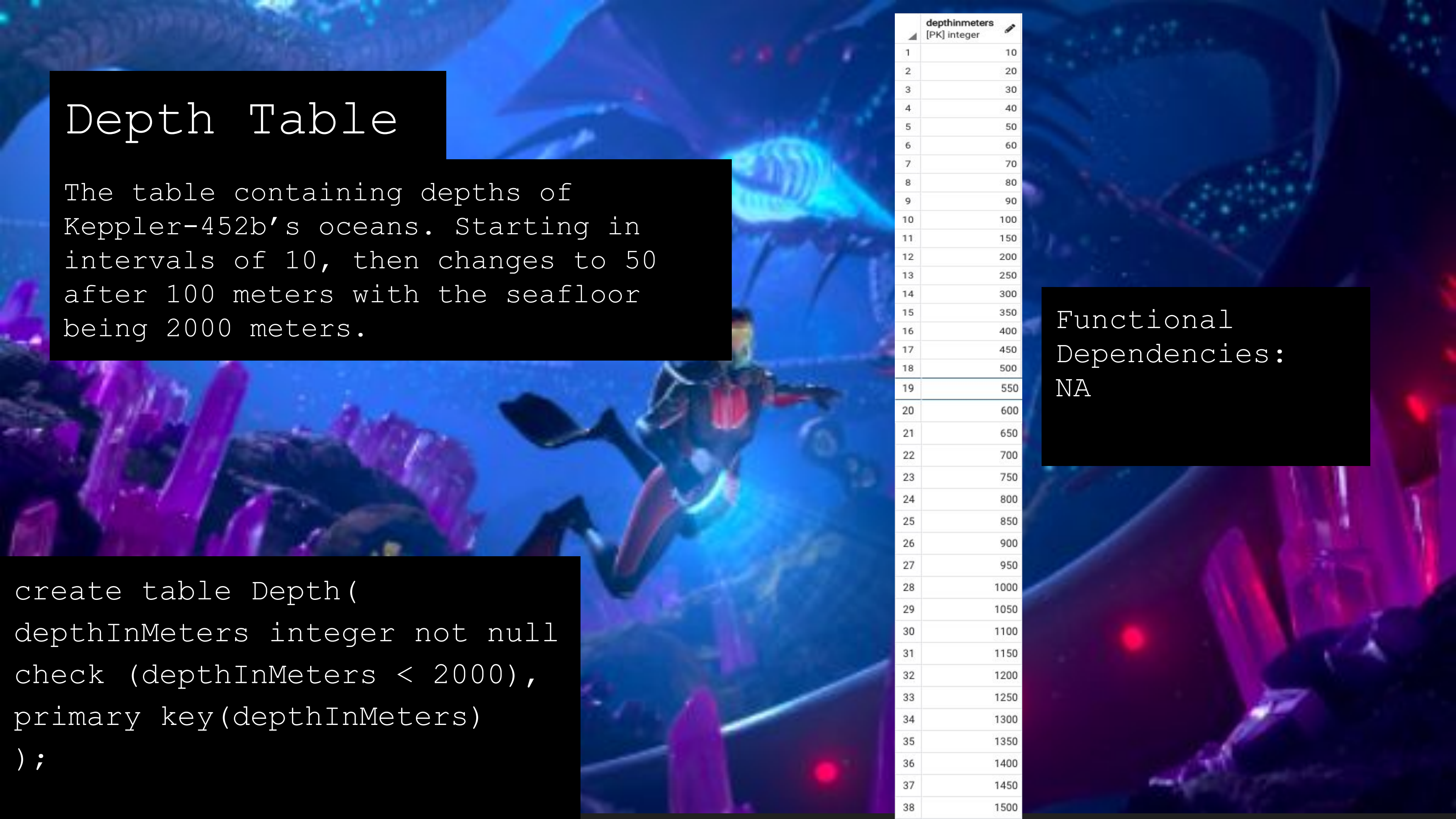
The table containing person ID, first name, last name and date of birth of everyone deployed on Expedition 1.

```
create table People (
 pid integer not null,
 firstName text not null,
 lastName text not null,
 DOB date not null,
 primary key(pid)
);
```

Functional Dependencies:
PID -> firstName, lastName, DOB



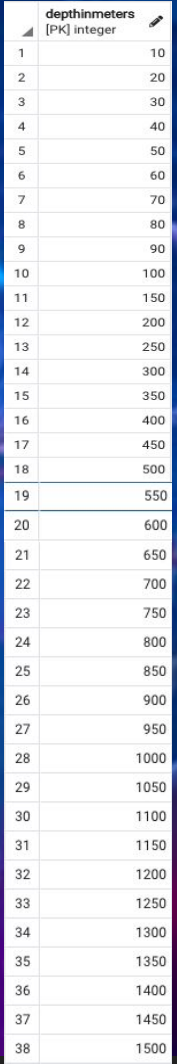
	pid [PK] integer	firstname text	lastname text	dob date
1	1	James	Cameron	1956-05-04
2	2	John	Marston	1969-07-23
3	3	Arthur	Morgan	1970-12-09
4	4	Wavey	Will	1990-11-09
5	5	John	Travolta	2000-03-14
6	6	Vincent	Vayga	1978-04-14
7	7	Marty	Mcfly	1976-05-19
8	8	Doc	Brown	1999-08-28
9	9	Mario	Gorlami	1982-10-31
10	10	Antonio	Margaretti	1952-09-20
11	11	Dominic	Decoco	1968-01-20
12	12	Aldo	Reign	1956-02-09
13	13	Donny	Donowitz	1988-06-22
14	14	Joel	Miller	2001-01-01
15	15	Tommy	Miller	1960-08-01
16	16	Han	Solo	1986-03-02
17	17	Tom	Segura	1973-05-23
18	18	Joe	Rogan	1988-07-11
19	19	Theo	von	1945-09-09
20	20	Andrew	Tate	1986-01-11
21	21	George	Lopez	1942-10-04
22	22	Jules	Winnfield	1993-04-08
23	23	Mia	Wallace	1992-11-26
24	24	Bruce	Wayne	1980-06-06
25	25	Tyler	Durden	1955-04-20
26	26	Edward	Cullen	1983-08-23
27	27	Bella	Swan	1980-06-06
28	28	Tony	Hawk	1955-08-12
29	29	Anton	Chigurh	1990-12-25
30	30	Llewelyn	Moss	1992-01-06
31	31	Saul	Goodman	1985-09-08



Depth Table

The table containing depths of Keppler-452b’s oceans. Starting in intervals of 10, then changes to 50 after 100 meters with the seafloor being 2000 meters.

```
create table Depth(
depthInMeters integer not null
check (depthInMeters < 2000),
primary key(depthInMeters)
);
```

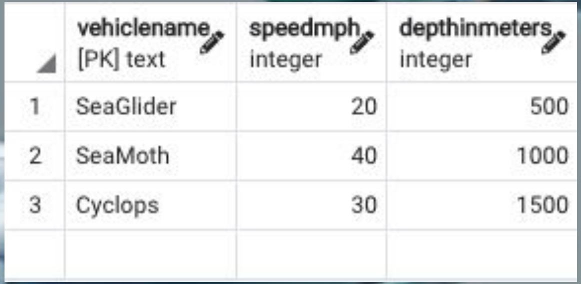


	depthinmeters [PK] integer
1	10
2	20
3	30
4	40
5	50
6	60
7	70
8	80
9	90
10	100
11	150
12	200
13	250
14	300
15	350
16	400
17	450
18	500
19	550
20	600
21	650
22	700
23	750
24	800
25	850
26	900
27	950
28	1000
29	1050
30	1100
31	1150
32	1200
33	1250
34	1300
35	1350
36	1400
37	1450
38	1500

Functional Dependencies:
NA

Vehicles Table

The table containing the vehicles used to explore the deep ocean. This table features the vehicle names, the speed in miles per hour, and the lowest possible depth the vehicle can travel.



	vehiclename [PK] text	speedmph integer	depthinmeters integer
1	SeaGlider	20	500
2	SeaMoth	40	1000
3	Cyclops	30	1500

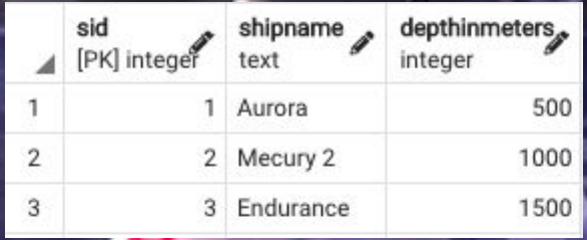
```
create table Vehicles(
vehicleName text not null,
speedMPH integer not null,
depthInMeters integer not null,
primary key(vehicleName),
foreign key(depthInMeters)
references depth(depthInMeters)
);
```

Functional Dependencies:
vehicleName -> speedMPH, depthInMeters

SunkenShips Table

The table containing the ships that are sunken from Expedition 1. This table features the ship ID, the ship name, and how many meters deep the ship is in the ocean.

```
create table sunkenShips(
 sid integer not null,
 shipName text not null,
 depthInMeters integer not
null,
 primary key(sid),
 foreign key(depthInMeters)
references
Depth(depthInMeters)
);
```

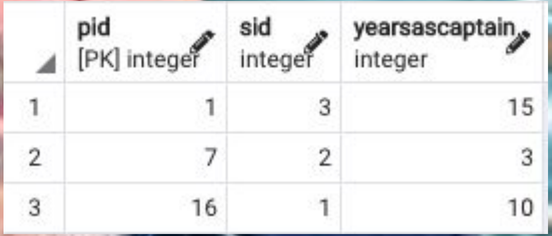


	sid [PK] integer	shipname text	depthinmeters integer
1	1	Aurora	500
2	2	Mecury 2	1000
3	3	Endurance	1500

Functional Dependencies:
sid -> shipName, depthInMeters

Captains Table

The table containing the captains of each ship from Expedition 1. This table features the person ID, the ship ID, and the amount of years the person has been a captain.



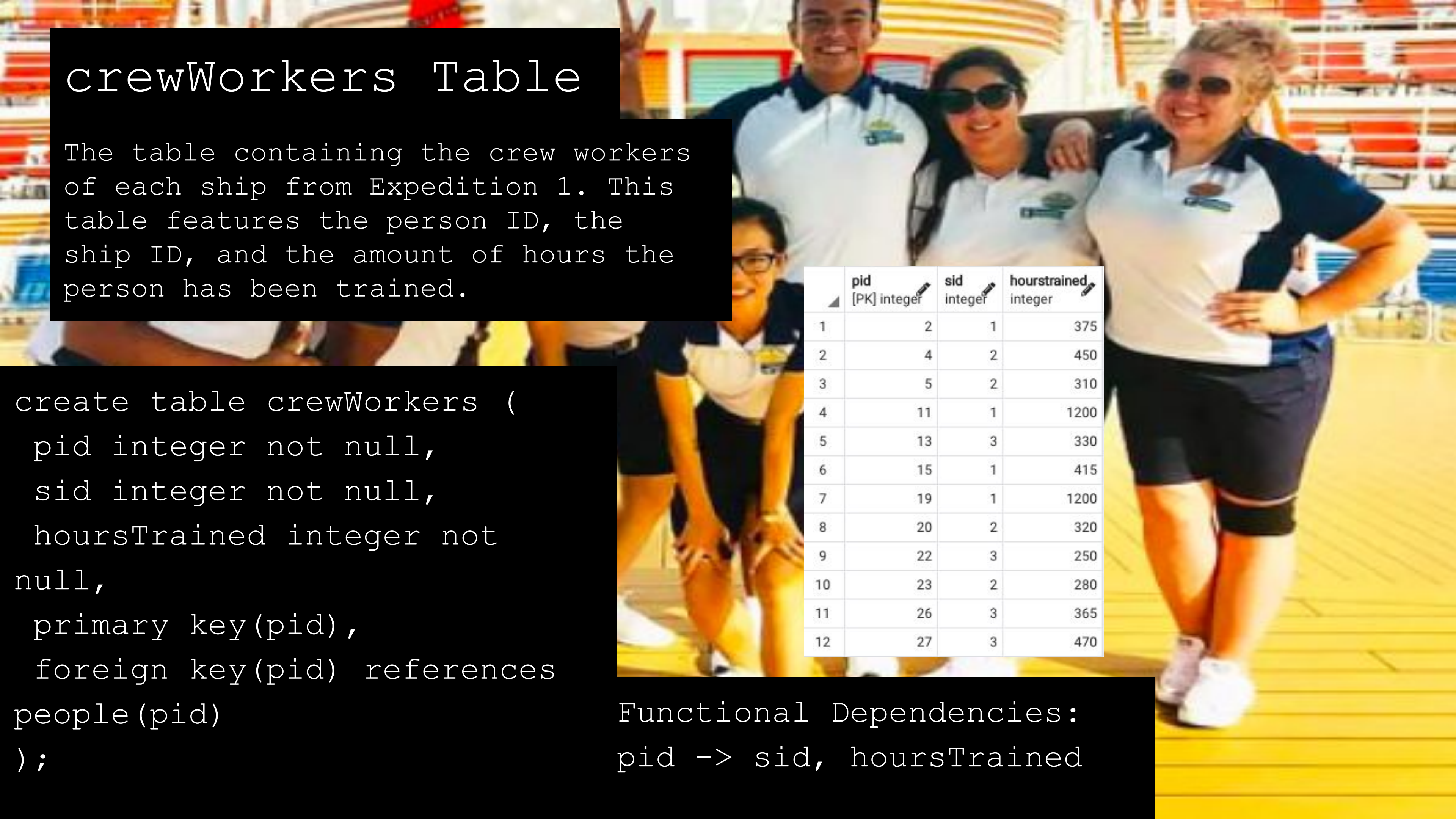
	pid [PK] integer	sid integer	yearsascaptain integer
1	1	3	15
2	7	2	3
3	16	1	10

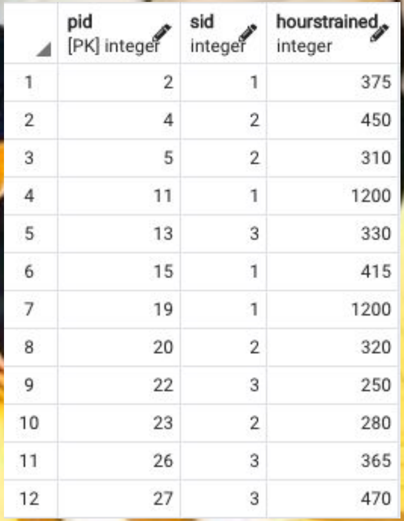
```
create table Captains (
 pid integer not null,
 sid integer not null,
 yearsAsCaptain integer not null,
 primary key(pid)
 foreign key(pid) references
people(pid)
);
```

Functional Dependencies:
pid -> sid, yearsAsCaptain

crewWorkers Table

The table containing the crew workers of each ship from Expedition 1. This table features the person ID, the ship ID, and the amount of hours the person has been trained.





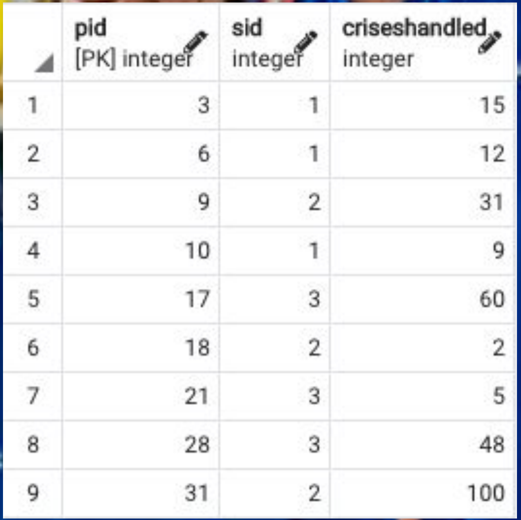
	pid [PK] integer	sid integer	hourstrained integer
1	2	1	375
2	4	2	450
3	5	2	310
4	11	1	1200
5	13	3	330
6	15	1	415
7	19	1	1200
8	20	2	320
9	22	3	250
10	23	2	280
11	26	3	365
12	27	3	470

```
create table crewWorkers (
 pid integer not null,
 sid integer not null,
 hoursTrained integer not 
null,
 primary key(pid),
 foreign key(pid) references 
people(pid)
);
```

Functional Dependencies:
pid -> sid, hoursTrained

medicalStaff Table

The table containing the medical staff of each ship from Expedition 1. This table features the person ID, the ship ID, and the amount of crises the person has been handled.



	pid [PK] integer	sid integer	criseshandled integer
1	3	1	15
2	6	1	12
3	9	2	31
4	10	1	9
5	17	3	60
6	18	2	2
7	21	3	5
8	28	3	48
9	31	2	100

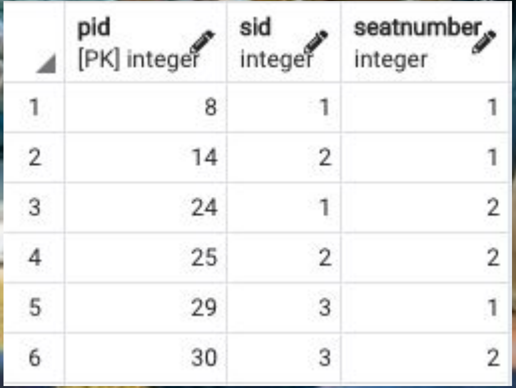
```
create table medicalStaff (
 pid integer not null,
 sid integer not null,
 crisesHandled integer not
null,
 primary key(pid)
 foreign key(pid) references
people(pid)
);
```

Functional Dependencies:

pid -> sid, crisesHandled

Passengers Table

The table containing the passengers of each ship from Expedition 1. This table features the person ID, the ship ID, and the persons seat number



	pid [PK] integer	sid integer	seatnumber integer
1	8	1	1
2	14	2	1
3	24	1	2
4	25	2	2
5	29	3	1
6	30	3	2

```
create table Passengers (
 pid integer not null,
 sid integer not null,
 seatNumber integer not null,
 primary key(pid)
 foreign key(pid) references
people(pid)
);
```

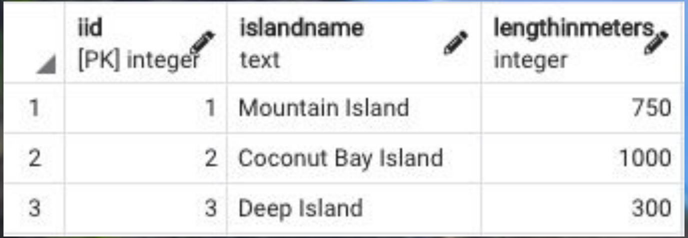
Functional Dependencies:

pid -> sid, seatNumber

Islands Table

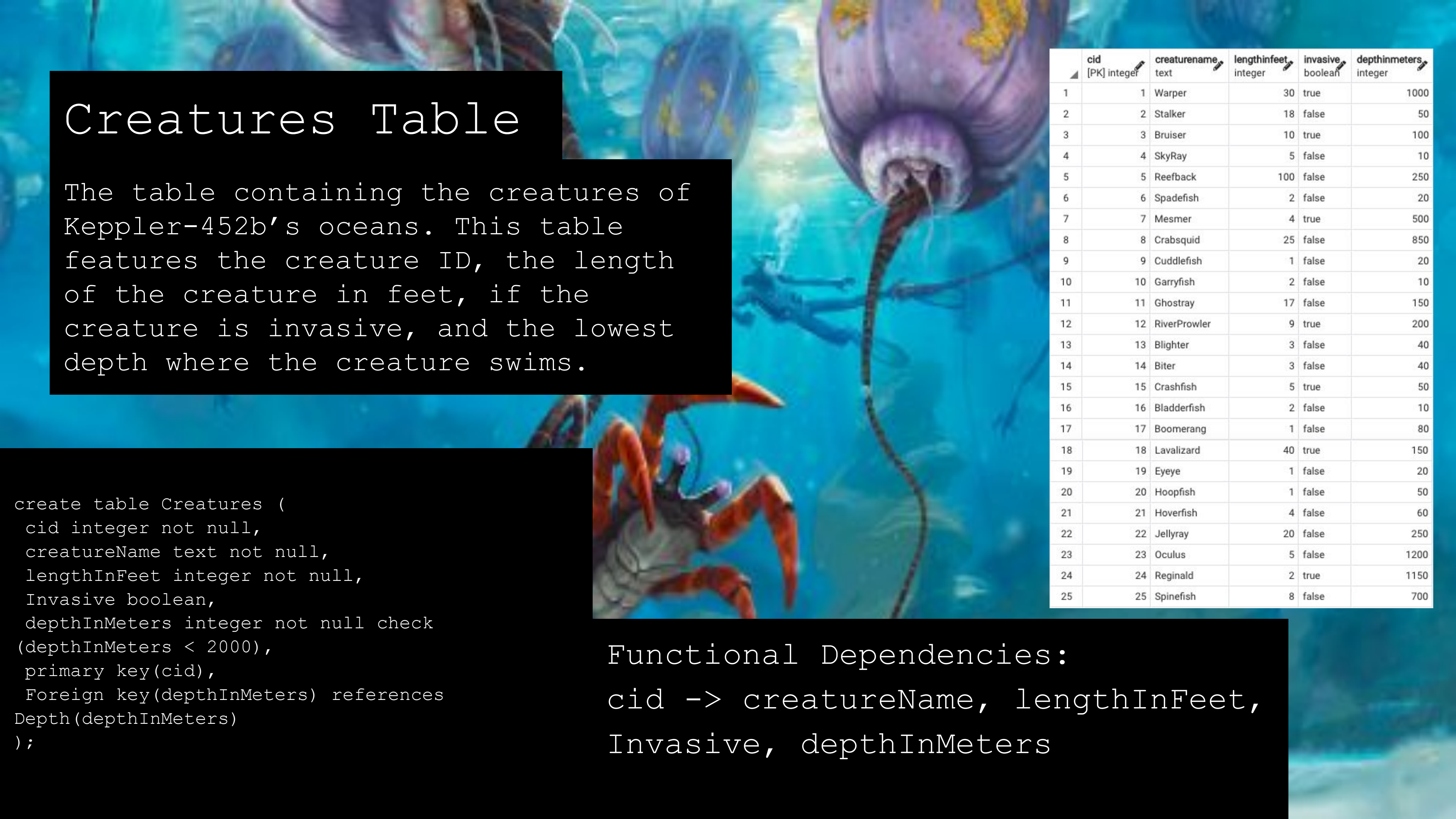
The table containing the islands of Keppler-452b. This table features the island ID, the island name, and the length of the island in meters.

```
create table Islands(
iid integer not null,
islandName text not null,
lengthInMeters integer
not null,
primary key(iid)
);
```



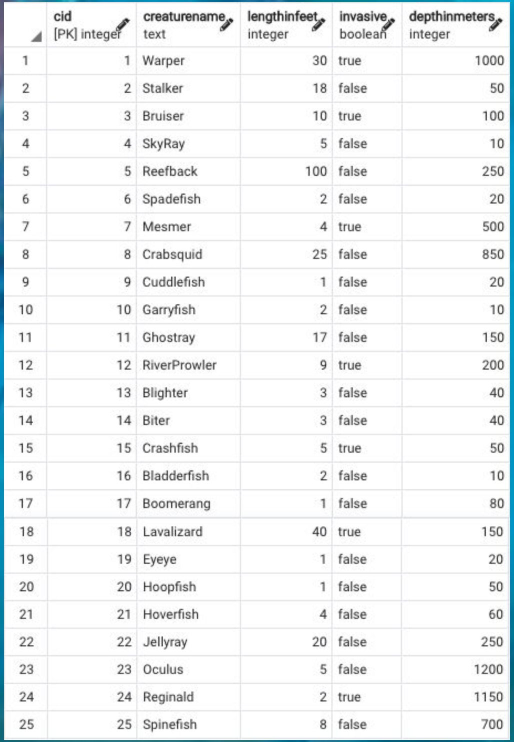
	iid [PK] integer	islandname text	lengthinmeters integer
1	1	Mountain Island	750
2	2	Coconut Bay Island	1000
3	3	Deep Island	300

Functional Dependencies:
iid -> IslandName, lengthInMeters



Creatures Table

The table containing the creatures of Keppler-452b’s oceans. This table features the creature ID, the length of the creature in feet, if the creature is invasive, and the lowest depth where the creature swims.

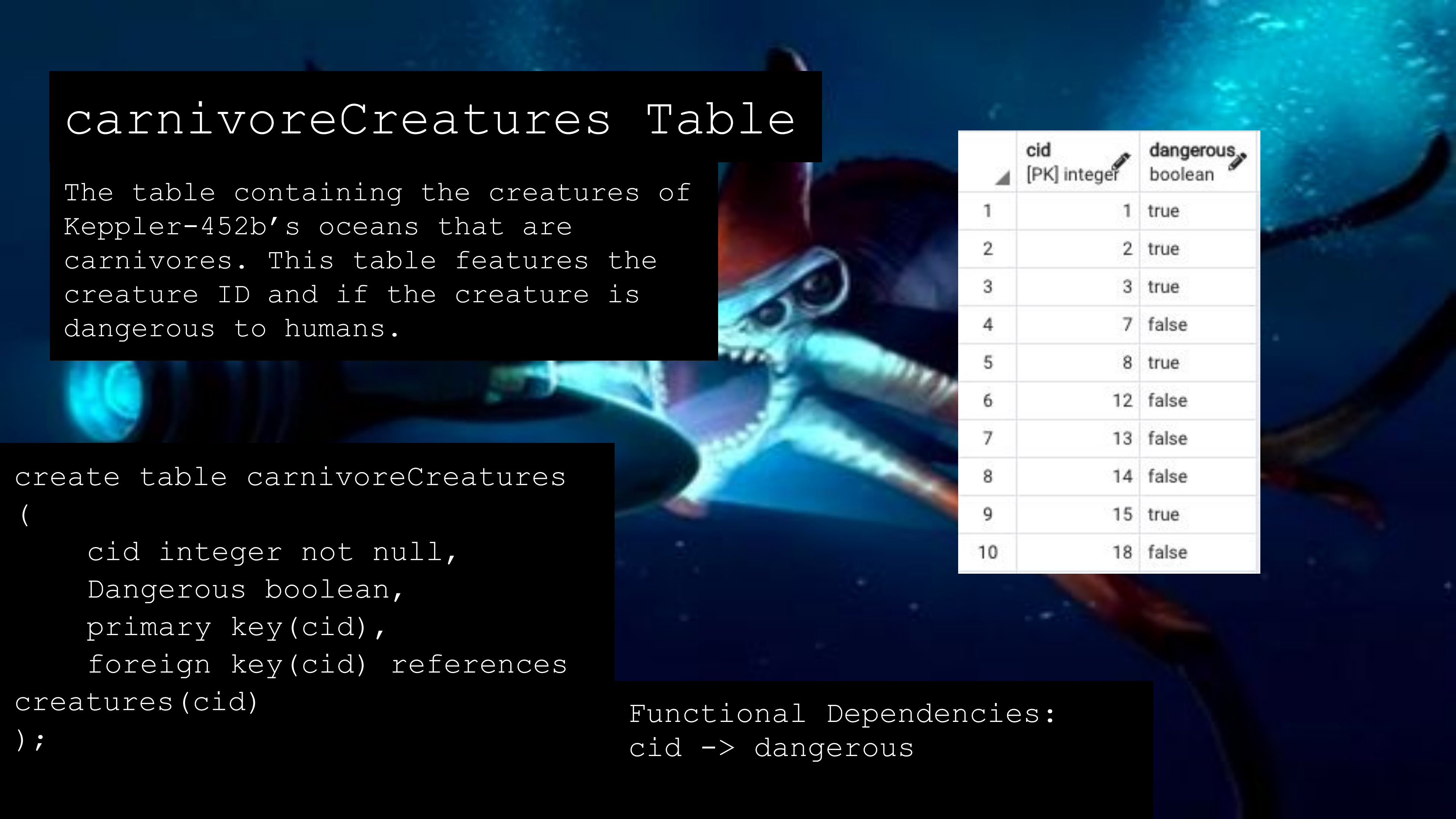


	cid [PK] integer	creaturename text	lengthinfeet integer	invasive boolean	depthinmeters integer
1	1	Warper	30	true	1000
2	2	Stalker	18	false	50
3	3	Bruiser	10	true	100
4	4	SkyRay	5	false	10
5	5	Reefback	100	false	250
6	6	Spadefish	2	false	20
7	7	Mesmer	4	true	500
8	8	Crabsquid	25	false	850
9	9	Cuddlefish	1	false	20
10	10	Garryfish	2	false	10
11	11	Ghostray	17	false	150
12	12	RiverProwler	9	true	200
13	13	Blighter	3	false	40
14	14	Biter	3	false	40
15	15	Crashfish	5	true	50
16	16	Bladderfish	2	false	10
17	17	Boomerang	1	false	80
18	18	Lavalizard	40	true	150
19	19	Eyeye	1	false	20
20	20	Hoopfish	1	false	50
21	21	Hoverfish	4	false	60
22	22	Jellyray	20	false	250
23	23	Oculus	5	false	1200
24	24	Reginald	2	true	1150
25	25	Spinefish	8	false	700

```
create table Creatures (
 cid integer not null,
 creatureName text not null,
 lengthInFeet integer not null,
 Invasive boolean,
 depthInMeters integer not null check
(depthInMeters < 2000),
 primary key(cid),
 Foreign key(depthInMeters) references
Depth(depthInMeters)
);
```

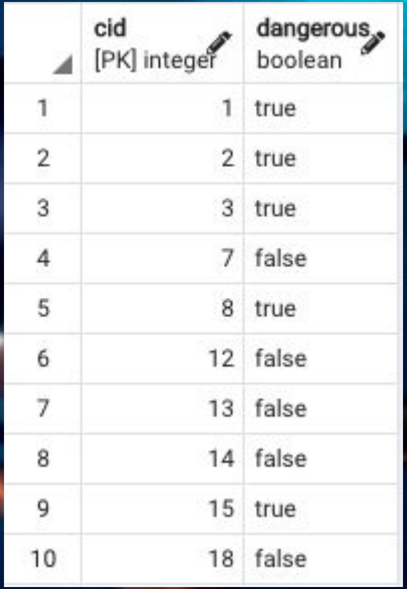
Functional Dependencies:

cid -> creatureName, lengthInFeet, Invasive, depthInMeters



carnivoreCreatures Table

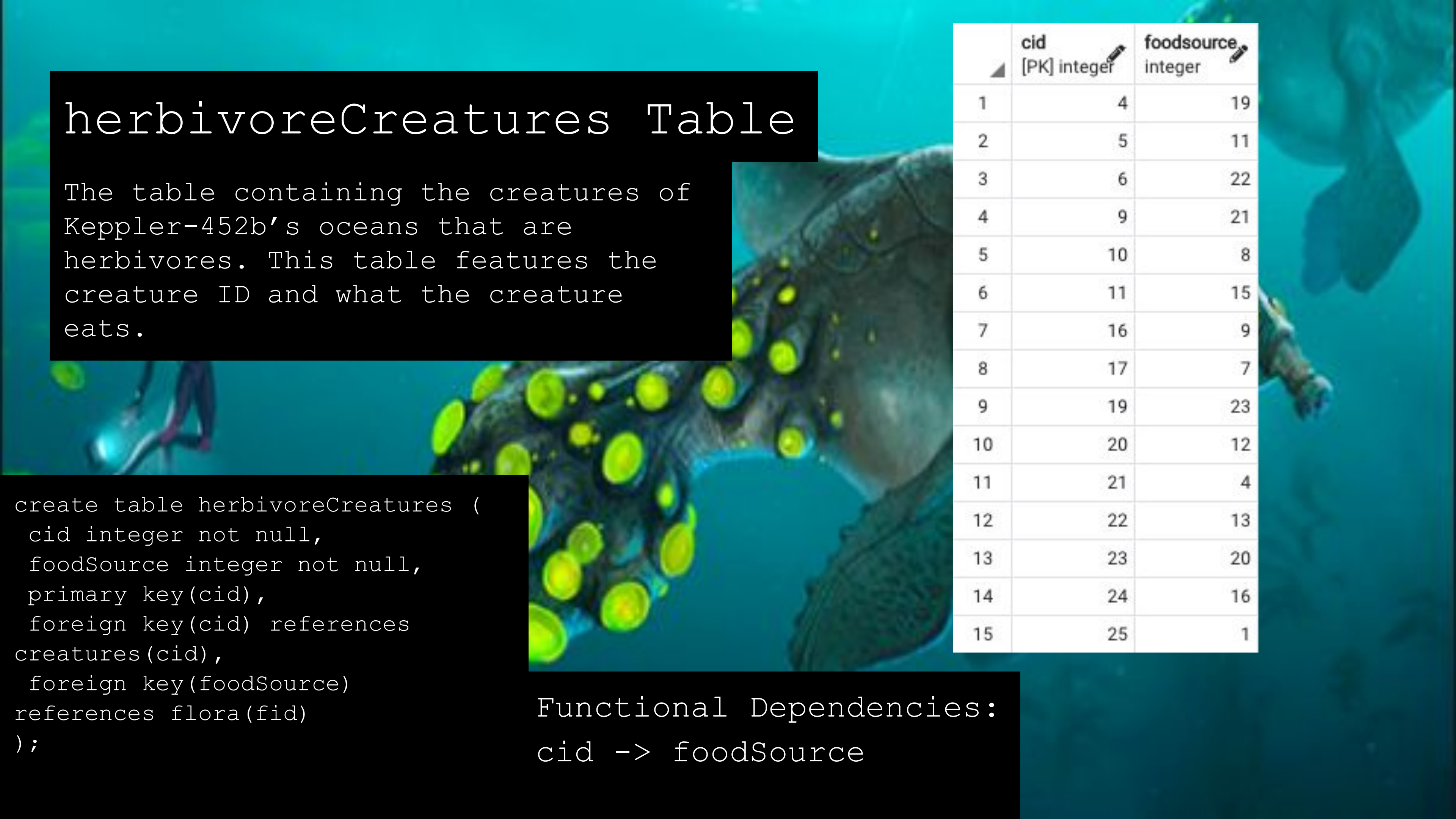
The table containing the creatures of Keppler-452b's oceans that are carnivores. This table features the creature ID and if the creature is dangerous to humans.



	cid [PK] integer	dangerous boolean
1	1	true
2	2	true
3	3	true
4	7	false
5	8	true
6	12	false
7	13	false
8	14	false
9	15	true
10	18	false

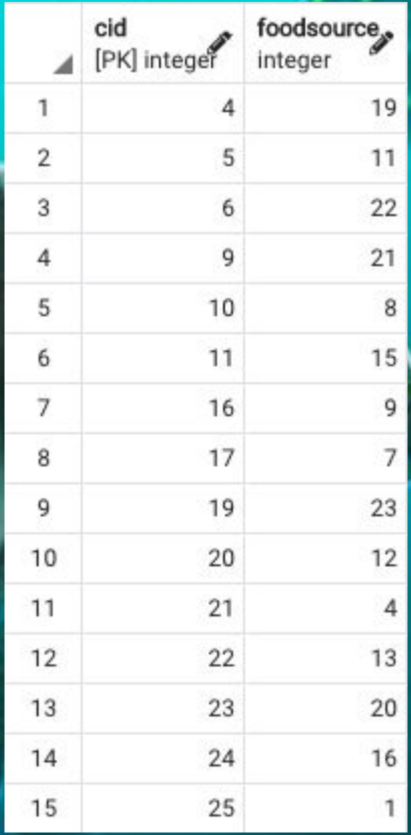
```
create table carnivoreCreatures
(
    cid integer not null,
    Dangerous boolean,
    primary key(cid),
    foreign key(cid) references
creatures(cid)
);
```

Functional Dependencies:
cid -> dangerous



herbivoreCreatures Table

The table containing the creatures of Keppler-452b's oceans that are herbivores. This table features the creature ID and what the creature eats.



	cid [PK] integer	foodsource integer
1	4	19
2	5	11
3	6	22
4	9	21
5	10	8
6	11	15
7	16	9
8	17	7
9	19	23
10	20	12
11	21	4
12	22	13
13	23	20
14	24	16
15	25	1

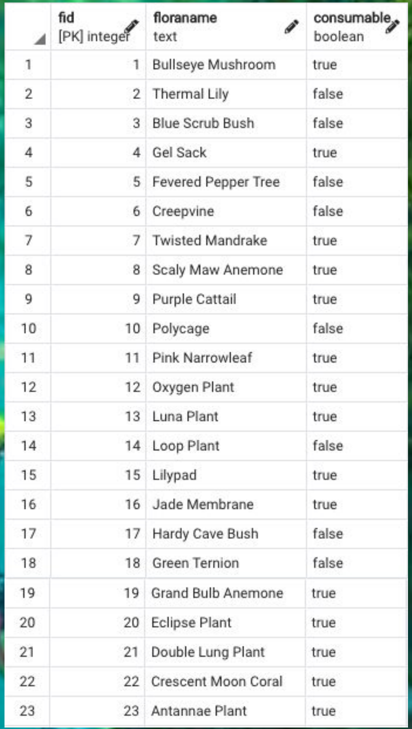
```
create table herbivoreCreatures (
 cid integer not null,
 foodSource integer not null,
 primary key(cid),
 foreign key(cid) references
creatures(cid),
 foreign key(foodSource)
references flora(fid)
);
```

Functional Dependencies:

cid -> foodSource

Flora Table

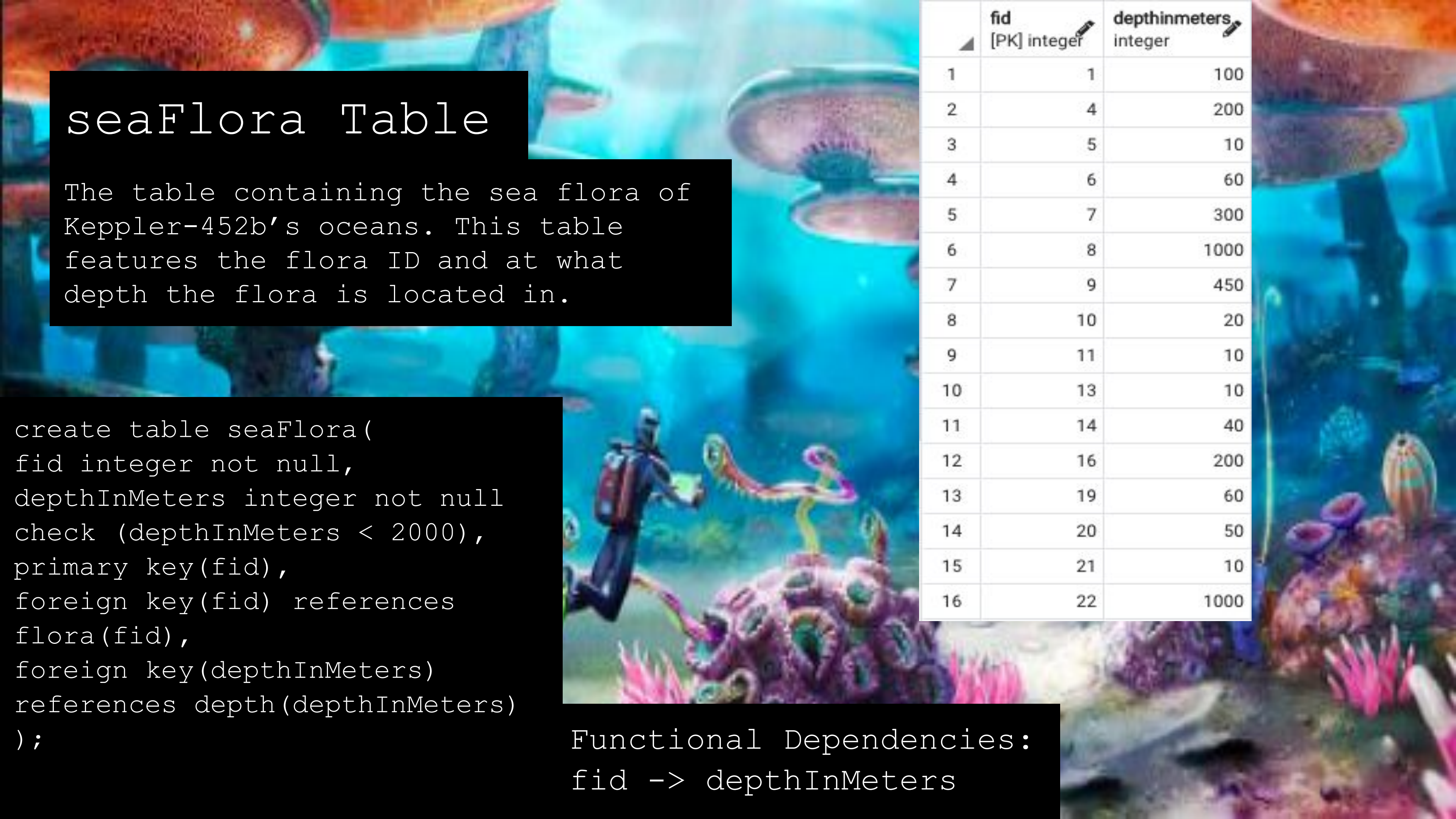
The table containing the flora of Keppler-452b’s oceans and islands. This table features the flora ID, the flora name, and if the flora is consumable or not.



	fid [PK] integer	floraname text	consumable boolean
1	1	Bullseye Mushroom	true
2	2	Thermal Lily	false
3	3	Blue Scrub Bush	false
4	4	Gel Sack	true
5	5	Fevered Pepper Tree	false
6	6	Creepvine	false
7	7	Twisted Mandrake	true
8	8	Scaly Maw Anemone	true
9	9	Purple Cattail	true
10	10	Polycage	false
11	11	Pink Narrowleaf	true
12	12	Oxygen Plant	true
13	13	Luna Plant	true
14	14	Loop Plant	false
15	15	Lilypad	true
16	16	Jade Membrane	true
17	17	Hardy Cave Bush	false
18	18	Green Ternion	false
19	19	Grand Bulb Anemone	true
20	20	Eclipse Plant	true
21	21	Double Lung Plant	true
22	22	Crescent Moon Coral	true
23	23	Antannae Plant	true

```
create table flora (
 fid integer not null,
 floraName text not null,
 Consumable boolean,
 primary key(fid),
 Foreign key(fid) references
flora(fid)
);
```

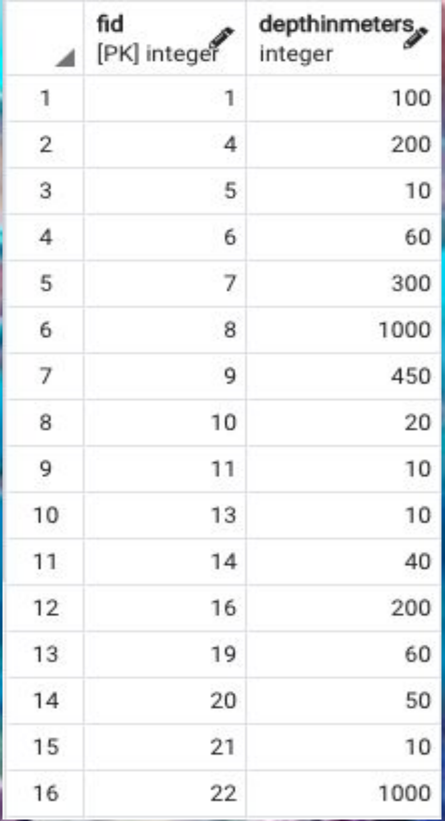
Functional Dependencies:
fid -> floraName, Consumable



seaFlora Table

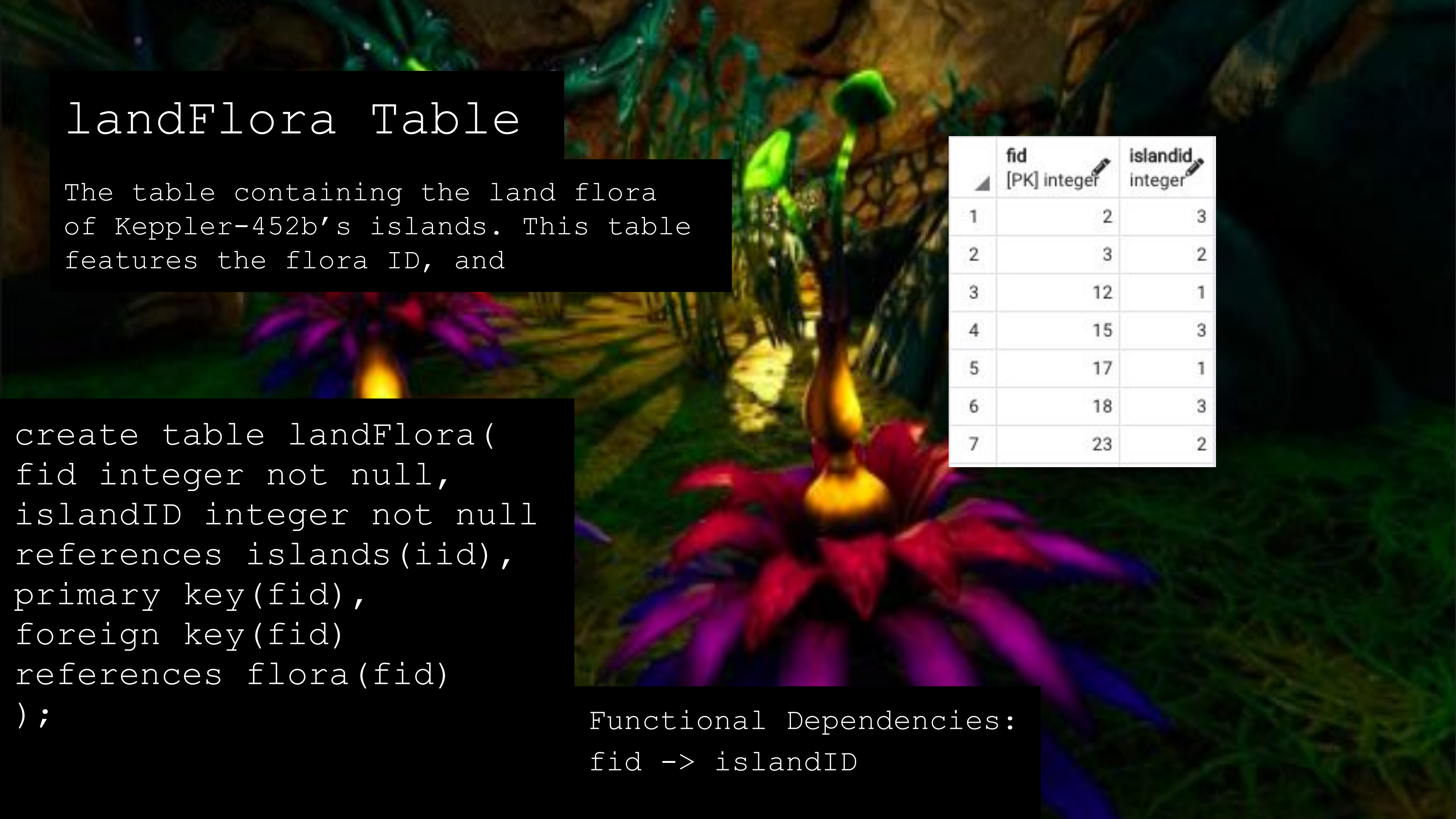
The table containing the sea flora of Keppler-452b’s oceans. This table features the flora ID and at what depth the flora is located in.

```
create table seaFlora(
fid integer not null,
depthInMeters integer not null
check (depthInMeters < 2000),
primary key(fid),
foreign key(fid) references
flora(fid),
foreign key(depthInMeters)
references depth(depthInMeters)
);
```



	fid [PK] integer	depthinmeters integer
1	1	100
2	4	200
3	5	10
4	6	60
5	7	300
6	8	1000
7	9	450
8	10	20
9	11	10
10	13	10
11	14	40
12	16	200
13	19	60
14	20	50
15	21	10
16	22	1000

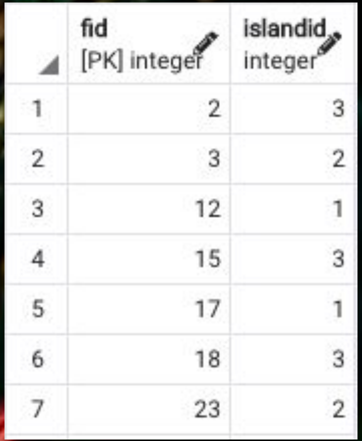
Functional Dependencies:
fid -> depthInMeters



landFlora Table

The table containing the land flora of Keppler-452b’s islands. This table features the flora ID, and

```
create table landFlora(
fid integer not null,
islandID integer not null
references islands(iid),
primary key(fid),
foreign key(fid)
references flora(fid)
);
```

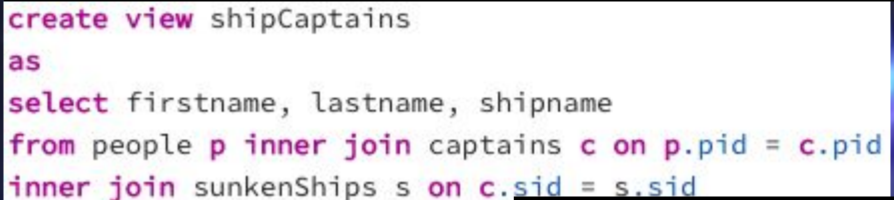


	fid [PK] integer	islandid integer
1	2	3
2	3	2
3	12	1
4	15	3
5	17	1
6	18	3
7	23	2

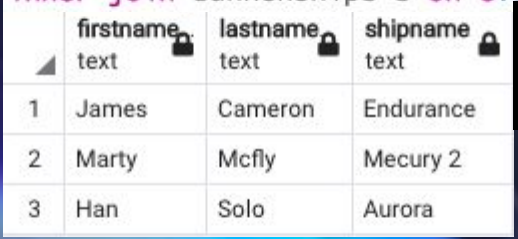
Functional Dependencies:
fid -> islandID

shipCaptains View

Gets the first name, last name, and ship name of the captain of the three Expedition 1 ships.



```
create view shipCaptains
as
select firstname, lastname, shipname
from people p inner join captains c on p.pid = c.pid
inner join sunkenShips s on c.sid = s.sid
```

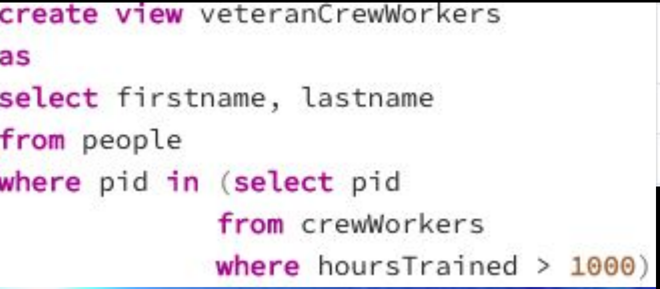


	firstname text	lastname text	shipname text
1	James	Cameron	Endurance
2	Marty	Mcfly	Mecury 2
3	Han	Solo	Aurora

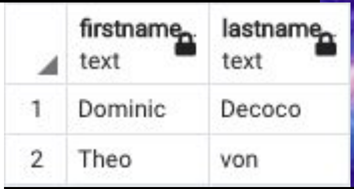
Select * from shipCaptains

veteranCrewWorkers View

Gets the first name and last name of the crew workers who have over 1000 hours of training

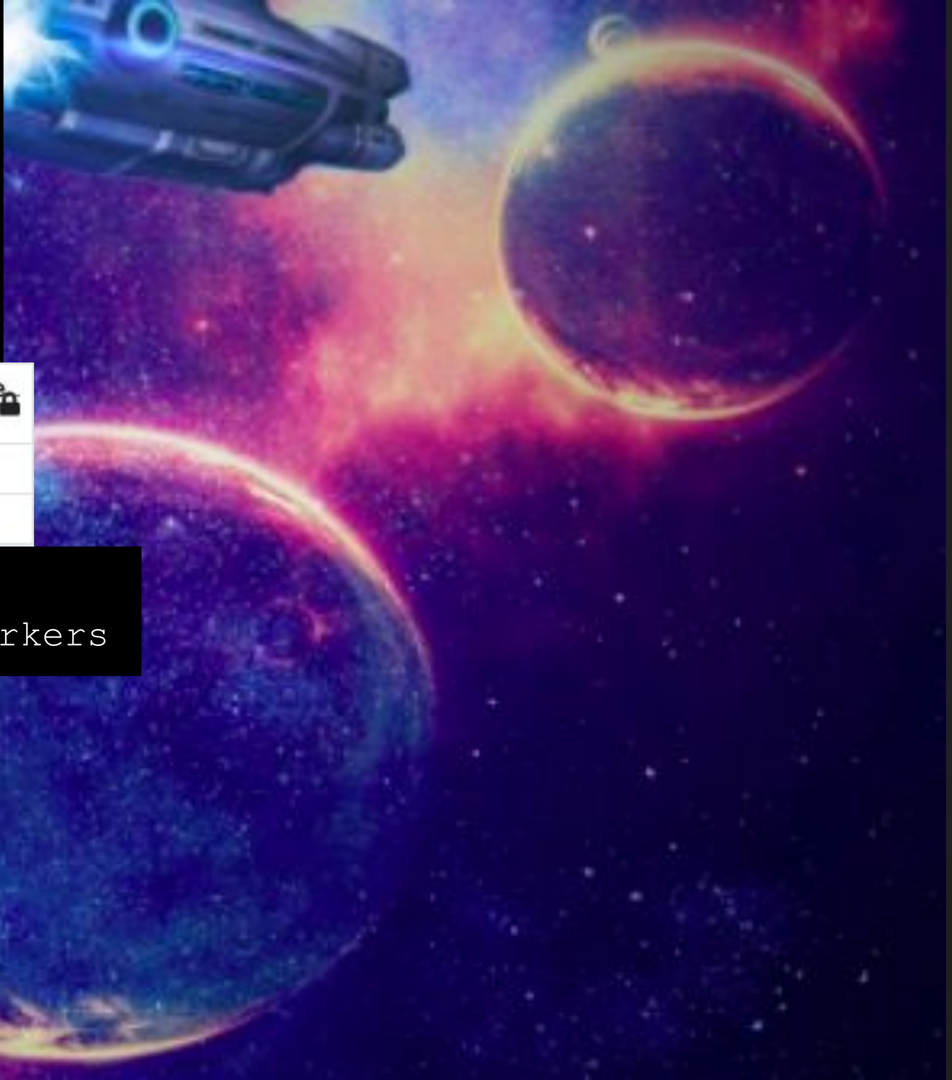


```
create view veteranCrewWorkers
as
select firstname, lastname
from people
where pid in (select pid
              from crewWorkers
              where hoursTrained > 1000)
```



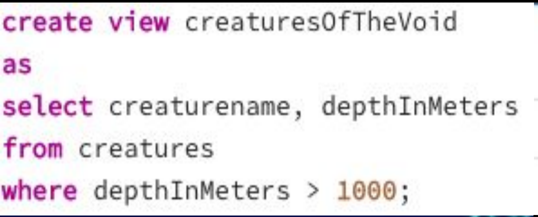
	firstname text	lastname text
1	Dominic	Decoco
2	Theo	von

```
Select * from
veteranCrewWorkers
```

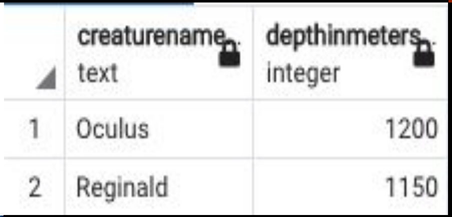


creaturesOfTheVoid View

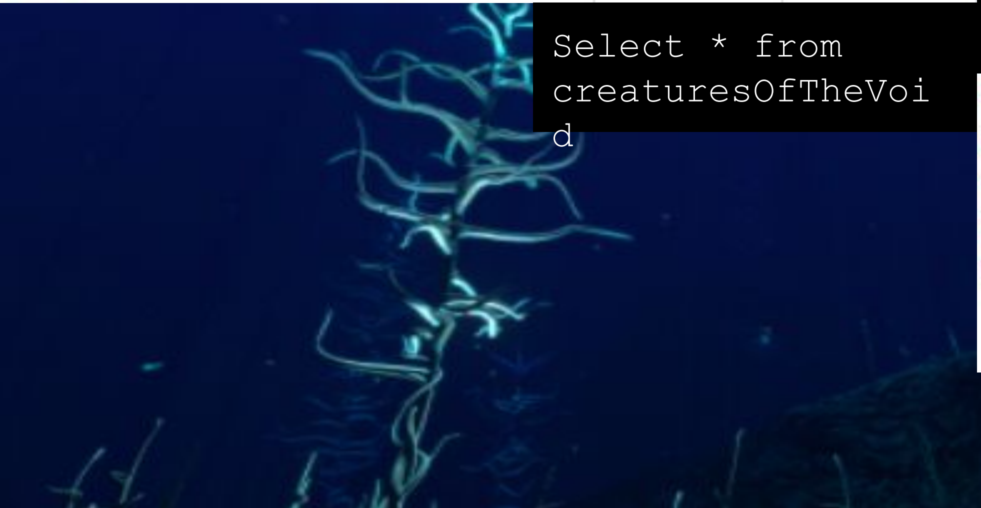
Gets the name of the creatures who live over 1000 meters deep in the ocean



```
create view creaturesOfTheVoid
as
select creaturename, depthInMeters
from creatures
where depthInMeters > 1000;
```



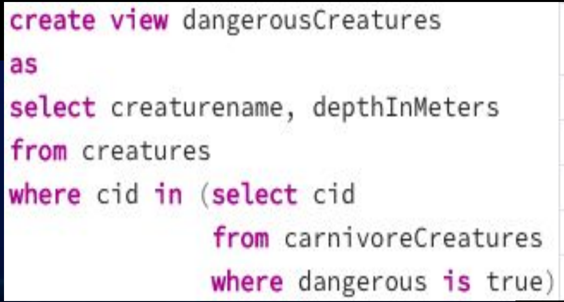
	creaturename text	depthinmeters integer
1	Oculus	1200
2	Reginald	1150



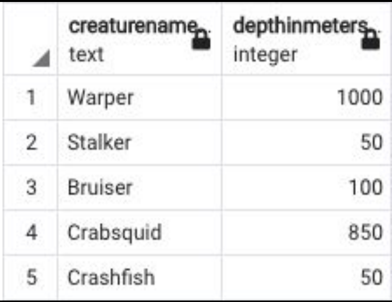
Select * from creaturesOfTheVoid

dangerousCreatures View

Gets the name of the creatures who are considered dangerous to humans



```
create view dangerousCreatures
as
select creaturename, depthInMeters
from creatures
where cid in (select cid
              from carnivoreCreatures
              where dangerous is true)
```

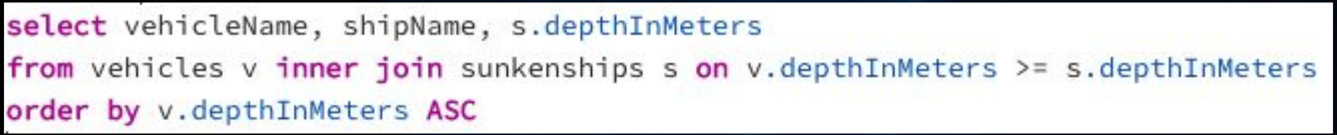


	creaturename text	depthinmeters integer
1	Warper	1000
2	Stalker	50
3	Bruiser	100
4	Crabsquid	850
5	Crashfish	50

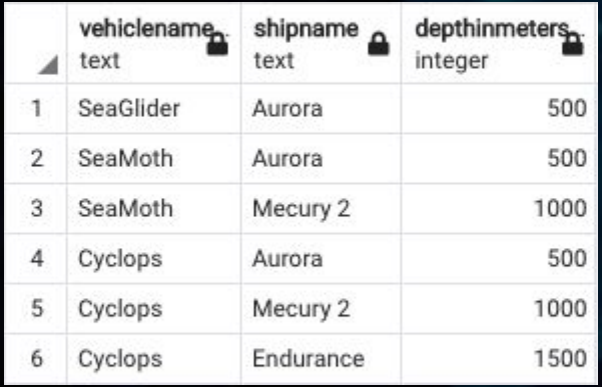
Select * from dangerousCreatures

Reports:

1. All possible vehicles to use when diving to each sunken ship



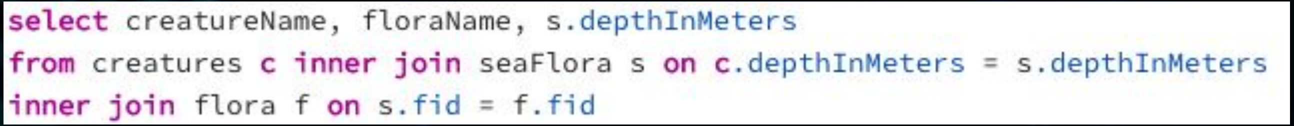
```sql
select vehicleName, shipName, s.depthInMeters
from vehicles v inner join sunkenships s on v.depthInMeters >= s.depthInMeters
order by v.depthInMeters ASC
```



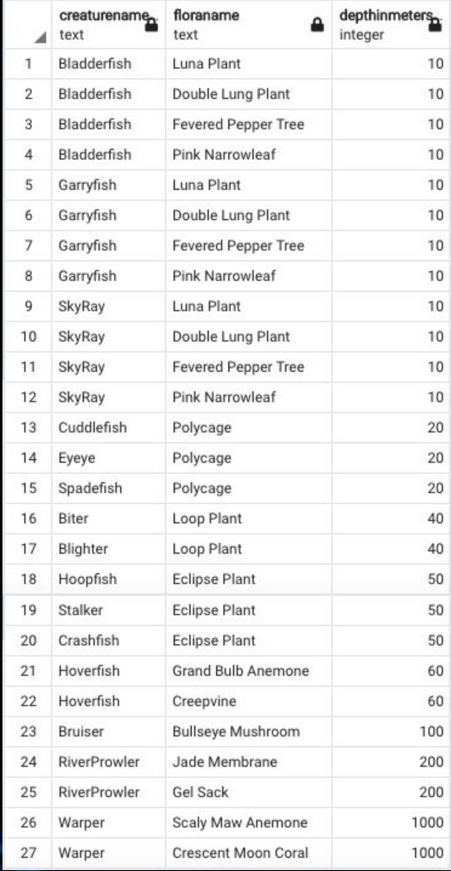
	vehiclename text	shipname text	depthinmeters integer
1	SeaGlider	Aurora	500
2	SeaMoth	Aurora	500
3	SeaMoth	Mecury 2	1000
4	Cyclops	Aurora	500
5	Cyclops	Mecury 2	1000
6	Cyclops	Endurance	1500

Reports:

2. Returns the creatures and flora that coexists with each other based on their depth in meters.



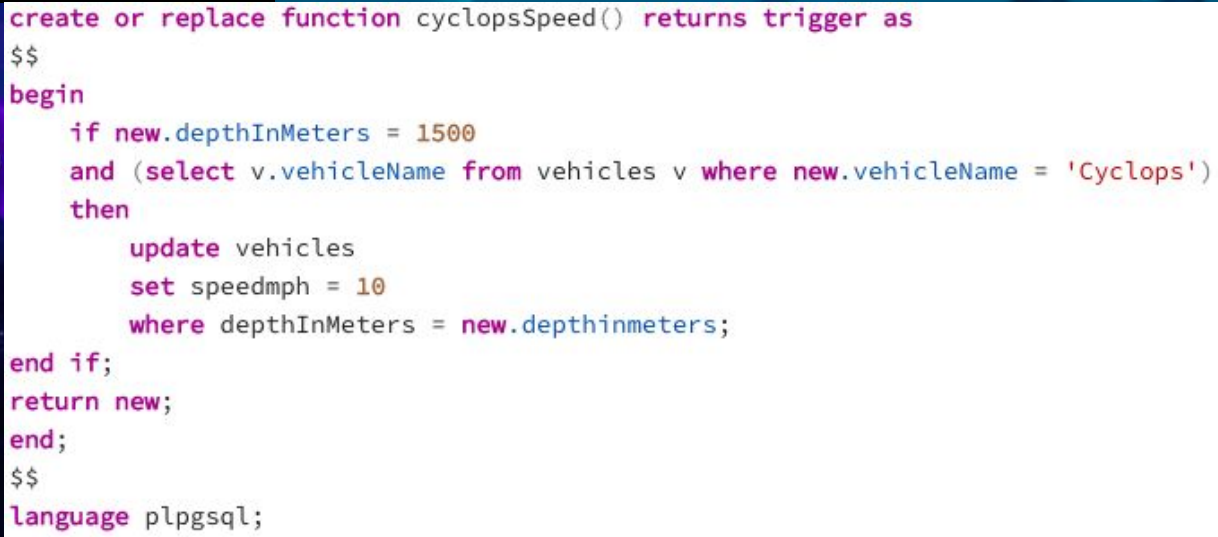
```sql
select creatureName, floraName, s.depthInMeters
from creatures c inner join seaFlora s on c.depthInMeters = s.depthInMeters
inner join flora f on s.fid = f.fid
```



	creaturename text	floraname text	depthinmeters integer
1	Bladderfish	Luna Plant	10
2	Bladderfish	Double Lung Plant	10
3	Bladderfish	Fevered Pepper Tree	10
4	Bladderfish	Pink Narrowleaf	10
5	Garryfish	Luna Plant	10
6	Garryfish	Double Lung Plant	10
7	Garryfish	Fevered Pepper Tree	10
8	Garryfish	Pink Narrowleaf	10
9	SkyRay	Luna Plant	10
10	SkyRay	Double Lung Plant	10
11	SkyRay	Fevered Pepper Tree	10
12	SkyRay	Pink Narrowleaf	10
13	Cuddlefish	Polycage	20
14	Eyeye	Polycage	20
15	Spadefish	Polycage	20
16	Biter	Loop Plant	40
17	Blighter	Loop Plant	40
18	Hoopfish	Eclipse Plant	50
19	Stalker	Eclipse Plant	50
20	Crashfish	Eclipse Plant	50
21	Hoverfish	Grand Bulb Anemone	60
22	Hoverfish	Creepvine	60
23	Bruiser	Bullseye Mushroom	100
24	RiverProwler	Jade Membrane	200
25	RiverProwler	Gel Sack	200
26	Warper	Scaly Maw Anemone	1000
27	Warper	Crescent Moon Coral	1000

Stored Procedure:

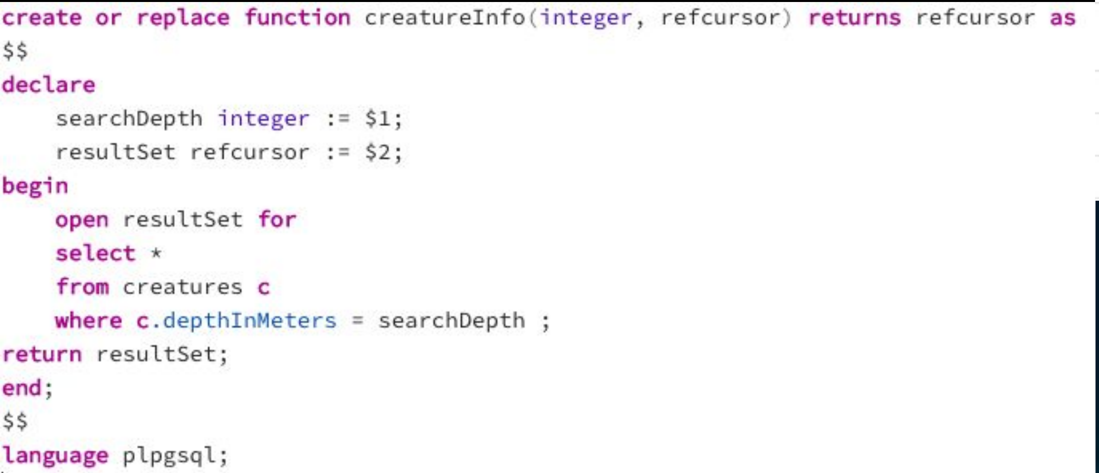
cyclopsSpeed: Returns a trigger that slows down the vehicle "Cyclops" to 10 miles per hour when depthInMeters is 1500. This is done to reduce the sound that the vehicle makes to avoid dangerous unknown creatures of the void.



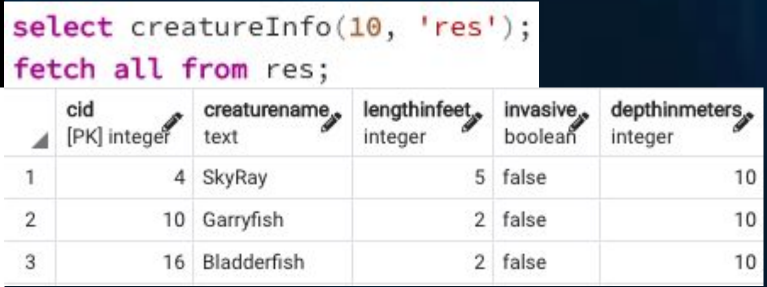
```
create or replace function cyclopsSpeed() returns trigger as
$$
begin
    if new.depthInMeters = 1500
    and (select v.vehicleName from vehicles v where new.vehicleName = 'Cyclops')
    then
        update vehicles
        set speedmph = 10
        where depthInMeters = new.depthinmeters;
end if;
return new;
end;
$$
language plpgsql;
```

Stored Procedure:

creatureInfo: Returns all information on the creature by searching for its depth in meters.



```
create or replace function creatureInfo(integer, refcursor) returns refcursor as
$$
declare
    searchDepth integer := $1;
    resultSet refcursor := $2;
begin
    open resultSet for
    select *
    from creatures c
    where c.depthInMeters = searchDepth ;
return resultSet;
end;
$$
language plpgsql;
```

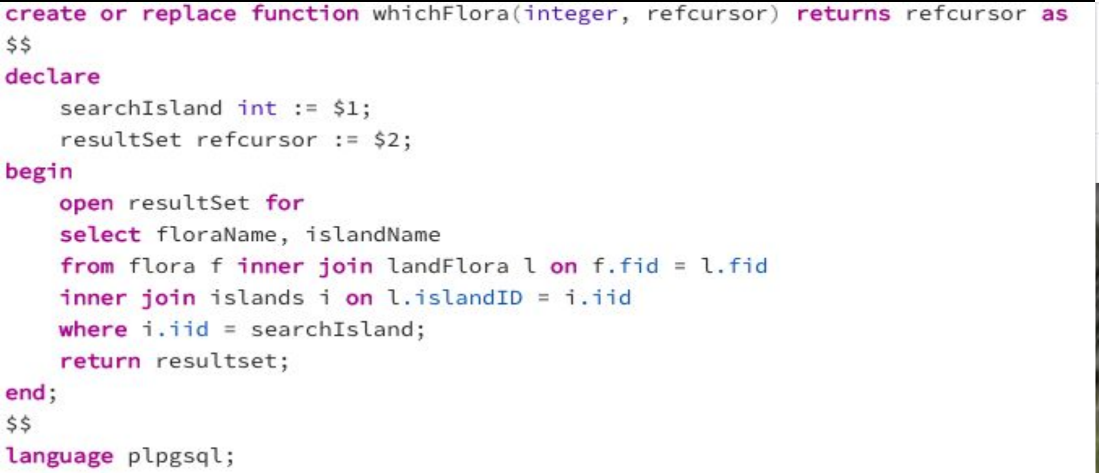


```
select creatureInfo(10, 'res');
fetch all from res;
```

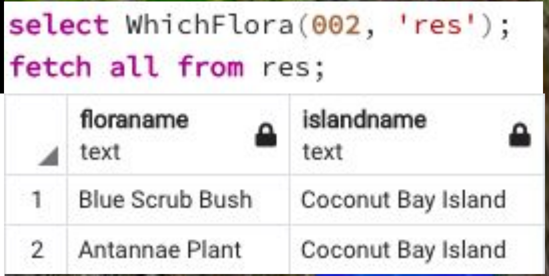
	cid [PK] integer	creaturename text	lengthinfeet integer	invasive boolean	depthinmeters integer
1	4	SkyRay	5	false	10
2	10	Garryfish	2	false	10
3	16	Bladderfish	2	false	10

Stored Procedure:

whichFlora: Returns the flora name and island name by searching through island ID



```
create or replace function whichFlora(integer, refcursor) returns refcursor as
$$
declare
    searchIsland int := $1;
    resultSet refcursor := $2;
begin
    open resultSet for
    select floraName, islandName
    from flora f inner join landFlora l on f.fid = l.fid
    inner join islands i on l.islandID = i.iid
    where i.iid = searchIsland;
    return resultset;
end;
$$
language plpgsql;
```

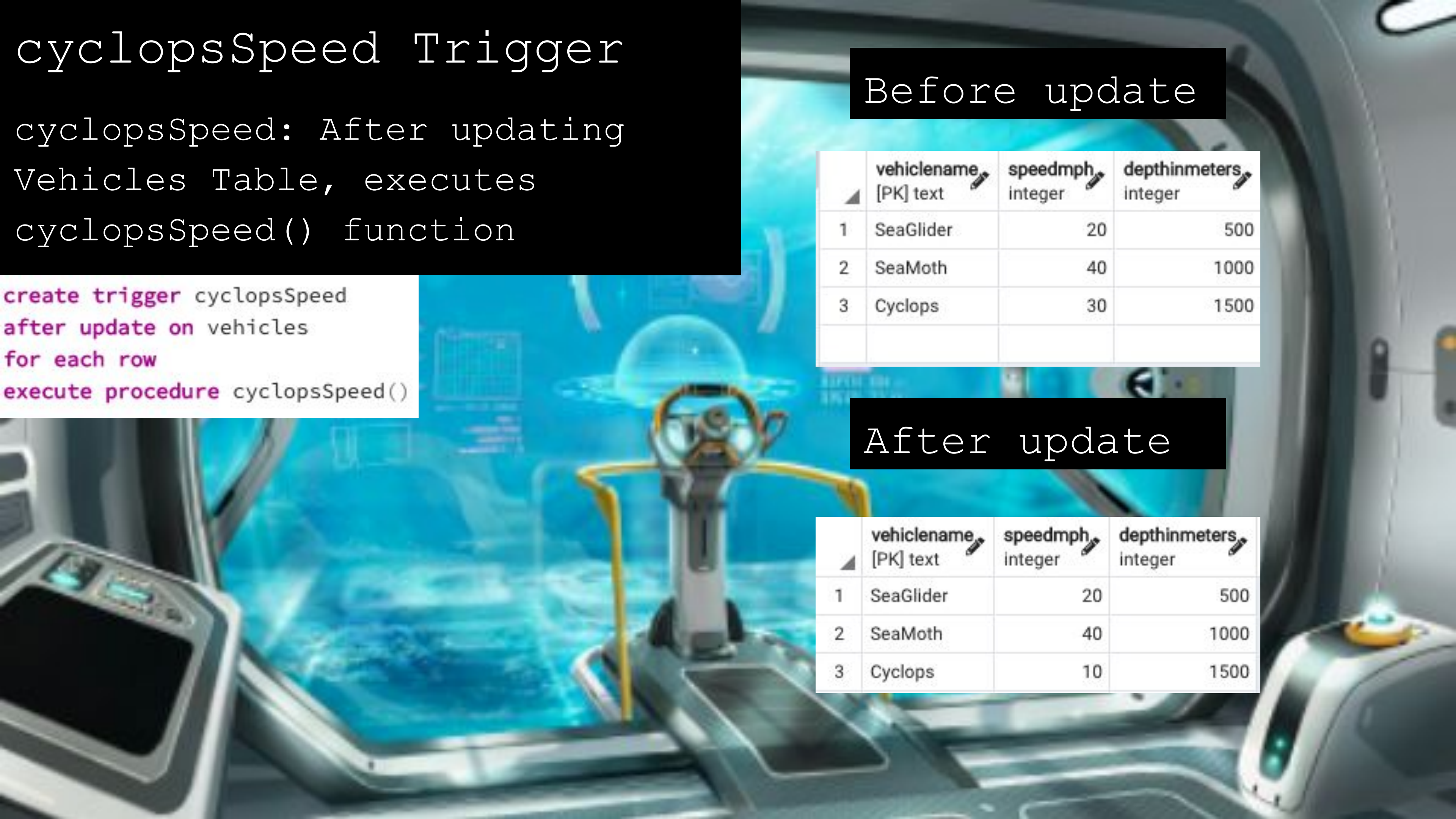


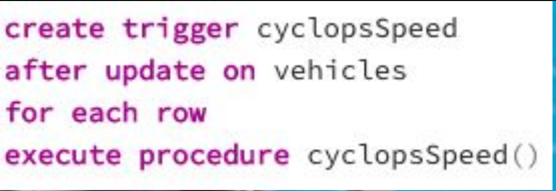
```
select WhichFlora(002, 'res');
fetch all from res;
```

	floraname text	islandname text
1	Blue Scrub Bush	Coconut Bay Island
2	Antannae Plant	Coconut Bay Island

cyclopsSpeed Trigger

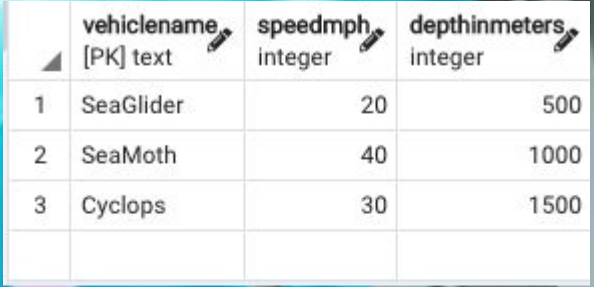
cyclopsSpeed: After updating Vehicles Table, executes cyclopsSpeed() function





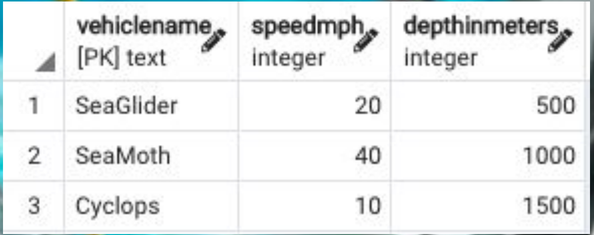
```
create trigger cyclopsSpeed
after update on vehicles
for each row
execute procedure cyclopsSpeed()
```

Before update



	vehiclename [PK] text	speedmph integer	depthinmeters integer
1	SeaGlider	20	500
2	SeaMoth	40	1000
3	Cyclops	30	1500

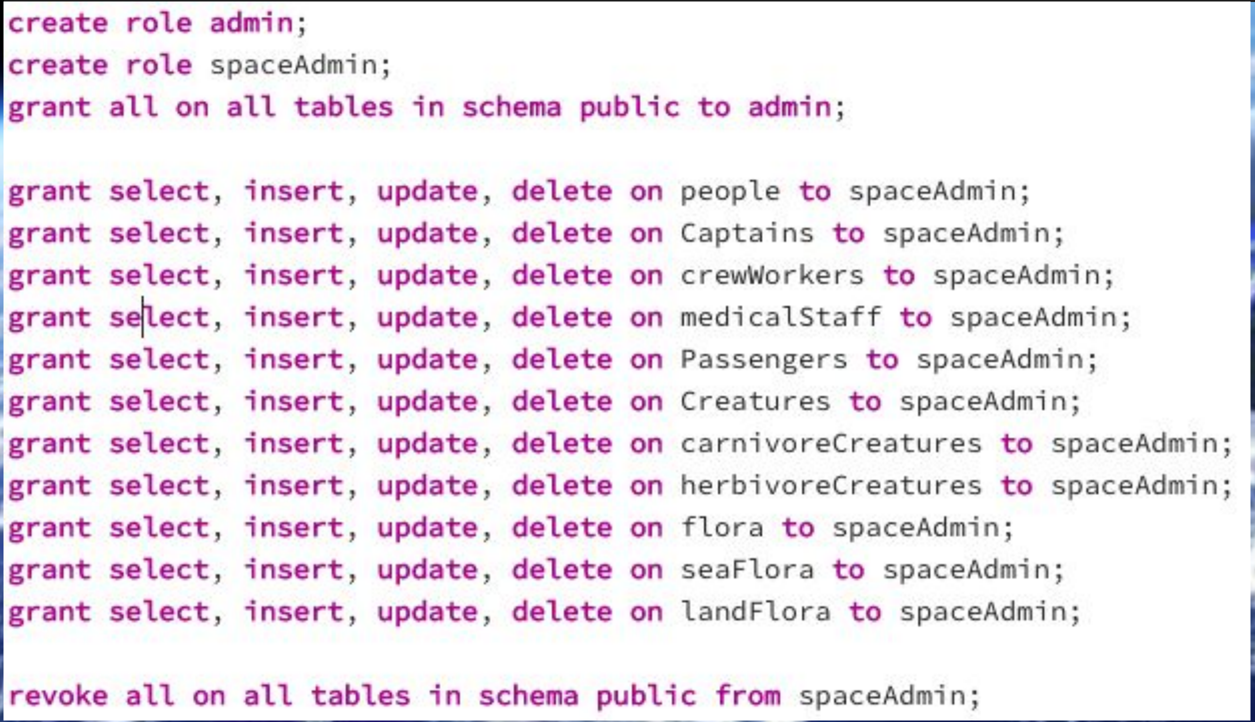
After update



	vehiclename [PK] text	speedmph integer	depthinmeters integer
1	SeaGlider	20	500
2	SeaMoth	40	1000
3	Cyclops	10	1500

Security

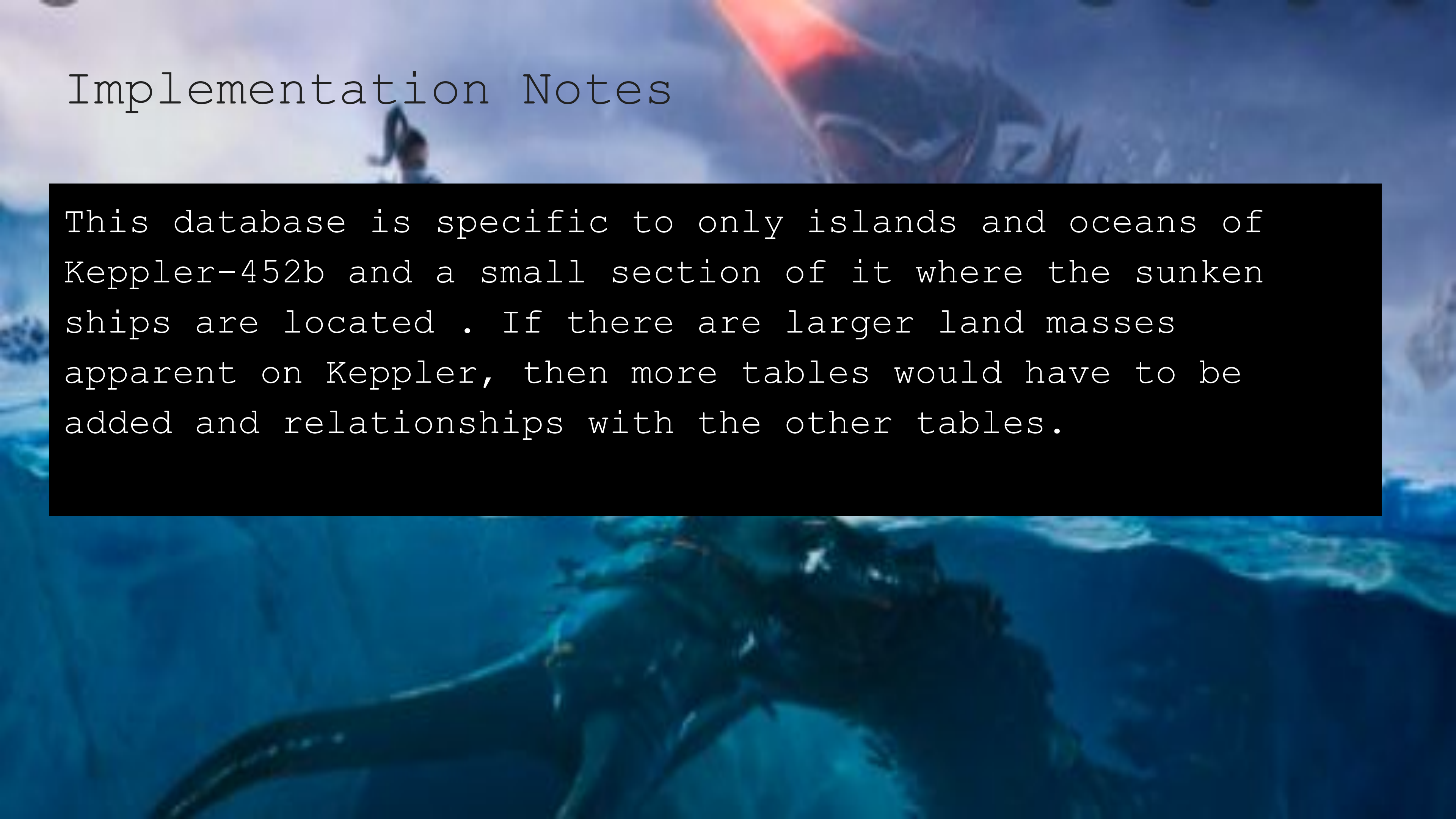
Granting access to tables for spaceAdmin. For The supervisor on the space station.



```
create role admin;
create role spaceAdmin;
grant all on all tables in schema public to admin;

grant select, insert, update, delete on people to spaceAdmin;
grant select, insert, update, delete on Captains to spaceAdmin;
grant select, insert, update, delete on crewWorkers to spaceAdmin;
grant select, insert, update, delete on medicalStaff to spaceAdmin;
grant select, insert, update, delete on Passengers to spaceAdmin;
grant select, insert, update, delete on Creatures to spaceAdmin;
grant select, insert, update, delete on carnivoreCreatures to spaceAdmin;
grant select, insert, update, delete on herbivoreCreatures to spaceAdmin;
grant select, insert, update, delete on flora to spaceAdmin;
grant select, insert, update, delete on seaFlora to spaceAdmin;
grant select, insert, update, delete on landFlora to spaceAdmin;

revoke all on all tables in schema public from spaceAdmin;
```



Implementation Notes

This database is specific to only islands and oceans of Keppler-452b and a small section of it where the sunken ships are located . If there are larger land masses apparent on Keppler, then more tables would have to be added and relationships with the other tables.