DROP TABLE ParkPod CASCADE;

DROP TABLE ParkBay CASCADE;

DROP TABLE ParkPod\_ParkBay CASCADE;

DROP TABLE Booking CASCADE;

DROP TABLE BillingAccount CASCADE;

DROP TABLE Member CASCADE;

DROP TABLE Driver CASCADE;

DROP TABLE Owner CASCADE;

DROP TABLE PayPal CASCADE;

DROP TABLE Member\_BillingAccount CASCADE;

DROP TABLE CarType CASCADE;

DROP TABLE Owner\_ParkBay CASCADE;

DROP TABLE CreditCard CASCADE;

DROP TABLE BankAccount CASCADE;

DROP TABLE Car CASCADE;

DROP TABLE Car\_CarType CASCADE;

DROP TABLE Review CASCADE;

DROP TABLE Review\_Rating CASCADE;

DROP TABLE ParkTag CASCADE;

DROP TABLE CurrentTag CASCADE;

DROP TABLE TagIssuedToCar CASCADE;

CREATE TABLE ParkPod(

device\_id CHAR(10) PRIMARY KEY CHECK (device\_id SIMILAR TO '[0-9]\*'), --Constraint to check exactly a 10 digit number

phone CHAR(10) NOT NULL CHECK (device\_id SIMILAR TO '[0-9]\*') --Constraint to check exactly a 10 digit number

);

CREATE TABLE ParkBay(

site VARCHAR(20) NOT NULL,

city VARCHAR(20) NOT NULL,

street VARCHAR(50) NOT NULL,

number SMALLINT NOT NULL CHECK (number > 0), --house numbers must be larger than 0

/\* should week\_start/end fields be represented in dates? if we are representing it using values from 0 to 24, wouldn’t it only represent one day of the week or weekend? \*/

week\_start SMALLINT CHECK (week\_start >= 0 AND week\_start <= 24), --can hold all values 0 to 24 (representing 24hr time)

week\_stop SMALLINT CHECK (week\_stop >= 0 AND week\_stop <= 24), --can hold all values 0 to 24 (representing 24hr time)

weekend\_start SMALLINT CHECK (weekend\_start >= 0 AND weekend\_start <= 24), --can hold all values 0 to 24 (representing 24hr time)

weekend\_stop SMALLINT CHECK (weekend\_stop >= 0 AND weekend\_stop <= 24), --can hold all values 0 to 24 (representing 24hr time)

--include checks for size that no dimension is 0 or negative

height NUMERIC(4,2) CHECK (height > 0), --metres up to 2 decimal places

length NUMERIC(4,2) CHECK (length > 0), --metres up to 2 decimal places

width NUMERIC(4,2) CHECK (width > 0), --metres up to 2 decimal places

latitude INTEGER CHECK (latitude <= 900000 AND latitude >= -900000),

longitude INTEGER CHECK (longitude <= 1800000 AND longitude >= -1800000),

--degrees (up to 90), minutes, seconds each represented by 2 digits

--degrees (up to 180), minutes, seconds each represented by 2 digits

PRIMARY KEY(site, city, street, number)

);

CREATE TABLE ParkPod\_ParkBay (

device\_id CHAR(10) UNIQUE NOT NULL REFERENCES ParkPod(device\_id) ON DELETE CASCADE,

site VARCHAR(20) NOT NULL ,

city VARCHAR(20) NOT NULL ,

street VARCHAR(50) NOT NULL ,

number SMALLINT NOT NULL,

FOREIGN KEY (site, city, street, number) REFERENCES ParkBay(site, city, street, number) ON DELETE CASCADE,

PRIMARY KEY(device\_id, site, city, street, number),

UNIQUE(site, city, street, number)

);

CREATE TABLE Booking(

site VARCHAR(20) NOT NULL,

city VARCHAR(20) NOT NULL,

street VARCHAR(50) NOT NULL,

number SMALLINT NOT NULL CHECK (number >= 0),

book\_date DATE NOT NULL CHECK (book\_date > CURRENT\_DATE),

hour SMALLINT NOT NULL CHECK (hour >= 0 and hour <= 24), --can hold all values 0 to 24 (representing 24hr time)

duration INTEGER,

FOREIGN KEY (site, city, street, number) REFERENCES ParkBay(site, city, street, number) ON DELETE CASCADE,

PRIMARY KEY(site, city, street, number, book\_date, hour)

);

CREATE TABLE Member(

email VARCHAR(20) PRIMARY KEY CHECK (email SIMILAR TO '%\_@\_\_%.\_\_%'), -- checks that the email contains an '@'

nickname VARCHAR(20) NULL UNIQUE, --Optional but unique

title VARCHAR(6) CHECK (title='MR' OR title='MRS' OR title='MISS' OR title='MASTER'), --Either MR, MRS, MISS, MASTER

family\_name VARCHAR(20),

given\_name VARCHAR(20),

address\_number SMALLINT CHECK (address\_number > 0),

street VARCHAR(20),

city VARCHAR(20)

);

CREATE TABLE Driver(

email VARCHAR(20) NOT NULL PRIMARY KEY CHECK (email SIMILAR TO '%\_@\_\_%.\_\_%'),

licence\_number INTEGER NOT NULL,

--here we have made licence\_expiry of data type DATE so that the check can be made with CURRENT\_DATE

licence\_expiry DATE NOT NULL CHECK (licence\_expiry > CURRENT\_DATE),

FOREIGN KEY (email) REFERENCES Member(email) ON DELETE CASCADE

);

CREATE TABLE Owner(

email VARCHAR(20) UNIQUE NOT NULL,

FOREIGN KEY (email) REFERENCES Member(email) ON DELETE CASCADE

);

CREATE TABLE Owner\_ParkBay(

email VARCHAR(20) NULL, -- NULL if the ParkBay has no owner

site VARCHAR(20) NOT NULL ,

city VARCHAR(20) NOT NULL ,

street VARCHAR(50) NOT NULL ,

number SMALLINT NOT NULL,

FOREIGN KEY (site, city, street, number) REFERENCES ParkBay(site, city, street, number) ON DELETE CASCADE,

FOREIGN KEY (email) REFERENCES Owner(email) ON DELETE CASCADE,

UNIQUE(site, city, street, number)

);

CREATE TABLE BillingAccount(

bill\_number INTEGER CHECK (bill\_number >= 0) NOT NULL UNIQUE,

email VARCHAR(20),

FOREIGN KEY (email) REFERENCES Member(email) ON DELETE CASCADE,

PRIMARY KEY (email, bill\_number)

);

CREATE TABLE PayPal(

bill\_number INTEGER CHECK (bill\_number >= 0) NOT NULL UNIQUE,

email VARCHAR(20) NOT NULL,

--Assume that the PayPal email is the same as the Member email

--FOREIGN KEY (email, bill\_number, acc\_type) REFERENCES BillingAccount(email, bill\_number, acc\_type) ON DELETE CASCADE,

FOREIGN KEY (email, bill\_number) REFERENCES BillingAccount(email, bill\_number) ON DELETE CASCADE,

PRIMARY KEY (email, bill\_number)

);

CREATE TABLE CreditCard(

bill\_number INTEGER CHECK (bill\_number >= 0) NOT NULL UNIQUE,

email VARCHAR(20) NOT NULL,

card\_number CHAR(16) NOT NULL CHECK (card\_number SIMILAR TO '[0-9]\*'),

card\_name VARCHAR(20),

--The brand name must be one of the following 3 brands

card\_brand VARCHAR(14) CHECK (card\_brand='MASTERCARD' OR card\_brand='VISA' OR card\_brand='AMERICANEXPRESS'),

card\_expiry DATE CHECK (card\_expiry > CURRENT\_DATE),

FOREIGN KEY (email, bill\_number) REFERENCES BillingAccount(email, bill\_number) ON DELETE CASCADE,

PRIMARY KEY (email, bill\_number)

);

CREATE TABLE BankAccount(

bill\_number INTEGER CHECK (bill\_number >= 0) NOT NULL UNIQUE,

email VARCHAR(20) NOT NULL,

account\_number INTEGER NOT NULL CHECK(account\_number > 0),

bsb INTEGER NOT NULL CHECK(bsb > 0),

account\_name VARCHAR(20) NOT NULL ,

FOREIGN KEY (email, bill\_number) REFERENCES BillingAccount(email, bill\_number) ON DELETE CASCADE,

PRIMARY KEY (email, bill\_number)

);

--Check that 1 to 3 accounts exist, requires trigger (see below)

CREATE TABLE Member\_BillingAccount(

bill\_number INTEGER CHECK (bill\_number >= 0) NOT NULL UNIQUE,

email VARCHAR(20) NOT NULL,

prefer\_status CHAR(1) NULL CHECK (prefer\_status='Y' OR prefer\_status=NULL ),

FOREIGN KEY (email) REFERENCES Member(email) ON DELETE CASCADE,

FOREIGN KEY (bill\_number) REFERENCES BillingAccount(bill\_number) ON DELETE CASCADE,

PRIMARY KEY (email, bill\_number),

UNIQUE(email, prefer\_status)

);

--This is our attempt at a trigger to stop a Member from having more than 3

--BillingAccounts. It follows the same structure as in the lectures but

--currently WHEN doesn’t support subqueries

--CREATE FUNCTION abort\_Add\_Account() RETURNS trigger AS

--$abort\_Account$

--BEGIN

--RAISE EXCEPTION 'Member already has 3 billing accounts';

--END;

--$abort\_Account$ LANGUAGE plpgsql;

--CREATE TRIGGER max\_3\_accs

-- BEFORE INSERT OR UPDATE ON BillingAccount

--FOR EACH ROW

-- WHEN((SELECT COUNT (B.email)

--FROM BillingAccount B

-- WHERE B.email = NEW.email)

-->= 3)

--EXECUTE PROCEDURE

--abort\_Add\_Account();

CREATE TABLE Car(

regno INTEGER NOT NULL CHECK (regno >= 0),

car\_name VARCHAR(20) NOT NULL,

email VARCHAR(20) NOT NULL,

FOREIGN KEY (email) REFERENCES Driver(email) ON DELETE CASCADE,

PRIMARY KEY (car\_name, email)

);

CREATE TABLE CarType(

make VARCHAR(20) NOT NULL ,

model VARCHAR(20) NOT NULL ,

--include checks for size that no dimension is 0 or negative

height NUMERIC(4,2) CHECK (height > 0), --metres up to 2 decimal places

length NUMERIC(4,2) CHECK (length > 0), --metres up to 2 decimal places

width NUMERIC(4,2) CHECK (width > 0), --metres up to 2 decimal places

PRIMARY KEY (make, model)

);

CREATE TABLE Car\_CarType(

car\_name VARCHAR(20) NOT NULL,

email VARCHAR(20) NOT NULL,

make VARCHAR(20) NOT NULL ,

model VARCHAR(20) NOT NULL ,

FOREIGN KEY (car\_name, email) REFERENCES Car(car\_name, email) ON DELETE CASCADE,

FOREIGN KEY (make, model) REFERENCES CarType(make, model) ON DELETE CASCADE,

PRIMARY KEY (car\_name, email, make, model),

--Ensures a car can be only 1 type of car

CONSTRAINT pair UNIQUE (car\_name, email)

);

CREATE TABLE Review (

email VARCHAR(20) NOT NULL,

review\_id INTEGER UNIQUE CHECK (review\_id > 0), --Identifies a new review when created

date\_reviewed DATE,

num\_likes INTEGER CHECK (num\_likes >= 0),

num\_dislikes INTEGER CHECK (num\_likes >= 0),

comment VARCHAR(100) NULL,

numerical\_review SMALLINT CHECK (numerical\_review >= 0 AND numerical\_review <= 5),

FOREIGN KEY (email) REFERENCES Driver(email) ON DELETE CASCADE,

PRIMARY KEY (email, review\_id)

);

--Could add a constraint to check that the owner of a review can not rate their own review, requires a subquery

CREATE TABLE Review\_Rating(

email VARCHAR(20) NOT NULL,

review\_id INTEGER NOT NULL CHECK (review\_id >= 0), --Identifies a new rating when created

date\_review\_rated DATE,

--A rating can be rated between 0 to 100 %

rating\_of\_review

SMALLINT CHECK (rating\_of\_review >= 0 AND rating\_of\_review <=100),

--Any driver can rate a review, the data held here is the email of the rater not the creator of the review

FOREIGN KEY (email) REFERENCES Driver(email) ON DELETE CASCADE,

FOREIGN KEY (review\_id) REFERENCES Review(review\_id) ON DELETE CASCADE,

PRIMARY KEY(email, review\_id, date\_review\_rated)

);

CREATE TABLE ParkTag(

--Constraint to check exactly a 10 digit number

tag\_id CHAR(10) PRIMARY KEY CHECK (tag\_id SIMILAR TO '[0-9]\*'),

car\_name VARCHAR(20) NOT NULL,

email VARCHAR(20) NOT NULL,

FOREIGN KEY (car\_name, email) REFERENCES Car(car\_name, email) ON DELETE CASCADE

);

CREATE TABLE CurrentTag(

tag\_id CHAR(10) UNIQUE NOT NULL,

car\_name VARCHAR(20) UNIQUE NOT NULL,

email VARCHAR(20) UNIQUE NOT NULL,

FOREIGN KEY (tag\_id) REFERENCES ParkTag(tag\_id) ON DELETE CASCADE,

FOREIGN KEY (car\_name, email) REFERENCES Car(car\_name, email) ON DELETE CASCADE,

PRIMARY KEY (tag\_id, car\_name, email)

);

CREATE TABLE TagIssuedToCar(

tag\_id CHAR(10) UNIQUE NOT NULL,

car\_name VARCHAR(20) NOT NULL,

email VARCHAR(20) NOT NULL,

FOREIGN KEY (tag\_id) REFERENCES ParkTag(tag\_id) ON DELETE CASCADE,

FOREIGN KEY (car\_name, email) REFERENCES Car(car\_name, email) ON DELETE CASCADE,

PRIMARY KEY (tag\_id)

);

INSERT INTO ParkPod VALUES(1234567890, 0412345678);

INSERT INTO ParkPod VALUES(1234567891, 0412345678);

INSERT INTO ParkBay VALUES('rooftop space', 'Sydney', 'Paramatta Road', '1234', '7', '24', NULL, NULL, 2.00, 4.75, 2.5, 338650, 1512094);

INSERT INTO ParkPod\_ParkBay VALUES(1234567890, 'rooftop space', 'Sydney', 'Paramatta Road', '1234');

--Checks that atmost 1 ParkPod can be attached to a ParkBay

--INSERT INTO ParkPod\_ParkBay VALUES(1234567891, 'rooftop space', 'Sydney', 'Paramatta Road', '1234');

--Checks that atmost 1 parkBay can be assigned to a ParkPod

--INSERT INTO ParkPod\_ParkBay VALUES(1234567890, 'other space', 'Sydney', 'Paramatta Road', '1234');

--For an empty relationship then no entry will be in the relationship table ParkPod\_ParkBay

INSERT INTO ParkBay VALUES('floor space', 'Sydney', 'Paramatta Road', '1234', '7', '24', NULL, NULL, 2.00, 4.75, 2.5, 338650, 1512094);

INSERT INTO ParkBay VALUES('new space', 'Sydney', 'Paramatta Road', '1234', '7', '24', NULL, NULL, 2.00, 4.75, 2.5, 338650, 1512094);

INSERT INTO ParkBay VALUES('old space', 'Sydney', 'Paramatta Road', '1234', '7', '24', NULL, NULL, 2.00, 4.75, 2.5, 338650, 1512094);

INSERT INTO Booking VALUES('rooftop space', 'Sydney', 'Paramatta Road', '1234', '2015-10-05', 7, 24);

INSERT INTO Booking VALUES('rooftop space', 'Sydney', 'Paramatta Road', '1234', '2015-10-05', 2, 22);

--This checks that a Booking (weak entity) cannot be made unless the ParkBay (strong entity) already exists

--INSERT INTO Booking VALUES('rooftop space', 'Melbourne', 'Paramatta Road', '1234', '2015-10-05', 7, 24);

INSERT INTO Member VALUES ('fake@hotmail.com', NULL, 'MASTER', 'lastname', 'givenname', 22, 'A street', 'Sydney');

INSERT INTO Member VALUES ('new@hotmail.com', 'McFly', 'MISS', 'lastname', 'givenname', 1, 'B street', 'Sydney');

--Fails the title CHECK that the title must be one of 4 pre-determined titles

--INSERT INTO Member VALUES ('fake@hotmail.com', NULL, 'SIR', 'lastname', 'givenname', 22, 'A street', 'Sydney');

--Here we test that a member can be both a driver and an owner as expected

INSERT INTO Driver VALUES('fake@hotmail.com', 1234, '2015-10-05');

INSERT INTO Driver VALUES('new@hotmail.com', 4554, '2015-10-10');

INSERT INTO Owner VALUES('fake@hotmail.com');

INSERT INTO Owner VALUES('new@hotmail.com');

--Owner can own many ParkBays but a ParkBay can be owned by only 0 or 1 owners

--This means email for Owner\_ParkBay can be NULL (no owner) but no ParkBay may have 2 Owners

INSERT INTO Owner\_ParkBay VALUES('fake@hotmail.com', 'floor space', 'Sydney', 'Paramatta Road', '1234');

INSERT INTO Owner\_ParkBay VALUES('fake@hotmail.com', 'rooftop space', 'Sydney', 'Paramatta Road', '1234');

INSERT INTO Owner\_ParkBay VALUES('new@hotmail.com', 'new space', 'Sydney', 'Paramatta Road', '1234');

INSERT INTO Owner\_ParkBay VALUES(NULL, 'old space', 'Sydney', 'Paramatta Road', '1234');

INSERT INTO BillingAccount VALUES (1111, 'fake@hotmail.com');

INSERT INTO BillingAccount VALUES (1112, 'fake@hotmail.com');

--Checks disjoint BillingAccount

--INSERT INTO BillingAccount VALUES (1112, 'new@hotmail.com');

INSERT INTO PayPal VALUES (1111, 'fake@hotmail.com');

INSERT INTO PayPal VALUES (1112, 'fake@hotmail.com');

INSERT INTO CreditCard VALUES (1112, 'fake@hotmail.com', '1111222233334444', 'jeff', 'VISA', '2016-12-12');

INSERT INTO Member\_BillingAccount VALUES (1112, 'fake@hotmail.com', 'Y');

--Cannot have more than 1 preffered Billing Account

--INSERT INTO Member\_BillingAccount VALUES (1111, 'fake@hotmail.com', 'Y');

INSERT INTO Car VALUES (1234, 'herbie', 'fake@hotmail.com');

INSERT INTO CarType VALUES('holden', 'commidore', 1,1,1);

INSERT INTO CarType VALUES('holden', 'camry', 1,1,1);

INSERT INTO Car\_CarType VALUES('herbie', 'fake@hotmail.com', 'holden', 'commidore');

--Checks that a car cannot be more than one type of car

--INSERT INTO Car\_CarType VALUES('herbie', 'fake@hotmail.com', 'holden', 'camry');

--Checks that a car cannot have NULL for the make or model and thus must have a single CarType

--INSERT INTO Car\_CarType VALUES('herbie', 'fake@hotmail.com', 'holden', NULL);

INSERT INTO Review VALUES ('fake@hotmail.com', 8000, '2016-12-25', 20, 5, NULL, 4);

INSERT INTO Review\_Rating VALUES ('new@hotmail.com', 8000, '2016-12-30', 99);

INSERT INTO Review\_Rating VALUES ('fake@hotmail.com', 8000, '2016-12-31', 89);

INSERT INTO ParkTag VALUES ('1234567890', 'herbie', 'fake@hotmail.com');

--Checks that a car cannot be more than one type of car

--INSERT INTO Car\_CarType VALUES('herbie', 'fake@hotmail.com', 'holden', 'camry');

--Checks that a car cannot have NULL for the make or model and thus must have a single CarType

--INSERT INTO Car\_CarType VALUES('herbie', 'fake@hotmail.com', 'holden', NULL);

INSERT INTO ParkTag VALUES ('1111111111', 'herbie', 'fake@hotmail.com');

--Checks that a ParkTag cannot exist unless attached to a Car

--INSERT INTO ParkTag VALUES ('1234554321');

--Zero to many Parktags can be issued but only once for an id.

--Each car must have a unique current ParkTag

INSERT INTO TagIssuedToCar VALUES ('1234567890', 'herbie', 'fake@hotmail.com');

INSERT INTO TagIssuedToCar VALUES ('1111111111', 'herbie', 'fake@hotmail.com');

INSERT INTO CurrentTag VALUES ('1234567890', 'herbie', 'fake@hotmail.com');

--Check to see that a car cannot be assigned 2 different park tags simultaneously

--INSERT INTO CurrentTag VALUES ('1111111111', 'herbie', 'fake@hotmail.com');