CI204 DATABASES II ASSIGNEMENT 2

University of Brighton |

John Vos

2019

Contents

[Table Booking: 4](#_Toc8900193)

[SQL Statements: 4](#_Toc8900194)

[Create table: 4](#_Toc8900195)

[Insert into: 4](#_Toc8900196)

[Table Staff: 0](#_Toc8900197)

[SQL Statements: 0](#_Toc8900198)

[Create table: 0](#_Toc8900199)

[Insert into: 0](#_Toc8900200)

[Table Staff Booking: 0](#_Toc8900201)

[SQL Statements: 0](#_Toc8900202)

[Create table: 0](#_Toc8900203)

[Insert into: 0](#_Toc8900204)

[Table Staff Pay: 2](#_Toc8900205)

[SQL Statements: 2](#_Toc8900206)

[Insert Into: 2](#_Toc8900207)

[Table Event Details: 3](#_Toc8900208)

[SQL Statements: 3](#_Toc8900209)

[Create table: 3](#_Toc8900210)

[Insert into: 3](#_Toc8900211)

[Table Event Contact Details: 0](#_Toc8900212)

[SQL Statements: 0](#_Toc8900213)

[Create table: 0](#_Toc8900214)

[Insert into: 0](#_Toc8900215)

[Table Truck: 0](#_Toc8900216)

[SQL Statements: 0](#_Toc8900217)

[Create table: 0](#_Toc8900218)

[Insert into: 0](#_Toc8900219)

[Table Truck Booking: 1](#_Toc8900220)

[SQL Statements: 1](#_Toc8900221)

[Create table: 1](#_Toc8900222)

[Insert into: 1](#_Toc8900223)

[Table Accident Log: 3](#_Toc8900224)

[SQL Statements: 3](#_Toc8900225)

[Create table: 3](#_Toc8900226)

[Insert into: 3](#_Toc8900227)

[Table Recipe: 0](#_Toc8900228)

[SQL Statements: 0](#_Toc8900229)

[Create table: 0](#_Toc8900230)

[Insert into: 0](#_Toc8900231)

[Table Ingredients: 1](#_Toc8900232)

[SQL Statements: 1](#_Toc8900233)

[Create table: 1](#_Toc8900234)

[Insert into: 1](#_Toc8900235)

[Table Recipe Ingredients: 0](#_Toc8900236)

[SQL Statements: 0](#_Toc8900237)

[Create table: 0](#_Toc8900238)

[Insert into: 0](#_Toc8900239)

[Table Ingredients to Recipe: 2](#_Toc8900240)

[SQL Statements: 2](#_Toc8900241)

[Create table: 2](#_Toc8900242)

[Insert into: 2](#_Toc8900243)

[Must: 3](#_Toc8900244)

[1. Record /edit/cancel bookings for multiple trucks at outdoor events 3](#_Toc8900245)

[Record booking: 3](#_Toc8900246)

[Edit booking: 4](#_Toc8900247)

[Cancel booking: 5](#_Toc8900248)

[2. For all staff, ensure that a level 2 food hygiene certificate is in place 6](#_Toc8900249)

[3. For each item sold, produce a list of ingredients 7](#_Toc8900250)

[4. Identify all truck bookings for any given date 8](#_Toc8900251)

[5. Identify all bookings for a given truck 9](#_Toc8900252)

[Should: 10](#_Toc8900253)

[6. Record any accidents / reportable incidents that take place, including the event at which they happened 10](#_Toc8900254)

[7. For all staff acting as supervisors, ensure that a level 3 food hygiene certificate is in place 11](#_Toc8900255)

[8. For all events involving alcohol, identify a team member as the license holder. 0](#_Toc8900256)

[9. Identify any temporary staff who are not currently working this weekend (i.e. may be available for a last- minute booking). 1](#_Toc8900257)

[Could: 2](#_Toc8900258)

[10. Record members of staff who worked at specific events, including who was acting as the supervisor. This should include the number of hours worked by each staff member, and their rate of pay. 2](#_Toc8900259)

[11. Produce a list of ingredients for each item sold, with known allergens at the top of the list - this should include sub-ingredients, like the tahini ingredients on the hummus recipe described previously. 0](#_Toc8900260)

[Would be nice if: 1](#_Toc8900261)

[12. Identify weekends in the next month with available trucks (to offer last minute deals) 1](#_Toc8900262)

[Functionality: 2](#_Toc8900263)

[PART 1) 2](#_Toc8900264)

[PART 2) 2](#_Toc8900265)

[PART 3) 3](#_Toc8900266)

[Create a simple application to provide an interface capable of executing the above functionality (select queries and procedures): 4](#_Toc8900267)

[ERD: 7](#_Toc8900268)

# Table Booking:

## SQL Statements:

### Create table:

CREATE TABLE tbl\_booking (

booking\_ID int NOT NULL,

event\_ID int NOT NULL,

booking\_name varchar(50) NOT NULL,

booking\_date date NOT NULL,

booking\_duration int NOT NULL,

repeat\_event bit,

PRIMARY KEY (booking\_ID),

FOREIGN KEY (event\_ID) REFERENCES tbl\_event\_details(event\_ID)

);

### Insert into:

/\*inserts 10 cells of data into booking\*/

INSERT INTO tbl\_booking(booking\_ID, event\_ID, booking\_name, booking\_date, booking\_duration, repeat\_event)

VALUES

(1, 1, 'Fuss - Budget Wedding', '2019-03-15', 1, 0),

(2, 2, 'Lewes Farmers Market', '2019-03-01', 1,1),

(3, 3, 'Brighton Grande Parade', '2019-04-22', 3, 1),

(4, 4, 'Lewes Farmers Market', '2019-04-01', 1, 1),

(5, 5, 'Carl Birthday Party', '2019-05-16', 1, 0),

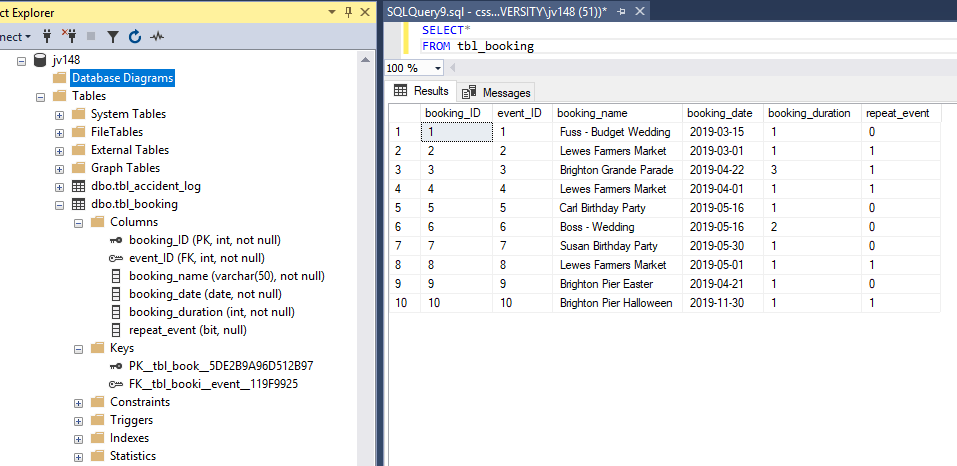
(6, 6, 'Boss - Wedding', '2019-05-16', 2, 0),

(7, 7, 'Susan Birthday Party', '2019-05-30', 1, 0),

(8, 8, 'Lewes Farmers Market', '2019-05-01', 1, 1),

(9, 9, 'Brighton Pier Easter', '2019-04-21', 1, 0),

(10, 10, 'Brighton Pier Halloween', '2019-11-30', 1, 1);



# Table Staff:

## SQL Statements:

### Create table:

CREATE TABLE tbl\_staff(

staff\_ID int NOT NULL,

staff\_first\_name varchar(50) NOT NULL,

staff\_second\_name varchar(50) NOT NULL,

staff\_email varchar(50) NOT NULL,

comments varchar(50),

alcohol\_vending\_licence bit NOT NULL,

food\_hygiene\_certificate bit NOT NULL,

food\_hygiene\_certificate\_level int NOT NULL,

PRIMARY KEY (staff\_ID),

);

### Insert into:

/\* inserts 10 cells of data into staff\*/

INSERT INTO tbl\_staff(staff\_ID, staff\_first\_name, staff\_second\_name, staff\_email, comments, alcohol\_vending\_licence, food\_hygiene\_certificate, food\_hygiene\_certificate\_level)

VALUES

(1, 'John', 'Vos', 'staff1email.com', 'cannot work sundays', 1, 1, 2),

(2, 'James', 'Shaw', 'staff2email.com', NULL, 1, 1, 2),

(3, 'Jack', 'Whitchurch', 'staff3email.com', 'cannot handle nuts, allergic', 1, 1, 2),

(4, 'Ross', 'Jones', 'staff4email.com', NULL, 1, 1, 2),

(5, 'Susan', 'Betsy', 'staff5email.com', NULL, 1, 1, 2),

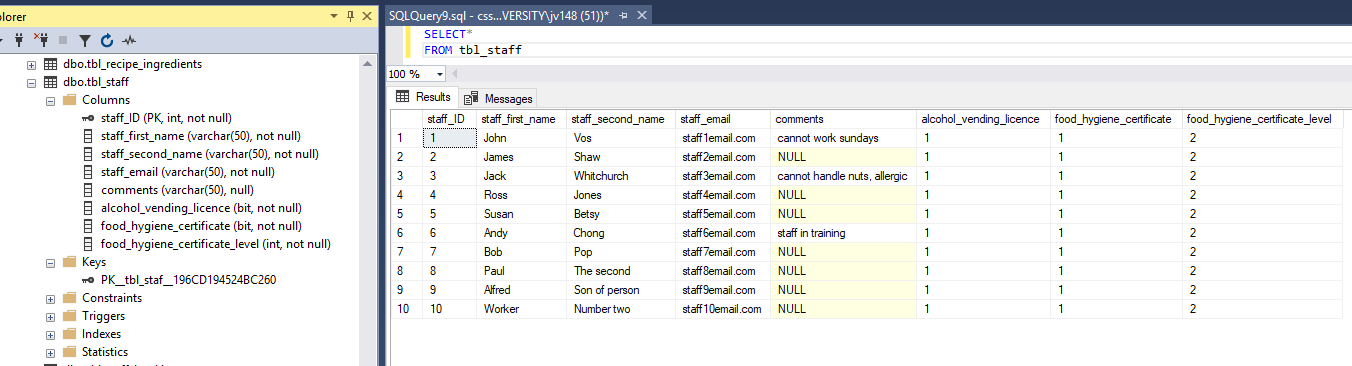
(6, 'Andy', 'Chong', 'staff6email.com', 'staff in training', 1, 1, 2),

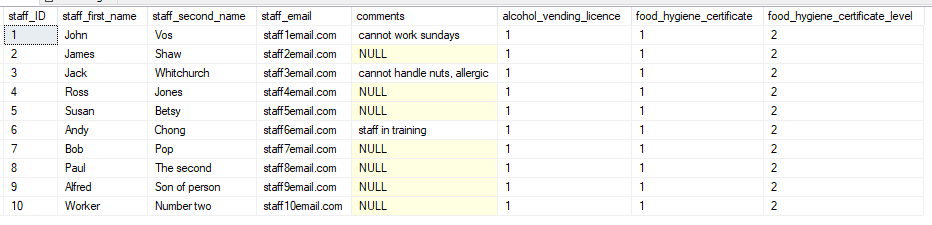
(7, 'Bob', 'Pop', 'staff7email.com', NULL, 1, 1, 2),

(8, 'Paul', 'The second', 'staff8email.com', NULL, 1, 1, 2),

(9, 'Alfred', 'Son of person', 'staff9email.com', NULL, 1, 1, 2),

(10, 'Worker', 'Number two', 'staff10email.com', NULL, 1, 1, 2);





# Table Staff Booking:

## SQL Statements:

### Create table:

CREATE TABLE tbl\_staff\_booking(

staff\_ID int NOT NULL,

booking\_ID int NOT NULL,

staff\_hours int NOT NULL,

staff\_role varchar(15),

alcohol\_vending\_licence bit,

PRIMARY KEY (staff\_ID, booking\_ID),

FOREIGN KEY (staff\_ID) REFERENCES tbl\_staff(staff\_ID),

FOREIGN KEY (booking\_ID) REFERENCES tbl\_booking(booking\_ID),

FOREIGN KEY (staff\_role) REFERENCES tbl\_staff\_pay(staff\_role)

);

### Insert into:

/\* inserts 10 cells of data into staff booking\*/

INSERT INTO tbl\_staff\_booking(staff\_ID, booking\_ID, staff\_hours, staff\_role, alcohol\_vending\_licence)

VALUES

(1, 1, 8, 'supervisor', 1),

(2, 1, 8, 'supervisor', 0),

(3, 2, 6, 'supervisor', 0),

(3, 3, 8, 'staff', 1),

(4, 3, 8, 'staff', 0),

(5, 4, 8, 'staff', 0),

(6, 5, 8, 'staff', 0),

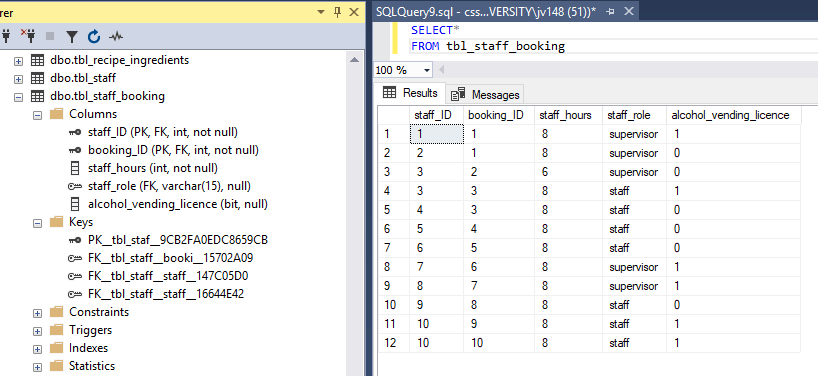
(7, 6, 8, 'supervisor', 1),

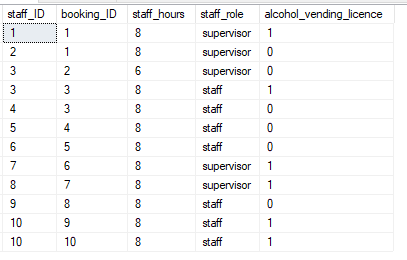
(8, 7, 8, 'supervisor', 1),

(9, 8, 8, 'staff', 0),

(10, 9, 8, 'staff', 1),

(10, 10, 8, 'staff', 1);





# Table Staff Pay:

## SQL Statements:

CREATE TABLE tbl\_staff\_pay (

staff\_role varchar(15) NOT NULL,

rate\_of\_pay smallmoney NOT NULL,

PRIMARY KEY(staff\_role),

)

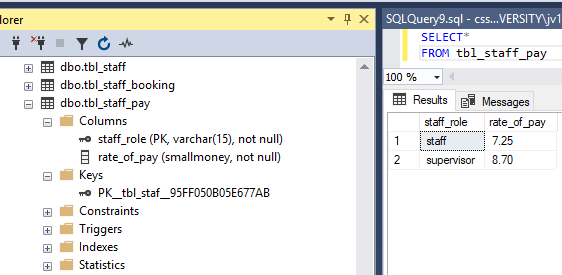
## Insert Into:

INSERT INTO tbl\_staff\_pay (staff\_role, rate\_of\_pay)

VALUES

('supervisor', 7.25\*1.2),

('staff', 7.25);



# Table Event Details:

## SQL Statements:

### Create table:

CREATE TABLE tbl\_event\_details(

event\_ID int NOT NULL,

event\_name varchar(50) NOT NULL,

start\_time time NOT NULL,

event\_address\_1 nvarchar(50) NOT NULL,

event\_address\_2 nvarchar(50),

event\_post\_code nvarchar(8) NOT NULL,

event\_type varchar(2) NOT NULL,

number\_of\_covers int,

alcohol\_present bit,

PRIMARY KEY (event\_ID),

);

### Insert into:

/\*inserts 10 cells of data into event\_details\*/

INSERT INTO tbl\_event\_details(event\_ID, event\_name, start\_time, event\_address\_1, event\_address\_2, event\_post\_code, event\_type, number\_of\_covers, alcohol\_present)

VALUES

(1, 'ABCD Road', '09:00:00', 'ABCD Road', 'Somewhere', 'UW8 9EP', 'OC', 200, 1),

(2, 'Cliffe High St', '06:30:00', 'Cliffe High St', 'Lewes', 'BN7 2AP', 'DS', NULL, 0),

(3, 'Brighton Grande Parade', '10:30:00', 'LOPK Street', 'Elsewhere', 'POO 1IK', 'DS', NULL,1),

(4, 'Lewes Farmers Market', '06:30:00', 'Cliffe High St', 'Lewes', 'BN7 2AP', 'DS', NULL, 1),

(5, 'Carl Birthday Party', '11:00:00', 'QWERTY blv', 'Everywhere', 'KR3 ST5', 'OC', 25, 1),

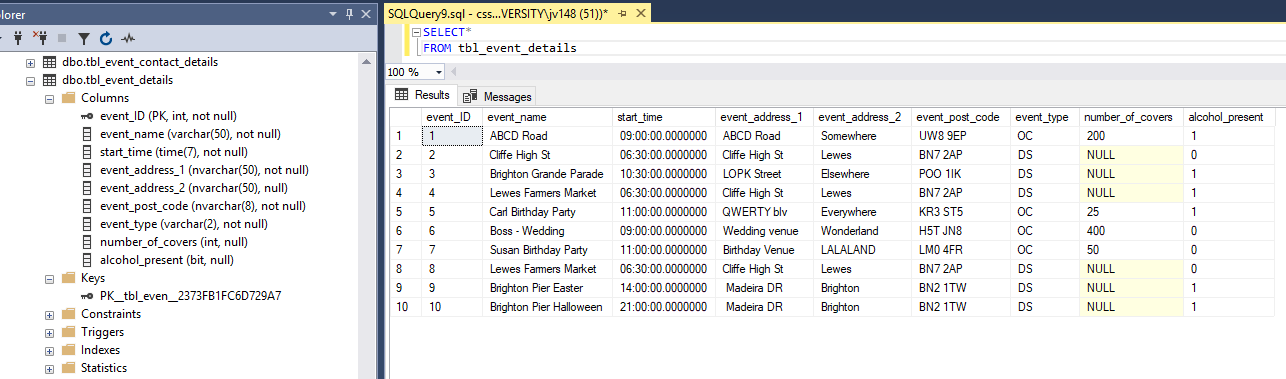
(6,'Boss - Wedding', '09:00:00', 'Wedding venue', 'Wonderland', 'H5T JN8', 'OC', 400, 0),

(7, 'Susan Birthday Party', '11:00:00', 'Birthday Venue', 'LALALAND', 'LM0 4FR', 'OC', 50, 0),

(8,'Lewes Farmers Market', '06:30:00', 'Cliffe High St', 'Lewes', 'BN7 2AP', 'DS', NULL, 0),

(9,'Brighton Pier Easter', '14:00:00', ' Madeira DR', 'Brighton', 'BN2 1TW', 'DS', NULL, 1),

(10,'Brighton Pier Halloween', '21:00:00', ' Madeira DR', 'Brighton', 'BN2 1TW', 'DS', NULL, 1);



# Table Event Contact Details:

## SQL Statements:

### Create table:

CREATE TABLE tbl\_event\_contact\_details(

event\_ID int NOT NULL,

event\_contact\_name varchar(50) NOT NULL,

event\_contact\_tel\_numb char(10) NOT NULL,

event\_contact\_email varchar(40) NOT NULL,

PRIMARY KEY (event\_ID),

FOREIGN KEY (event\_ID) REFERENCES tbl\_event\_details(event\_ID)

);

### Insert into:

/\* inserts 10 cells of data into staff booking\*/

INSERT INTO tbl\_event\_contact\_details(event\_ID, event\_contact\_name, event\_contact\_tel\_numb, event\_contact\_email)

VALUES

(1, 'rogan fuss', 07700900597, 'evencontact1@mail.com'),

(2, 'trey parks', 07700900029, 'evencontact2@mail.com'),

(3, 'alaya ahmad', 07632960579, 'evencontact3@mail.com'),

(4, 'trey parks', 07700900029, 'evencontact4@mail.com'),

(5, 'lochlan gardner', 07632960395, 'evencontact5@mail.com'),

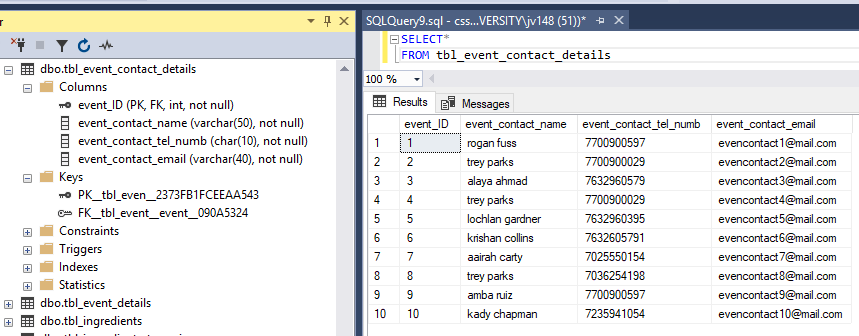
(6, 'krishan collins', 07632605791, 'evencontact6@mail.com'),

(7, 'aairah carty', 07025550154, 'evencontact7@mail.com'),

(8, 'trey parks', 07036254198, 'evencontact8@mail.com'),

(9, 'amba ruiz', 07700900597, 'evencontact9@mail.com'),

(10, 'kady chapman', 07235941054, 'evencontact10@mail.com');



# Table Truck:

## SQL Statements:

### Create table:

CREATE TABLE tbl\_truck(

truck\_ID varchar(4) NOT NULL,

truck\_VIN varchar(17) NOT NULL,

PRIMARY KEY (truck\_ID),

);

### Insert into:

/\*inserts 5 cells of data into truck\*/

INSERT INTO tbl\_truck(truck\_ID, truck\_VIN)

VALUES

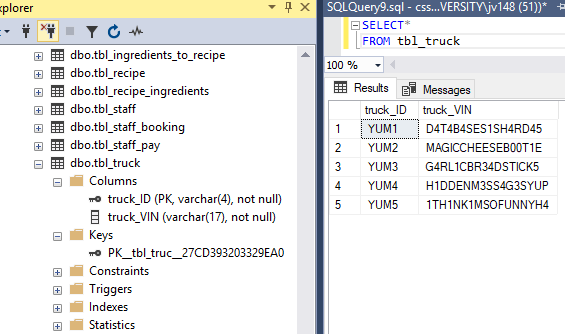
('YUM1', 'D4T4B4SES1SH4RD45'),

('YUM2', 'MAGICCHEESEB00T1E'),

('YUM3', 'G4RL1CBR34DSTICK5'),

('YUM4', 'H1DDENM3SS4G3SYUP'),

('YUM5', '1TH1NK1MSOFUNNYH4');



# Table Truck Booking:

## SQL Statements:

### Create table:

CREATE TABLE tbl\_truck\_booking(

truck\_ID varchar(4) NOT NULL,

booking\_ID int NOT NULL,

recipe\_ID VARCHAR(3) NOT NULL,

booking\_date date NOT NULL,

PRIMARY KEY (truck\_ID, booking\_ID, recipe\_ID),

FOREIGN KEY (truck\_ID) REFERENCES tbl\_truck(truck\_ID),

FOREIGN KEY (recipe\_ID) REFERENCES tbl\_recipe(recipe\_ID),

FOREIGN KEY (booking\_ID) REFERENCES tbl\_booking(booking\_ID)

);

### Insert into:

INSERT INTO tbl\_truck\_booking(truck\_ID, booking\_ID, recipe\_ID, booking\_date)

VALUES

('YUM1', 1, 'HF1', '2019-03-15'),

('YUM1', 1, 'HF2', '2019-03-15'),

('YUM2', 1, 'HT6', '2019-03-15'),

('YUM2', 1, 'JS1', '2019-03-15'),

('YUM3', 2, 'CK6', '2019-03-01'),

('YUM3', 2, 'VB1', '2019-03-01'),

('YUM4', 3, 'VT1', '2019-04-22'),

('YUM4', 3, 'LM3', '2019-04-22'),

('YUM1', 4, 'PC3', '2019-04-01'),

('YUM1', 4, 'VM7', '2019-04-01'),

('YUM2', 5, 'HF1', '2019-05-16'),

('YUM2', 5, 'HF2', '2019-05-16'),

('YUM3', 6, 'HT6', '2019-05-16'),

('YUM3', 6, 'JS1', '2019-05-16'),

('YUM4', 7, 'CK6', '2019-05-30'),

('YUM4', 7, 'VB1', '2019-05-30'),

('YUM1', 8, 'VT1', '2019-05-01'),

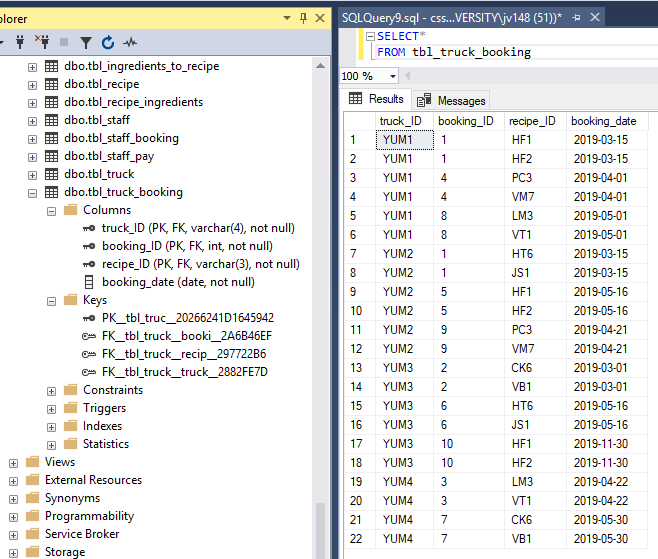
('YUM1', 8, 'LM3', '2019-05-01'),

('YUM2', 9, 'PC3', '2019-04-21'),

('YUM2', 9, 'VM7', '2019-04-21'),

('YUM3', 10, 'HF1', '2019-11-30'),

('YUM3', 10, 'HF2', '2019-11-30');



# Table Accident Log:

## SQL Statements:

### Create table:

CREATE TABLE tbl\_accident\_log(

accident\_ID int NOT NULL,

booking\_ID int NOT NULL,

staff\_ID int NOT NULL,

date\_of\_incident date NOT NULL,

date\_reported date NOT NULL,

comments varchar(200) NOT NULL,

PRIMARY KEY (accident\_ID),

FOREIGN KEY (booking\_ID) REFERENCES tbl\_booking(booking\_ID),

FOREIGN KEY (staff\_ID) REFERENCES tbl\_staff(staff\_ID)

);

### Insert into:

/\* inserts 5 cells of data into accident log\*/

INSERT INTO tbl\_accident\_log(accident\_ID, booking\_ID, staff\_ID, date\_of\_incident, date\_reported, comments)

VALUES

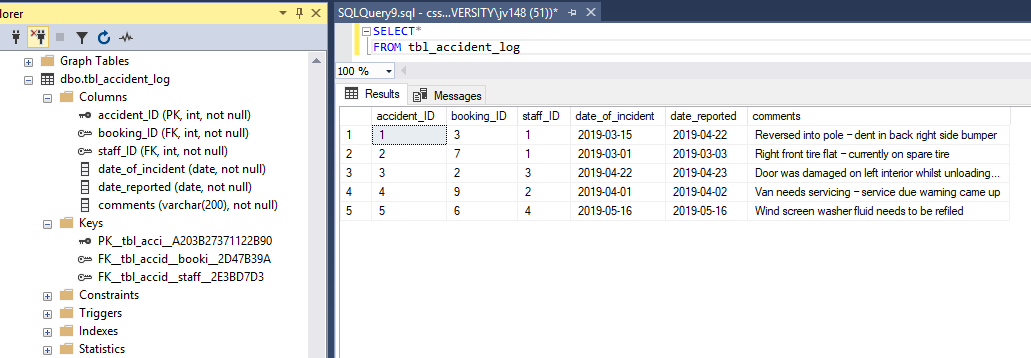
(1, 3, 1, '2019-03-15', '2019-04-22', 'Reversed into pole – dent in back right side bumper'),

(3, 2, 3, '2019-04-22', '2019-04-23', 'Door was damaged on left interior whilst unloading tables for event'),

(4, 9, 2, '2019-04-01', '2019-04-02', 'Van needs servicing – service due warning came up'),

(2, 7, 1, '2019-03-01', '2019-03-03', 'Right front tire flat – currently on spare tire'),

(5, 6, 4, '2019-05-16', '2019-05-16', 'Wind screen washer fluid needs to be refiled');



# Table Recipe:

## SQL Statements:

### Create table:

CREATE TABLE tbl\_recipe(

recipe\_ID varchar(3) NOT NULL,

recipe\_name varchar(40) NOT NULL,

PRIMARY KEY (recipe\_ID),

);

### Insert into:

/\*inserts 10 cells of data into recipe\*/

INSERT INTO tbl\_recipe(recipe\_ID, recipe\_name)

VALUES

('HF1', 'hummus falafel wrap'),

('HF2', 'falafel and tabbouleh'),

('HT6', 'tabbouleh roasted veg'),

('JS1', 'apple and banana smoothie'),

('CK6', 'evening cocktails'),

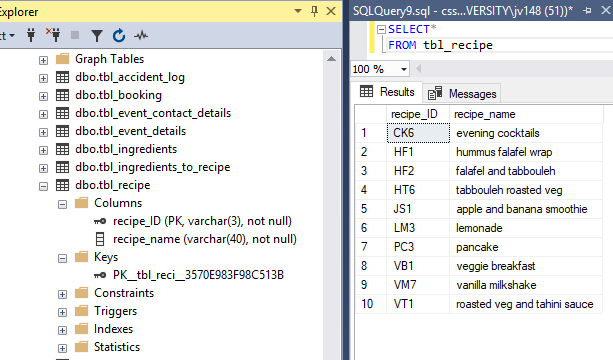
('VB1', 'veggie breakfast'),

('VT1', 'roasted veg and tahini sauce'),

('LM3', 'lemonade'),

('PC3', 'pancake'),

('VM7', 'vanilla milkshake');



# Table Ingredients:

## SQL Statements:

### Create table:

CREATE TABLE tbl\_ingredients(

ingredient\_ID int NOT NULL,

contains\_allergen bit NOT NULL,

ingredient varchar(200) NOT NULL,

PRIMARY KEY (ingredient\_ID)

);

### Insert into:

/\*inserts 28 cells of data into ingredients\*/

INSERT INTO tbl\_ingredients(ingredient\_ID, contains\_allergen , ingredient)

VALUES

(1, 1, 'cereal (wheat, rye, barely, oats)'),

(2, 1, 'crustaceans (prawns, lobster, and crayfish)'),

(3, 1, 'eggs'),

(4, 1, 'fish'),

(5, 1, 'peanuts'),

(6, 1, 'soybeans'),

(7, 1, 'milk (including lactose)'),

(8, 1, 'nuts (almonds, hazelnuts, walnuts, cashews, pecan nuts, brazil nuts, pistachio nuts, and macadamia)'),

(9, 1, 'celery (including celeriac)'),

(10, 1, 'mustard'),

(11, 1, 'sesame'),

(12, 1, 'sulphur dioxide / sulphites, where added and at a level above 10mg/kg or 10mg/L in the finished product. This can be used as a preservative in dried fruit'),

(13, 1, 'lupin, which includes lupin seeds and flour and can be found in types of bread, pastries and pasta'),

(14, 1, 'molluscs like, mussels, whelks, oysters, snails and squid'),

(15, 0, 'tahini'),

(16, 0, 'banana'),

(17, 0, 'chicken breast'),

(18, 0, 'lemon'),

(19, 0, 'cayenne pepper'),

(20, 0, 'sesame seeds'),

(21, 0, 'lamb mince'),

(22, 0, 'thyme'),

(23, 0, 'olive oil'),

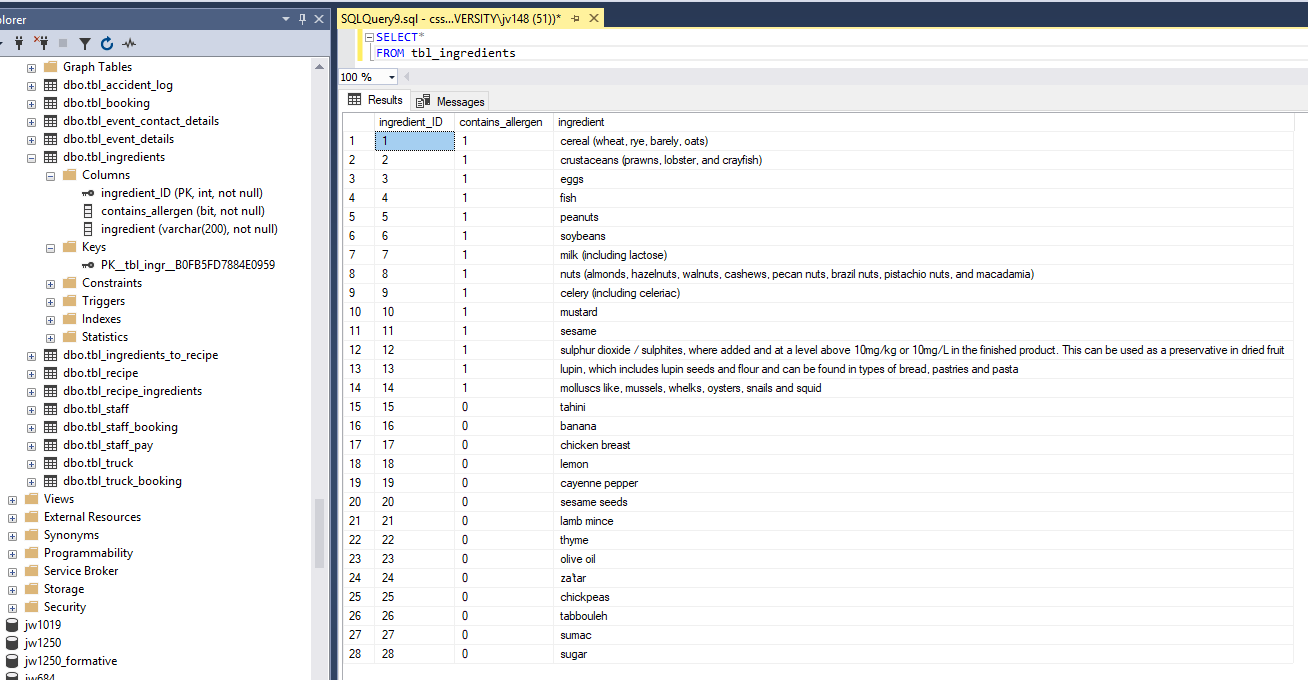
(24, 0, 'za''tar'),

(25, 0, 'chickpeas'),

(26, 0, 'tabbouleh'),

(27, 0, 'sumac'),

(28, 0, 'sugar');



# Table Recipe Ingredients:

## SQL Statements:

### Create table:

CREATE TABLE tbl\_recipe\_ingredients(

ingredient\_ID int NOT NULL,

recipe\_ID varchar(3) NOT NULL,

quantity varchar(15) NOT NULL,

PRIMARY KEY (ingredient\_ID, recipe\_ID),

FOREIGN KEY (ingredient\_ID) REFERENCES tbl\_ingredients(ingredient\_ID),

FOREIGN KEY (recipe\_ID) REFERENCES tbl\_recipe(recipe\_ID)

);

### Insert into:

/\*inserts 31 cells of data into recipe ingredients\*/

INSERT INTO tbl\_recipe\_ingredients(ingredient\_ID, recipe\_ID, quantity)

VALUES

(25, 'HF1', '500g'),

(23, 'HF1', '25g'),

(15, 'HF1', '1 clove'),

(19, 'HF1', '1 ground'),

(20, 'HF1', '1 pinch'),

(22, 'HF1', '1 branch'),

(25, 'HF2', '500g'),

(23, 'HF2', '25g'),

(15, 'HF2', '1 clove'),

(19, 'HF2', '1 ground'),

(20, 'HF2', '1 pinch'),

(22, 'HF2', '1 branch'),

(26, 'HF2', '2 spoons'),

(25, 'HT6', '500g'),

(23, 'HT6', '25g'),

(15, 'HT6', '1 clove'),

(19, 'HT6', '1 ground'),

(20, 'HT6', '1 pinch'),

(22, 'HT6', '1 branch'),

(27, 'JS1', '4 apples'),

(16, 'JS1', '4 bananas'),

(24, 'JS1', '500ml'),

(27, 'VB1', '2 apples'),

(16, 'VB1', '2 bananas'),

(1, 'VB1', '1 bowl'),

(25, 'VT1', '500g'),

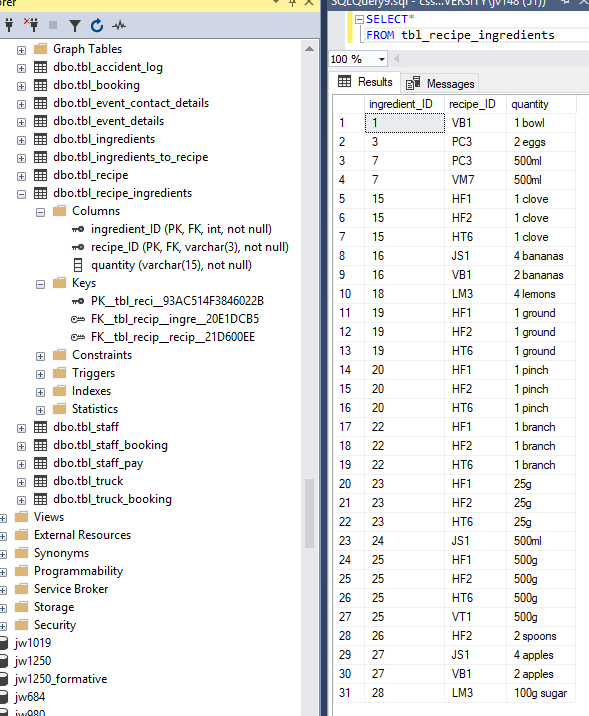
(18, 'LM3', '4 lemons'),

(28, 'LM3', '100g sugar'),

(7, 'PC3', '500ml'),

(3, 'PC3', '2 eggs'),

(7, 'VM7', '500ml');



# Table Ingredients to Recipe:

## SQL Statements:

### Create table:

CREATE TABLE tbl\_ingredients\_to\_recipe(

main\_ingredient int FOREIGN KEY REFERENCES tbl\_ingredients(ingredient\_ID),

sub\_ingredient int FOREIGN KEY REFERENCES tbl\_ingredients(ingredient\_ID),

quantity varchar(15) NOT NULL,

PRIMARY KEY(main\_ingredient, sub\_ingredient)

);

### Insert into:

INSERT INTO tbl\_ingredients\_to\_recipe(main\_ingredient, sub\_ingredient, quantity)

VALUES

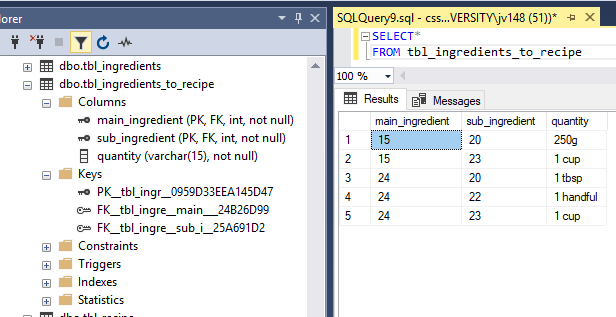
(15, 20, '250g'),

(15, 23, '1 cup'),

(24, 22, '1 handful'),

(24, 20, '1 tbsp'),

(24, 23, '1 cup');



# Must:

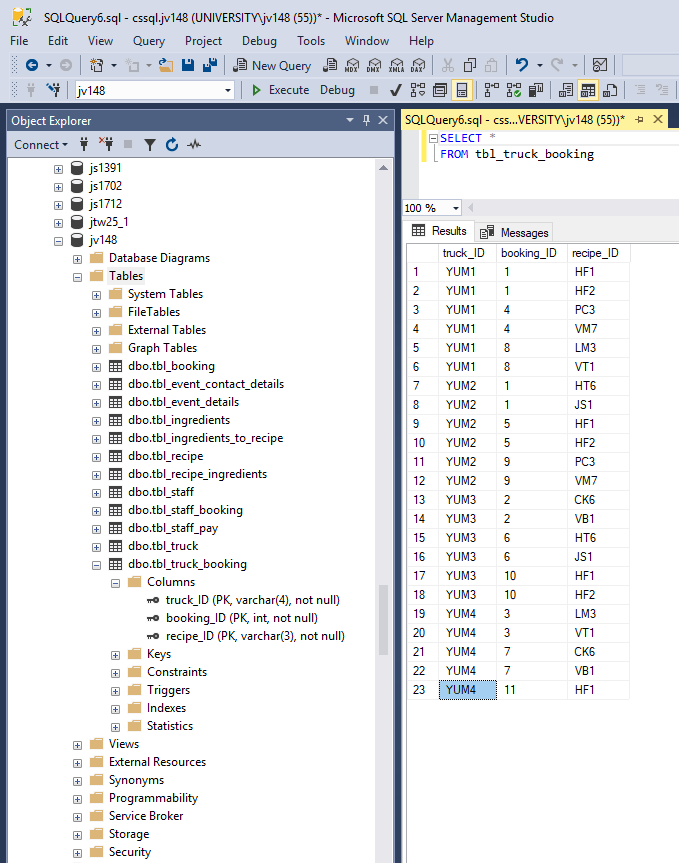
## Record /edit/cancel bookings for multiple trucks at outdoor events

### Record booking:

INSERT INTO tbl\_truck\_booking(truck\_ID, booking\_ID, recipe\_ID)

VALUES

('YUM4', 11, 'HF1');



### Edit booking:

UPDATE tbl\_truck\_booking

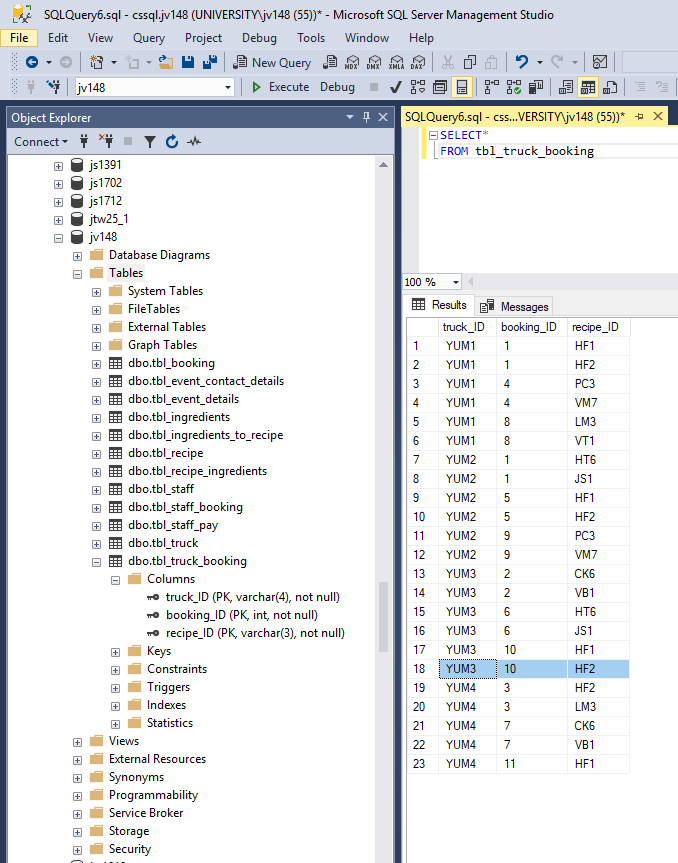
SET recipe\_ID = 'HF2'

WHERE

truck\_ID = 'YUM4'

AND booking\_ID = 3

AND recipe\_ID = 'VT1'



### Cancel booking:

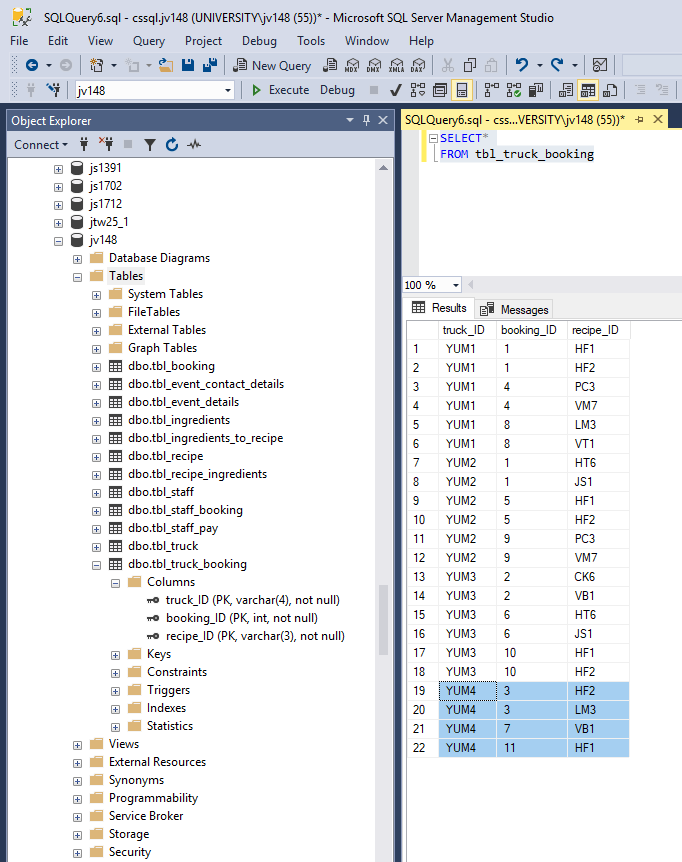
DELETE

FROM tbl\_truck\_booking

WHERE truck\_ID = 'YUM4'

AND booking\_ID = 7

AND recipe\_ID = 'CK6'



## For all staff, ensure that a level 2 food hygiene certificate is in place

CREATE TRIGGER food\_hygiene\_certificate

ON tbl\_staff

FOR INSERT AS DECLARE @food\_hygiene INTEGER

SELECT @food\_hygiene = food\_hygiene\_certificate

FROM inserted

IF @food\_hygiene < 2

BEGIN

PRINT('a food hygiene certificate level 2 or higher is required')

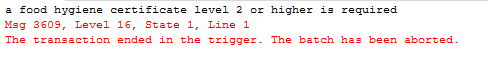
ROLLBACK TRANSACTION

END

INSERT INTO tbl\_staff(staff\_ID, staff\_email, comments, alcohol\_vending\_licence, food\_hygiene\_certificate, food\_hygiene\_certificate\_level)

VALUES

(11, 'staff11@mail.com', 'in training for food certificate', 1, 1, 0);



## For each item sold, produce a list of ingredients

SELECT tbl\_ingredients.ingredient\_ID, contains\_allergen, ingredient

FROM tbl\_recipe

INNER JOIN tbl\_recipe\_ingredients

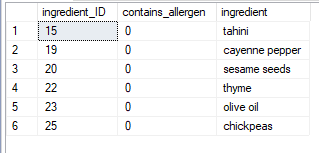
ON tbl\_recipe.recipe\_ID = tbl\_recipe\_ingredients.recipe\_ID

INNER JOIN tbl\_ingredients

ON tbl\_ingredients.ingredient\_ID = tbl\_recipe\_ingredients.ingredient\_ID

WHERE

tbl\_recipe.recipe\_ID = 'HF1'



SELECT tbl\_ingredients.ingredient\_ID, contains\_allergen, ingredient

FROM tbl\_recipe

INNER JOIN tbl\_recipe\_ingredients

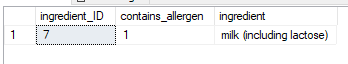
ON tbl\_recipe.recipe\_ID = tbl\_recipe\_ingredients.recipe\_ID

INNER JOIN tbl\_ingredients

ON tbl\_ingredients.ingredient\_ID = tbl\_recipe\_ingredients.ingredient\_ID

WHERE

tbl\_recipe.recipe\_ID = 'VM7'



SELECT tbl\_ingredients.ingredient\_ID, contains\_allergen, ingredient

FROM tbl\_recipe

INNER JOIN tbl\_recipe\_ingredients

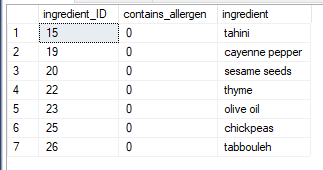
ON tbl\_recipe.recipe\_ID = tbl\_recipe\_ingredients.recipe\_ID

INNER JOIN tbl\_ingredients

ON tbl\_ingredients.ingredient\_ID = tbl\_recipe\_ingredients.ingredient\_ID

WHERE

tbl\_recipe.recipe\_ID = 'HF2'



## Identify all truck bookings for any given date

SELECT tbl\_truck.truck\_ID, truck\_VIN, tbl\_booking.booking\_name, tbl\_event\_details.event\_name

FROM tbl\_truck

INNER JOIN tbl\_truck\_booking

ON tbl\_truck.truck\_ID = tbl\_truck\_booking.truck\_ID

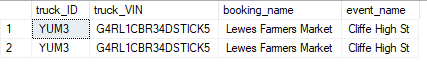
INNER JOIN tbl\_booking

ON tbl\_booking.booking\_ID = tbl\_truck\_booking.booking\_ID

INNER JOIN tbl\_event\_details

ON tbl\_booking.event\_ID = tbl\_event\_details.event\_ID

WHERE booking\_date = '2019-03-01'



## Identify all bookings for a given truck

SELECT DISTINCT tbl\_booking.booking\_ID, event\_name, booking\_date

FROM tbl\_booking

INNER JOIN tbl\_event\_details

ON tbl\_booking.event\_ID = tbl\_event\_details.event\_ID

INNER JOIN tbl\_truck\_booking

ON tbl\_truck\_booking.booking\_ID = tbl\_booking.booking\_ID

WHERE truck\_ID = 'YUM1'



SELECT DISTINCT tbl\_booking.booking\_ID, event\_name, booking\_date

FROM tbl\_booking

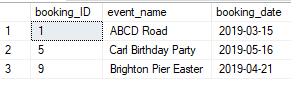
INNER JOIN tbl\_event\_details

ON tbl\_booking.event\_ID = tbl\_event\_details.event\_ID

INNER JOIN tbl\_truck\_booking

ON tbl\_truck\_booking.booking\_ID = tbl\_booking.booking\_ID

WHERE truck\_ID = 'YUM2'



SELECT DISTINCT tbl\_booking.booking\_ID, event\_name, booking\_date

FROM tbl\_booking

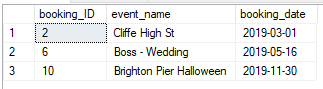
INNER JOIN tbl\_event\_details

ON tbl\_booking.event\_ID = tbl\_event\_details.event\_ID

INNER JOIN tbl\_truck\_booking

ON tbl\_truck\_booking.booking\_ID = tbl\_booking.booking\_ID

WHERE truck\_ID = 'YUM3'



# Should:

## Record any accidents / reportable incidents that take place, including the event at which they happened

SELECT date\_of\_incident, tbl\_staff.staff\_ID, tbl\_staff.staff\_first\_name + ' ' + tbl\_staff.staff\_second\_name AS staff\_name, event\_name, tbl\_staff.comments

FROM tbl\_accident\_log

INNER JOIN tbl\_staff

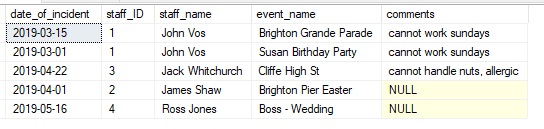
ON tbl\_accident\_log.staff\_ID = tbl\_staff.staff\_ID

INNER JOIN tbl\_booking

ON tbl\_accident\_log.booking\_ID = tbl\_booking.booking\_ID

INNER JOIN tbl\_event\_details

ON tbl\_booking.event\_ID = tbl\_event\_details.event\_ID



## For all staff acting as supervisors, ensure that a level 3 food hygiene certificate is in place

CREATE TRIGGER supervisors\_hygiene\_cert

ON tbl\_staff\_booking

FOR

INSERT

AS

DECLARE

@supervisor\_hygiene int,

@staff\_role varchar(10),

@staff\_ID int

SELECT @staff\_ID = staff\_ID FROM inserted

SELECT @staff\_role = staff\_role FROM inserted

SELECT @supervisor\_hygiene = food\_hygiene\_certificate FROM tbl\_staff

WHERE staff\_ID = @staff\_ID

IF @staff\_role LIKE 'supervisor' AND @supervisor\_hygiene != 3

BEGIN

PRINT('food hygiene certificate level 3 is needed to be a supervisor')

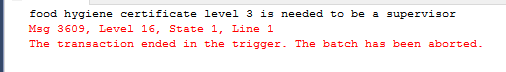
ROLLBACK TRANSACTION

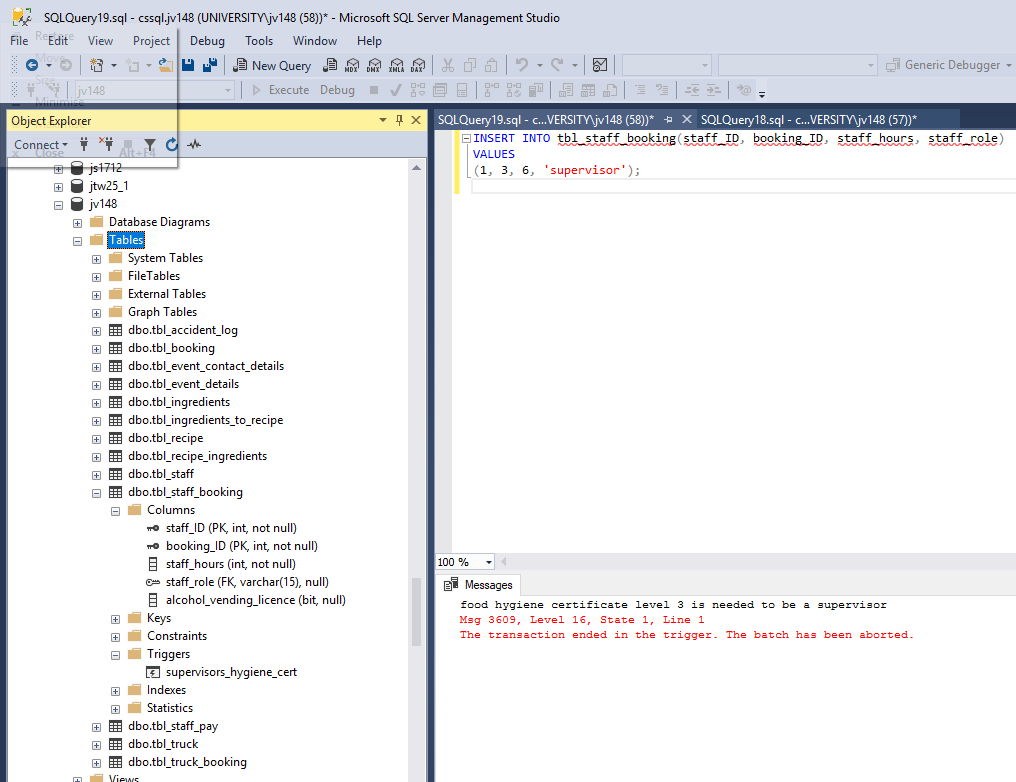
END

INSERT INTO tbl\_staff\_booking(staff\_ID, booking\_ID, staff\_hours, staff\_role)

VALUES

(1, 3, 6, 'supervisor');





## For all events involving alcohol, identify a team member as the license holder.

CREATE TRIGGER alcohol\_license\_present

ON tbl\_staff\_booking

FOR INSERT

AS

BEGIN

DECLARE

@staff\_ID int,

@booking\_ID int,

@event\_ID int,

@license\_needed bit,

@alcohol\_vending\_license\_present bit,

@alcohol\_staff\_license\_holder bit,

@count int = 0

SELECT @staff\_ID = staff\_ID FROM inserted

SELECT @booking\_ID = booking\_ID FROM inserted

SELECT @event\_ID = event\_ID FROM tbl\_booking WHERE booking\_ID = @booking\_ID

SELECT @alcohol\_staff\_license\_holder = alcohol\_vending\_licence FROM inserted

SELECT @license\_needed = alcohol\_present FROM tbl\_event\_details

SELECT @alcohol\_staff\_license\_holder = alcohol\_vending\_licence FROM tbl\_staff

SELECT @count = COUNT(alcohol\_vending\_licence) FROM tbl\_staff\_booking WHERE booking\_ID = @booking\_ID AND alcohol\_vending\_licence != NULL

if @license\_needed = 1 AND (@alcohol\_staff\_license\_holder != 1 OR @alcohol\_vending\_license\_present !=1) AND @count <1

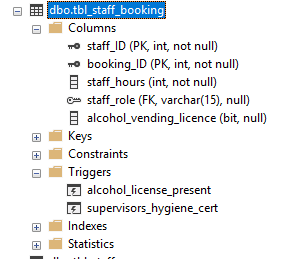
BEGIN

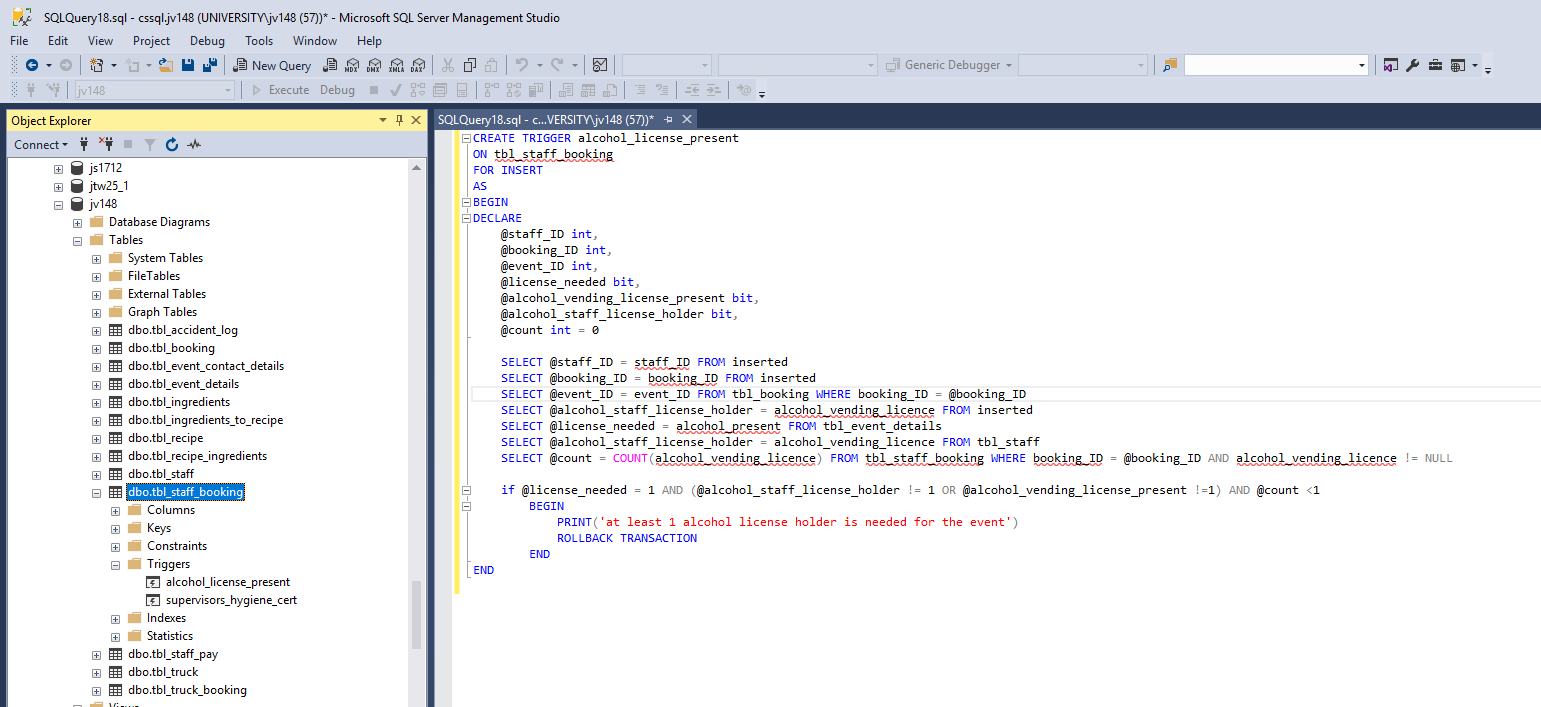
PRINT('at least 1 alcohol license holder is needed for the event')

ROLLBACK TRANSACTION

END

END





## Identify any temporary staff who are not currently working this weekend (i.e. may be available for a last- minute booking).

SELECT staff\_ID, staff\_first\_name + ' ' + staff\_second\_name AS NAME, staff\_email, food\_hygiene\_certificate, alcohol\_vending\_licence

FROM tbl\_staff

WHERE NOT EXISTS(

SELECT \*

FROM tbl\_staff\_booking

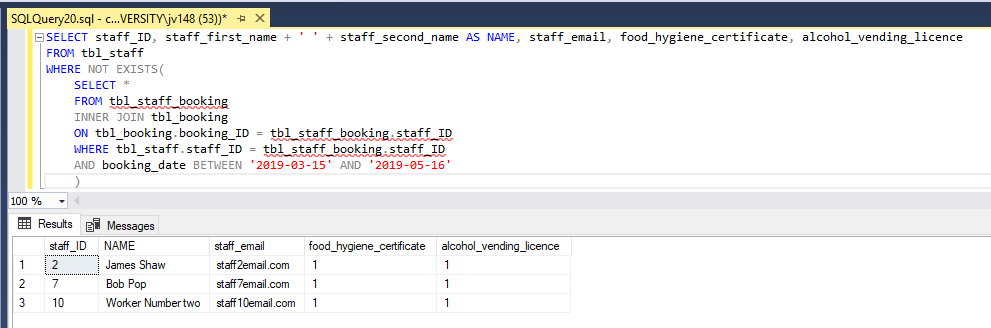
INNER JOIN tbl\_booking

ON tbl\_booking.booking\_ID = tbl\_staff\_booking.staff\_ID

WHERE tbl\_staff.staff\_ID = tbl\_staff\_booking.staff\_ID

AND booking\_date BETWEEN '2019-03-15' AND '2019-05-16'

)



# Could:

## Record members of staff who worked at specific events, including who was acting as the supervisor. This should include the number of hours worked by each staff member, and their rate of pay.

SELECT tbl\_staff.staff\_ID, staff\_first\_name + ' ' + staff\_second\_name AS staff\_name, tbl\_staff\_pay.staff\_role, tbl\_staff\_booking.staff\_hours,

tbl\_staff\_pay.rate\_of\_pay, staff\_hours\*rate\_of\_pay AS salary

FROM tbl\_staff

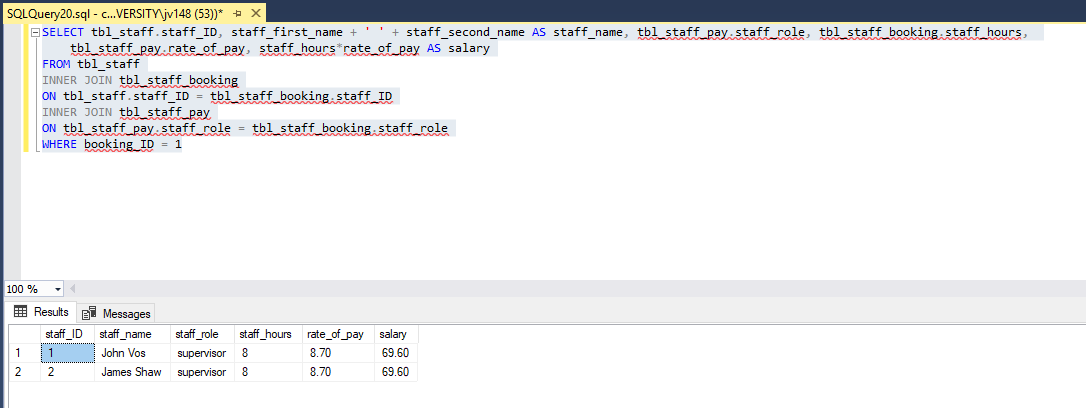
INNER JOIN tbl\_staff\_booking

ON tbl\_staff.staff\_ID = tbl\_staff\_booking.staff\_ID

INNER JOIN tbl\_staff\_pay

ON tbl\_staff\_pay.staff\_role = tbl\_staff\_booking.staff\_role

WHERE booking\_ID = 1



SELECT tbl\_staff.staff\_ID, staff\_first\_name + ' ' + staff\_second\_name AS staff\_name, tbl\_staff\_pay.staff\_role, tbl\_staff\_booking.staff\_hours,

tbl\_staff\_pay.rate\_of\_pay, staff\_hours\*rate\_of\_pay AS salary

FROM tbl\_staff

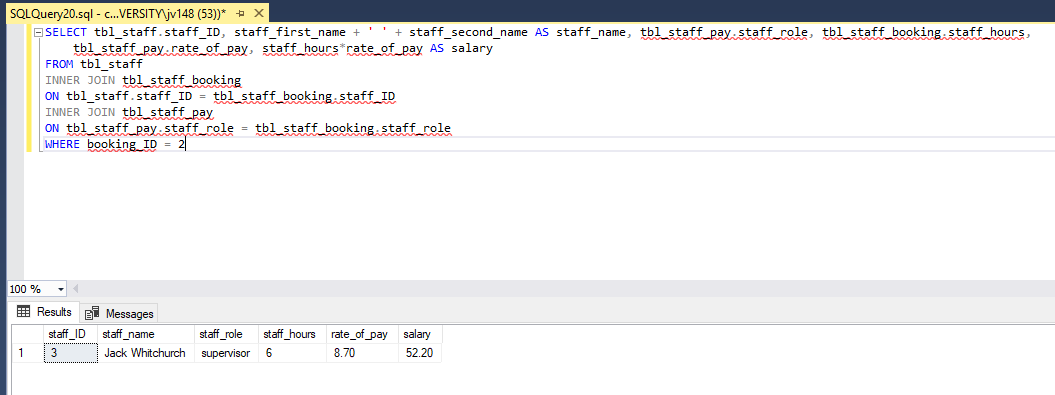
INNER JOIN tbl\_staff\_booking

ON tbl\_staff.staff\_ID = tbl\_staff\_booking.staff\_ID

INNER JOIN tbl\_staff\_pay

ON tbl\_staff\_pay.staff\_role = tbl\_staff\_booking.staff\_role

WHERE booking\_ID = 2



## Produce a list of ingredients for each item sold, with known allergens at the top of the list - this should include sub-ingredients, like the tahini ingredients on the hummus recipe described previously.

SELECT tbl\_recipe\_ingredients.ingredient\_ID, ingredient, contains\_allergen

FROM tbl\_ingredients

INNER JOIN tbl\_recipe\_ingredients

ON tbl\_recipe\_ingredients.ingredient\_ID = tbl\_ingredients.ingredient\_ID

WHERE recipe\_ID = 'HF1'

UNION

SELECT DISTINCT sub.ingredient\_ID, sub.ingredient, sub.contains\_allergen

FROM tbl\_ingredients main

JOIN tbl\_ingredients sub

ON sub.ingredient\_ID= main.ingredient\_ID

WHERE main.ingredient\_ID

IN(

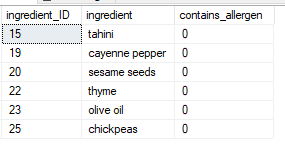
SELECT ingredient\_ID

FROM tbl\_recipe\_ingredients

WHERE recipe\_ID = 'HF1'

)

ORDER BY contains\_allergen DESC



# Would be nice if:

## Identify weekends in the next month with available trucks (to offer last minute deals)

DECLARE

@date date,

@booking\_ID varchar(4),

@booking\_date date

SET @date = CONVERT(DATE, '2019-05-01')

; WITH tbl\_truck\_booking AS

(

SELECT

@date AS [date],

DATENAME (MONTH, @date) AS [month],

DATENAME (DW, @date) AS [day],

@booking\_ID AS [booking],

@booking\_date AS [booking date]

UNION ALL

SELECT

DATEADD(DAY, 1, [date]) AS [date],

DATENAME (MONTH,DATEADD(DAY, 1, [date])) AS [month],

DATENAME (DW ,DATEADD(DAY, 1, [date])) AS [day],

@booking\_ID AS [booking ID],

@booking\_date AS [booking date]

FROM

tbl\_truck\_booking

WHERE

YEAR(DATEADD(DAY, 1 ,[date]) ) = YEAR(@Date)

AND MONTH(DATEADD(DAY, 1, [date])) = MONTH(@Date)

)

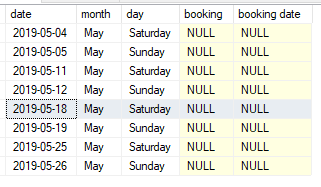
SELECT \*

FROM tbl\_truck\_booking

WHERE [Day] IN ('Saturday', 'Sunday')

ORDER BY [DATE]

OPTION (MAXRECURSION 367)



# Functionality:

## PART 1)

A stored procedure is required that will create a new booking using parameters giving the details of the date and length of event in days, a named contact and the total number of workers required.

CREATE PROCEDURE add\_booking

@booking\_ID int,

@event\_ID int,

@booking\_name varchar(50),

@booking\_date date,

@booking\_duration int,

@repeat\_event bit

AS

DECLARE @errors int

INSERT INTO tbl\_booking(booking\_ID, event\_ID, booking\_name, booking\_date, booking\_duration, repeat\_event)

VALUES

(@booking\_ID, @event\_id, @booking\_name, @booking\_date, @booking\_duration, @repeat\_event)

SET @errors = @@ERROR

IF @errors <> 0

RETURN 99

ELSE

RETURN 00

GO

## PART 2)

A stored procedure is required to produce a list of bookings and staff members (identifying the supervisor) for the forthcoming month.

CREATE PROCEDURE bookings\_per\_month

AS

BEGIN

SELECT tbl\_booking.booking\_ID, booking\_date, staff\_first\_name, staff\_second\_name, staff\_role

FROM tbl\_booking

INNER JOIN tbl\_staff\_booking

ON tbl\_staff\_booking.booking\_ID = tbl\_booking.booking\_ID

INNER JOIN tbl\_staff

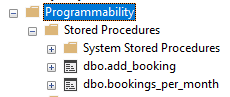
ON tbl\_staff\_booking.staff\_ID = tbl\_staff.staff\_ID

WHERE booking\_date BETWEEN GETDATE() AND DATEADD(MONTH, 1, GETDATE())

END

GO

RETURN



## PART 3)

A trigger is required that is actioned when a booking is cancelled. This will produce all staff who are booked to be working at this event, with their contact details so that that they can be alerted to change.

CREATE TRIGGER booking\_cancelation\_prompt

ON tbl\_booking

INSTEAD OF DELETE

AS

DECLARE

@deleted\_booking\_ID int

SELECT @deleted\_booking\_ID = booking\_ID FROM deleted

BEGIN

SELECT staff\_first\_name, staff\_second\_name, staff\_email

FROM tbl\_staff

INNER JOIN tbl\_staff\_booking

ON tbl\_staff\_Booking.staff\_ID = tbl\_staff.staff\_ID

WHERE booking\_ID = @deleted\_booking\_ID

DELETE FROM tbl\_truck\_booking

WHERE booking\_ID = @deleted\_booking\_ID

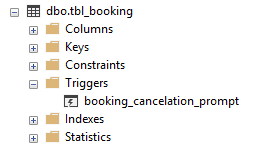
DELETE FROM tbl\_staff\_booking

WHERE booking\_ID = @deleted\_booking\_ID

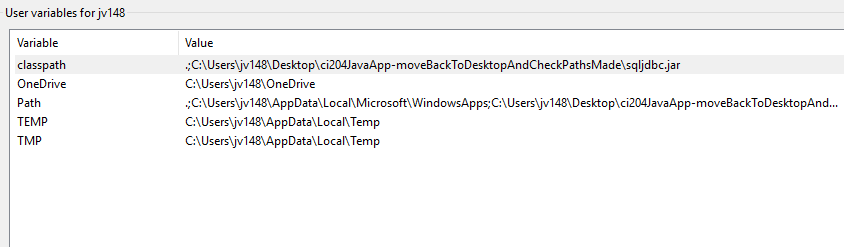
DELETE FROM tbl\_booking

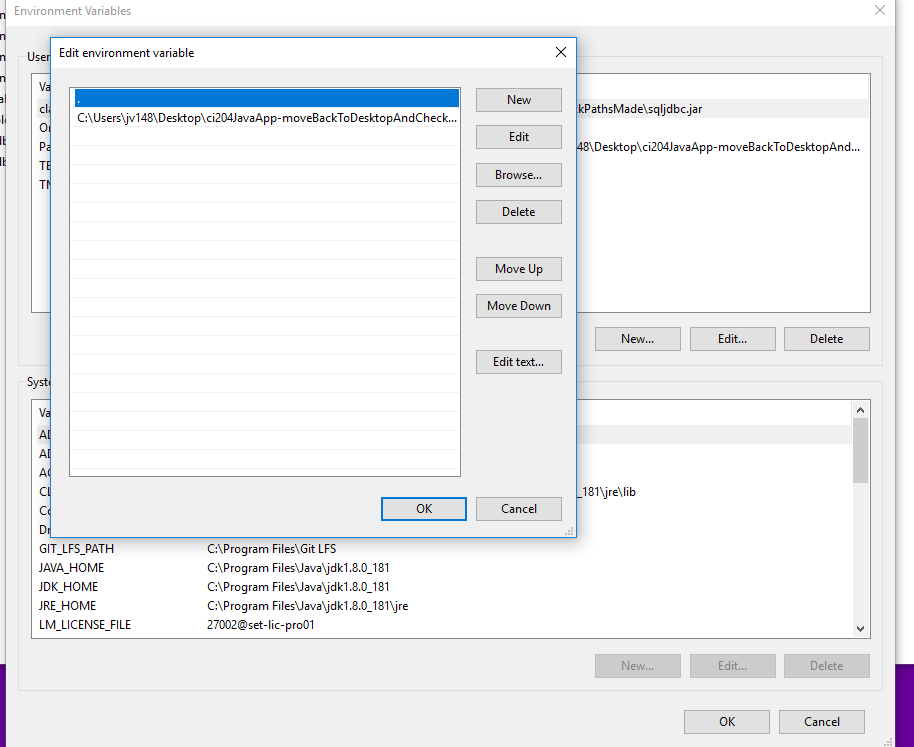
WHERE booking\_ID = @deleted\_booking\_ID

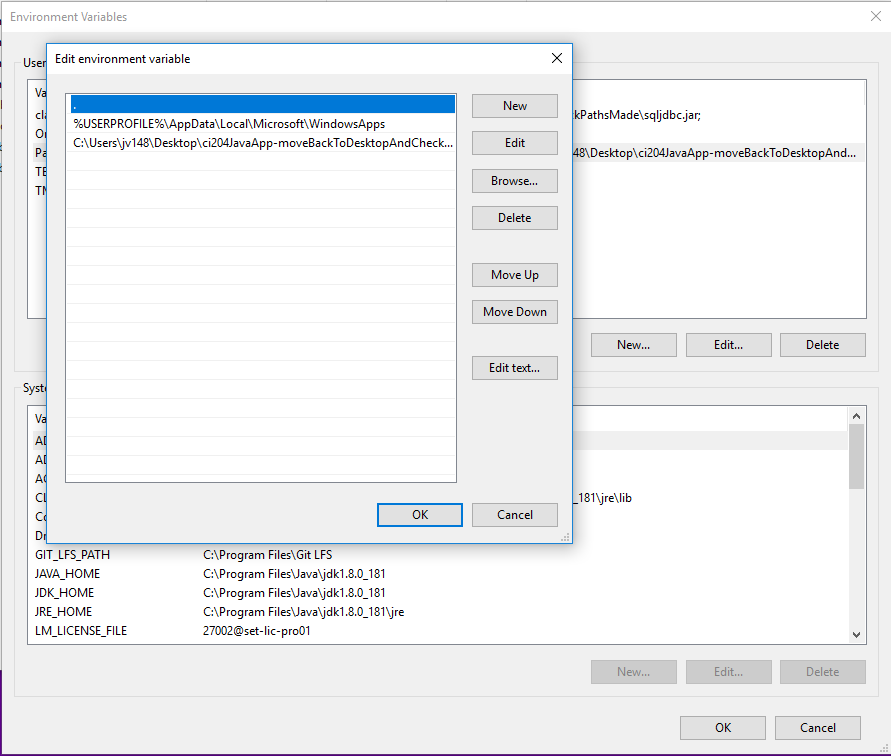
END

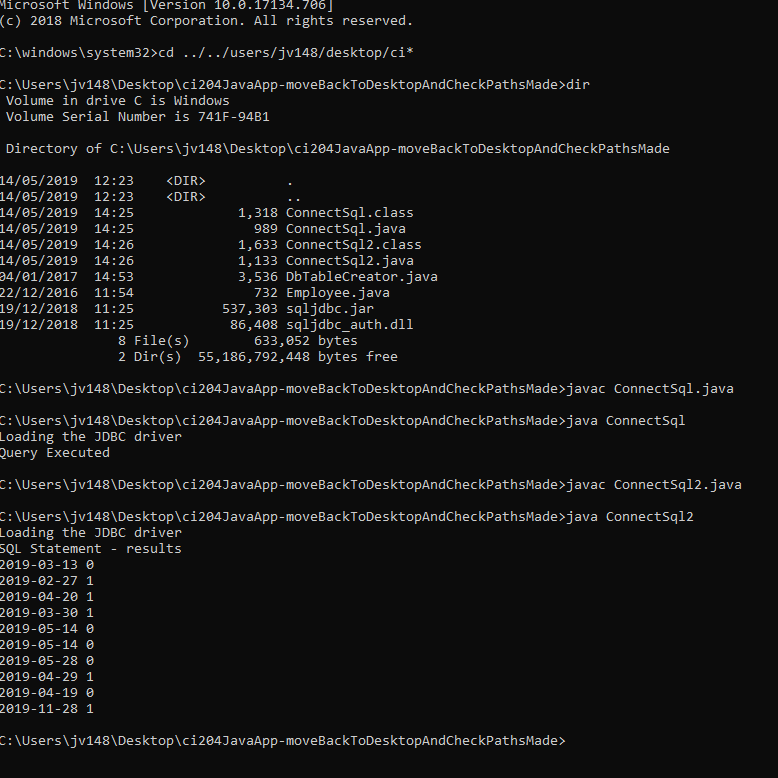


# Create a simple application to provide an interface capable of executing the above functionality (select queries and procedures):









# ERD:

