COMP 1630

Relational Database Design and SQL

Practical Final

Time Permitted: 3 Hours

TOTAL Score
/ 65

INSTRUCTIONS:

This exam is **Open Book**.

What is Allowed

- ☑ You may consult any of your class notes and/or previous labs/projects.
- ☑ You may consult the Internet.
- ☑ You may consult your textbook.

What is NOT Allowed

- ☒ You may NOT share any portion of your solutions with others.
- ☒ You may NOT use anyone else's solutions as your own.
- ☒ You may NOT have someone else complete any portion of your exam.
- ☒ You may NOT discuss your solution with others.
- ☒ You may NOT work with anyone else to complete the exam.

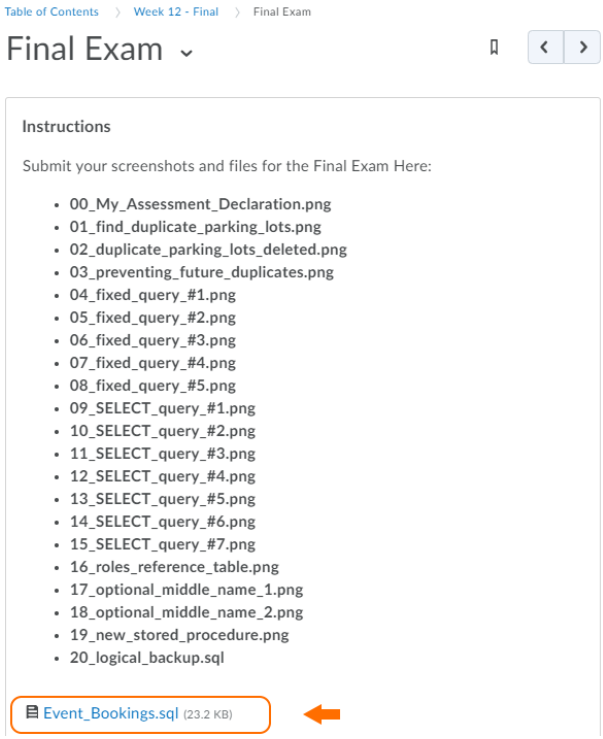
This exam consists of the following sections to complete:

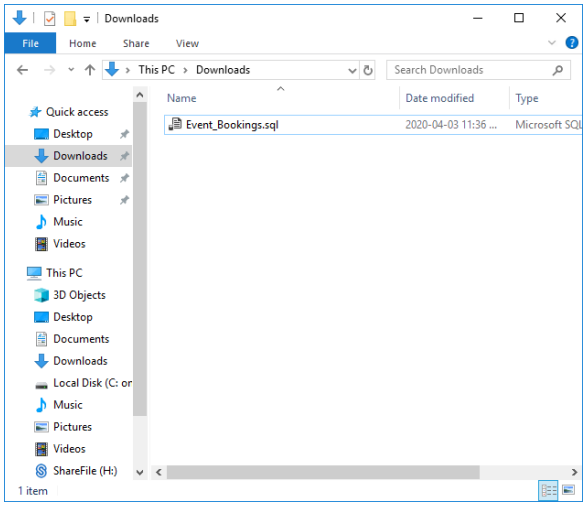
SECTION 1:	Removing Duplicate Rows	8 Marks
SECTION 2:	SQL Statements - Spot the Errors	10 Marks
SECTION 3:	Queries	21 Marks
SECTION 4:	Database Design	10 Marks
SECTION 5:	Optional Middle Name	8 Marks
SECTION 6:	Stored Procedure	5 Marks
SECTION 7:	Backup	3 Marks

Before starting the Exam:

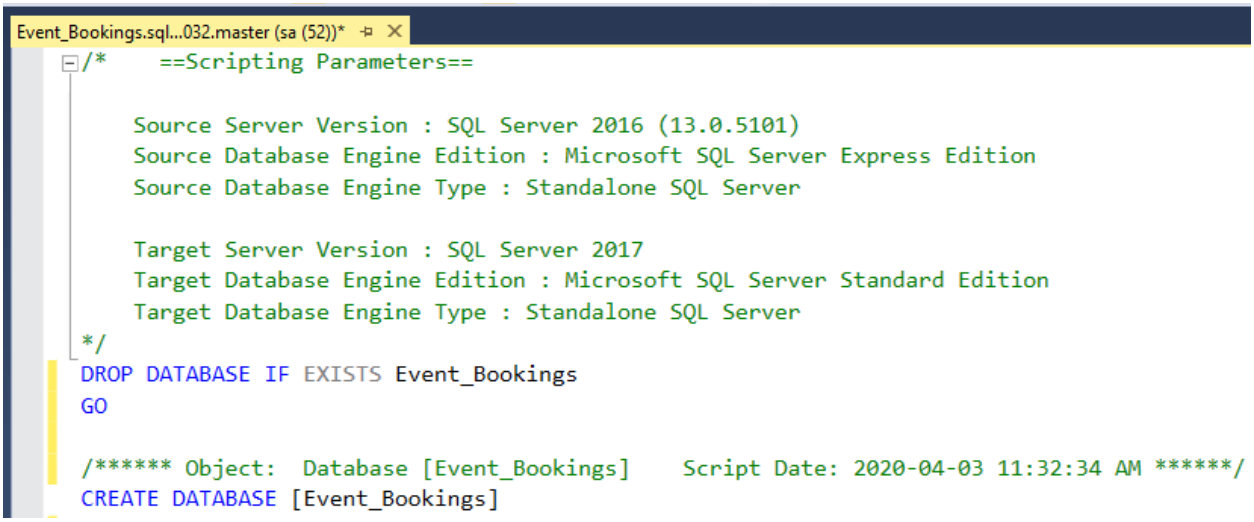
Import the `Event_Bookings` database.

Step 1: Download `Event_Bookings.sql` from the D2L learning hub (learn.bcit.ca).



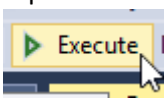


Step 2: Open the file using SQL Server Management Studio (SSMS).



```
Event_Bookings.sql...032.master (sa (52))*
/*    ==Scripting Parameters==

    Source Server Version : SQL Server 2016 (13.0.5101)
    Source Database Engine Edition : Microsoft SQL Server Express Edition
    Source Database Engine Type : Standalone SQL Server

    Target Server Version : SQL Server 2017
    Target Database Engine Edition : Microsoft SQL Server Standard Edition
    Target Database Engine Type : Standalone SQL Server
*/
DROP DATABASE IF EXISTS Event_Bookings
GO

/****** Object:  Database [Event_Bookings]    Script Date: 2020-04-03 11:32:34 AM ******/
CREATE DATABASE [Event_Bookings]
```

Step 3: Execute the entire script.



Event_Bookings.sql...032.master (sa (52))*

```
/*    ==Scripting Parameters==

    Source Server Version : SQL Server 2016 (13.0.5101)
    Source Database Engine Edition : Microsoft SQL Server Express Edition
    Source Database Engine Type : Standalone SQL Server

    Target Server Version : SQL Server 2017
    Target Database Engine Edition : Microsoft SQL Server Standard Edition
    Target Database Engine Type : Standalone SQL Server
*/
DROP DATABASE IF EXISTS Event_Bookings
GO

/****** Object:  Database [Event_Bookings]    Script Date: 2020-04-03 11:32:34 AM ******/
CREATE DATABASE [Event_Bookings]
```

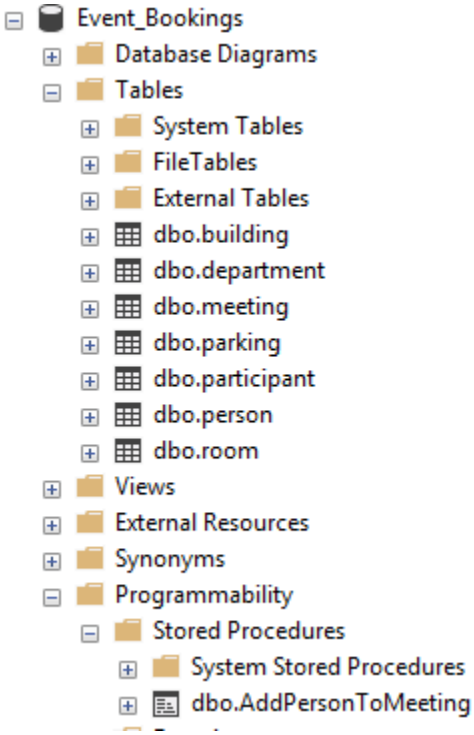
121 %

Messages

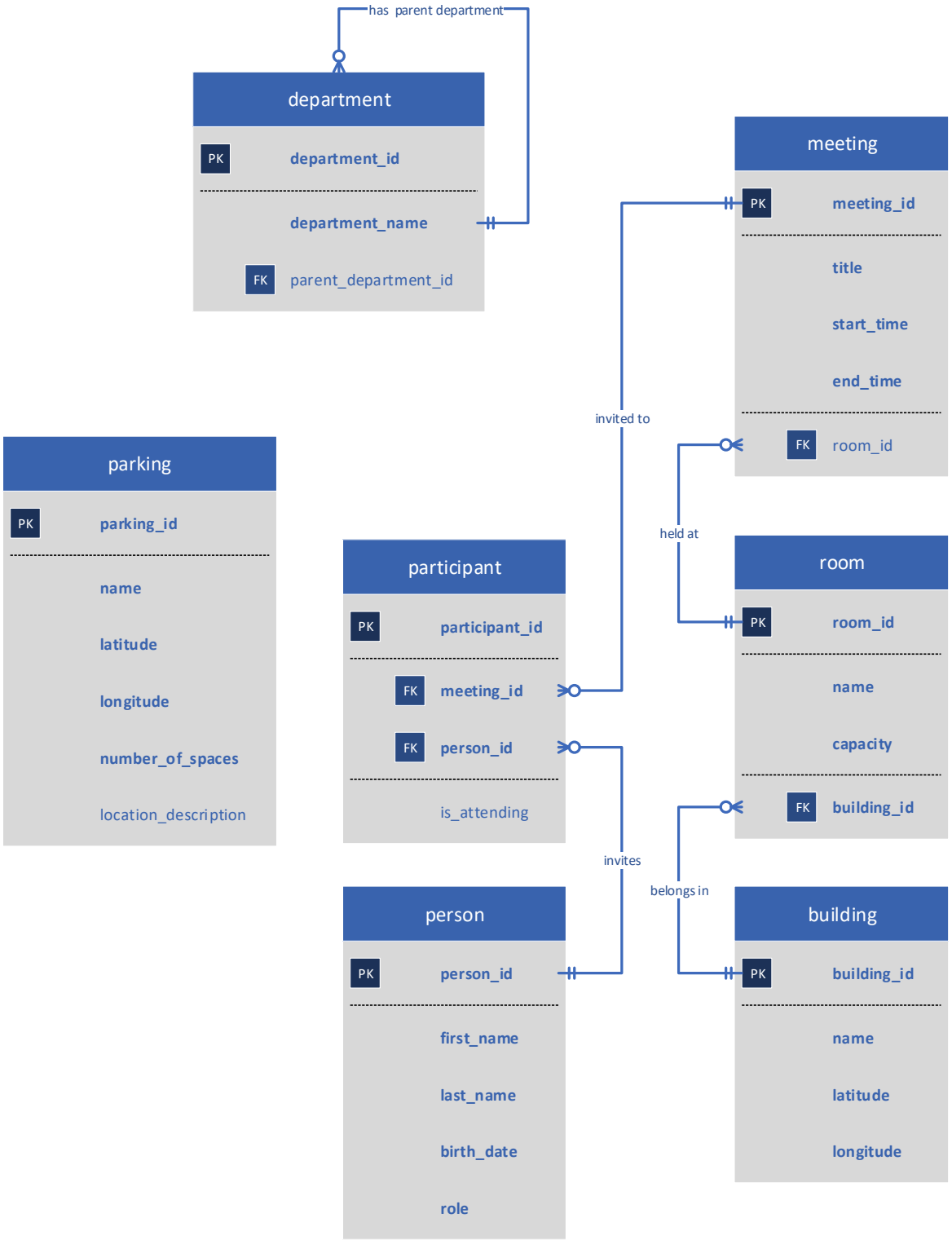
```
(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)
```

Step 4: Verify that you have the following tables in your database:

1. building
2. meeting
3. parking
4. participant
5. person
6. room
7. department



Event_Bookings ERD



As part of your Final Exam, please print, complete, take a picture of and submit this ***BCIT Assessment Declaration*** form.

Submit a picture of your completed form.
<u>Required in your picture</u>:

- Your completed ***BCIT Assessment Declaration***.
  - Assessment: **Final Exam**
  - Assessment Date: **Apr 6, 2020**
  - Course Name: **Relational Database & SQL**
  - Course Number: **COMP 1630**
  - Student Number, Name, Signature and Date are required.

Filename: **00_My_Assessment_Declaration.jpg**

Submit

British Columbia Institute of Technology

School of Computing and Academic Studies

Assessment Declaration

Assessment: Final Exam Assessment Date: Apr 6, 2020

Course Name: Relational Database + SQL Course Number: COMP 1630

I, _____ (print name) hereby declare that the above assessment has been completed and submitted as required by myself as an individual. I have honoured the principles of academic integrity and have upheld British Columbia Institute of Technology's Student Code of Academic Integrity in the completion of this assessment.

Student Name: _____

Student Number: _____

Signature: _____ Date: Apr 6, 2020

British Columbia Institute of Technology

School of Computing and Academic Studies

Assessment Declaration

Assessment: _____ Assessment Date: _____

Course Name: _____ Course Number: _____

I, _____ (print name) hereby declare that the above assessment has been completed and submitted as required by myself as an individual. I have honoured the principles of academic integrity and have upheld British Columbia Institute of Technology's Student Code of Academic Integrity in the completion of this assessment.

Student Name: _____

Student Number: _____

Signature: _____ Date: _____

SECTION 1: Removing Duplicate Rows

SECTION 1
Score
/ 8

In our `Event_Bookings` database, the `parking` table contains a list of all the parking lots on the campus.

parking_id	name	latitude	longitude	number_of_spaces	location_description
1	north_lot	49.91747	-122.93413	120	The North Lot…
2	center_lot	49.91335	-122.93972	500	The biggest lot…
3	south_east_lot	49.91086	-122.93492	50	Reserved for Staff …
4	south_visitor_lot	49.91061	-122.9358	25	Short term free …
5	east_visitor_lot	49.91296	-122.93445	10	Short term free …
6	south_visitor_lot	49.91061	-122.9358	25	Short term free …
7	center_lot	49.91335	-122.93972	500	The biggest lot …
8	bookworms_lot	49.9108	-122.93979	600	The campus's new…
9	north_lot	49.91747	-122.93413	120	The North Lot...
10	center_lot	49.91335	-122.93972	500	The biggest lot…

Unfortunately, the table was created without any contraints and someone has accidentally added the same information multiple times.

We see now, that this is a bad design; so let's fix this.

Step 1: Find which parking lot names have duplicates and how many duplicate rows exist.

Write a query to find all the duplicate parking lot names. [/3 marks]

Your query should produce the following output:

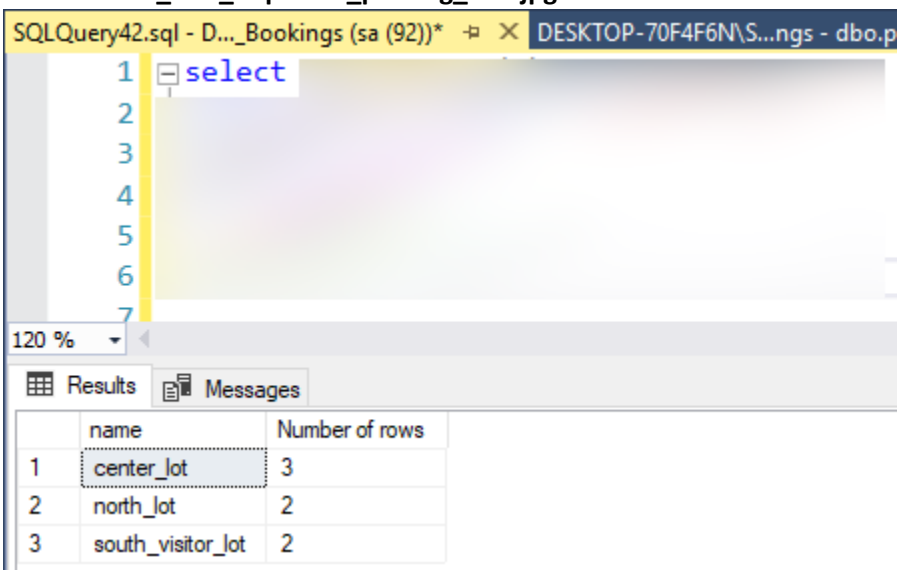
Name	Number of rows
center_lot	3
north_lot	2
south_visitor_lot	2

Submit your screenshot.
<u>Required in your screenshot</u>:

- Your SELECT query that finds the duplicates (pay attention to the column names).
- The results of your query.

Submit

Filename: **01_find_duplicate_parking_lots.jpg**



Step 2: Clean up the duplicate data.

Write and execute a DELETE statement to delete the specific rows that are duplicates. [/ 2 marks]

From Step 1, we now know which parking lot names have duplicates. Find rows to delete, by finding the `parking_id` that matches those rows. For example: if you think that a parking lot is duplicated and has `parking_ids` of 2, 5 and 9 - you decide to delete `parking_ids` 5 and 9. It is important to keep the original row (in this example keep `parking_id` =2).

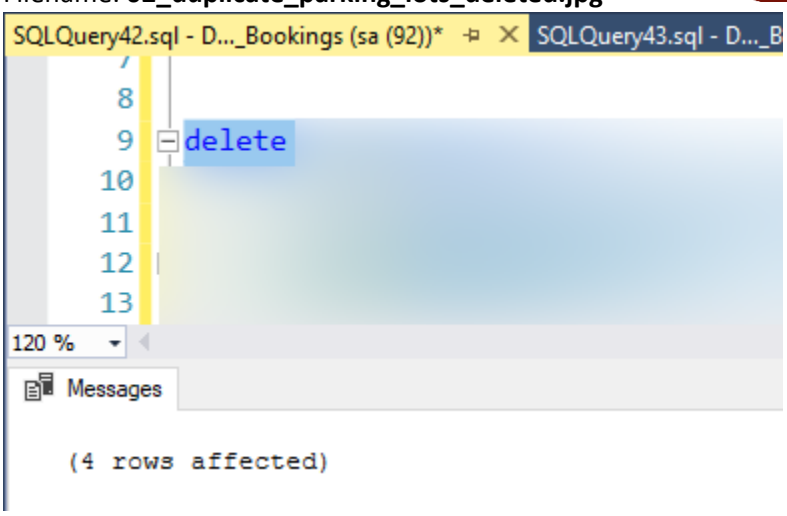
Don't over complicate this DELETE statement. Find the `parking_ids` that need to be deleted and write a single DELETE statement that deletes *these specific* `parking_ids`.
(This is the *one exception* to the specific/genral rule)

Submit your screenshot.
<u>Required in your screenshot</u>:

- Your DELETE query that deletes the duplicates.
- The number of rows affected.

Submit

Filename: **02_duplicate_parking_lots_deleted.jpg**



Step 3: Prevent future duplicate data.

With a properly designed database, we could have prevented these duplicates from entering our database in the first place. In order to prevent future duplicates, let's add a constraint to the `parking` table to prevent duplicates.

Add a constraint on the `parking` table to prevent duplicates on the `name` column. [/ 3 marks]

IMPORTANT: The name of the of the constraint must be in the following format:

Unique_Parking_Name_ + **Your First Initial** + _ + **Your Last Name**

As an example, for me, I would create the constraint as:
Unique_Parking_Name _**P_Guichon**

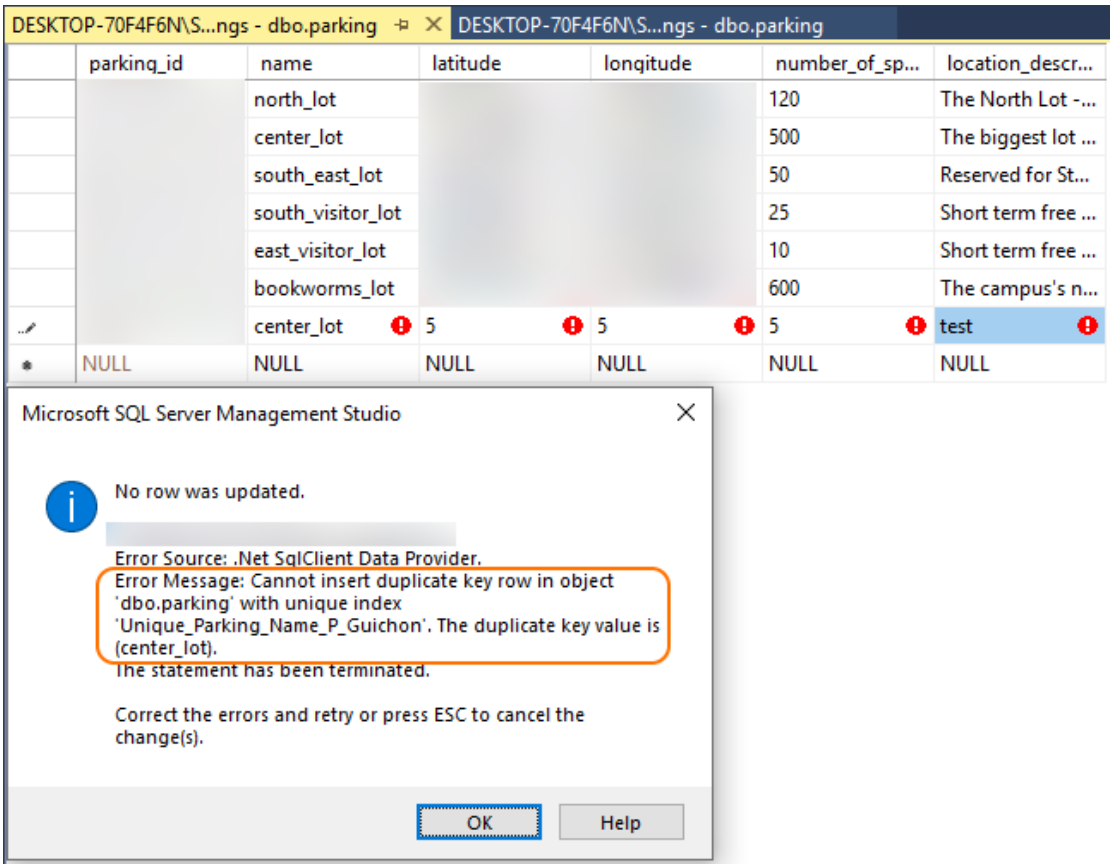
Once you have added the constraint to your `parking` table. Prove that the constraint works, by trying to add a row with the same name as one of the existing rows.

Submit your screenshot.
<u>Required in your screenshot</u>:

- The row you are trying to add, which should ***fail*** (because it is a duplicate).
- The error message showing the contraint name.

Filename: **03_preventing_future_duplicates.jpg**

Submit



SECTION 2: SQL Statements - Spot the Errors.

SECTION 2
Score:
/ 10

All of the SQL Statements below have errors.

For each query determine what the errors are and fix them.
[2 marks each]

The following queries should be run against the `Event_Bookings` database.

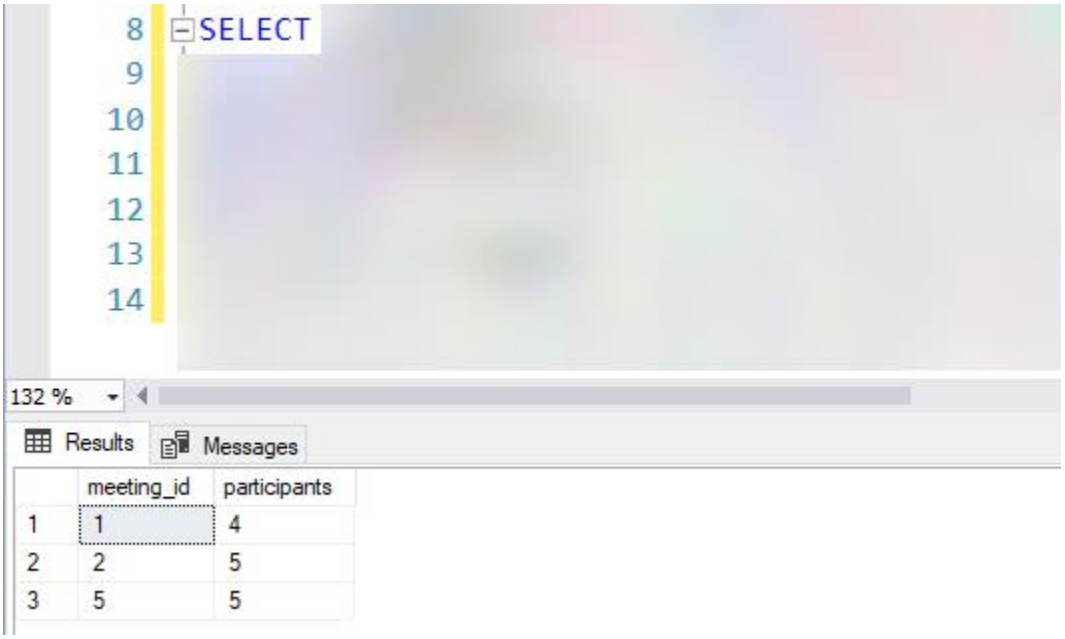
1. Fix the error(s) in the following SELECT query. [2 marks]

```
SELECT meeting_id, count(*) AS 'participants'
FROM participant
WHERE count(*) >= 4
GROUP BY meeting_id;
```

Submit your screenshot.
<u>Required in your screenshot</u>:

- Your fixed query.
- The result set.
  Filename: **04_fixed_query_#1.jpg**

Submit



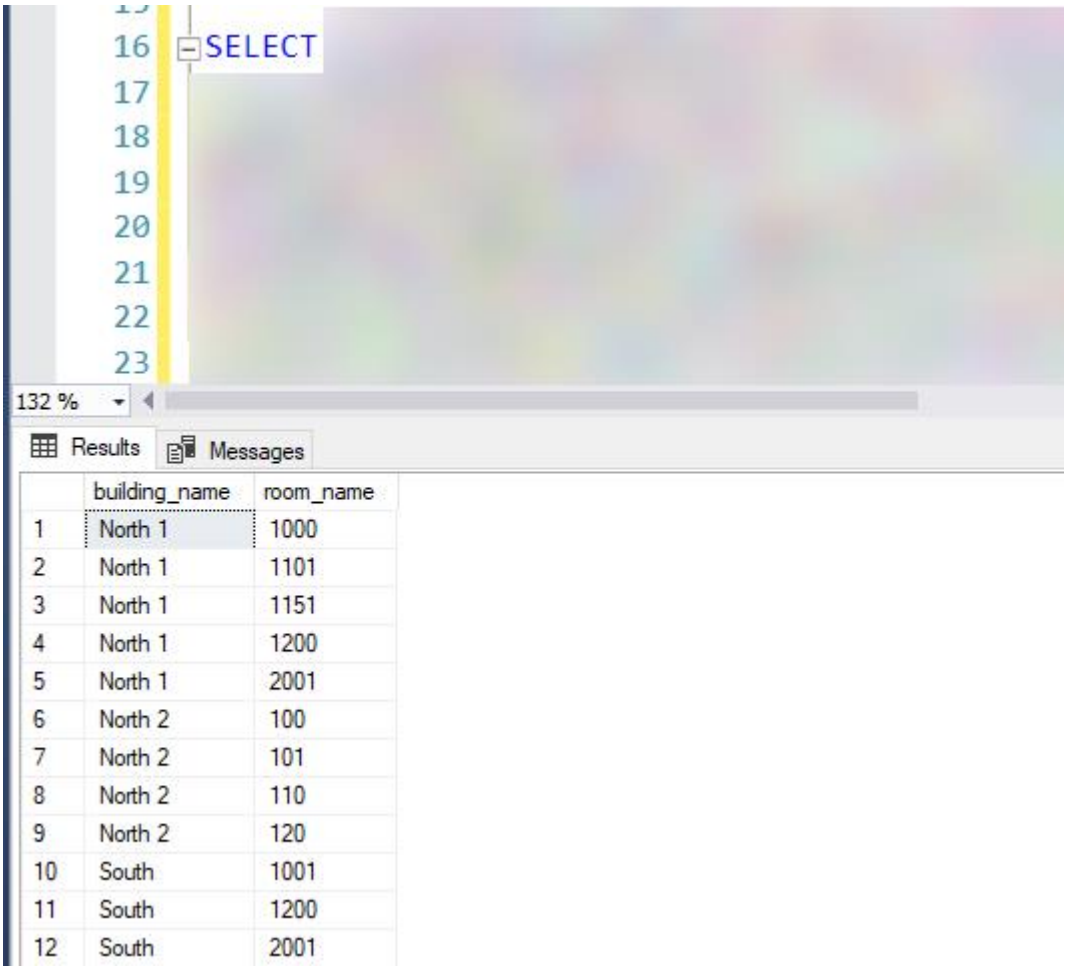
2. Fix the error(s) in the following SELECT query. [2 marks]

```
SELECT name AS 'building_name', name AS 'room_name'
FROM room
INNER JOIN building
  ON building.building_id = room.building_id;
```

Submit your screenshot.
<u>Required in your screenshot</u>:

- Your fixed query.
- The result set.
  Filename: **05_fixed_query_#2.jpg**

Submit

- 

	building_name	room_name
1	North 1	1000
2	North 1	1101
3	North 1	1151
4	North 1	1200
5	North 1	2001
6	North 2	100
7	North 2	101
8	North 2	110
9	North 2	120
10	South	1001
11	South	1200
12	South	2001

(Note: There are more rows than are shown in this screenshot)

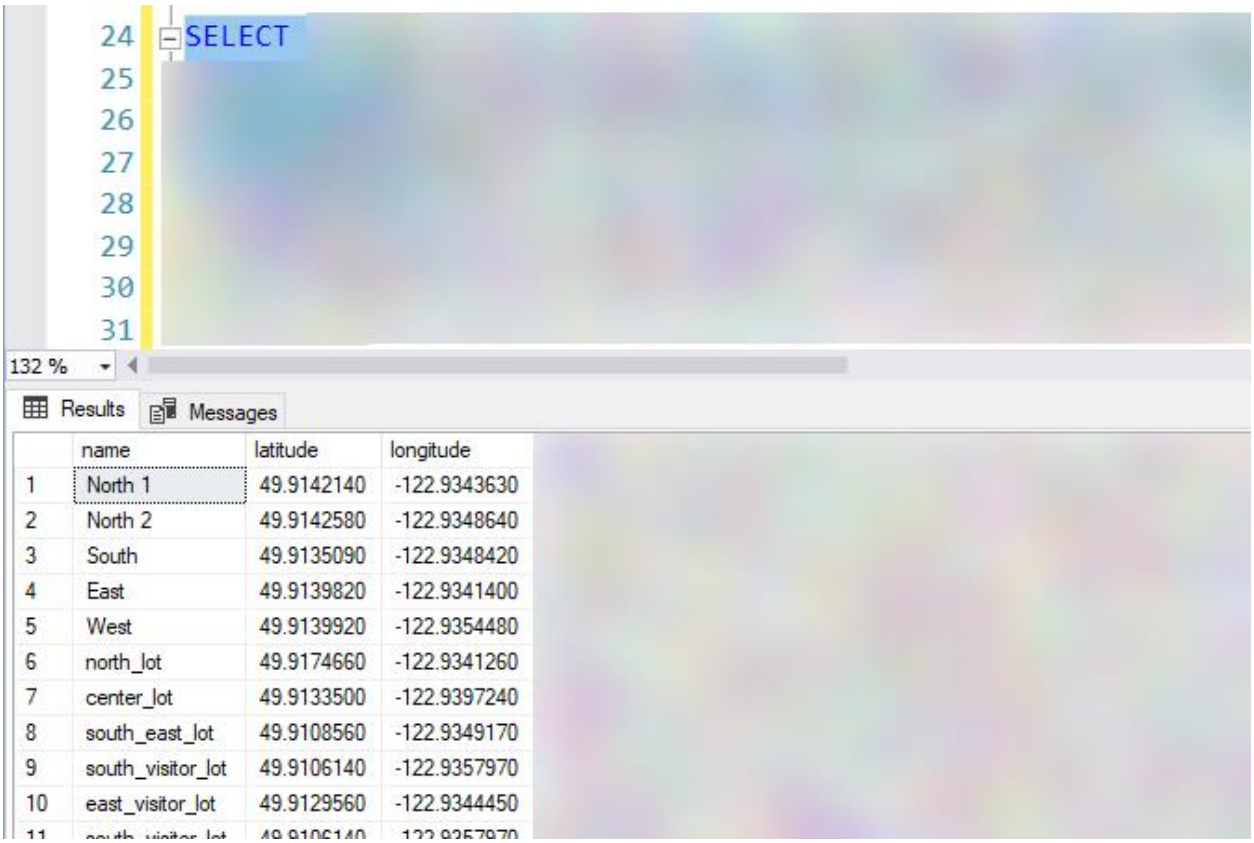
3. Fix the error(s) in the following SELECT query. [2 marks]

```
SELECT building.name, building.latitude, building.longitude
FROM building
UNION ALL
SELECT parking.name, parking.latitude, parking.longitude,
parking.location_description
FROM parking;
```

Submit your screenshot.
<u>Required in your screenshot</u>:

- Your fixed query.
- The result set.
  Filename: **06_fixed_query_#3.jpg**

Submit



	name	latitude	longitude
1	North 1	49.9142140	-122.9343630
2	North 2	49.9142580	-122.9348640
3	South	49.9135090	-122.9348420
4	East	49.9139820	-122.9341400
5	West	49.9139920	-122.9354480
6	north_lot	49.9174660	-122.9341260
7	center_lot	49.9133500	-122.9397240
8	south_east_lot	49.9108560	-122.9349170
9	south_visitor_lot	49.9106140	-122.9357970
10	east_visitor_lot	49.9129560	-122.9344450

(Note: There are more rows than are shown in this screenshot)

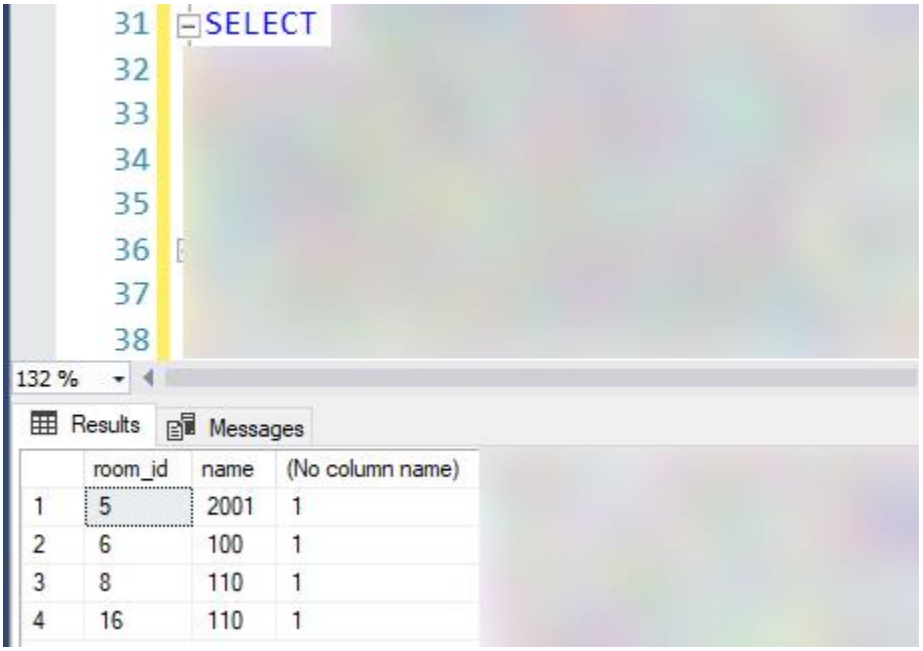
4. Fix the error(s) in the following SELECT query. [2 marks]

```
SELECT room_id, name, count(*)
FROM meeting
INNER JOIN room
ON room.room_id = meeting.room_id
GROUP BY room_id;
```

Submit your screenshot.
<u>Required in your screenshot</u>:

- Your fixed query.
- The result set.
  Filename: **07_fixed_query_#4.jpg**

Submit



	room_id	name	(No column name)
1	5	2001	1
2	6	100	1
3	8	110	1
4	16	110	1

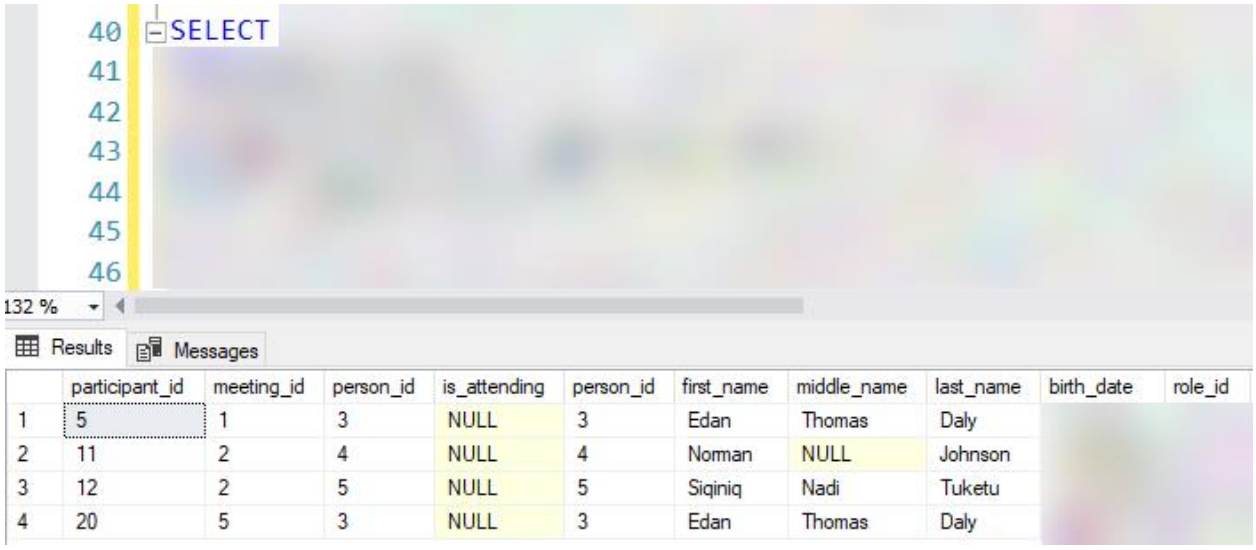
5. Fix the error(s) in the following SELECT query. [2 marks]

```
SELECT *
FROM participant
INNER JOIN person
ON person_id = person_id
WHERE is_attending = NULL;
```

Submit your screenshot.
<u>Required in your screenshot</u>:

- Your fixed query.
- The result set.
  Filename: **08_fixed_query_#5.jpg**

Submit



	participant_id	meeting_id	person_id	is_attending	person_id	first_name	middle_name	last_name	birth_date	role_id
1	5	1	3	NULL	3	Edan	Thomas	Daly		
2	11	2	4	NULL	4	Norman	NULL	Johnson		
3	12	2	5	NULL	5	Siqiniq	Nadi	Tuketu		
4	20	5	3	NULL	3	Edan	Thomas	Daly		

SECTION 3: Queries

SECTION 3
Score:
/ 21

Queries:

With our `Event_Bookings` database and tables, think about how to answer the following questions by writing queries (SELECT statements).

IMPORTANT: Make sure that your queries are *specific* and would still work if additional rows were added to the tables.
Example: If asked to find everyone who lives in 'BC' from the following table:

person_id	firstName	lastName	house	street	city	prov
1	Paul	Waldman	2333	Roger St	Nanaimo	BC
2	Lynn	William	3028	Blue Rocks Rd	Mahone Bay	NS
3	Bruce	William	3417	Haaglund Rd	Grand Forks	BC
4	Jacki	Ballweg	1573	Burdett Av	Victoria	BC
5	Kathy	Bromberg	538	Lock St	Guelph	ON

This is **not an acceptable answer** (even if it produces the correct output):

```
SELECT firstName, lastName
FROM person
WHERE person_id IN (1,3,4);
```

You **must** write the SELECT statement as *generic* queries - example:

```
SELECT firstName, lastName
FROM person
WHERE prov = 'BC';
```

For each question, write the correct SELECT query required to produce the result set and retrieve the information to answer the question.

[3 marks each]

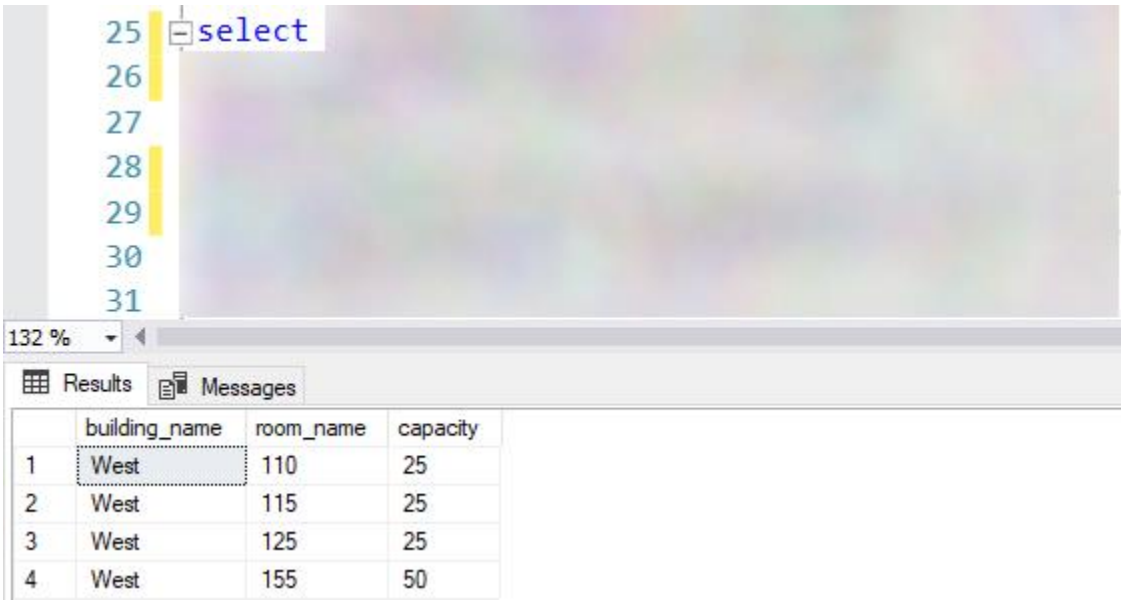
1. List all the rooms in the 'West' Building. [3 marks]

building_name	room_name	capacity
West	110	25
West	115	25
West	125	25
West	155	50

Submit your screenshot.
<u>Required in your screenshot</u>:

- Your SELECT query.
- The result set.
  Filename: **09_SELECT_query_#1.jpg**

Submit



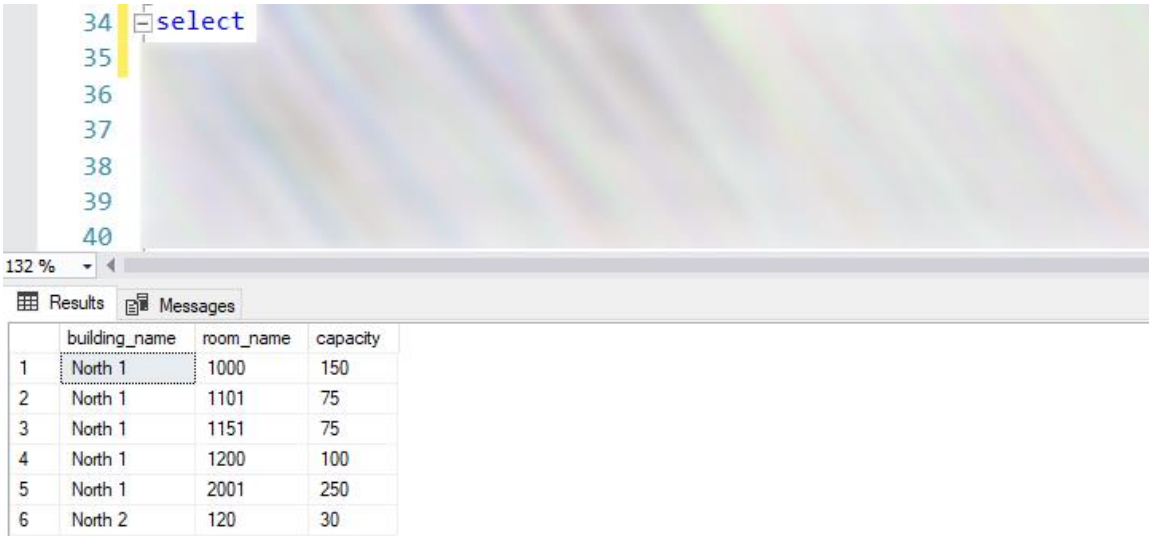
2. List all the rooms in a building whose name contains 'North' with a capacity of 30 or more. [3 marks]

building_name	room_name	capacity
North 1	1000	150
North 1	1101	75
North 1	1151	75
North 1	1200	100
North 1	2001	250
North 2	120	30

Submit your screenshot.
<u>Required in your screenshot</u>:

- Your SELECT query.
- The result set.
  Filename: **10_SELECT_query_#2.jpg**

Submit



	building_name	room_name	capacity
1	North 1	1000	150
2	North 1	1101	75
3	North 1	1151	75
4	North 1	1200	100
5	North 1	2001	250
6	North 2	120	30

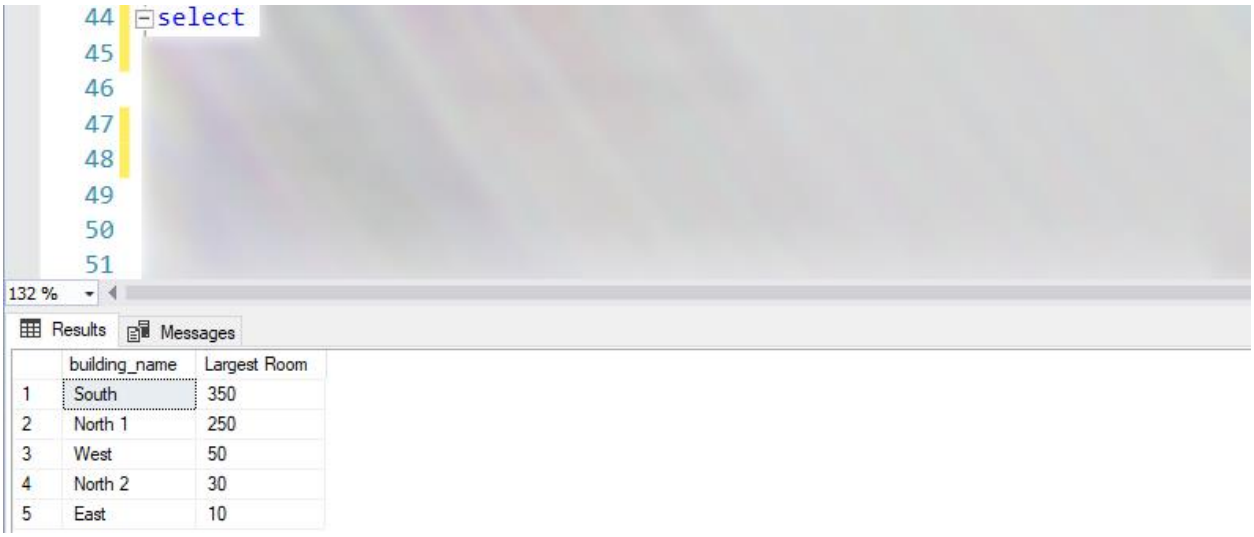
3. List the largest (highest capacity) room in each of the buildings, sorted by highest capacity. [3 marks]

building_name	Largest Room
South	350
North 1	250
West	50
North 2	30
East	10

Submit your screenshot.
<u>Required in your screenshot</u>:

- Your SELECT query.
- The result set.
  Filename: **11_SELECT_query_#3.jpg**

Submit



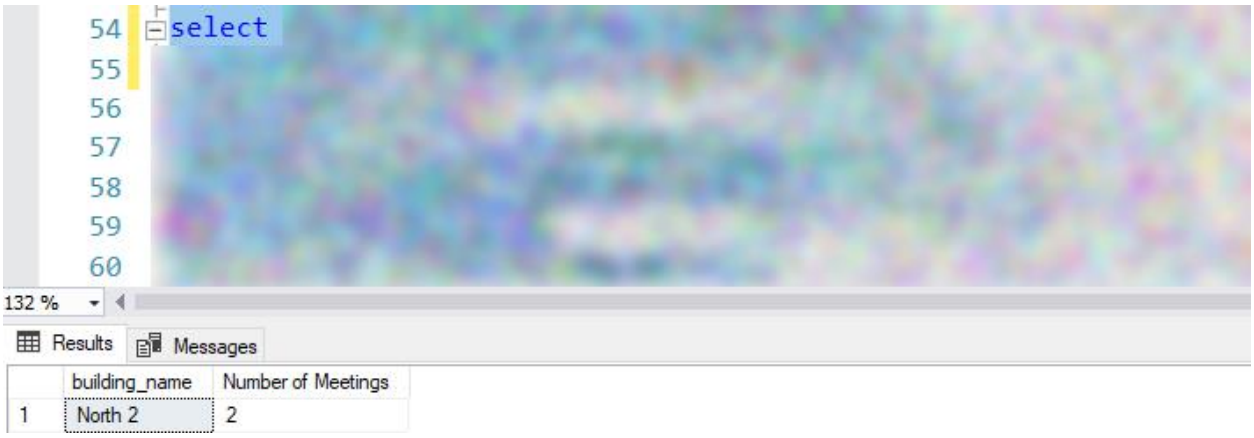
4. Which building has the most frequently used rooms (most meetings)?
   List only the most frequent. [3 marks]

building_name	Number of Meetings
North 1	2

Submit your screenshot.
<u>Required in your screenshot</u>:

- Your SELECT query.
- The result set.
  Filename: **12_SELECT_query_#4.jpg**

Submit



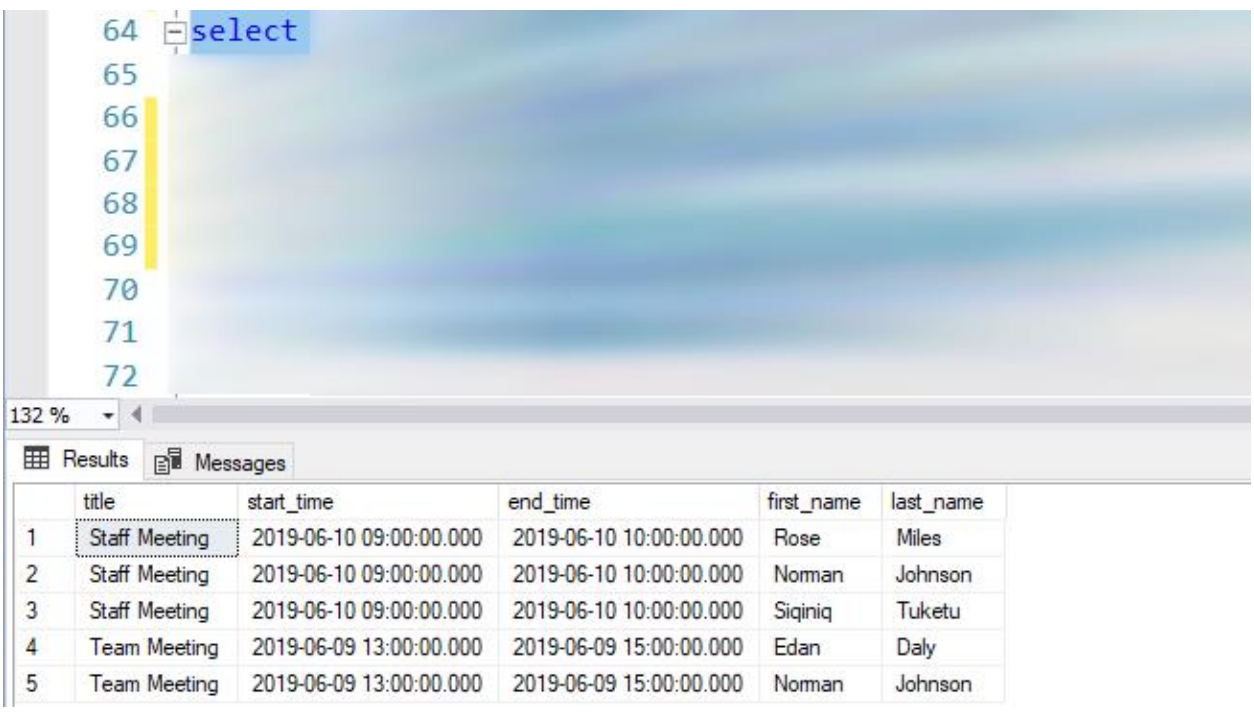
5. List all the people and meeting times who are attending a meeting without a room (online meeting). [3 marks]
   Note: There are 3 possible values for `is_attending`:
   - 1 - Is attending
   - 0 - Is NOT attending
   - NULL - Unknown; has not responded yet.

title	start_time	end_time	first_name	last_name
Staff Meeting	2019-06-10 9:00 AM	2019-06-10 10:00 AM	Rose	Miles
Staff Meeting	2019-06-10 9:00 AM	2019-06-10 10:00 AM	Norman	Johnson
Staff Meeting	2019-06-10 9:00 AM	2019-06-10 10:00 AM	Siqiniq	Tuketu
Team Meeting	2019-06-09 1:00 PM	2019-06-09 3:00 PM	Edan	Daly
Team Meeting	2019-06-09 1:00 PM	2019-06-09 3:00 PM	Norman	Johnson

Submit your screenshot.
<u>Required in your screenshot</u>:

- Your SELECT query.
- The result set.
  Filename: **13_SELECT_query_#5.jpg**

Submit



	title	start_time	end_time	first_name	last_name
1	Staff Meeting	2019-06-10 09:00:00.000	2019-06-10 10:00:00.000	Rose	Miles
2	Staff Meeting	2019-06-10 09:00:00.000	2019-06-10 10:00:00.000	Norman	Johnson
3	Staff Meeting	2019-06-10 09:00:00.000	2019-06-10 10:00:00.000	Siqiniq	Tuketu
4	Team Meeting	2019-06-09 13:00:00.000	2019-06-09 15:00:00.000	Edan	Daly
5	Team Meeting	2019-06-09 13:00:00.000	2019-06-09 15:00:00.000	Norman	Johnson

6. Create a schedule for all people who have confirmed (is attending).
   Include online conferences. [3 marks]
   Note: There are 3 possible values for `is_attending`:
   - 1 - Is attending
   - 0 - Is NOT attending
   - NULL - Unknown; has not responded yet.

title	start_time	first_name	last_name	building_name	room_name
Tech Conference 2019	2019-06-06 9:00 AM	Rose	Miles	North 1	2001
Tech Conference 2019	2019-06-06 9:00 AM	Solveiga	Armani	North 1	2001
Tech Conference 2019	2019-06-06 9:00 AM	Siqiniq	Tuketu	North 1	2001
Staff Lunch	2019-06-07 11:00 AM	Rose	Miles	North 2	100
Staff Lunch	2019-06-07 11:00 AM	Solveiga	Armani	North 2	100
Workshop - Level 1	2019-06-07 9:00 AM	Solveiga	Armani	North 2	110
Workshop - Level 1	2019-06-07 9:00 AM	Edan	Daly	North 2	110
Workshop - Level 1	2019-06-07 9:00 AM	Siqiniq	Tuketu	North 2	110
Workshop - Level 2	2019-06-07 1:00 PM	Norman	Johnson	West	110
Workshop - Level 2	2019-06-07 1:00 PM	Siqiniq	Tuketu	West	110
Staff Meeting	2019-06-10 9:00 AM	Rose	Miles	NULL	NULL
Staff Meeting	2019-06-10 9:00 AM	Norman	Johnson	NULL	NULL
Staff Meeting	2019-06-10 9:00 AM	Siqiniq	Tuketu	NULL	NULL
Team Meeting	2019-06-09 1:00 PM	Edan	Daly	NULL	NULL
Team Meeting	2019-06-09 1:00 PM	Norman	Johnson	NULL	NULL

Submit your screenshot.

<u>Required in your screenshot</u>:

- Your SELECT query.
- The result set.
  Filename: **14_SELECT_query_#6.jpg**

Submit

```
75 select
76
77
78
79
80
81
82
83
```

132 %

Results | Messages

	title	start_time	end_time	first_name	last_name	building_name	room_name
1	Tech Conference 2019	2019-06-06 09:00:00.000	2019-06-06 17:00:00.000	Rose	Miles	North 1	2001
2	Tech Conference 2019	2019-06-06 09:00:00.000	2019-06-06 17:00:00.000	Solveiga	Armani	North 1	2001
3	Tech Conference 2019	2019-06-06 09:00:00.000	2019-06-06 17:00:00.000	Siqiniq	Tuketu	North 1	2001
4	Staff Lunch	2019-06-07 11:00:00.000	2019-06-07 14:00:00.000	Rose	Miles	North 2	100
5	Staff Lunch	2019-06-07 11:00:00.000	2019-06-07 14:00:00.000	Solveiga	Armani	North 2	100
6	Workshop - Level 1	2019-06-07 09:00:00.000	2019-06-07 11:00:00.000	Solveiga	Armani	North 2	110
7	Workshop - Level 1	2019-06-07 09:00:00.000	2019-06-07 11:00:00.000	Edan	Daly	North 2	110
8	Workshop - Level 1	2019-06-07 09:00:00.000	2019-06-07 11:00:00.000	Siqiniq	Tuketu	North 2	110
9	Workshop - Level 2	2019-06-07 13:00:00.000	2019-06-07 15:00:00.000	Norman	Johnson	West	110
10	Workshop - Level 2	2019-06-07 13:00:00.000	2019-06-07 15:00:00.000	Siqiniq	Tuketu	West	110
11	Staff Meeting	2019-06-10 09:00:00.000	2019-06-10 10:00:00.000	Rose	Miles	NULL	NULL
12	Staff Meeting	2019-06-10 09:00:00.000	2019-06-10 10:00:00.000	Norman	Johnson	NULL	NULL
13	Staff Meeting	2019-06-10 09:00:00.000	2019-06-10 10:00:00.000	Siqiniq	Tuketu	NULL	NULL
14	Team Meeting	2019-06-09 13:00:00.000	2019-06-09 15:00:00.000	Edan	Daly	NULL	NULL
15	Team Meeting	2019-06-09 13:00:00.000	2019-06-09 15:00:00.000	Norman	Johnson	NULL	NULL

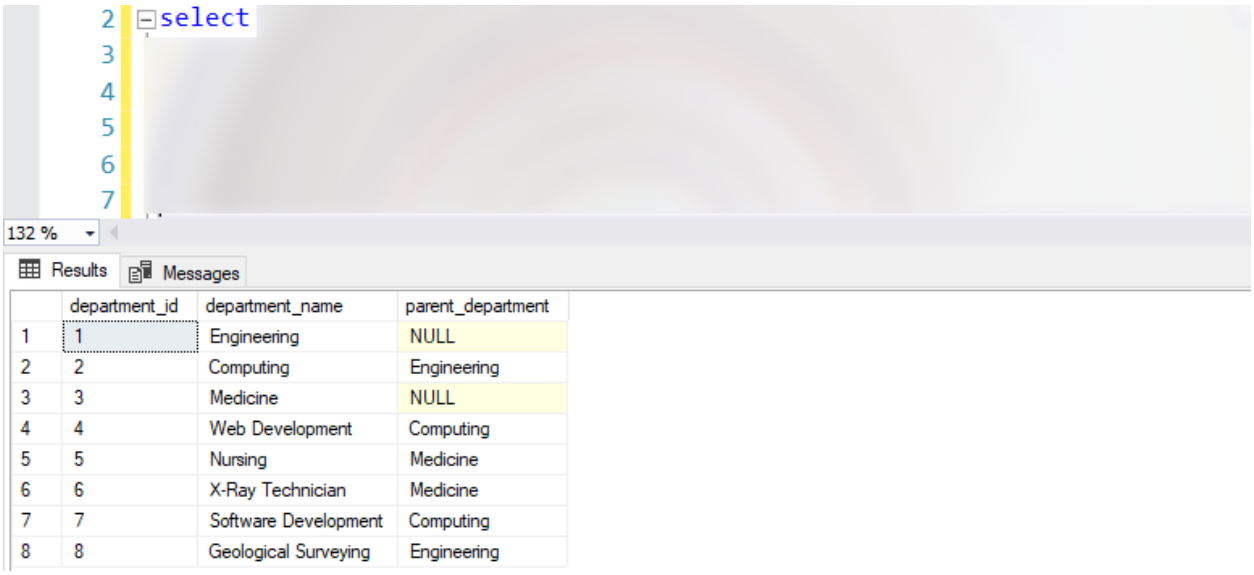
7. Create a list of all the departments and their parent departments. [3 marks]

department_id	department_name	parent_department
1	Engineering	NULL
2	Computing	Engineering
3	Medicine	NULL
4	Web Development	Computing
5	Nursing	Medicine
6	X-Ray Technician	Medicine
7	Software Development	Computing
8	Geological Surveying	Engineering

Submit your screenshot.
<u>Required in your screenshot</u>:

- Your SELECT query.
- The result set.
  Filename: **15_SELECT_query_#7.jpg**

Submit



	department_id	department_name	parent_department
1	1	Engineering	NULL
2	2	Computing	Engineering
3	3	Medicine	NULL
4	4	Web Development	Computing
5	5	Nursing	Medicine
6	6	X-Ray Technician	Medicine
7	7	Software Development	Computing
8	8	Geological Surveying	Engineering

SECTION 4: Good Database Design

SECTION 4
Score:
/ 10

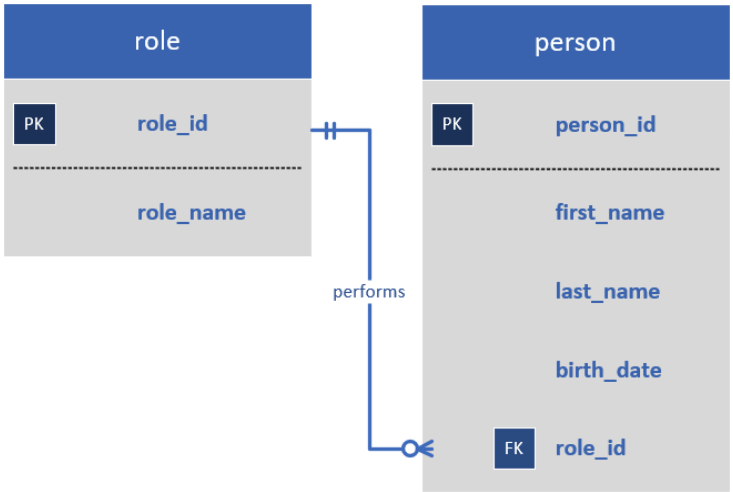
There is something that I don't like about the `Events_Booking` database.

Let's look at the `person` table. There is a `role` column where the values are duplicated. Notice how "Assistant" and "Sales" show up multiple times for multiple people.

This is a bad design; so let's fix this.

person_id	first_name	last_name	birth_date	role
1	Rose	Miles	1954-10-21	Manager
2	Solveiga	Armani	1989-03-27	Sales
3	Edan	Daly	1984-06-15	Assistant
4	Norman	Johnson	1984-11-08	Inventory
5	Siqiniq	Tuketu	1978-06-22	Sales
6	Robert	Sanders	1985-05-13	Assistant

Create another table called `role` according to the ERD below:



Use the following data types:

Column	Data type	Extra
role_id	int	Identity, Primary Key
role_name	nvarchar(50)	Allow Nulls - No

IMPORTANT: Make sure that `role_id` is the Identity Column and a Primary Key for the table.

Replace the old `role` column in the `person` table with correct foreign key values from the new `role` table. Hint: It might be easier to drop the old `role` column and create a new `role_id` column.

Create and enforce all appropriate Foreign Keys for the role table.

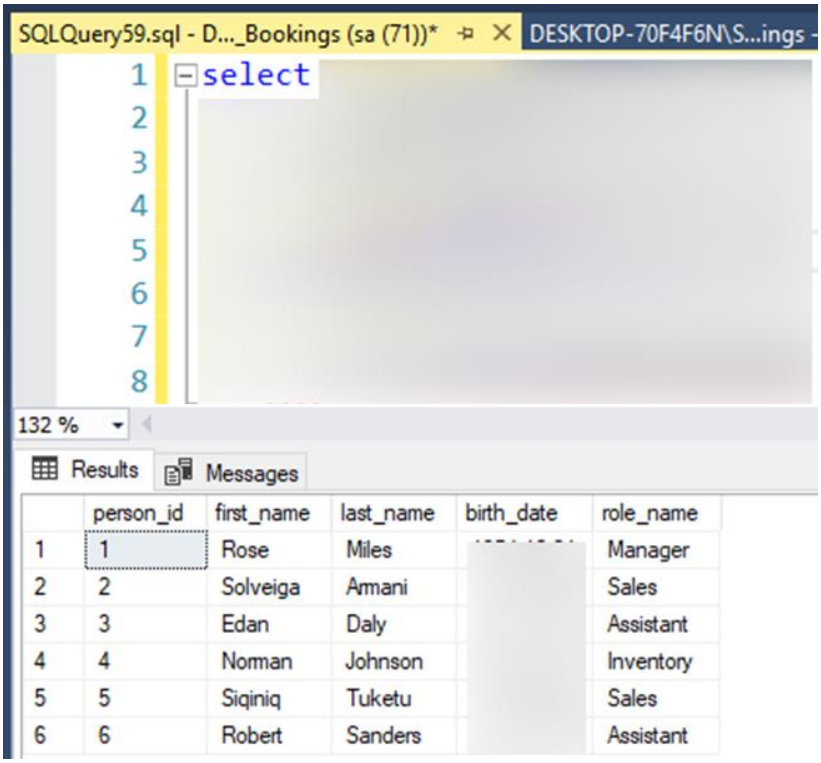
Write a SELECT statement to query the data from the `person` and new `role` table to get back the original result set.

person_id	first_name	last_name	birth_date	role_name
1	Rose	Miles	1954-10-21	Manager
2	Solveiga	Armani	1989-03-27	Sales
3	Edan	Daly	1984-06-15	Assistant
4	Norman	Johnson	1984-11-08	Inventory
5	Siqiniq	Tuketu	1978-06-22	Sales
6	Robert	Sanders	1985-05-13	Assistant

Submit your screenshot.
<u>Required in your screenshot</u>:

- Your SELECT statement.
- The result set.
  Filename: **16_roles_reference_table.jpg**

Submit



SECTION 5: Optional Middle Name

SECTION 5
Score:
/ 8

You have been asked by your manager to modify the `person` table so that we can record people's middle name (if they have one).

Add another column in the `person` table for middle name. [2 marks]

Once you've added the new `middle_name` column, enter the following middle names for the people.

person_id	first_name	middle_name	last_name	birth_date	role
1	Rose		Miles	1954-10-21	Manager
2	Solveiga		Armani	1989-03-27	Sales
3	Edan	Thomas	Daly	1984-06-15	Assistant
4	Norman		Johnson	1984-11-08	Inventory
5	Siqiniq	Nita	Tuketu	1978-06-22	Sales
6	Robert	James	Sanders	1985-05-13	Assistant

Use the following data type:

Column	Data type	Extra
middle_name	nvarchar(100)	Allow Nulls - Yes

Once you have added the middle_name column and added the people's correct middle names create the following SELECT queries.

For each question, write the correct SELECT query required to produce the result set and retrieve the information to answer the question.

[3 marks each]

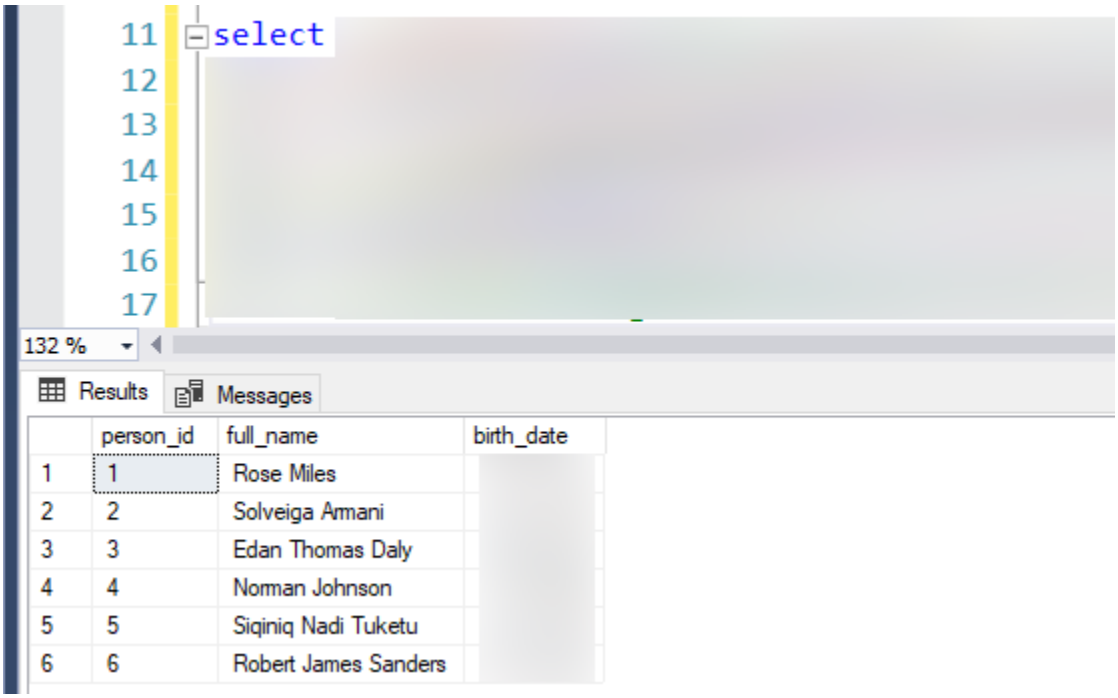
1. Create a list for all people, their full name (first_name, middle_name and last_name) and birth date. [3 marks]

person_id	full_name	birth_date
1	Rose Miles	1954-10-21
2	Solveiga Armani	1989-03-27
3	Edan Thomas Daly	1984-06-15
4	Norman Johnson	1984-11-08
5	Siqiniq Nita Tuketu	1978-06-22
6	Robert James Sanders	1985-05-13

Submit your screenshot.
<u>Required in your screenshot</u>:

- Your SELECT statement.
- The result set.
  Filename: **17_optional_middle_name_1.jpg**

Submit



2. Create a list of people without middle names, who were born after Jan 1st, 1984 (1984-01-01) This time, list the people by last name then first name (with a comma in between). [3 marks]

person_id	full_name	birth_date
2	Armani, Solveiga	1989-03-27
4	Johnson, Norman	1984-11-08

Submit your screenshot.
<u>Required in your screenshot</u>:

- Your SELECT statement.
- The result set.
  Filename: **18_optional_middle_name_2.jpg**

Submit

SECTION 6: Stored Procedure

SECTION 6
Score:
/ 5

The `Event_Bookings` database contains the following stored procedure:

```sql
CREATE PROCEDURE [dbo].[AddPersonToMeeting]
        @first_name nvarchar(50),
        @last_name nvarchar(50),
        @meeting_title nvarchar(50),
        @is_attending INT
AS
BEGIN
        DECLARE @person_id INT;
        DECLARE @meeting_id INT;
        DECLARE @participants INT;
        DECLARE @room_capacity INT;
        DECLARE @room_id INT;

        SELECT @meeting_id = meeting_id,
               @room_id = room_id
        FROM meeting
        WHERE title = @meeting_title;

        IF @@ROWCOUNT = 0
               THROW 50015, 'Meeting does not exist', 2

        SELECT @participants = count(*)
        FROM participant
        WHERE meeting_id = @meeting_id;

        SELECT @room_capacity = capacity
        FROM room
        WHERE room_id = @room_id

        IF @participants >= @room_capacity
               THROW 50007, 'Not enough room in this meeting for more people.', 2

        INSERT INTO participant(meeting_id, person_id, is_attending)
        VALUES
        (@meeting_id, @person_id, @is_attending);
END
GO
```

This stored procedure adds people to an existing meeting.

You have been asked to modify this stored procedure. Add the required code to the stored procedure to validate that the person exists ***before*** adding them to the meeting.

Replace the existing stored procedure with your new and improved version.
Your changes must not change or break any of the existing functionality. [/ 5 marks]

If the person does not exist:

Throw error number: **50001** and a
Message of: **"Person does not exist"**

Test your new stored procedure to make sure it works (generates the correct error when the person does not exist.)

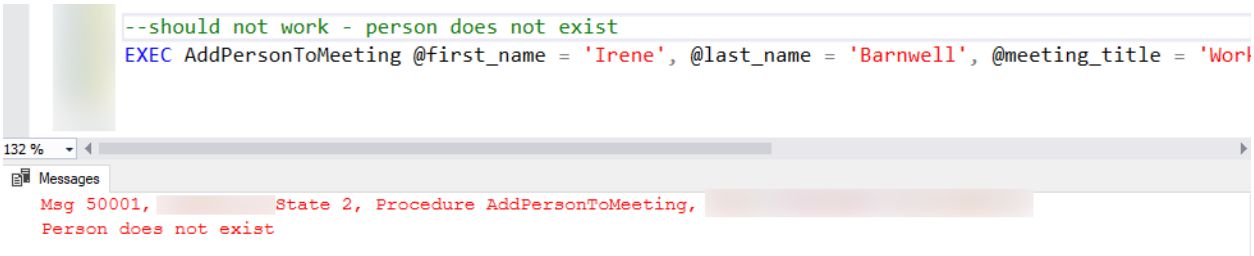
```
--should not work - person does not exist
EXEC AddPersonToMeeting @first_name = 'Irene', @last_name = 'Barnwell', @meeting_title =
'Workshop - Level 1', @is_attending = 1;
```

Submit your screenshot.
<u>Required in your screenshot</u>:

- The EXEC command, which should ***fail*** (because "Irene" does not exist).
- The error message showing "Person does not exist".

Filename: **19_ new_stored_procedure.jpg**

Submit

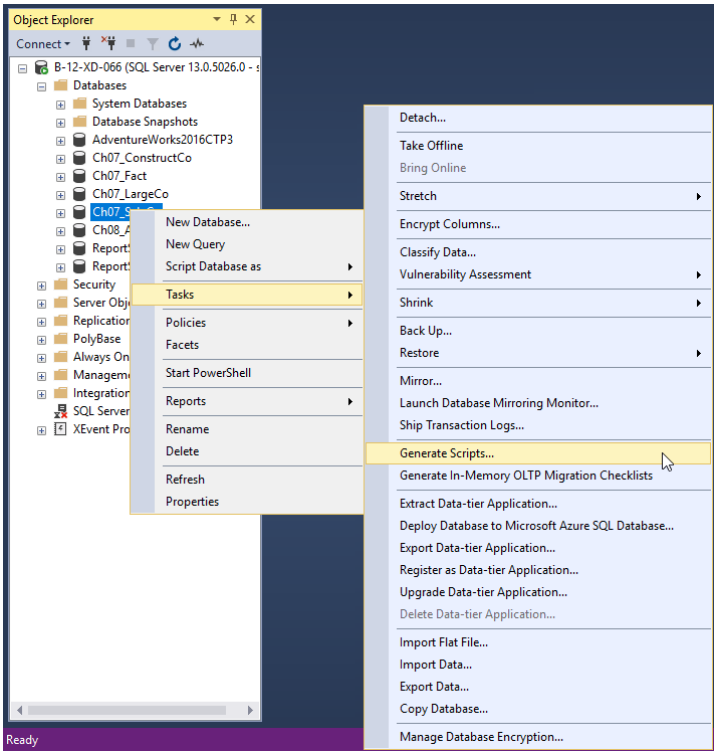


SECTION 7: Logical Backup

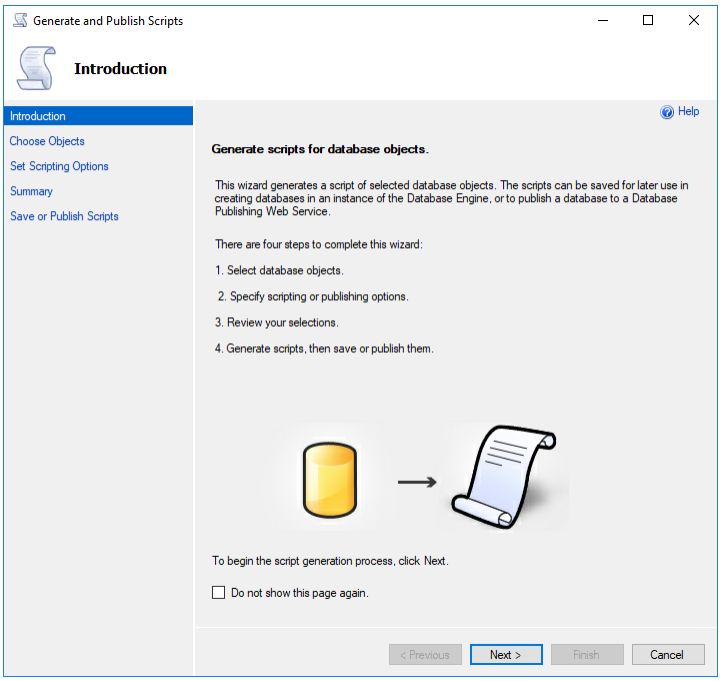
SECTION 7
Score:
/ 3

Perform a Logical Backup of your database.

Right click on the `Event_Bookings` database, then click "Tasks", and click "Generate Scripts..."

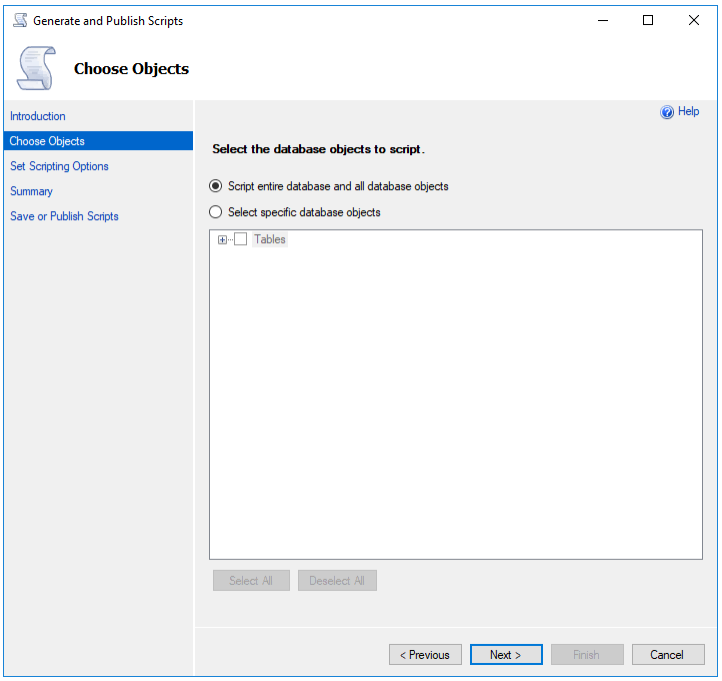


Click Next.

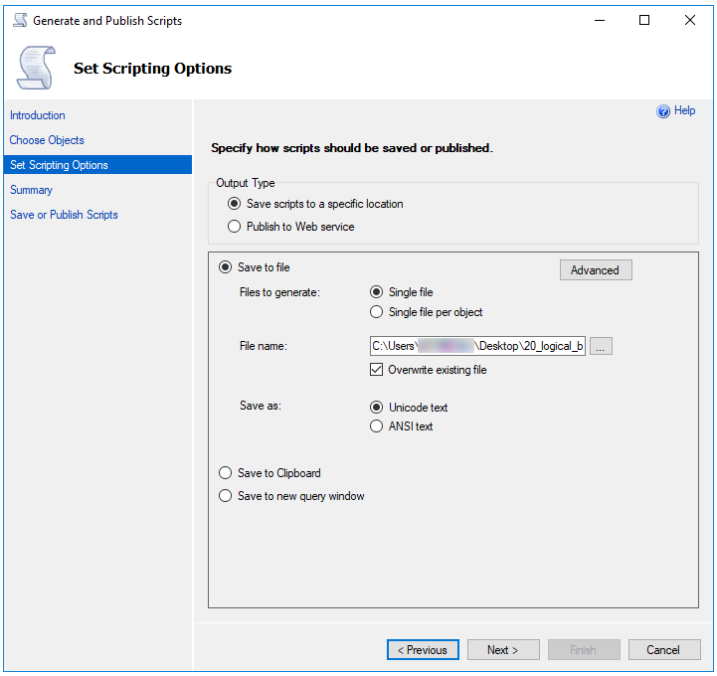


Make sure "Script entire database and all database objects" is selected.

Click Next

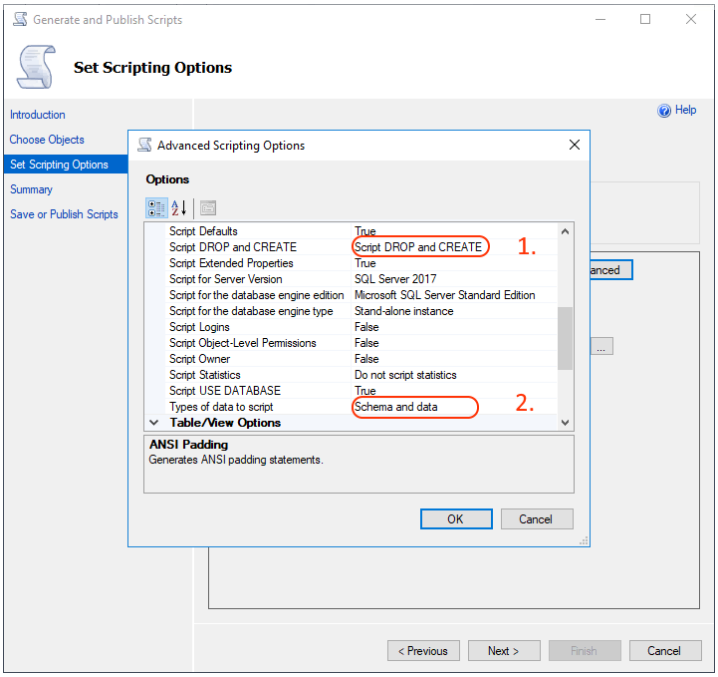


Make sure that you select "Save to file".
Click Advanced



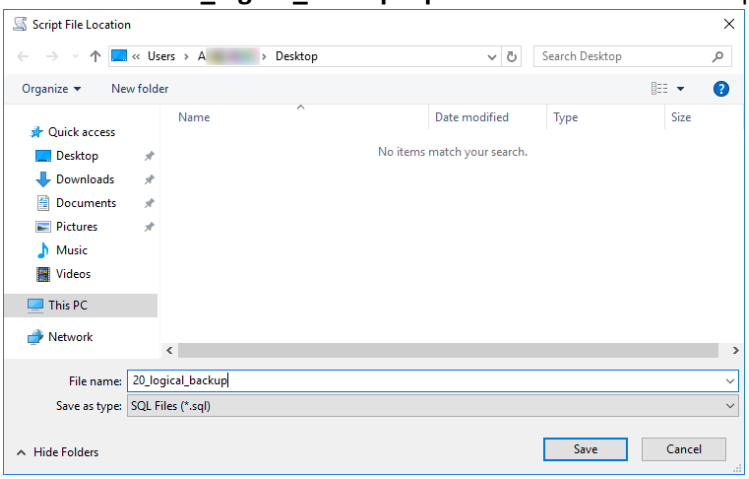
Make sure that you:

1. Select "Script DROP and CREATE".
2. Select "Schema and data".



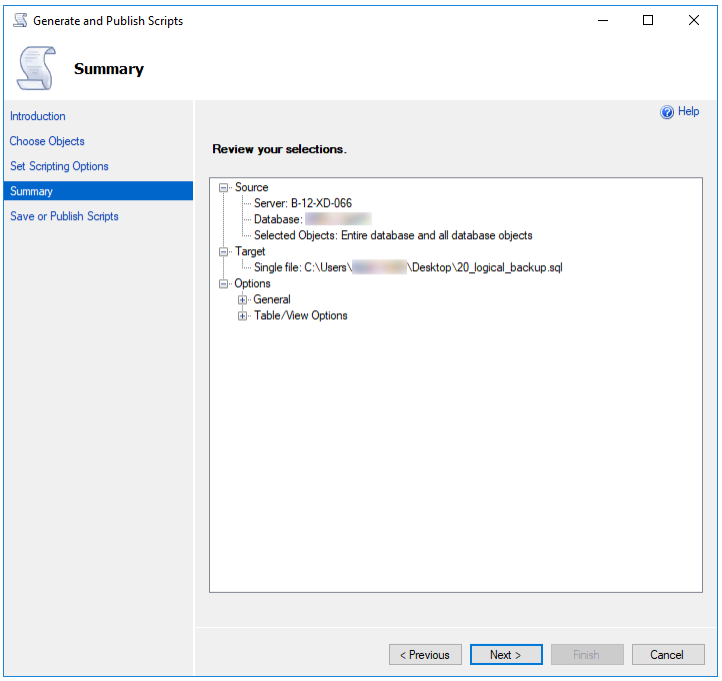
Click OK.

Name the file **20_logical_backup.sql** and save it to the Desktop:

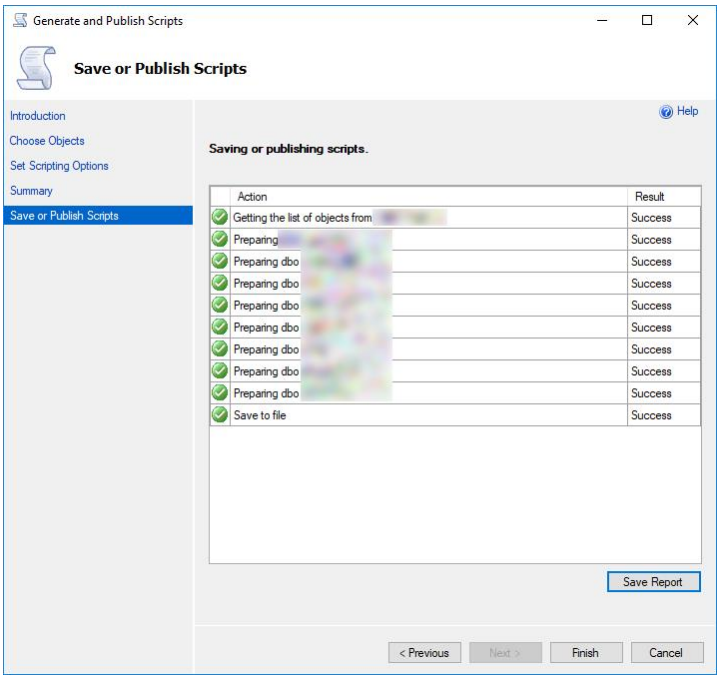


Click Save

Click Next:

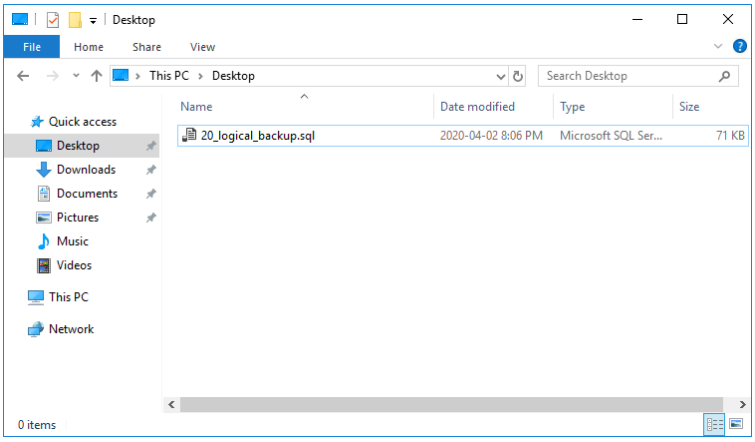


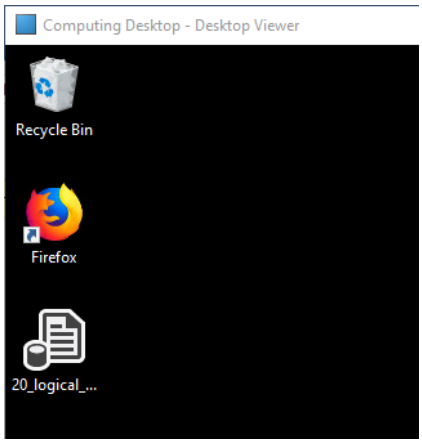
Click Next:



Click Finish

Find the file on your Desktop using Windows File Explorer or on the Desktop:





Submit your backup file.

Filename: **20_logical_backup.sql**

Submit

Congratulations! You are done!