HOMEWORK 1==============================================================================================================

HW1 - 1.sql: ----------------------------------------------------------------------------------------------------------------------

SELECT \*

FROM Hotel

WHERE city = 'London';

HW1 - 2.sql: ----------------------------------------------------------------------------------------------------------------------

SELECT \*

FROM Room

WHERE (type IN ('double', 'family') AND price < 40.00)

ORDER BY price;

--ORDER BY price DESC;

-- SELECT \*

-- FROM Hotel

-- WHERE city = 'London';

-- SELECT s.FirstName, s.LastName

-- FROM student s

-- JOIN studentcourses sc ON s.Id = sc.StudentId

-- WHERE sc.courseId = 'CS140';

HW1 - 3.sql: ----------------------------------------------------------------------------------------------------------------------

SELECT DISTINCT COUNT(\*)

FROM Hotel;

HW1 - 4.sql: ----------------------------------------------------------------------------------------------------------------------

SELECT SUM(price)

FROM Room

WHERE type = 'double';

HW1 - 5.sql: ----------------------------------------------------------------------------------------------------------------------

SELECT price, type

FROM Room

WHERE hotelNo = 1;

HW1 - 6.sql: ----------------------------------------------------------------------------------------------------------------------

--NOTE: I got 3 points off b/c I didn't consider roomNo = roomNo basically. Look at key.

--SELECT \*

SELECT SUM(price)

FROM Hotel h

JOIN Room r ON h.hotelNo = r.hotelNo AND h.hotelName = 'Grosvenor Hotel'

WHERE (h.hotelNo, r.roomNo) IN (SELECT hotelNo, roomNo

FROM Booking

WHERE dateFrom <= CURDATE() AND dateTo > CURDATE());

--NOTE: This works too, but I was originally aiming for the above

-- ...BUT, I screwed up the "(h.hotelNo, r.roomNo) IN" part

-- -- SELECT \*

-- SELECT SUM(price)

-- FROM Hotel h

-- JOIN Room r ON h.hotelName = 'Grosvenor Hotel' AND h.hotelNo = r.hotelNo

-- JOIN Booking b

-- WHERE b.dateFrom <= CURDATE() AND b.dateTo > CURDATE() AND r.roomNo = b.roomNo;

HW1 - 7.sql: ----------------------------------------------------------------------------------------------------------------------

SELECT hotelNo, COUNT(\*) AS count

FROM Room

GROUP BY hotelNo;

HW1 - 8.sql: ----------------------------------------------------------------------------------------------------------------------

SELECT h.hotelName, COUNT(\*) AS count

FROM Hotel h

JOIN Room r ON h.hotelNo = r.hotelNo AND h.city = 'London'

GROUP BY h.hotelNo;

HW1 - 9.sql: ----------------------------------------------------------------------------------------------------------------------

UPDATE Room

SET price = (price\*0.05 + price);

SELECT \*

FROM Room;

HOMEWORK 2==============================================================================================================

HW2 - 1.sql: ----------------------------------------------------------------------------------------------------------------------

DROP VIEW IF EXISTS HotelRoom;

DROP VIEW IF EXISTS BookingGuest;

CREATE VIEW HotelRoom

AS SELECT r.\*

FROM Hotel h, Room r

WHERE h.hotelNo = r.hotelNo AND h.hotelName = 'Grosvenor Hotel';

CREATE VIEW BookingGuest

AS SELECT b.roomNo, g.guestName

FROM Booking b, Guest g

WHERE b.guestNo = g.guestNo

AND ( b.dateFrom <= CURDATE() AND b.dateTo > CURDATE() )

AND b.hotelNo =

(SELECT h.hotelNo

FROM Hotel h

WHERE h.hotelName = 'Grosvenor Hotel'

); --NOTE: "b.hotelNo = (...)" works b/c only 1 row is returned!! There is only one hotel with that name!

-- Otherwise, I'd have to use "b.hotelNo IN (...)" if more than one row returned. Else, I get error

SELECT hr.\*, bg.guestName

FROM HotelRoom hr

LEFT JOIN BookingGuest bg ON hr.roomNo = bg.roomNo;

--LEFT JOIN BookingGuest bg; --NOTE: This was giving me errors without the ON clause above.

-- Error message wasn't very insightful (syntax is bad, mmkay?)

-- SELECT \*

-- FROM HotelRoom;

-- SELECT \*

-- FROM BookingGuest;

HW2 - 2.sql: ----------------------------------------------------------------------------------------------------------------------

--NOTE: Since "Booking" contained foreign keys to other parent tables, I had to DROP it first!!!

drop table if exists Booking;

drop table if exists Hotel;

drop table if exists Room;

drop table if exists Guest;

create table Hotel

(

hotelNo int(4),

hotelName varchar(30),

city varchar(20),

PRIMARY KEY(hotelNo)

);

create table Room

(

roomNo int(3)

CHECK (roomNo BETWEEN 1 AND 100),

hotelNo int(4),

type varchar(9),

CHECK (type IN ('Single', 'Double', 'Family')),

price dec(5,2)

CHECK (price BETWEEN 10.00 AND 100.00),

PRIMARY KEY (roomNo)

);

create table Guest

(

guestNo int(7),

guestName varchar(30),

guestAddress varchar(50),

PRIMARY KEY (guestNo)

);

create table Booking

(

hotelNo int(4),

guestNo int(7),

dateFrom date,

/\*CHECK (dateFrom > CURDATE()),\*/

dateTo date,

/\*CHECK (dateTo > CURDATE()),\*/

roomNo int(3)

CHECK (roomNo BETWEEN 1 AND 100),

/\*

NOTE: In order to get foreign keys to work, I had to

1) Put all parent tables ABOVE Booking.

2) Make referenced attributes primary keys in parent tables

\*/

FOREIGN KEY(hotelNo) REFERENCES Hotel(hotelNo)

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY(guestNo) REFERENCES Guest(guestNo)

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY(roomNo) REFERENCES Room(roomNo)

ON DELETE CASCADE

ON UPDATE CASCADE

/\*

CONSTRAINT e

CHECK

(

NOT EXISTS

(

SELECT b1.\*, COUNT(\*) AS numOfConflictsWithOtherBookings

FROM Booking b1, Booking b2

WHERE b1.hotelNo = b2.hotelNo

AND b1.roomNo = b2.roomNo

AND

(

(b1.dateFrom > b2.dateFrom AND b1.dateFrom < b2.dateTo)

OR

(b2.dateFrom > b1.dateFrom AND b2.dateFrom < b1.dateTo)

)

GROUP BY b1.dateFrom, b1.dateTo

ORDER BY b1.dateFrom, b1.dateTo;

)

)

CONSTRAINT f

CHECK

(

NOT EXISTS

(

SELECT b1.\*, COUNT(\*) AS numOfOverlappingBookingsByGuests

FROM Booking b1, Booking b2

WHERE b1.guestNo = b2.guestNo

AND

(

(b1.dateFrom > b2.dateFrom AND b1.dateFrom < b2.dateTo)

OR

(b2.dateFrom > b1.dateFrom AND b2.dateFrom < b1.dateTo)

)

GROUP BY b1.guestNo, b1.dateFrom, b1.dateTo

ORDER BY b1.dateFrom, b1.dateTo;

)

)

\*/

);

--TEST CODEs =================================================================================

--NOTE: Inserts below WORK with constraints AND foreign key dependencies

-- insert into Hotel values(1, 'Grosvenor Hotel', 'London');

-- insert into Hotel values(2, 'Holiday Inn', 'Knoxville');

-- insert into Hotel values(3, 'Hilton', 'Knoxville');

-- insert into Hotel values(4, 'Marriott', 'Memphis');

-- insert into Room values(100, 2, 'Double', 99.99);

-- insert into Room values(99, 2, 'Double', 99.99);

-- insert into Guest values(33, 'Cinderella', 'Orlando, FL');

-- insert into Guest values(666, 'Satan', 'My Flesh Palace, HELL');

-- insert into Booking values(1, 33, '2021-02-17', '2021-02-22', 100);

-- insert into Booking values(1, 33, '2021-02-15', '2021-02-18', 100);

-- insert into Booking values(1, 33, '2021-02-02', '2021-02-28', 100);

-- insert into Booking values(1, 33, '2021-02-01', '2021-02-03', 100);

--NOTE: With foreign keys added to Bookings, these fail b/c I need to establish parent tables first!

-- Correction above

-- insert into Booking values(1, 33, '2021-02-17', '2021-02-22', 100);

-- insert into Booking values(1, 33, '2021-02-15', '2021-02-18', 100);

-- insert into Booking values(1, 33, '2021-02-02', '2021-02-28', 100);

-- insert into Booking values(1, 33, '2021-02-01', '2021-02-03', 100);

-- --(e)

-- SELECT b1.\*, COUNT(\*) AS numOfConflictsWithOtherBookings

-- FROM Booking b1, Booking b2

-- WHERE b1.hotelNo = b2.hotelNo

-- AND b1.roomNo = b2.roomNo

-- AND

-- (

-- (b1.dateFrom > b2.dateFrom AND b1.dateFrom < b2.dateTo)

-- OR

-- (b2.dateFrom > b1.dateFrom AND b2.dateFrom < b1.dateTo)

-- )

-- GROUP BY b1.dateFrom, b1.dateTo

-- ORDER BY b1.dateFrom, b1.dateTo;

-- --(f)

-- SELECT b1.\*, COUNT(\*) AS numOfOverlappingBookingsByGuests

-- FROM Booking b1, Booking b2

-- WHERE b1.guestNo = b2.guestNo

-- AND

-- (

-- (b1.dateFrom > b2.dateFrom AND b1.dateFrom < b2.dateTo)

-- OR

-- (b2.dateFrom > b1.dateFrom AND b2.dateFrom < b1.dateTo)

-- )

-- GROUP BY b1.guestNo, b1.dateFrom, b1.dateTo

-- ORDER BY b1.dateFrom, b1.dateTo;

-- insert into Room values(100, 2, 'penthouse', 249.99); --'penthouse' not valid type. Error success.

-- insert into Room values(100, 2, 'Family', 249.99); --249.99 not BETWEEN 10.00 AND 100.00. Error success.

-- insert into Room values(101, 2, 'Family', 15.00); --room 101 not BETWEEN 1 AND 100. Error success.

-- insert into Booking values(1, 21, '2021-02-02', '2021-10-01', 101);--room 101 not BETWEEN 1 AND 100. Error success.

-- insert into Booking values(1, 21, '2021-02-02', '2021-02-11', 99);--room 101 not BETWEEN 1 AND 100. Error success.

--IGNORE CODE BELOW THIS LINE =========================================================================

-- -- SELECT b1.\*

-- SELECT b1.\*, COUNT(\*) as numOfConflictingBooks

-- FROM Booking b1, Booking b2

-- WHERE

-- /\*(b1.dateFrom BETWEEN b2.dateFrom AND b2.dateTo)\*/

-- (b1.dateFrom > b2.dateFrom AND b1.dateFrom < b2.dateTo)

-- OR

-- /\*(b2.dateFrom BETWEEN b1.dateFrom AND b1.dateTo)\*/

-- (b2.dateFrom > b1.dateFrom AND b2.dateFrom < b1.dateTo)

-- GROUP BY b1.roomNo;

-- --HAVING numOfSimultaneousBooksIncludingSelf > 1;

HW2 - 3.sql: ----------------------------------------------------------------------------------------------------------------------

DROP TABLE IF EXISTS BookingOld;

-- a.

CREATE TABLE BookingOld

( hotelNo int(4),

guestNo int(7),

dateFrom date,

dateTo date,

roomNo int(3)

);

-- b.

INSERT INTO BookingOld

( SELECT \*

FROM Booking

WHERE dateTo < '2003-1-1'

);

-- c.

DELETE FROM Booking

WHERE dateTo < '2003-1-1';

-- NOTE: Not sure about this print. Check Piazza for Dr. VZ's answer.

SELECT \*

FROM BookingOld;

SELECT \*

FROM Booking;

HW2 - 4.sql: ----------------------------------------------------------------------------------------------------------------------

DROP VIEW IF EXISTS CurrentGuestCount;

CREATE VIEW CurrentGuestCount

AS SELECT hotelNo, COUNT(\*) as guestCount

FROM Booking

WHERE dateFrom <= CURDATE() AND dateTo > CURDATE()

GROUP BY hotelNo;

-- print result

SELECT \*

FROM CurrentGuestCount;

-- test code --

-- SELECT \*

-- FROM Booking

-- ORDER BY dateTo;

HW2 - 5.sql: ----------------------------------------------------------------------------------------------------------------------

DROP VIEW IF EXISTS HotelData;

-- problem 5

CREATE VIEW HotelData

AS SELECT g.guestName, h.hotelName

FROM Hotel h, Guest g, Booking b

WHERE h.hotelNo = b.hotelNo

AND b.guestNo = g.guestNo

AND ( b.dateFrom <= CURDATE() AND b.dateTo > CURDATE() )

ORDER BY h.hotelName;

-- TEST CODE

-- CREATE VIEW HotelData

-- AS SELECT g.guestName, h.hotelName, b.dateFrom, b.dateTo

-- FROM Hotel h, Guest g, Booking b

-- WHERE h.hotelNo = b.hotelNo

-- AND b.guestNo = g.guestNo

-- AND ( b.dateFrom <= CURDATE() AND b.dateTo > CURDATE() )

-- ORDER BY h.hotelName;

-- print final result

SELECT \*

FROM HotelData;

-- OLD CODE THAT DIDN'T WORK: KEEPING FOR REFERENCE ------------------------------------------------------

-- CREATE VIEW HotelData

-- AS SELECT g.guestName, h.hotelName

-- FROM Hotel h, Guest g, Booking b

-- WHERE h.hotelNo = b.hotelNo AND b.guestNo = g.guestNo;

-- WHERE g.guestNo IN

-- (

-- SELECT b.guestNo

-- FROM Booking b

-- WHERE b.dateFrom <= CURDATE() AND b.dateTo > CURDATE()

-- );

-- AND

-- h.hotelNo IN

-- (

-- SELECT b.hotelNo

-- FROM Booking b

-- WHERE b.dateFrom <= CURDATE() AND b.dateTo > CURDATE()

-- );

HW2 - 6.sql: ----------------------------------------------------------------------------------------------------------------------

DROP VIEW IF EXISTS CheckingOutToday;

CREATE VIEW CheckingOutToday

AS SELECT g.\*, b.roomNo, DATEDIFF(b.dateTo, b.dateFrom) as numDays,

DATEDIFF(b.dateTo, b.dateFrom) \* r.price as totalCost

FROM Booking b, Guest g, Room r

WHERE b.guestNo = g.guestNo

AND b.roomNo = r.roomNo

AND b.hotelNo = r.hotelNo

AND b.hotelNo =

(

SELECT h.hotelNo

FROM Hotel h

WHERE h.hotelName = 'Grosvenor Hotel'

)

AND b.dateTo = CURDATE();

-- print final result

SELECT \*

FROM CheckingOutToday;

HW2 - 7.sql: ----------------------------------------------------------------------------------------------------------------------

-- REFERENCE: pdf page 37/43 from "Ch07-SQL-DDL.pdf"

-- (a) Valid:: hotelNo and roomNo are correctly selected columns in the View. Lastly, "hotelNo = 1"...

-- ...in WHERE clause is valid b/c hotelNo is not a column based on a aggregate function.

-- If bookingCount had been used in WHERE, then it would have been invalid.

-- (b) Invalid: Columns in Views based on aggregate functions are not allowed to be used as arguments...

-- ... to any aggregate function within any query on the corresponding View.

-- (c) Valid: Columns in Views based on aggregate functions are only allowed in SELECT and ORDER BY...

-- ...clauses. Thus, this query is valid since it adheres to this rule.

-- PLEASE IGNORE EVERYTHING BELOW THIS LINE =======================================================

--TEST CODE for kicks

-- DROP VIEW IF EXISTS RoomBookingCount;

-- CREATE VIEW RoomBookingCount (hotelNo, roomNo, bookingCount)

-- AS SELECT b.hotelNo, r.roomNo, COUNT(\*)

-- FROM Room r, Booking b

-- WHERE r.roomNo = b.roomNo AND r.hotelNo = b.hotelNo

-- GROUP BY b.hotelNo, r.roomNo;

-- SELECT b.hotelNo, r.roomNo

-- FROM Room r, Booking b

-- WHERE r.roomNo = b.roomNo AND r.hotelNo = b.hotelNo;

-- --A: This first GROUP BY seems to only pick the first hotelNo/roomNo pair it finds and uses that

-- SELECT b.hotelNo, r.roomNo

-- FROM Room r, Booking b

-- WHERE r.roomNo = b.roomNo AND r.hotelNo = b.hotelNo

-- GROUP BY b.hotelNo;

-- --B: This GROUP BY actually groups by hotelNo first, then it looks inside each of those groups and...

-- --...groups by roomNo. Good stuff.

-- SELECT b.hotelNo, r.roomNo

-- FROM Room r, Booking b

-- WHERE r.roomNo = b.roomNo AND r.hotelNo = b.hotelNo

-- GROUP BY b.hotelNo, r.roomNo;

-- --C: This GROUP BY behaves the same as A:, but groups by roomNo instead.

-- SELECT b.hotelNo, r.roomNo

-- FROM Room r, Booking b

-- WHERE r.roomNo = b.roomNo AND r.hotelNo = b.hotelNo

-- GROUP BY r.roomNo;

-- SELECT hotelNo, roomNo

-- FROM RoomBookingCount

-- WHERE hotelNo = 1;

-- SELECT hotelNo, SUM(bookingCount)

-- FROM RoomBookingCount

-- GROUP BY hotelNo;

-- SELECT \*

-- FROM RoomBookingCount

-- ORDER BY bookingCount;

HW2 - 8.sql: ----------------------------------------------------------------------------------------------------------------------

--NOTE: The first SELECT prints out result,...

--...while the 2nd one (i.e., ...INTO...) surpresses this output

--SELECT @avgStaffCntPerBranch := AVG(staffCntPerBranch)

SELECT AVG(staffCntPerBranch) INTO @avgStaffCntPerBranch

FROM

(

SELECT COUNT(\*) AS staffCntPerBranch

FROM Staff

GROUP BY branchNo

) as requiredSubqueryAlias; --NOTE. this last alias, "as required...", is needed or else error

SELECT branchNo, COUNT(\*) - @avgStaffCntPerBranch AS staffDiff

FROM Staff

GROUP BY branchNo

HAVING COUNT(\*) > @avgStaffCntPerBranch;

--NOTE: I accidentlly used WHERE instead of HAVING. Got error....

--TEST CODE

-- SELECT @avgStaffCntPerBranch;

-- SELECT COUNT(\*) AS staffCntPerBranch

-- FROM Staff

-- GROUP BY branchNo;

-- NOTE: This is wrong syntax. See above for correct way. The subquery must be in FROM clause

-- SELECT @avgStaffCntPerBranch := AVG

-- (

-- SELECT COUNT(\*)

-- FROM Staff

-- GROUP BY branchNo

-- );

-- OLD CODE: Storing this for reference --------------------------------------------

-- DROP VIEW IF EXISTS groupByBranchNo;

-- CREATE VIEW groupByBranchNo

-- AS SELECT COUNT(\*) as avgMe

-- FROM Staff

-- GROUP BY branchNo;

-- SELECT @avgStaffCntPerBranch := AVG(avgMe) FROM groupByBranchNo;

-- SELECT \*

-- FROM groupByBranchNo;

-- SELECT @avgStaffCntPerBranch;