

COMP207 – Main assignment: The casino database

There are 100 points you can get in this assignment. Recall that the assignment contributes 25% to your overall grade in the course, meaning each point corresponds to 0.25% of your final grade.

The assignment is to create a database for a casino company: The initial task will focus on building the database (i.e., tables and so on), while the latter tasks focus on creating useful queries. The number of points for each question is marked on them.

For question 1, the number of points you get for your submission is public, i.e. it is given to you by CodeGrade a few minutes after you submit.

For questions 2-7, the points are given for two sub-tasks:

- **Public data:** For each question, 2 points are given for getting the right output on the public test data described in an additional file and you will be told how many of these your current submission gets you a few minutes after you submit. The question will specify what this right output is, and you will also be able to see it when you upload your solution to CodeGrade (since you can upload any number of times before the deadline, I would suggest using it to test your solution against the test data)
- **Hidden data:** The remaining points are given for getting the right output on another data set. The latter set is kept hidden to avoid you hardcoding the right output in the queries. It will follow the same form as the test data though and unless you hardcode your solution for the test data, it is very likely that if you get points for one part you will get points for the other.

Because I won't have much time to help each of you individually (I think there will be around 450 of you so you do the math), I would prefer that you put any questions you had on the discussion board. That said, **you are NOT meant to include your code on the discussion board** – simply reference the code you uploaded to CodeGrade if needed. This also means that I would prefer that you try first on your own and/or check the discussion board (because with so many of you, most questions will have been asked and answered at the time most of you think of them and it is faster to look it up for you than waiting for me to answer it again!) and then ask.

How much help can you expect if you ask for it: I am willing to help you a lot if you get stuck in questions 1-3 (but please, spend some time on them first – nearly everybody – think >95% - won't have much trouble with them). For questions 4-5 I will come up with suggestions and ideas to help guide you to a solution (but again, please try first! The questions already have some hints so check those first). Questions 6-7 are meant to be hard. If you do the questions in order, these questions will get your grade from 75 to 100. According to the university's marking scheme, you need to be able to answer every question perfectly to get a grade in the range 70-80(!). Therefore, I will only help clarify what the questions are asking you to do but not actually help you solve them.

Each question from 2-7 requires you to create a view with a specified (in the question) name. You may use as many views as you wish to solve each question (well, except question 1, since it would not be helpful), but there should be one with the specified name, which is the one that is getting checked for having the right output. E.g., you could have **view1ForQuestion2**, **view2ForQuestion2** and **view3ForQuestion2**, if you feel it would help to do question 2 – most other names are fine too (but it might be helpful to you if the names were descriptive)!

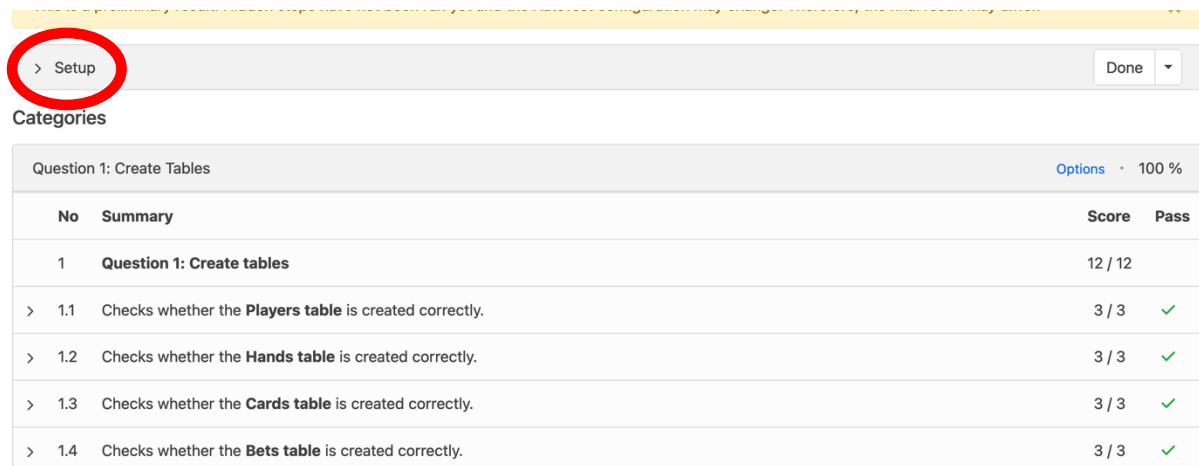
ChatGPT: You are not allowed to use ChatGPT.

Group work: While you might discuss with your friends/colleagues, you must hand in a solution only you worked on, and plagiarism checks will be run.

How to get error messages from CodeGrade

The location of the error messages your submission gets is perhaps not obvious so please check this!

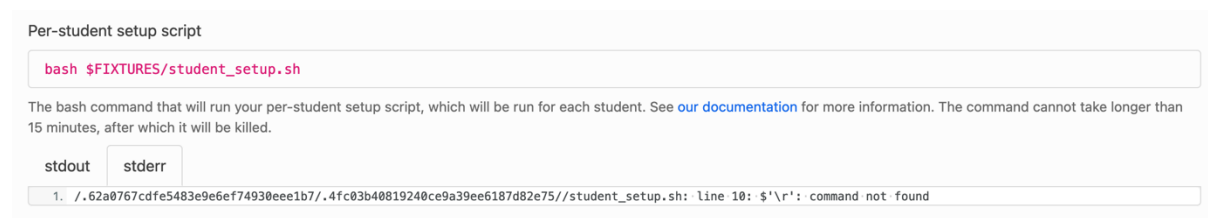
After having run your submission, you will get an output like this (without a **red** circle):



The screenshot shows the CodeGrade interface. At the top, there is a navigation bar with a tab labeled '> Setup' which is circled in red. To the right of this tab is a 'Done' button with a dropdown arrow. Below the navigation bar, the section is titled 'Categories'. A table displays the results for 'Question 1: Create Tables'. The table has columns for 'No', 'Summary', 'Score', and 'Pass'. The first row shows '1' for the question number and 'Question 1: Create tables' for the summary, with a score of '12 / 12'. Below this, there are four sub-questions (1.1 to 1.4) each with a score of '3 / 3' and a green checkmark in the 'Pass' column.

No	Summary	Score	Pass
1	Question 1: Create tables	12 / 12	
> 1.1	Checks whether the Players table is created correctly.	3 / 3	✓
> 1.2	Checks whether the Hands table is created correctly.	3 / 3	✓
> 1.3	Checks whether the Cards table is created correctly.	3 / 3	✓
> 1.4	Checks whether the Bets table is created correctly.	3 / 3	✓

Click on the > in the **red** circle and scroll down some. You will see something like this:



The screenshot shows the 'Per-student setup script' section. It contains a text box with the command `bash $FIXTURES/student_setup.sh`. Below this, there is a note: 'The bash command that will run your per-student setup script, which will be run for each student. See our documentation for more information. The command cannot take longer than 15 minutes, after which it will be killed.' There are two tabs: 'stdout' and 'stderr'. The 'stderr' tab is selected, showing an error message: '1. /.62a0767cdf5483e9e6ef74930eee1b7/.4fc03b40819240ce9a39ee6187d82e75//student_setup.sh: line 10: \$'\r': command not found'.

That stderr will contain your error message.

Deadline and feedback

The deadline for the assignment is **Wednesday the 6th of November at 17:00**. General feedback for the assignment will be given on Tuesday the 19st of November. The relatively long period between those is because you can get an ELP and they only last until I give feedback. To accommodate people with ELPs as well as I can, I will therefore first give feedback nearly 2 weeks after the deadline (2 weeks is the maximum amount of time an ELP can give you).

If you have specific concerns about your grade or similar for the assignment, then, after the general feedback has been released, I will answer questions about your solution and grade over email.

Format

The assignment should be done in .sql format (i.e., the output format from MySQL's workbench) – it is really just a basic text file with the SQL commands written in it and you could do it by writing the file directly in a basic text editor if you wish (Notepad in Windows or TextEdit on Mac – if you select Make Plain Text in Format).

The name of the file should be casino.sql: You can hand in precisely 1 file, and it must have precisely that name (you can submit as many times as you wish until the deadline, but only the most recent version counts).

And each line should contain only the following:

1. CREATE TABLE statements for question 1 (4 in total)
2. CREATE VIEW statements for questions 2-7 (the number of views depends on how you solve the questions and how many you solve, if not all). Note, that you may use any positive number of views to solve each question, but each question's specified view should have the properties requested.
3. SQL comments, i.e. the part of lines after "-- ", i.e. double - followed by space. You do not need to make any but may do so if you wish.

In particular, **do not include CREATE DATABASE and USE statements**. They will make the tests on the hidden data not work (technically, I create two databases based on your construction, one with the public data given in the additional file and one with the hidden data. If you use CREATE DATABASE or USE statements, in essence only 1 is made and it will be marked as if you did not do the other one). I have written tests that test for this so if you do not do something really complex to avoid these, you should at least be warned about it.

You can include INSERT statements, but the database will be emptied before checking, so it serves little purpose (often, many people will submit with the test data already inserted, but, because the databases will be emptied, it will not matter).

Make sure that you can run the full file through MySQL when using the Casino database (starting with an empty Casino database) and after having done so, the Casino database should contain the tables and views required from the questions you solved (and perhaps some more views if you feel it would be convenient). This means that **you should remove any statement that causes errors before handing in the assignment** because MySQL stops when it encounters an error (meaning that the last statements are not executed)! If you do not, you risk getting a far lower grade than otherwise (because the part of your hand-in after the first error will not be graded).

You can submit any number of times before the deadline: We are using CodeGrade for checking these things and whenever you submit, you will see whether your file works for the public dataset. I suggest using it...

Do *not* do the following!

Any of the following should **not** be done:

- End by removing the database (i.e. DROP DATABASE Casino; or similar). It would be the same as handing in an empty file. It will be very easy to see on CodeGrade since everything will stop working.
- Create comments like "-----". MySQL Workbench will accept it, but the command line version of MySQL does not, which is what is used to check your file... Just insert an extra space after the second -.
- Use any other order for the columns than what is specified. Since the insert command does not state which columns they insert into, you will put the information in the wrong column and then get hard-to-understand issues when you attempt to solve the questions.

- Use PARTITION OVER or other new commands. It was not taught in class and is specific to newer versions of MySQL (like the one you would install on your own laptop). Unfortunately, CodeGrade is using an old version of Ubuntu (or at least did last year), where the newest versions of MySQL do not work. Therefore, it will not work when we grade you and you will fail that (and later questions – MySQL will report an error on that line and will not run the rest of your file).
- It is very unlikely that you would want to remove any views after you have made them. Unless you have a very good reason, do NOT remove them. Doing so anyway typically leads to strange error messages.

Question 1 – (Easy) (worth 16 points – 4 points for each table – you get 3 as part of the question and 1 as part of a check that you did not use USE or CREATE DATABASE)

Make the set of tables that match the following set of schemas.

- **Players**(birth_day, first_name, last_name, is_dealer,g_id)
- **Hands**(time,game_type, g_id*,r_id,h_id)
- **Cards**(rnk,suit,h_id*)
- **Bets**(amount ,h_id*)

Each underlined attribute should be the primary key for the table (it happens to be the last attribute in the first 2 tables) and each attribute with * should have a foreign key to the table with a primary key of just that name, e.g. if the tables were R(a,b) and S(b*,c), b in R and c in S should be the primary keys and b in S should reference b in R as a foreign key. More directly, g_id* in Hands should reference g_id in Player and h_id in Cards and in Bets should each reference h_id in Hands.

Only use data types in the following list: INT, VARCHAR(20), DATE, DATETIME and BOOL. Instead of specifying the datatypes explicitly, ensure that the test data defined in the additional file gets inserted correctly (it seems very likely that you would also guess the same datatypes as these suggest – a few exceptions: rank is at least 1 and at most 13 and game_type is either 'Blackjack' or 'Poker' and suit is one of 'Spades', 'Diamonds', 'Hearts' or 'Clubs') and use DATE, BOOL or DATETIME if all entries are dates, booleans or date-times. If you follow all of these requirements, each attribute should have a clear, unique datatype (which happens to likely be what you would guess it to be).

Question 2 – (Easy)

(worth 24 points – 2 points for getting the right output on the test data and another 22 for the hidden data – see the first page for more detail!)

Find the total amount of bets placed in October 2024. Note that a hand can be associated with multiple bets (in poker) and that the dealers (only playing blackjack) do not directly bet – in other words, a hand is associated with a non-negative number of bets.

More precisely, you are asked to create a view **October2024Bets** with a single column, `total_bets`, in which the lone entry is the sum of every bet made on a hand played in October 2024. You may assume that some bets were placed on some hand in that month (i.e. including in the private data).

Note that in the test data, there are some hands to count in October 2024, but also some in each of 1) October 2023, 2) September 2024, 3) November 2024. The bets in October 2024 are respectively of amount 2, 1, 4, 1, 4, 5, 5, and 42, which sums to 64. Note that in Poker, you can bid multiple times on a hand (raising), while the casino lets you bet on your blackjack hand once each (this is not likely to matter for your queue).

HINT: Recall that `SUM(amount)` will sum up all amounts you have in the output, so you just need to make a query that finds the bets placed on hands in October 2024 and then use that.

The view should be called **October2024Bets** and be such that the output of

```
SELECT * FROM October2024Bets;
```

when run on the Casino database (after inserting the test data given in an additional file) should be:

<code>total_bets</code>
64

Question 3 – (Easy) (worth 10 points – 2 points for getting the right output on the test data and another 8 for the hidden data – see the first page for more detail!)

While in the UK, you need to be at least 18 years old to gamble, the casino wants to be safe and will manually consider everybody in the range 18 to 20 years (or more precisely 7305 days old, see below) whenever they play a hand.

Given two dates (or in this case likely a date and a datetime), X and Y, you can determine the number of days between them using DATEDIFF(X,Y). Due to leap years, you are (typically) 20 years old exactly when you are 7305 days old (due to how leap years work around centuries, this is not actually true: E.g. if you were born in the year 1897 you would need just 7304 days to become 20 years old due to 1900 not being a leap year). To make this question simpler, you are meant to determine whether a player is at least 7305 days old when they play a hand.

In the example data, Hank and Eve are below 20 each time they play, while Grace is below 20 the first time, but above 20 all remaining times (she got older in-between).

For each hand played, display the h_id, the (first and last) name of the player and whether they were at least 7305 days old when they played the hand (the latter column should be called Above20).

The view should be called **Above20** and be such that the output of

```
SELECT * FROM Above20 ORDER BY h_id;
```

when run on the Casino database (after inserting the test data given in an additional file) should be:

h_id	first_name	last_name	Above20
1	Grace	Moore	0
2	Alice	Johnson	1
3	Ivy	Anderson	1
4	Charlie	Davis	1
5	Hank	Taylor	0
6	Alice	Johnson	1
7	Hank	Taylor	0
8	Alice	Johnson	1
9	Grace	Moore	1
10	Ivy	Anderson	1
11	Jane	Smith	1
12	Bob	Brown	1
13	John	Doe	1
14	Jane	Smith	1
15	Grace	Moore	1
16	Jane	Smith	1
17	Charlie	Davis	1
18	Eve	Wilson	0
19	Frank	Miller	1
20	Jack	Thomas	1

Question 4 – (Medium) (worth 15 points – 2 points for getting the right output on the test data and another 13 for the hidden data – see the first page for more detail!)

A manager wants to be able to wish each player happy birthday on their birthdays (whether dealer or outsider). Output all the players (birth_day,first_name,last_name) sorted by how far in the future their birthday is, from the 6th of November (i.e. the day you are meant to hand in) – so any on the 6th of November would be the first, followed by any on the 9th of November and so on and any on the 31st of December would be before any on the 1st of January and so on. Finally, any on the 5th of November would be last.

HINT: Given a date D, you can get which month it is in using MONTH(D) and the day it is in that month using DAYOFMONTH(D). There is a similar function for the day in the year, but if you used that you run into issues with people born in a leap years (specifically birthdays in March or later would be off-by-1). One way to do the query is to find the people that still have their birthday this year in some view where you make a constant 1 in a new column and then find the ones that first have their birthday next year and give them the constant 2 in the same new column and putting those together with UNION. You can then order the UNION by the new column, the month and the dayofmonth of the birthdays.

Note, **your query is NOT meant to use the current date, but specifically the 6th of November** – it is likely easy to convert it to the current date, using CURDATE() but it would make the query change depending on the day it gets checked.

The view you create should be called **UpcomingBirthdays** and it should be such that

```
SELECT * FROM UpcomingBirthdays;
```

when run on the Casino database (after inserting the test data given in an additional file) should be:

birth_day	first_name	last_name
1985-11-07	Jane	Smith
1993-12-22	Frank	Miller
1980-01-15	Bob	Brown
1992-03-07	Alice	Johnson
1987-04-25	Ivy	Anderson
1990-05-12	John	Doe
1995-07-30	Charlie	Davis
1994-08-14	Jack	Thomas
2004-09-18	Grace	Moore
2004-10-30	Hank	Taylor
2005-11-05	Eve	Wilson

Question 5 – (Medium-Hard)

(worth 10 points – 2 points for getting the right output on the test data and another 8 for the hidden data – see the first page for more detail!)

Over the next two questions, we will look at blackjack (i.e. the Hands and Cards tables for h_id's such that game_type='Blackjack' in Hands).

Blackjack is a game between two players, specifically a player from outside the casino and a dealer (the latter working for the casino). Each round, the outsider is dealt two cards and can then repeatedly add in another card until they decide to stop or their hand has a value of 22 or more (in which case they go bust). The dealer stops whenever their hand has a value of 17 or more (and also go bust on 22 or more). The winner is the player with the largest hand value when they have both stopped with bust counting as a hand value of 0.

- The cards with ranks between 2 and 10 (both included) work in a simple manner where the value of the cards is just their rank
- Cards with ranks 11-13 (referred to as jack, queen and king respectively) each have a value of 10.
- The cards with rank 1 (referred to as ace) are special in that their value depends on what would be best for the player with it in hand: Each ace can be 11 or 1 depending on what would be best (since $2 \times 11 = 22$, it is never best to have two aces of value 11).

In this question, we are looking at blackjack hands with id numbers at most 5. Those hands are such that they have no cards of rank above 10 or rank 1 (in both the public and private data).

You are meant to find the hand value of each such hand (taking busts into account).

HINT: It might be easier to first make a view that simply adds up the ranks for each hand, then one or more building on that to take care of reducing busts to 0.

As an example, h_id = 1 contains (2,'Diamonds',1),(5,'Diamonds',1),(3,'Spades',1). Observe that $2+5+3=10$, so h_id 1 has a Hand_value of 10. Another example would be h_id 3, which has the cards (9,'Diamonds',3),(10,'Clubs',3),(3,'Clubs',3). Since that sums to $9+10+3=22$, the hand value is 0.

The view you create should be called **SimpleBlackjack** and it should be such that

```
SELECT * FROM SimpleBlackjack WHERE h_id<=5 ORDER BY h_id;
```

when run on the Casino database (after inserting the test data given in an additional file) should be:

h_id	Hand_value
1	10
2	20
3	0
4	21
5	0

Question 6 – (Hard)

(worth 15 points – 2 points for getting the right output on the test data and another 13 for the hidden data – see the first page for more detail!)

In this question, we are looking at blackjack again (like in question 5), but looking at the remaining blackjack hands where `h_id > 5` (that can contain cards with rank 11-13 and rank 1).

Besides the remaining rank of cards, we also want the output to be for each `r_id`, is the outsider's hand worth more than the dealers? (for each blackjack round, i.e. for each `r_id` where `game_type='blackjack'`, there are two hands: One by an outsider and one by a dealer).

For simplicity, no round has the same hand value for both players. As an example, in round 1, consisting of hands 1 and 2, we have a hand value of 10 for the outsider and a hand value of 20 for the dealer, meaning that the dealer wins. Another example could be round 4 in which the outsider has a hand value of 20 and the dealer has one of 17, meaning the outsider wins.

Hint: It is likely easier to split this into multiple views (recall that you are allowed to use any number of views for each question): One that gives the value of each hand (**NotSoSimpleBlackjack** e.g.), that likely could call **SimpleBlackjack** from the last question to good effect, and then, based on that, one that finds the winner of each round.

The view should be called **Blackjack** and it should be such that

```
SELECT * FROM Blackjack ORDER BY r_id;
```

when run on the Casino database (after inserting the test data given in an additional file) should be:

r_id	Outsider_wins
1	0
2	0
3	0
4	1
7	1

Question 7 – (Hard) (worth 10 points – 2 points for getting the right output on the test data and another 8 for the hidden data – see the first page for more detail!)

In this question, we are looking at the poker hands. In this case, they are all 5 different card hands.

A poker hand has one of the following values:

- Straight-flush: The hand is both straight and flush, see below
- Four of a kind: There are 4 of some card rank
- Full house: There are 3 of some card rank and two of another
- Flush: All suits are the same (but it is not a straight-flush)
- Straight: The card ranks are consecutive (e.g. the card ranks form the set {3, 4, 5, 6, 7}) OR the 5 cards ranks form the set {1,10,11,12,13}. Finally, the hand is not also a flush (because it would then be a straight-flush).
- Three of a kind: There are 3 of a card rank
- Two pairs: There are two of one card rank and two of another (but it is not a full house)
- Pair: There are two of one card rank (but not any of the above)
- High card: None of the above

While the above is the ordering of hands in poker (earlier being better), since we are not comparing hands but simply trying to find their value, it does not matter.

You are simply meant to, for each hand, determine its value. The output is supposed to be `h_id`, `handvalue1`, `handvalue2`, where `handvalue1` and `handvalue2` depends on the type of hand, see below:

- Each hand which is a Straight-flush, Flush or Straight, should be such that `handvalue1` = 'Straight' if the hand is straight and `handvalue1` = "" otherwise. Also, if it is a Flush, `handvalue2` = 'Flush' and `handvalue2` = "" otherwise.
- Each hand which is a Four of a kind, Full house, Three of a kind, Two pairs or pair, should be such that `handvalue1` is how many there are of the rank of which there are the most. Also, `handvalue2` is how many there are of the rank of which there are the second most (or equivalently, how many ranks there are two or more of). E.g. Four of a kind has `handvalue1`=4 and `handvalue2`=1 and Full house has `handvalue1`=3 and `handvalue2`=2.
- Each hand which is a High card has `handvalue1` = 'High' and `handvalue2` = 'Card'.

While it is unlikely to be a good idea to make just one view for this question (you could if you felt it was the better solution, but I am fairly sure it is not), one of the explicit challenges in this question is that there is no suggested set of views to make.

If you look in the public test data, each hand has its value written on it as a comment. So, e.g. `(1,'Hearts',9),(10,'Hearts',9),(11,'Hearts',9),(12,'Hearts',9),(13,'Hearts',9)` means that `h_id`=9 is a straight-flush, so `handvalue1`='Straight' and `handvalue2`='Flush', while `(1,'Diamonds',19),(2,'Diamonds',19),(11,'Diamonds',19),(12,'Diamonds',19),(13,'Hearts',19)` means that `h_19` is High card and has `handvalue1`='High' and `handvalue2`='Card'.

The view should be called **Poker** and be such that the output of

```
SELECT * FROM Poker ORDER BY h_id;
```

when run on the Casino database (after inserting the test data given in an additional file) should be:

h_id	handvalue1	handvalue2
9	Straight	Flush
10	4	1
11	3	2
12		Flush
13	Straight	
14	3	1
15	2	2
18	2	1
19	High	Card
20	High	Card