1. Project Description: Course Layout and Teaching load Allocations

This project aims to facilitate course layout, planning and teaching load allocations at a university.
Which involves designing and creating a database and writing and executing several queries that correspond to the operations specified in the following subsections.

1.1 Business Overview

The university offers several courses.
A course may consist of several teaching activities.
The two most important tasks are

- to plan a course in terms of deciding the hours needed for various teaching activities called "course layout and planning",
- and actual allocation of teaching activities to the teachers, called "teaching load allocations".

1.2 Detailed Descriptions

Course Layout

Each course has a unique course code, course name, högskolepoäng (HP),
minimum and maximum number of students must also be mentioned in the course layout.

Below is an example of a course layout.

Table 1. Course Layout Example.
This example is only meant to illustrate some teaching activities and other information that may be associated with course layouts,
it is NOT meant to show any rows or columns of a Table.

Course code	Course Name	HP	Min Students	Max Students
IV1351	Data Storage Paradigms	7.5	50	250
IX1500	Discrete Mathematics	7.5	50	150

Course Instances:

A course instance is an instance of a particular course layout given at a particular period of a year,
for example a course given during period 1 in year 2025 gets a new unique course instance.
An academic year is divided into four periods, P1, P2, P3 and P4.
For simplicity, you may assume that the academic year is from January to December.
A course instance also has the total number of students registered.

Teaching Activities:

We are mainly interested in the following teaching activities that may be associated with course layouts or course instances :
Lecture, Lab, Tutorial, Seminar, examination, administration and Others.
However, the database must be flexible enough to add more activities in the future.

Some teaching activities may require some preparation time for teachers (let's call it a multiplication factor).

Below is an example of multiplication factors for different teaching activities.

Table 2. Multiplication Factors Example.
This example is only meant to illustrate example of multiplication factors for some teaching activities,
it is NOT meant to show any rows or columns of a Table.

Activity Name	Factor
Lecture	3.6
Lab	2.4
Tutorial	2.4
Seminar	1.8

For example, for the course IV1351, in the course instance, Table 3, lecture hours are given 20 hours.
After applying the above rule/multiplication factor, the total hours for the lectures including preparation time would be 72 hours.
In other words, a teacher actually spends 72 hours for giving 20 hours lecture.

Planned Activities:

A course may contain several activities, we are mainly interested in the number of hours needed for the following teaching activities:
Lectures, Labs, Tutorials, Seminars and Others. It is important to note that some courses may have only some specific teaching activities.
Each course instance also gets two additional activities that correspond to examination and administration hours which depend on the number of registered students and högskolepoäng (HP).
Here is an example to calculate the examination and administration hours for a particular course instance (these are derived attributes).

Examination hour = 32 + 0.725 · No. Students

Admin hours = 2HP + 28 + 0.2 · No. Students

Below is an example of course instances with planned activities.

Table 3. Course Instance Example.
This example is only meant to illustrate some teaching activities and other information that may be associated with course instances,
it is NOT meant to show any rows or columns of a Table.

Course code	Course Instance ID	HP	Period	# Students	Lecture Hours	Tutorial Hours	Lab Hours	Seminar Hours	Other Overhead Hours	Admin	Exam
IV1351	2025-50273	7.5	P2	200	20	80	40	80	650	177	83
IX1500	2025-50413	7.5	P1	150	44	0	0	64	200	141	73

Departments:

There are several departments at the university.
Each department has a manager who is also a teacher/employee.

Teachers/Employees:

Teachers or employees are affiliated with departments.
They are allocated teaching activities based on their interests or skill sets.
It is also important to have their contact details and salary, which is later used to calculate the teaching cost of a course or total teaching cost for a department.
It is also advisable to have their job titles/designations. Each employee has a manager or a supervisor.

Allocations:

A teacher can get involved in teaching activities for many course instances,
and a course instance can have many teachers involved with.

However, a teacher must not be allocated more than four different course instances simultaneously during a particular period..

1.3 Conceptual Model:

A conceptual model is provided that describes the requirements above.

Note that the provided conceptual model is an abstract and possibly may not capture all specific details of the system.

Note: Watch the conceptual model lecture (only recorded, not given live).
The conceptual model lecture page contains several videos, which together lasts about three hours.

1.4 Requirements on Project Task Application

The database must store all data described above, in sections 1.1 and 1.2, but no other data.
The database will be used to retrieve reports and statistics of all possible kinds, but a user interface is not required for that purpose.
It will instead be done by manually querying the database.

The database will not be used for any financial purpose like bookkeeping, salary or bank contacts.
What is written above regarding salary is only about calculating the total teaching cost of a course instance or a department over a particular period.

2. Grading

There are three tasks, which are described below. Each task gives max 10p, and is divided into two parts giving 5p each.
The first two tasks have a mandatory part (5p) and a higher grade part (5p).
The third task has two higher grade parts worth 5p each.
To pass the project all mandatory tasks must be passed.
The higher grade tasks are optional, and contribute to the final grade as specified in the Course Layout.
No partial score is given, either a part of a task is accepted and gives five points, or it isn't accepted and does not give any points.
A task is passed when both seminar and written report are passed.
Reporting is explained in detail below.

3. How to Get Help

Questions about the project can be asked these ways:

- At tutorials, tutorials exist only to provide help with project tasks. You are welcome to ask about anything related to project work at tutorials.
- In Piazza. Don't send direct emails to teachers, instead post questions in Piazza since then all students can be helped by the answers.
  Mind that posts can be anonymous to peer students in Piazza.

4. How to Report

Each solved task shall be reported both in writing and at a seminar.
A task is passed only when both written report and seminar have been accepted.
You are encouraged to collaborate and discuss with as many other students as you wish when doing the project,
group discussions always give a better result than individual work. The assignments are done in groups of three students;
please sign up for a group with name "Project Groups", which you can find by following the "People" link to the left.
You are also allowed to work alone, but that is not recommended.

4.1 Written Report

The written report must follow this report template, iv1351-report-template_new.pdf.
Note that the template is a pdf, a similar document must be created.
Layout changes are allowed, it is only the content that's important.
The template pdf is created using latex, if you want to try that you can use the following tex file, iv1351-report-template_new.tex, which was used to create the template pdf.
The report shall be submitted under the corresponding seminar, the report for task 1 is submitted under seminar 1, etc.
All submitted assignments should be on behalf of a project group, and will be plagiarism checked.

4.2 Seminar

The seminar schedule is found on the Seminar Schedule page,
and you can see which group you belong to on the seminar groups page.
To pass a seminar, you must have solved the specified tasks and participate in the discussion during the seminar.
It's not required that the solution is correct in all parts or that all statements made at the seminar are correct.
The seminars are divided into two parts, one for peer assessment and one for presentation.

4.2.1 First Seminar Part, Peer Assessment

You'll be divided into groups of three, each group will assess the solutions of members of one other group.
The results of the assessments shall be uploaded to Canvas and emailed to the assessed student.
You're not required to bring a printout of your report.

4.2.2 Second Seminar Part, Presentation

- One randomly chosen student from each group will give a five-minute presentation of their own solution to the entire seminar.

Prepare this presentation before the seminar.
- One randomly chosen student from the assessing group will give comments on this solution and there will be a brief discussion.
  Five minutes are allocated for comments and discussion together.

5. Tasks

The assignments are done in groups of three students; please sign up for a group with name "Project Groups",
which you can find by following the "People" link to the left.
You are also allowed to work alone, but that is not recommended.

5.1. Task 1, Logical and Physical Model

How To Prepare

Understand the lectures on normalisation (given live and recorded)
and on logical and physical models (only recorded, not given live).
You can start working on the task before the normalisation lecture, but you have to understand that lecture before completing the task.
Read the document seminar1-tips-and-tricks.pdf.

When To Solve

You're recommended to start working on this task as soon as possible, but remember to first do the preparations mentioned above.
The deadline for submitting the report is found on the seminar 1 assignment page.

Intended Learning Outcomes

- Model needs for information based on an organizational description and convert the model to a functioning database.
- Use relational databases and query languages.

Mandatory Part

Translate the given conceptual model into the same kind of model that was created in the lecture on logical and physical models, that is a logical model with enough physical aspects to enable creating a database.
The model that is created must cover the entire description of the project in the section above.
The diagram must be made in a crow foot notation (for example, IE notation).
Also create a database based on the model;
the database must be exactly as described in the model.
If you discover flaws when creating the database, and want to change it,
also the model must be changed.
You are advised to also look at tasks 2 already now,
to get an understanding of how your database will be used.
Below are guidelines for what should be written in the report.
In addition, it is also required to insert enough data in the database you have created.

- In the Method chapter of your report, mention diagram editor(s) and other tool(s) you used, and explain the procedure you followed to create the model.

You do not have to mention all eleven steps covered in the videos on on logical and physical models, but it must be clear how you proceeded.
Do not explain the result of each step of your procedure, only explain the steps themselves.

- In the Result chapter of your report, show and briefly explain your model.
  Also include a link to a git repository where you have stored SQL scripts that create your database.
  There shall be one script that creates the database, and another script that inserts data.
  HINT: You might want to use an online generator to create the data, for example https://generatedata.com/Links, instead of creating the data manually.
  You're also not required to write the script that creates the database manually.
  If you create the database using a GUI tool, you may generate the script using for example pg_dump (note that pg_dump is a Postgres tool, but there are similar tools for MySQL).
  You may explain the SQL in the scripts if you wish, and think it clarifies the model, but it is not required to write such an explanation.
- In the Discussion chapter of your report, evaluate your model.
  Suggested assessment criteria are found in seminar1-assessment-criteria.pdf, you do not have to cover them all.
  These same criteria will also be used to grade your project report.

Higher Grade Part

All three requirements below must be met to get the higher grade points.
You can not get part of the points by meeting just some of the requirements.

1. **The report must include a relevant and extensive Discussion** chapter about the mandatory part of the task.
2. Make sure that the models created in this task contain all data required by the project, and don't require an application using the database to manage any data at all.
   It could, for example, be tempting to create a database that doesn't store the number 4, which is mentioned in the text each teacher can teach up to 4 courses in a particular period of an academic year,
   and instead hardcode the number 4 into the application calling the database.
   Carefully read the description of the project in section one above, and make sure your models include all data mentioned in that text.
   You don't have to consider any data that's not clearly mentioned in section one. Discuss advantages and disadvantages of storing all data in the database, as is done here,
   instead of having some data in the application.
   This discussion shall be placed in the Discussion chapter of the report.
3. **This task is to handle the fact that a course layout can change and also a teacher's salary can change**.
   Say for example, a course is given both in P1 And P2, for P2 HP is changed from 7.5 to 15. In this situation, it must be possible to find the correct HP for the course instance for P1 and for P2.
   You have to solve this task by creating a model which allows adding a new layout, when something changes in a course layout.
   Discuss advantages and disadvantages of keeping multiple versions of data, as is done here.
   This discussion shall be placed in the Discussion chapter of the report.
   HINT: A solution to this problem often includes storing the date_time or version of a certain layout.
   You're however not required to do that, there are also other possible solutions.

5.2. Task 2, SQL

How to prepare

Understand the lecture on SQL (given live and recorded).
Read the document seminar2-tips-and-tricks.pdf.

When To Solve

You're recommended to start working on this task as soon as possible after seminar 1, but remember to first do the preparations mentioned above.
The deadline for submitting the report is found on the seminar 2 assignment page.

Intended Learning Outcomes

- Use relational databases and query languages.
- Describe and explain basic concepts, principles and theories in the area of data/databases/data storage and in information administration and database design

Mandatory Part

The goal here is to create OLAP, Online Analytical Processing, queries and views.
Such queries serve to analyze the business and to create reports.
You're also required to analyze the efficiency of one of the queries with EXPLAIN ANALYZE.
Even though that's not standard SQL, it's available in both PostgreSQL and MySQL.
You also have to make sure the database contains sufficient data to check that all queries work as intended.
If needed, update the script that inserts data, created in task two.
You're allowed to change the database you created in task two if needed.
The queries that shall be created are explained below, only OLAP queries will be created here.
The OLTP (Online Transaction Processing) queries used by the business itself will be created in task 3, together with the program executing them.

HINT: You might want to use an online generator to create the database content, for example https://generatedata.com, instead of creating the data manually.

The following queries will be executed manually, to generate analysis reports.

1. Planned hours calculations:
   Calculate the total hours (with the multiplication factor) along with the break-ups for each activity,
   for the current years' course instances.

Table 4. Expected output for query 1.
This example is only meant to illustrate the expected rows and columns.
It's perfectly fine to change text formatting, and also to change the values.

Course code	Course Instance ID	HP	Period	# Students	Lecture Hours	Tutorial Hours	Lab Hours	Seminar Hours	Other Overhead Hours	Admin	Exam	Total Hours
IV1351	2025-50273	7.5	P2	200	72	192	96	144	650	177	83	1414
IX1500	2025-50413	7.5	P1	150	159	0	0	116	270	141	73	759

2. Calculate actual allocated hours for a course:
   Calculate the total allocated hours with the multiplication factor along with the break-ups for each activity and for each teacher, for a current years' course instance.
   Table 5 below is an example result of such a query, illustrating the expected output.

Table 5. Expected output for query 2.
This example is only meant to illustrate the expected rows and columns.

It's perfectly fine to change text formatting, and also to change the values.

Course code	Course Instance ID	HP	Teacher's Name	Designation	Lecture Hours	Tutorial Hours	Lab Hours	Seminar Hours	Other Overhead Hours	Admin	Exam	Total
IV1351	2025-50273	7.5	Paris Carbone	Ass. Professor	72	0	0	0	100	43	61	276
IV1351	2025-50273	7.5	Leif Linbäck	Lecturer	0	0	0	64	100	0	62	226
IV1351	2025-50273	7.5	Niharika Gauraha	Lecturer	0	0	0	64	100	43	61	268
IV1351	2025-50273	7.5	Brian	PhD Student	0	0	50	0	100	0	0	150
IV1351	2025-50273	7.5	Adam	TA	0	0	50	50	0	0	0	100

3. **Calculate the total allocated hours (with multiplication factors) for a teacher, only for the current years' course instances.**
   Table 6 below is an example result of such a query, illustrating the expected output.

Table 6. Expected output for query 3.
This example is only meant to illustrate the expected rows and columns.
It's perfectly fine to change text formatting, and also to change the values.

Course code	Course Instance ID	HP	Period	Teacher's Name	Lecture Hours	Tutorial Hours	Lab Hours	Seminar Hours	Other Overhead Hours	Admin	Exam	Total
IX1500	2025-50413	7.5	P1	Niharika Gauraha	159	0	0	0	100	141	73	473
IV1351	2025-50273	7.5	P2	Niharika Gauraha	0	0	0	64	100	43	61	268
ID2214	2025-50341	7.5	P2	Niharika Gauraha	44	36	0	0	40	0	20	140
IV1350	2025-60104	7.5	P3	Niharika Gauraha	0	25	0	0	100	0	74	199

4. **List employee ids and names of all teachers who are allocated in more than a specific number of course instances during the current period.**
   Table 7 below is an example result of such a query, illustrating the expected output.

Table 7. Expected output for query 4.
This example is only meant to illustrate the expected rows and columns.
It's perfectly fine to change text formatting, and also to change the values.

Employment ID	Teacher's Name	Period	N.o of courses
500001	Paris Carbone	P1	1
500004	Leif Lindbäck	P1	1
500009	Niharika Gauraha	P1	2

Below is specified what shall be written in the report.

- In the **Method** chapter of your report, mention which DBMS you use, which tool is used to develop SQL queries, and how you have verified that your SQL queries work as intended.
  You shall just tell which method you used for verifying that the queries work as intended, not explain in detail how you verified the functionality of each query.
- In the **Result** chapter of your report, include a link to a git repository where you have stored a script with all queries.
  Also explain each query and show that all queries work as intended by including the output of each query.
  The git repository must also contain the scripts that create the database and insert data.
  It shall be possible to test your solution by executing first the script that creates the database, then the script that inserts data, and finally any of the queries created in this task.
- In the **Discussion** chapter of your report, evaluate your queries.
  Suggested assessment criteria are found in seminar2-assessment-criteria.pdf.
  You don't have to cover them all, but **you must at least cover the bullet on EXPLAIN, which means you have to analyze at least one of your queries.**
  The assessment criteria in this document will also be used to grade your project report.

Higher Grade Part

The following requirements must be met to get the higher grade points.
You can not get half of the points by meeting just one or two requirements.

1. **The report must include a relevant and extensive Discussion chapter** about the mandatory part of the task.
2. **Consider the following workload of queries, with their approximate daily frequencies:**
   1. Planned hours per course instance (8x/day)
   2. Actual allocated hours per teacher per course instance (12x/day)
   3. Teacher load per period (5x/day)
   4. Course instances with planned vs actual variance >15% (3x/day)
   5. Teachers allocated to more than N courses in current period (20x/day)
   - Decide which queries benefit most from indices
   - Which (if any) of the above queries should be turned into materialized views.
   - Justify your choices based on query frequency, expected data size, freshness etc.
   - Implement the chosen indices/views and run EXPLAIN ANALYZE before and after, comparing costs and runtimes.
   - In the discussion chapter, argue about trade-offs between normal views vs materialized views, index maintenance cost vs query performance, and what happens if workloads change.

Task 3, Programmatic Access

How To Prepare

- Before solving this task you have to understand the lectures on transactions (given live and recorded) and on Database Applications (only recorded, not given live).
- Read the document seminar3-tips-and-tricks.pdf

When To Solve

You're recommended to start working on this task as soon as possible after seminar 2, but remember to first do the preparations mentioned above.
The deadline for submitting the report is found on the seminar 3 assignment page.

Intended Learning Outcomes

- Describe how a program can access a database and write such a program

Mandatory Part

This project task has no mandatory part; you don't have to solve it unless you want points to improve your grade.

Higher Grade Part, Task A (gives 5p)

The assignment is to develop part of course layout and teaching allocation's website.
You're however only required to develop a very limited set of functionalities.
Also, since focus here is on database access, you're not required to develop the web interface, but a command line user interface is sufficient.
You're allowed to reuse as much code as you wish from all classes in the view layer of the JDBC bank example at the page Database Applications, but all code included in your program is your responsibility.
You're not allowed to blame any deficiency in your application on the bank program.
Your program shall be stored in a public git repository, for example on GitHub.
The program is required to handle ACID transactions properly, which means autocommit must be turned off, instead the program must call commit and rollback as required.
Handling transactions properly also means that SELECT FOR UPDATE must be used when required.
Finally, you have to make sure the database contains sufficient data to check that all queries work as intended.
If needed, update the script that inserts data, created in task one.
You're allowed to change the database you created in task one if needed.

The program must have the following functionality:

1. **Compute the teaching cost (as planned and actually allocated) of a particular course instance given in the current year.**
   You may assume (or calculate) the average salary for a teacher, for computation of the planned cost.

Table 8. Expected output for query 4.
This example is only meant to illustrate the expected rows and columns.
It's perfectly fine to change text formatting, and also to change the values.

Course Code	Course Instance	Period	Planned Cost (in KSEK)	Actual Cost (in KSEK)
IV1351	2025-50413	P1	600	745

2. **Modify the course instance:**
   Increase the number of registered students by 100 to the course instance you have selected at the previous step.
   Now compute the teaching cost for that course instance again and see how the teaching cost is affected.

3. **Allocate and deallocate teaching loads:**
   Allocate and deallocate teaching activities for various course instances for teachers.

   Remember that a teacher is not allowed to teach (or get involved in any teaching activities) in more than four course instances in a particular period, your program must check that this limit is not exceeded.
   You must demonstrate at least one allocation, one deallocation, and one case of throwing error for exceeding the limit.

4. **Add a new teaching activity called "Exercise",** this new activity must be associated with at least one course layout/instance and must have one teacher allocated for the same.
   Also, write a query to display the course layout/instance and allocation for the teacher which is affected by this new activity.

Below follows guidelines for what shall be written in the report.

- In the *Method* chapter of your report, mention which IDE(s) and other tool(s) you used and explain how you proceeded and reasoned when writing the program.
  Do not explain the result of each step you took, only explain the steps themselves.
- In the *Result* chapter of your report, briefly explain the program and in particular explain ACID transaction handling.
  Include links to your git repository, and make sure the repository is public.
  Also include a printout of a sample run.
  The git repository must also contain the scripts that create the database and insert data.
  It shall be possible to test your solution by executing first the script that creates the database, then the script that inserts data, and finally execute your program.
- The *Discussion* chapter of your report must include a **relevant and extensive** evaluation of your program.
  Suggested assessment criteria are found in seminar3-assessment-criteria.pdf, you do not have to cover them all.
  These same criteria will also be used to grade your project report.

Higher Grade Part, Task B (gives 10p, since solving this task means also task A is solved)

This task is identical to task A, but the program being developed must also be well-designed and have a properly layered architecture.
The required level of design and architecture is that of the JDBC bank example at the page Database Applications.
The following must be shown in the discussion chapter of the report.

- The code must be easy to understand. This is of course subjective, what is required is to show that you have tried sufficiently to make the code easy to understand.
- The MVC and Layer patterns must be used correctly.
  There must be enough layers, packages and classes. Neither controller nor model are allowed to contain any code related to the view (input or output).
  Also, the integration layer (the DAO) is only allowed to contain methods that create, read, update or delete rows in the database.
  There must not be any logic at all in the integration layer.
  As an example, this means you're not allowed to have a method in a DAO that checks if a teacher is not fully loaded (having 4 courses), then checks if the teacher can be allocated teaching activities from some other course instances etc.
  Instead, to handle this scenario, the controller must first call a method in a DAO that reads the allocations, then the controller checks if the teacher is available to take more activities, then finally calls a DAO method that creates an allocation.
- There must not be any duplicated code.