Examination Information

This folder contains multiple files, which you are going to fill out with the solutions to each assignment. This file describes the tasks you must perform, while the other files are referred to via the tasks described below. Each task has a percentage that indicates how much the task weighs toward your grade for the examination. **Remember that all SQL submitted at this exam must run in PostgreSQL – other dialects are unacceptable**. Mind the time you use on each task (each task has a time recommendation attached), but remember there are multiple subtasks for each task! The following sections describe the tasks you need to perform. **READ the entire document before solving the assignments!**

Each task can have subtasks – remember to read them all before you start. The tasks present in this exam:

- Task 1
  - Subtask 1 – ER and EER diagrams
  - Subtask 2 – Mapping to Tables
  - Subtask 3 – Querying the Database
- Task 2
  - Subtask 1 - Normalization
- Task 3
  - Subtask 1 – Database Comparisons
- Task 4
  - Subtask 1 – JSON objects
- Submission

Task 1: Relational Databases – Part 1 (3 subtasks) (60% ~ 72 min)

Subtask 1: ER and EER diagrams (30 min)

This subtask concerns generating an ER/EER diagram from a textual context. For this task, make sure you either open the Task1Subtask1.drawio file (if you have the offline version of draw.io) or download your diagram and override the Task1Subtask1.drawio file. If you use other tools, export it into a commonly readable format such as PNG, JPG, or PDF, and call the file Task1Subtask1 when done. You are allowed to draw the diagram by hand and attach a picture of it instead but be sure everything is easily readable, as we cannot grade anything we cannot read.

The text below is the output of an interview. This is your only documentation, so if something needs to be clarified, you must decide how to understand it - make a note in the diagram explaining your reasoning. Remember to be as precise as possible and include all the types of descriptions you have learned (some might not apply to the case, though). This includes strong and weak entities, inheritance, relationships, cardinalities, keys, etc.

Interview Results (in Danish):

Stellar Airframes Inc. blev grundlagt i 1992 af luftfartsingeniør Dr. Elizabeth Harper og entreprenør Richard Lawson i Seattle, Washington. Oprindeligt et lille værksted, der specialiserede sig i specialdele til eksperimentelle fly, fik virksomheden hurtigt anerkendelse for sin innovative brug af lette kompositmaterialer. I begyndelsen af 2000'erne sikrede de kontrakter med store flyselskaber og militærkunder for hvem de i dag bygger komplette fly.

Stellar Airframes Inc. skal bruge databasesystem til at understøtte deres flykonstruktionsindsats. Denne database skal håndtere detaljeret information relateret til forskellige aspekter af flykonstruktion. Her skal man kunne holde styr på Materialer, herunder deres navn, leverandør, leveringstid, og pris. Disse Materialer kan derefter samles til Komponenter som er skabt af flere Materialer. Det er vigtigt at man kan holde styr på hvilke Materialer og hvor mange der er blevet brugt til et enkelt Komponent. Derefter kommer den faktiske konstruktion af flyet. Flyet er konstrueret af et antal Komponenter, samt Medarbejdere der udfører arbejdet. Medarbejderne har en løn og et timeantal de bruger. Det vigtigt at det er muligt for firmaet at udregne hvad hver enkelte fly har kostet at konstruere så de kan sikre de stadig tjener penge på deres Konstruktioner. Generelt omtales fly som Konstruktioner i virksomheden.

Vedsiden af ovenstående, skal de tekniske tegnere (en Medarbejder) kunne uploade deres tegninger til databasen, for at de altid kan finde Dokumenterne for hvordan Konstruktioner og Komponenter skal samles. Dette er simple dokumenter, men der er brug for at kunne holde styr på hvilken medarbejder der har uploadet filen, samt holde en versionering tilgængelig af denne fil. Dermed skal man altid kunne gå tilbage og finde en teknisk tegning der er ældre selv om der er oprettet en ny. Komponenter der bliver bygget, peger altid på hvilken Teknisk Tegning der er blevet brugt til at bygge den, det samme gælder Konstruktioner.

Subtask 2: Mapping to Tables (30 min)

This subtask concerns mapping an ER diagram (or EER) to tables in a database.

For this task, make sure you open the Task1Subtask2and3.sql file. The file is empty, and you must fill it out with the database to create the script for the ER diagram you made in ***Task 1 Subtask 1.*** If you could not create the diagram, use the interview result text as the basis for your tables.

Subtask 3: Querying a database (12 min)

This subtask concerns itself with querying databases. Open Task1Subtask2and3.sql, and add the following queries at the end of the file:

1. Add at least one row to each table created in subtask 2.
2. Create a query summarizing the price of the first construction in your database.

Task 2: Relational Databases Part 2 – Normalization. (20% ~ 24 min)

This task concerns itself with normalizing a given dataset (the Excel image below) to a database schema. The expected format is shorthand, as described below.

The output of this task should be saved in the Task2.txt file, which is currently empty. Open the Task2.txt file directly and save it to that file using your editor. Feel free to write comments.

Real Estate Listings

ListingID	PropertyAddress	City	ZipCode	SellerName	SellerContact	BuyerName	AgentName	AgentContact	SalesDate	SalePrice	PropertyType	Bedrooms	Bathrooms	ListingDate
1	123 Main St	New York, NY	10001	John Doe	555-1234	Alice Smith, 555-5678	Robert Johnson	555-8765	1/15/23	950000	Apartment	3	2	12/1/22
2	456 Oak Ave	Los Angeles, CA	90001	Jane Roe	555-2345	Bob Johnson, Forgot phone	Emily Davis	555-9876	2/20/23	1200000	House	4	3	11/15/22
3	789 Pine Rd	Chicago, IL	60601	James Smith	555-3456	Carol White, 555-7890	David Brown	555-0987	3/10/23	850000	Condo	2	1	1/5/23

Tasks are as follows (read all steps first!):

- Normalize the dataset to at least the third normal form.
- Write the normalized schema in Task2.txt using the template/shorthand form (for example ***TableName(PK id, prop1, prop2, FK prop3)***)
    - Do not create SQL for this; it will take too much time.
- **Remember** the relationships with primary and foreign keys. It is helpful to show the relationships in shorthand format at the end (for example, **“Table1” M -> N “Table2”)**.
- Feel free to write notes/reasoning for your choices in the document.

Task 3: Database Comparisons (10% ~ 12 min)

Open the file called “Task3.txt” from the zip file. The file is empty. Write a few paragraphs that explain the following:

- What are the commonalities and differences between these three database types?
    - Relational Database
    - Time Series Database
    - Key-Value Based Databases

Task 4: JSON files (10% ~ 12 min)

Open the file called “Task4.json” from the zip file. The file is empty. Create a JSON file representing a travel itinerary with the following structure:

1. **Destination**
    - **destination**: A string for the travel destination (e.g., "Paris, France").
2. **Travel Dates**
    - **travel_dates**: An object with:
        - **departure**: A date string (YYYY-MM-DD) for the departure date (e.g., "2024-06-15").
        - **return**: A date string (YYYY-MM-DD) for the return date (e.g., "2024-06-25").
3. **Accommodations**
    - **accommodations**: An array of objects, each with:
        - **name**: A string for the accommodation name (e.g., "Hotel Le Meurice").
        - **address**: A string for the accommodation address (e.g., "228 Rue de Rivoli, 75001 Paris, France").
        - **check_in**: A date-time string (YYYY-MM-DDTHH:MM:SSZ) for check-in (e.g., "2024-06-15T14:00:00Z").
        - **check_out**: A date-time string (YYYY-MM-DDTHH:MM:SSZ) for check-out (e.g., "2024-06-25T11:00:00Z").
        - **pre-paid**: A Boolean of true or false.
        - **price**: An integer/number describing the combined price of the accommodation (e.g., 1000 or 5000).
4. **Activities**
    - **activities**: An array of objects, each with:
        - **activity_name**: A string for the activity name (e.g., "Eiffel Tower Visit").
        - **date**: A date string (YYYY-MM-DD) for the activity date (e.g., "2024-06-16").
        - **time**: A time string (HH:MM:SS) for the activity time (e.g., "10:00:00").
        - **location**: A string for the activity location (e.g., "Eiffel Tower, Paris, France").

Ensure your JSON file follows this structure. Remember to fill in the information in the objects, as it helps me check their validity.

Submission

To submit the assignment, make sure you have a folder containing the following files:

- Task1Subtask1.drawio (alternatively .jpg, .png, or .pdf)
- Task1Subtask2and3.sql
- Task2.txt
- Task3.txt
- Task4.json

Instructions:

1. Check that each file contains your work.
2. Zip the files without the folder containing them (mark all files and zip them together).
3. Name the zip file after your SDU Username, such as “abcd17.zip.
4. Go to the assignment location from where you downloaded the exam.
5. Upload the zip file.
6. For your security, download the file again and unzip it to verify that the version stored on the server works as intended (and that you included the correct files – yes, this happens!).