AY2019/2020 Semester 2: CS2102 Midterm test (20 marks)

- This is a NON-SECURE WITH INTERNET Examplify test.
- The test consists of 11 questions. You are required to answer all questions.
- Question 1 has no marks; you **must** type your full name into the answer box to declare that you will abide by the NUS Code of Student Conduct.
- The maximum score for Questions 2 to 11 is 2 marks each, and the maximum score for the test is 20 marks.
- The duration of the test is **90 minutes** and you must submit your answers using Examplify by **1:45pm**.
- As the test will be auto-graded, it is **important** that your entered answers conform to the requirements specified under **Test Instructions**.
- The instructions given here can be viewed anytime either via **Exam Controls -> Exam Notice** or via **Exam Controls -> Exam Attachment** (to view as it as a pdf attachment).

Test Cases

The password to decrypt cs2102-midterm.zip is **CS2102__16798402__Midterm__0327?**

Examplify Instructions

- You can download the test only once.
- Once you have submitted the test, you will not be able to make any changes to your answers. It is important that you ensure that you've answered all the questions and that the entered answers are indeed the intended answers for the specific questions before you click the 'Submit' button.

- **Troubleshooting** If you encounter any technical problems while using Examplify, please seek help from Mr. Uday using **Microsoft Teams' Examplify Tech Channel**.

Test Instructions

As this test will be **auto-graded**, it is important that your SQL answers conform to the requirements specified below. If the grading program detects that an entered answer does not conform to the specified requirements, that answer will receive 0 marks even if there is some correct solution embedded within the entered answer.

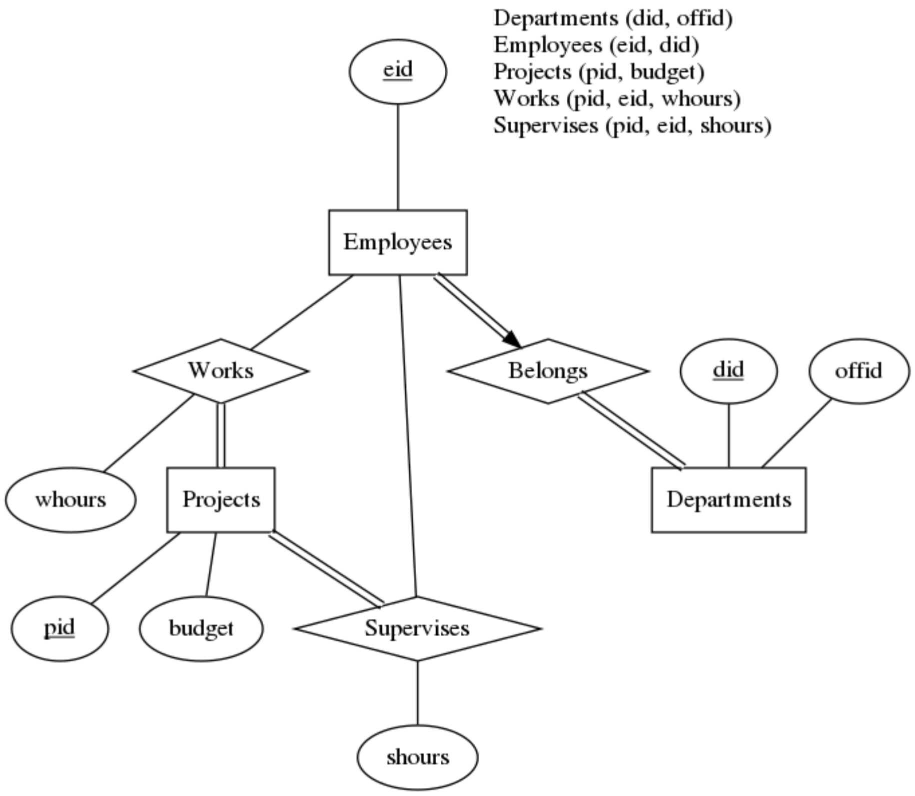
- For each question, you are to write a **single CREATE VIEW SQL statement** to answer the question. You MUST use the view schema provided for each question without changing any part of the view schema (i.e., view name, column names, or the order of the columns). For example, the provided view schema for Question 1 is "v1 (eid)", and if your answer to this question is the query "SELECT eid FROM Employees", then you must enter your answer in the answer box as follows:

```
CREATE VIEW v1 (eid) AS
SELECT eid
FROM Employees
;
```

- Each CREATE view statement must terminate with a semicolon.
- Each answer must be a syntactically valid SQL query without any typographical errors: a SQL answer that is syntactically invalid or contains very minor typographical error will receive 0 marks even if the answer is semantically correct.
- For each question, your answer view must not contain any duplicate records.
- You must not enter any extraneous text for your answer (e.g., "My answer is: CREATE VIEW v1 (eid) AS SELECT eid FROM Employees;") or enter multiple solutions (where your answer ends up being multiple SQL statements).
- Each question must be answered independently of other questions: the answer for a question must not refer to any other view that is created for another question.
- You are allowed to create the answer view using a single CTE statement (defining possibly multiple temporary tables). If your answer uses a CTE statement to create temporary table(s), you **must not** use any of the following table names:
  - Names that are used by the application's schema
  - Names that are used by the answer views: v1, v2, . . . , v9, v10, v11.
  - Any table name that begins with the prefix **zz**.
- Your answers must not use SQL constructs that are not covered in class (e.g., window functions).
- Your answers must be executable on PostgreSQL: your answers must not use constructs that are not supported in PostgreSQL.
- It is not necessary to write comments in your answer. If you choose to write comments, you should not use SQL comments that begin with two hyphen characters to avoid a potential copy-and-paste problem that results in a query being copied over as a single line with SQL code being commented out (e.g., "CREATE VIEW v1(eid) AS SELECT eid – yada yada FROM Employees;"). Using C-style SQL comments are fine (e.g., "CREATE VIEW v1(eid) AS SELECT eid /* yada yada */ FROM Employees;").

Database Schema

The questions are based on the following application about a company's operation. Its ER data model is shown below with the following constraints.



1. The company has at least one **department** (identified by **did**). Each department is located in an office identified by **offid**. The attribute offid has non-null values. Each department has one or more employees.
2. Each **employee** (identified by **eid**) must belong to exactly one department.
3. Each **project** (identified by **pid**) has a budget given by **budget**. The attribute budget has non-null values and its unit is in millions of dollars; e.g., a budget value of 10 means 10 million dollars.
4. Each employee can work on 0 or more projects.
5. There must be at least one employee working on each project.
6. For each project P that an employee E works on, the number of hours per week that E spends working on P is given by **whours**. The attribute whours has non-null values.
7. Each employee can supervise 0 or more projects.
8. There must be at least one employee supervising each project.
9. For each project P that an employee E supervises, the number of hours per week that E spends supervising P is given by **shours**. The attribute shours has non-null values.
10. An employee who supervises a project may or may not work on that project. An employee who works on a project may or may not supervise that project.

The above image showing the ER model and an overview of the relational schema is also viewable as a pdf attachment for each question.

Relational Schema

The following is the relational schema for this application.

```
CREATE TABLE Departments (
    did             INTEGER,
    offid           INTEGER NOT NULL,
    PRIMARY KEY (did)
);

CREATE TABLE Employees (
    eid             INTEGER,
    did             INTEGER NOT NULL,
    PRIMARY KEY (eid),
    FOREIGN KEY (did) REFERENCES Departments
);

CREATE TABLE Projects (
    pid             INTEGER,
    budget          INTEGER NOT NULL,
    PRIMARY KEY (pid)
);


CREATE TABLE Works (
    pid             INTEGER,
    eid             INTEGER,
    whours          INTEGER NOT NULL,
    PRIMARY KEY (pid,eid),
    FOREIGN KEY (eid) REFERENCES Employees,
    FOREIGN KEY (pid) REFERENCES Projects
);

CREATE TABLE Supervises (
    pid             INTEGER,
    eid             INTEGER,
    shours          INTEGER NOT NULL,
    PRIMARY KEY (pid,eid),
    FOREIGN KEY (eid) REFERENCES Employees,
    FOREIGN KEY (pid) REFERENCES Projects
);
```