1 Instructions

Assignment 3 will be done using `Examplify` and is due on **March 8 (Friday) at 11:59pm**. To start the assignment on `Examplify`, you need to use the assignment password "**march81159pm**". Once you start the assignment with `Examplify`, you are given a time limit of 1000 minutes (about 16.6 hours) to complete and submit the assignment. You can suspend the assignment (as well as the timer) by clicking on "Suspend Exam" from the "Exam Controls" menu. To resume the suspended assignment, you need to enter the assignment password. You can find screenshots on how to suspend and resume the assignment at https://wiki.nus.edu.sg/pages/viewpage.action?pageId=230556734.

This assignment consists of 10 one-mark questions. You can answer the questions in any order and you may edit any of your entered answers. However, once you have submitted the assignment, you will not be able to make any changes to your answers. Therefore, 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. You can find screenshots on how to submit the assignment at https://wiki.nus.edu.sg/pages/viewpage.action?pageId=187598226.

It is important that you submit your assignment by the assignment deadline (March 8 at 11:59pm) as you will not be able to submit after the deadline even if the Examplify timer has not counted down to zero.

You must answer each question with a SQL statement that satisfies the following conditions:

1. For each question, you are to write a single `CREATE VIEW` SQL statement to answer the question. For example, if your answer to Question 1 is the SQL query $Q1$ given by "SELECT eid, count(*) FROM Works GROUP BY eid;", then your answer must be entered in the form "CREATE VIEW v1 (columnList) AS Q1;" as follows:

```
CREATE VIEW v1 (eid, num) AS
SELECT eid, count(*)
FROM Works
GROUP BY eid
;
```

   The schema of the view definition (i.e., the name of the view and the names of its columns) will be provided to you for each question. You must use the provided view schema for each question and must not modify it in any way.

2. Since each answer must be a valid, single SQL statement of the form "CREATE VIEW viewName (columnList) AS Q;", it is illegal to create additional view within Q. However, Q could be a single CTE statement (using multiple temporary tables).

3. Each question must be answered independently of other questions: the answer for a question must not refer to any other view that created is for another question.

4. The query answer to each question must not have any duplicate records.

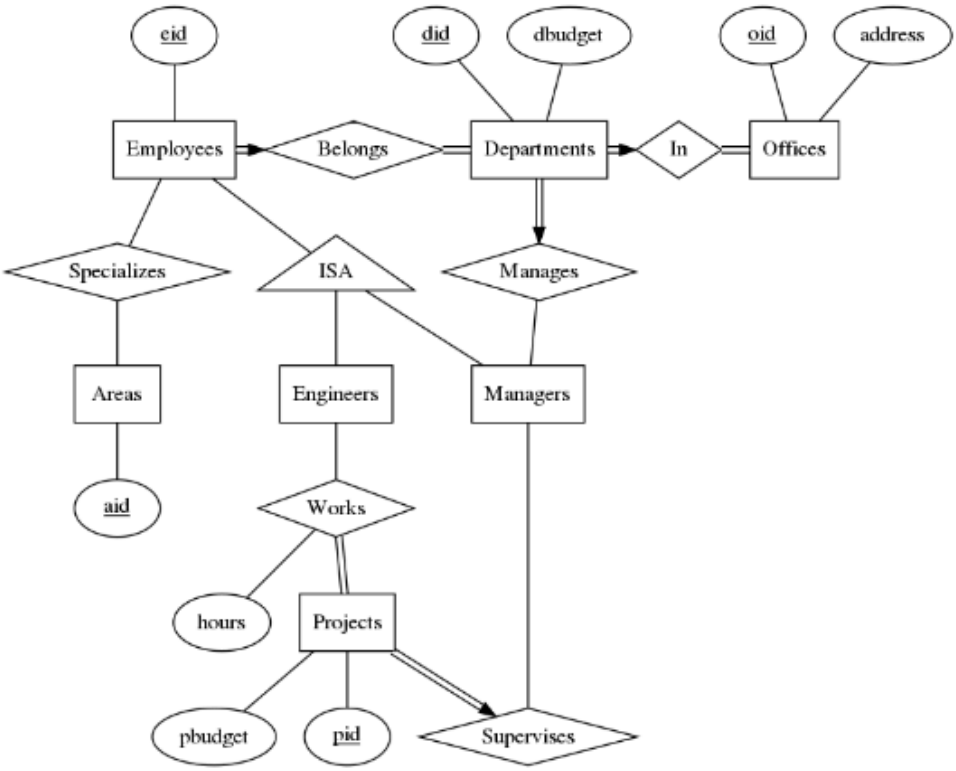
5. As your submitted answers will be auto-graded, it is important that each entered answer is a valid, single CREATE VIEW statement that ends with a semicolon. You should not enter any extraneous text (e.g., "My answer is: CREATE VIEW v1 (e) AS SELECT eid FROM Employees;") or enter multiple solutions (where your answer ends up being multiple SQL statements). If the grading program detects that an entered answer is not a valid, single CREATE VIEW statement, that answer will receive 0 marks even if there is some correct solution embedded within the entered answer.

6. If your answer uses a CTE statement to create temporary tables, you must not use any of the following table names:

   - Names that are used by the application's schema (e.g., Offices, Departments)
   - Names used by the 10 answer views: `v1`, `v2`, ..., `v9`, `v10`.
   - Any table name that begins with the prefix `zz`.

2 Database Schema

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



The company has at least one office. Each office (identified by `oid` with location specified by `address`) consists of one or more departments. Each department (identified by `did` with a budget `dbudget`) is located in one office and has one or more employees. Each employee (identified by `eid`) must belong to a department. The application focuses on two subclasses of employees: engineers and managers. An employee can be neither an engineer nor a manager, and no employee can be both an engineer and a manager. Each employee can specialize in 0 or more areas (identified by `aid`). Each department must be managed by exactly one manager, and each manager can manage 0 or more departments. Each engineer can work in 0 or more projects, and there must be at least one engineer working in each project. For each project P that an engineer E works on, the number of hours per week that E spends on P is given by `hours`. Each project (identified by `pid` with a budget `pbudget`) must be supervised by exactly one manager. A manager can supervise 0 or more projects. Attributes `hours`, `dbudget` and `pbudget` have non-null values.

2.1 Relational Schema

The following is the relational schema for this application. This information is also from available from the `init.sql` script described in Section 3.

```
CREATE TABLE Offices (
    oid             INTEGER,
    address         VARCHAR(60),
    PRIMARY KEY (oid)
);

-- eid = eid of department's manager
CREATE TABLE Departments (
    did             INTEGER,
    dbudget         INTEGER NOT NULL,
    oid             INTEGER NOT NULL,
    eid             INTEGER NOT NULL,
    PRIMARY KEY (did),
    FOREIGN KEY (oid) REFERENCES Offices
);

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

CREATE TABLE Engineers (
    eid             INTEGER,
    PRIMARY KEY (eid),
    FOREIGN KEY (eid) REFERENCES Employees
);

CREATE TABLE Managers (
    eid             INTEGER,
    PRIMARY KEY (eid),
    FOREIGN KEY (eid) REFERENCES Employees
);
```

```
-- eid = eid of project's supervisor
CREATE TABLE Projects (
    pid             INTEGER,
    pbudget             INTEGER NOT NULL,
    eid             INTEGER NOT NULL,
   PRIMARY KEY (pid),
   FOREIGN KEY (eid) REFERENCES Managers
);


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

CREATE TABLE Areas (
    aid             VARCHAR(5),
   PRIMARY KEY (aid)
);

CREATE TABLE Specializes (
    eid             INTEGER,
    aid             VARCHAR(5),
   PRIMARY KEY (eid,aid),
   FOREIGN KEY (eid) REFERENCES Employees,
   FOREIGN KEY (aid) REFERENCES Areas
);
```

3 Testing Your Answers

To test your answers using some database instances, download the encrypted file **cs2102-assign3.zip** from `IVLE Assignment Workbin`. Use the password **2102sql** to unzip this file.

The unzipped **cs2102-assign3** directory contains the following files:

- init.sql - SQL script to initialize the database with tables
- answers.sql - SQL script for your answers to the 10 questions
- status1.sql, . . ., status10.sql - SQL scripts to report a summary of test comparisons for each question
- test1.sql, . . ., test10.sql - SQL scripts to report the details of test comparisons for each question
- set-psql.txt - script to customize `psql`
- Backup/ - contains a backup copy of the original answers.sql script
- Misc/ - optional SQL scripts for loading data into the database for each test case

3.1 answers.sql

The only file that you need to edit for the purpose of testing is the file named `answers.sql`. This file consists of 10 `CREATE VIEW` statements representing your answers for the 10 questions. For example, the current view definition for Question 2 is given as

```
– – Question 2
create view v2 (eid, num) as
select 1,1
;
```

You must not modify the schema definition; i.e., the line "create view v2 (eid, num) as". If your answer for this question is "select eid, count(*) from Works group by eid", then you should replace the line "select 1,1" with your answer as follows:

```
– – Question 2
create view v2 (eid, num) as
select eid, count(*)
from Works
group by eid
;
```

3.2 init.sql

For this assignment, instead of using the default database named **postgres**, which might contain other tables previously created that could conflict with the assignment's tables, you will create and connect to a new database named **assign3**. After you've connected to the new database, execute the `init.sql` SQL script to create the application's tables and additional solution tables that will be used for testing.

psql users: If you're currently running `psql` and connected to the `postgres` database, execute the following commands to create and connect to the new database named `assign3`, followed by executing the `init.sql` script.

```
postgres=# create database assign3;
CREATE DATABASE
postgres=#
postgres=# \c assign3
You are now connected to database "assign3" as user "alice".
assign3=#
assign3=# \i init.sql
```

To execute a shell/commandline command from `psql`, you can use prefix the command with `psql`'s meta-command \!. For example, to show the current working directory, execute the command "\! pwd" (for MacOS/Linux) or "\! cd" (for Windows).

DBeaver users:

1. Create the new database named `assign3` as follows. In the "Database Navigator" window, right click postgres → Create New Database to activate the "Create database" window. Enter the value "assign3" for `Database name` and click "OK".
2. In the "Database Navigator" window, click on "assign3".
3. To load the SQL script `init.sql`, right click SQL Editor → Load SQL Script, navigate to the `cs2102-assign3` directory, and select `init.sql`.
4. To execute the loaded script `init.sql`, move to the SQL editor window and press ALT+X.

3.3 answers.sql, statusN.sql & testN.sql

Suppose that you have edited `answers.sql` with a view definition for v2 (i.e., your answer for Question 2). To test your answer for Question 2, perform the following steps:

1. First, execute the script `answers.sql` to install your answer's view definition for Question 2.
2. Next, execute the script `status2.sql` to obtain a summary of the test outcomes for the 13 provided test cases. The following example output indicates that your answer did not pass any of the test cases. Specifically, for test case 11, your answer's output is different from the correct solution as there are 4 correct tuples that are missing from your output and your output contains 3 extra tuples that are absent from the correct output.

```
            Summary of testing query 2
-------------------------------------------------
 Test case 1 - INCORRECT: MISSING = 3, EXTRA = 2
 Test case 2 - INCORRECT: MISSING = 5, EXTRA = 4
 Test case 3 - INCORRECT: MISSING = 5, EXTRA = 4
 Test case 4 - INCORRECT: MISSING = 5, EXTRA = 4
 Test case 5 - INCORRECT: MISSING = 5, EXTRA = 4
 Test case 6 - INCORRECT: MISSING = 7, EXTRA = 6
 Test case 7 - INCORRECT: MISSING = 6, EXTRA = 5
 Test case 8 - INCORRECT: MISSING = 6, EXTRA = 5
 Test case 9 - INCORRECT: MISSING = 5, EXTRA = 4
 Test case 10 - INCORRECT: MISSING = 5, EXTRA = 4
 Test case 11 - INCORRECT: MISSING = 4, EXTRA = 3
 Test case 12 - INCORRECT: MISSING = 3, EXTRA = 2
 Test case 13 - INCORRECT: MISSING = 3, EXTRA = 2
(13 rows)
```

3. To see the detailed test comparisons for your answer to Question 2, execute the script `test2.sql`. The following example output shows the detailed comparison for test case 11 which indicates that there are 4 correct tuples in your output (marked as "OK"), 3 extra tuples in your output (marked as "EXTRA"), and 4 correct tuples that are mising in your output (marked as "MISSING").

```
 eid | num | Test case 11
-----+-----+--------------
 108 |   1 | OK
 105 |   1 | OK
 109 |   1 | OK
 115 |   1 | OK
 113 |   1 | EXTRA
 114 |   1 | EXTRA
 101 |   3 | EXTRA
 101 |   4 | MISSING
 114 |   2 | MISSING
 106 |   0 | MISSING
 113 |   2 | MISSING
(11 rows)
```

psql users: As the default mode of `psql` is quite verbose (it echos each of the executed commmands), you can configure `psql` to be quieter by executing the `psql` commands in `psql-set.txt`. The execution of `set-psql.txt` needs only be done once for each `psql` session. For more information on `psql`'s configuration options/commands, refer to https://www.postgresql.org/docs/current/app-psql.html.

```
assign3=# \i set-psql.txt
assign3=# \i answers.sql
psql:answers.sql:2: NOTICE: view "zzanswer" does not exist, skipping
assign3=#
assign3=# \i status2.sql
psql:status2.sql:70: NOTICE: view "zzanswer" does not exist, skipping
Summary of testing query 2
......
assign3=#
assign3=# \i test2.sql
......
assign3=#
```

You can ignore `psql`'s warnings about missing views.

DBeaver users: Follow similiar instructions given in Section 3.2 to load and execute the required scripts. Ignore the file `set-psql.txt` which is relevant only for `psql`.

3.4 Debugging Queries

Suppose that your query for Question 2 is incorrect for test case 10. If you want to issue queries on test case 10's database for debugging, you can load its database by executing the SQL script Misc/load-db10.sql.

3.5 Re-initializing the Database

Should your `assign3` database be corrupted and you want to restore it to its initial state, perform the following steps. First, if you are currently connected to `assign3`, you need to disconnect from `assign3` by connecting to the default database named `postgres`. For `psql` users, execute the command "\c postgres" followed by "drop database assign3;". Follow the instructions in Section 3.2 to re-create and re-initialize `assign3`.

3.6 Important Points

1. Your answers in **answers.sql** are only for the purpose of testing. As this file will not be submitted, it is important that you remember to copy each of the tested answers from this file to the appropriate question's answer in `Examplify`. Furthermore, you must copy the entire view definition as a valid, single SQL statement. Taking Question 2 as an example, your answer for Question 2 in `Examplify` should start with the line "CREATE VIEW v2 (eid, num) AS" and end with the line containing the semicolon.

2. It is important to remember to execute the `answers.sql` script (step 1 in Section 3.3) before testing your answers. Otherwise, if you've modified `answers.sql` after your last execution of `answers.sql`, you will be testing your old answers (installed by the last execution of `answers.sql`) and not your revised answers in `answers.sql`.

3. As usual, the test cases are meant to help catch certain errors in your answers, and an answer that passes all the provided test cases does not necessarily mean that the answer is indeed correct. The grading of this assignment will be using both the provided test cases as well as additional test cases.