This assignment contains two parts, Part A and Part B. You will need to submit both parts through Gradescope. For Part A, you will need to print pages 2 through 4, handwrite your solutions, and then scan the document. For Part B, you will need to submit queries written in SQL, which will be autograded on Gradescope.

**Database schema for Parts A and B**

Consider the schema below.

```
Emp(eid: integer, ename: string, age: integer, salary: real)
Works(eid: integer, did: integer, pct_time: integer)
Dept(did: integer, dname: string, budget: real, managerid: integer)
```

The Emp and Dept relations represent employee entities and department entities with their respective attributes. The domain of each field is listed after the field name. The Works relation represents the relationships of an employee working in a department. In this schema, an employee can work in more than one department; the pct_time field shows the percentage of time that a given employee works in a given department. We assume that pct_time is always greater than 0.
Part A: Relational Algebra

You will need to write the following queries in relational algebra. Make sure that the final result contains the attributes (and only those attributes) specified by the question. Please make sure you write clearly. We cannot grade what we cannot read. If you wish, you can use the rename operator (\( \rho \)) to assign a name to an intermediate result, if that helps simplify the specification of your query. For example:

\[
\rho(R_1, \sigma_{age>50}Emp) \\
\pi_{ename}R_1
\]

You may omit the join condition when it is clear from the context. For example, you can write \( Emp \bowtie Works \) instead of \( Emp \bowtie_{did=did} Works \).

Submission instructions:

- Print pages 2–4 of this document, and handwrite your solutions in the corresponding spaces. Please write clearly!
- Scan your solutions into a single PDF. Make sure that the quality of the PDF is good and your writing is clear and readable.
- Upload your PDF to the corresponding assignment on Gradescope.
- Check here for helpful tips on scanning and uploading assignments to Gradescope: [https://gradescope.com/help#help-center-section-student-workflow](https://gradescope.com/help#help-center-section-student-workflow)

1. Retrieve the name and age of each employee who works in the ‘Legal’ department.

2. Retrieve the names of the employees who work in a department with a budget exceeding $1,000,000.
3. Retrieve the names of the employees who work in the ‘Software’ department more than 60% of time.

4. Retrieve the names of the employees who work in the ‘Software’ department more than 60% of time, or the ‘Hardware’ department more than 20% of the time.

5. Retrieve the names of the employees who work in the ‘Software’ department more than 60% of time, and the ‘Hardware’ department more than 20% of the time.
6. Retrieve the dids of departments with at least two employees.

7. Retrieve the names of the employees who work in all departments with budget greater than 12 million.

8. Retrieve the names of the employees who work only in departments with budget less than 1 million.
Part B: SQL

You will need to write SQL queries on the provided database schema. We have provided a sample dataset for the schema, so you can experiment with the queries on your local machine.

Setup

- Set up a database system on your machine. You may set up PostgreSQL, by following our instructions here:
  

  Or, you may start with a simpler DBMS, sqlite, which is typically pre-installed in OSX and Linux distributions. You may also download it here:

  https://www.sqlite.org/download.html

- Create a new database, called hw1. In postgres, you can use the command CREATE DATABASE hw1; once created, you can connect to your database with \c hw1. In sqlite, you can create the database by starting sqlite with the command sqlite3 hw1.db

- Create all the necessary tables. For example:

  ```
  create table Emp(
    eid integer primary key,
    ename varchar(30), age integer, salary real
  );
  ```

  Make sure you use the exact same names for the attributes as indicated in the schema specification on the first page of this document, otherwise your queries will not work with our autograder.

- Download the sample data from the course webpage:
  
  http://avid.cs.umass.edu/courses/645/s2018/hw/hw1/hw1data.zip

  The zip file contains 3 comma-separated files. You will use those to populate your tables. To do so, look up the `copy` command for postgres, and `.import` for sqlite.

- Once you import the data, you are ready to write SQL queries. When you are happy with each query, copy it into a file named Qx.sql where x is the number of the corresponding query. For example, you should store query Q1 in Q1.sql. **Do not include more than one query into a single file.**

- You will need to submit all of your query files to the corresponding assignment on Gradescope.
Submission instructions

- You will need to submit your work through Gradescope. When you upload your submission, the autograder will check it and will tell you which queries you got right and which you got wrong. You may submit as many times as you like until the deadline. Your final submission will be the one to be counted; make sure that it includes answers to all queries.

- When you select upload submission, you can drag and drop all your .sql files into the submission window, or upload them all together into a .zip file. You don’t have to submit all queries each time, but you can check any number of queries that you have ready.

- To help you get started, we include a sample submission file Q0.sql, which includes a query that retrieves all data from the Dept table: [http://avid.cs.umass.edu/courses/645/s2018/hw/hw1/Q0.sql](http://avid.cs.umass.edu/courses/645/s2018/hw/hw1/Q0.sql) Start by dragging and dropping that one file into the submission window.

- In the autograder results, you will note that the first test (on query Q0) passes and the others fail.
• To complete the assignment, you need to create and upload (in a single submission) files Q1.sql through Q12.sql. Each query needs to return the appropriate fields, and produce results in the requested order.

• **Important note:** Different DBMSs may demonstrate small differences in the acceptable SQL syntax. If your query works locally on your computer, but fails on the autograder, it may be due to such an issue. If you are using sqlite locally, you shouldn’t encounter this issue, as the autograder uses sqlite as its backend DB. Sqlite does not accept parentheses adjacent to `UNION` and `INTERSECT`. For example, instead of `(SELECT * FROM R) UNION (SELECT * FROM S)`, you would need to write `SELECT * FROM R UNION SELECT * FROM S`. If this does not resolve your problem and you are confident that your query is correct, please post a private question to the TAs on Piazza, indicating your query and the DBMS you are using.

• **Important note:** Submission will remain open for a week past the due date to accommodate grace days and late submissions. **DO NOT** upload submissions past the deadline if you do not wish to use up grace days (or receive a late submission penalty past the grace days). If you upload past the due date, it will count against your grace days quota. As a reminder, grace days count in 24 hour periods. So, submitting a few minutes after the deadline counts as a full grace day used.

• **DO NOT** try to trick the autograder with queries that drop tables and otherwise modify the data. The autograder already accounts for such tricks, and submissions will also be verified manually.

• **DO NOT** try to trick the autograder with queries that try to “fake” the right answer. For example, if through manual reasoning you determine that the answer to a particular query on the sample data should be “Bob”, submitting the query `SELECT ‘Bob’` will not work. The autograder uses a slightly modified dataset to verify your queries.
Write the following SQL queries. **Make sure you eliminate duplicates in the results of all queries.**

1. Q1: Retrieve the name and age of each employee who works in the ‘Legal’ department. The output should list the name first, then the age, and order the results first by name and then by age. Submit the query in file Q1.sql.

2. Q2: Retrieve the names of employees who work in a department with a budget exceeding $1,000,000. Print the output in sorted order of name. Submit the query in file Q2.sql.

3. Q3: Retrieve the names of the employees who work in the ‘Software’ department more than 60% of time. Print the output in sorted order of name. Submit the query in file Q3.sql.

4. Q4: Retrieve the names of the employees who work in the ‘Software’ department more than 60% of time or the ‘Hardware’ department more than 20% of time. Print the output in sorted order of name. Submit the query in file Q4.sql.

5. Q5: Retrieve the names of the employees who work in the ‘Software’ department more than 60% of time and the ‘Hardware’ department more than 20% of time. Print the output in sorted order of name. Submit the query in file Q5.sql.

6. Q6: Retrieve the dids of departments with **at least two** employees. Print the output in sorted order of did. Submit the query in file Q6.sql.
7. Q7: Retrieve the names of employees that work in all departments with budget greater than 12 million. Print the output in sorted order of name. Submit the query in file Q7.sql.

8. Q8: Retrieve the names of employees that work only in the departments with budget less than 1 million. Print the output in sorted order of name. Submit the query in file Q8.sql.

9. Q9: Retrieve the average salary of employees for each department. The output should list each department id first, followed by the average salary for that department. Print the output in sorted order of department id. (Note: to compute the average salary for each department, simply consider all the employees who work for it.) Submit the query in file Q9.sql.

10. Q10: Retrieve the names of managers who manage the departments with the largest budget. Print the output in sorted order of name. Submit the query in file Q10.sql.

11. Q11: If a manager manages more than one department, s/he controls the sum of all the budgets for those departments. Retrieve the managerids of managers who control more than 5 million. Print the output in order of managerid. Submit the query in file Q11.sql.

12. Q12: Retrieve the names of managers who manage only departments with budgets larger than 1 million, but at least one department with budget less than 5 million. Print the output in sorted order of name. Submit the query in file Q12.sql.