COMP 3005
Assignment #1
Due: Sept. 27@11:59PM

Instruction

1. Do the assignments independently. Copying is not allowed.
2. First replace Last in the Employee table with your own last name. If your last name is not shown correctly in the result, you will get 0 for the assignment.
3. Do this assignment directly on this document and rename it with your last + first name and submit to **brightspace**. Scanned handwritten documents *won't* be accepted. Make sure your uploaded file can be opened.
4. You need to use either Openstack or install Oracle VM on your personal computing running intel chips. Macbooks with M1 chips cannot use Oracle VM. Detail about how to install and use Oracle VM can be found in the file Oracle-VM2023.docx.

Part 1 Concepts (20 marks)

Explain the following concepts based on the definitions given in the lecture notes. Different answers found online will be marked wrong. The explanation should be complete; i.e, it does not contain any concept not explained here. Each concept is 2 marks.

1. Data

   Data refers to information that can be recorded that holds meaning. They are facts that that have implied meanings such as someone's contact information. We assume someone's name and phone number to be true so it is a fact.

2. Domain

   A domain is a space that defines valid values. In the context of databases, the domain is the set of valid values for attributes. It has a name, a type, and a set of values. Values for the attributes need to fall into the specified domain in order to be correct.

3. Attribute

   An attribute is a characteristic of something. In the context of a database, an attribute is a characteristic of a relation (a table). A set attributes identifies a row in a table.

4. Key

   In the context of databases, keys are the set of attribute(s) that uniquely identifies a row in a table. They are underlined in the table to indicate that they are the key to that table. They are one way to constrain in a relational model.

5. Primary Key

   In a database, a primary key is typically a chosen key that is the smallest of all the keys in terms of size. They are used to identify tuples in a relation. When tuple(s) in relations reference a tuple(s) in other relations, they use this type of key.

6. Key integrity constraints

   This type of constraint specifics uniqueness for values of a data item. For example, in a course table, no two courses can have the same course id, or else they would not be different.

7. Entity Integrity Constraint

Entity integrity constraints state that no primary keys can be NULL. Since primary keys are used to identify unique tuples in a relation, they cannot be NULL or we have no way of identifying some tupples.

8. Referential Integrity Constraint

   Referential integrity constraint is a constraint that is applied between two relations and is used to maintain consistency between them. It basically says that if a tuple in one relation refers to a tuple in another relation, it must exist. It involves the use of a foreign key, which is an attribute that refers to the primary key in a relation. The value of the foreign key column(s) must have the same domain as the primary key that they are referencing or NULL.

9. Data Definition Language (DDL)

   DDL is the language used to specify the schema constructs. This includes the creation, modification and access control of a schema.

10. Query Language (QL)

    QL is the language to specify database queries. These provide an alternative to programming, making it faster to write the queries.

Part 2 (80 marks)

Given the following employee and project databases with three relations. Use Oracle DBMS to create, populate and query this database. Test your statements on Oracle DBMS and take necessary screenshots to show your statements executed successfully. Copy and paste the corresponding SQL statements into this assignment.

1. Use SQL Data Definition Language to create the database. You should properly define primary keys and foreign keys with other 8 unique integrity constraints of different kinds. You can use check more than once. **Boldface** every integrity constraints. (10)
2. Use SQL Data Manipulation Language to populate the database with the data given in the tables. (10)
3. Use Relational Algebra (ALG) to express the following queries based on the above database. Submit your ALG expressions for each query as well as the query result. Each query is 4 marks and the result is 2 marks (60)

Employee

E#	Name	Age	Manager
E1	Adams	50	
E2	Blake	40	E1
E3	Clark	35	E1
E4	David	30	E3
E5	Emily	25	E4
E6	Last	20	E5

Workson

E#	P#	Hours
E1	P1	700
E2	P1	300
E2	P2	200
E3	P1	100
E3	P2	200
E3	P3	300
E4	P1	100
E4	P2	200
E4	P3	300
E6	P1	200
E6	P2	300
E6	P3	400
E6	P4	500

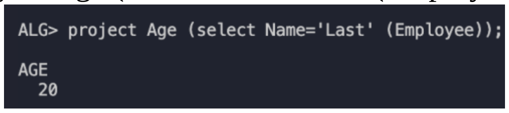
Project

P#	Name	Location
P1	CPU	B1
P2	GPU	B2
P3	GPU	B2
P4	SSD	B3

4. Use Relational Algebra (ALG) to express the following queries based on the above database. Submit your ALG expressions for each query as well as the query result. Each query is 4 marks and the result is 2 marks (60)

1) Get the age of Last.

Query: project Age (select Name='Last' (Employee));

Result: 

```
ALG> project Age (select Name='Last' (Employee));

AGE
  20
```

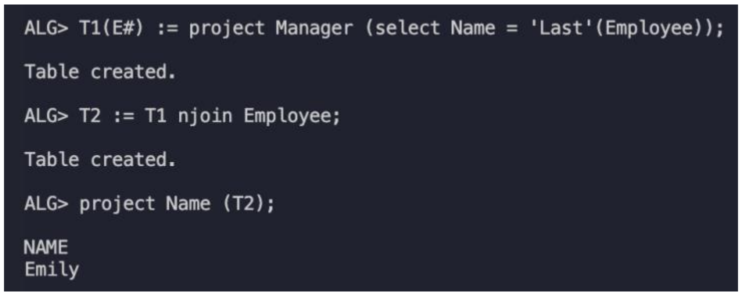
2) Get the name of Last's manager

Queries:

T1(E#) := project Manager (select Name = 'Last'(Employee));

T2 := T1 njoin Employee;

project Name (T2);

Result: 

```
ALG> T1(E#) := project Manager (select Name = 'Last'(Employee));

Table created.

ALG> T2 := T1 njoin Employee;

Table created.

ALG> project Name (T2);

NAME
Emily
```

3) Get the name of the employee who works on GPU project.

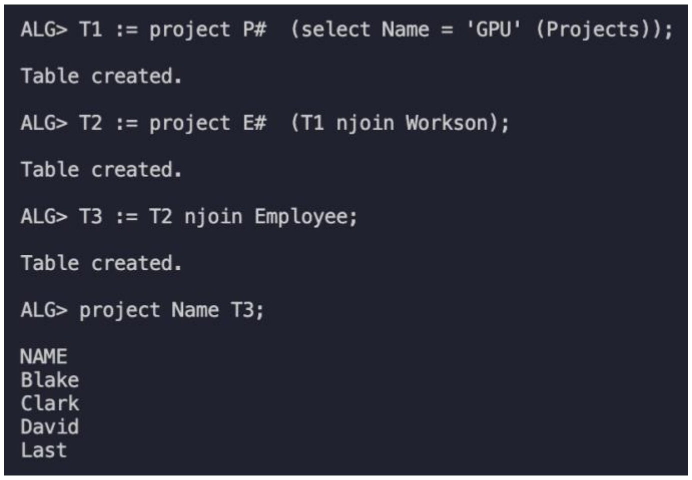
Queries:

T1 := project P# (select Name = 'GPU' (Projects));

T2 := project E# (T1 njoin Workson);

T3 := T2 njoin Employee;

project Name T3;

Result: 

```
ALG> T1 := project P#  (select Name = 'GPU' (Projects));

Table created.

ALG> T2 := project E#  (T1 njoin Workson);

Table created.

ALG> T3 := T2 njoin Employee;

Table created.

ALG> project Name T3;

NAME
Blake
Clark
David
Last
```

4) Get the name of the employee who does not work on any project.

Queries:

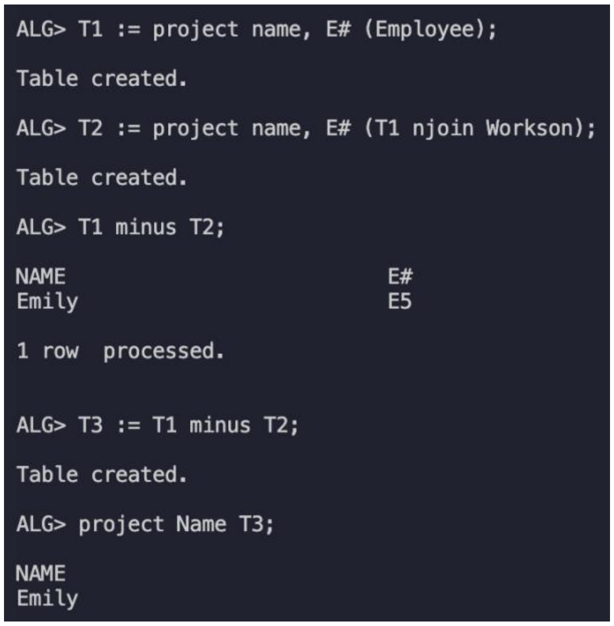
T1 := project name, E# (Employee);

T2 := project name, E# (T1 njoin Workson);

T3 := T1 minus T2;

project name T3;

Result:



```
ALG> T1 := project name, E# (Employee);

Table created.

ALG> T2 := project name, E# (T1 njoin Workson);

Table created.

ALG> T1 minus T2;

NAME                          E#
Emily                         E5

1 row  processed.


ALG> T3 := T1 minus T2;

Table created.

ALG> project Name T3;

NAME
Emily
```

5) Get the pair of employee name and project name such that the employee works on the project less than 300 hours.

Queries:

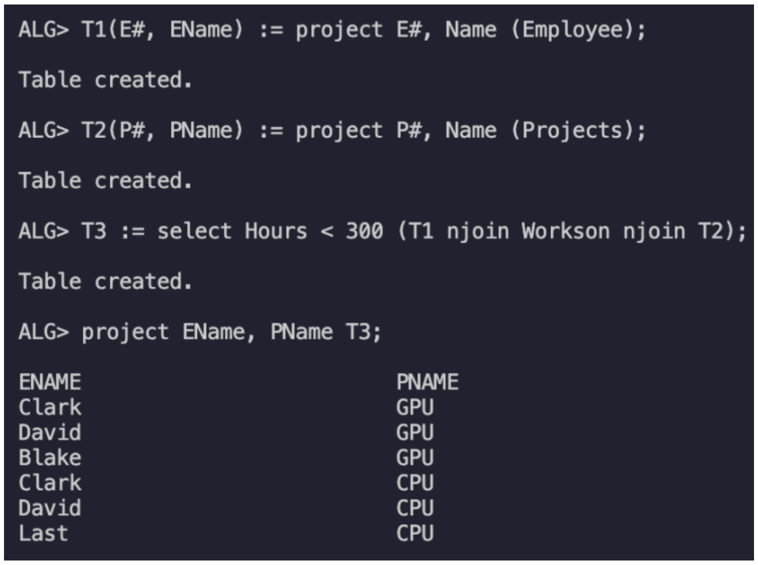
T1(E#, EName) := project E#, Name (Employee);

T2(P#, PName) := project P#, Name (Projects);

T3 := select Hours < 300 (T1 njoin Workson njoin T2);

project EName, PName T3;

Result:



```
ALG> T1(E#, EName) := project E#, Name (Employee);

Table created.

ALG> T2(P#, PName) := project P#, Name (Projects);

Table created.

ALG> T3 := select Hours < 300 (T1 njoin Workson njoin T2);

Table created.

ALG> project EName, PName T3;

ENAME                         PNAME
Clark                         GPU
David                         GPU
Blake                         GPU
Clark                         CPU
David                         CPU
Last                          CPU
```

6) Get the name of the employee who works on every project

Queries:

T1 := project E#, P# (Workson);

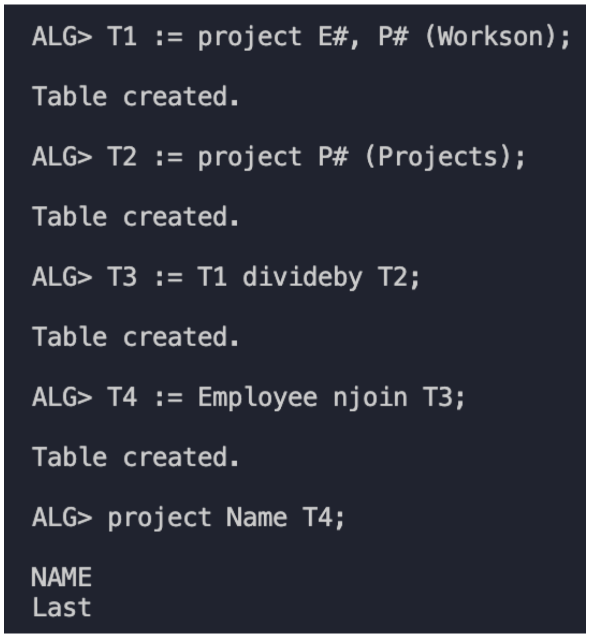
T2 := project P# (Projects);

T3 := T1 divideby T2;

T4 := Employee njoin T3;

project Name T4;

Result:



```
ALG> T1 := project E#, P# (Workson);

Table created.

ALG> T2 := project P# (Projects);

Table created.

ALG> T3 := T1 divideby T2;

Table created.

ALG> T4 := Employee njoin T3;

Table created.

ALG> project Name T4;

NAME
Last
```

7) Get the name of the employee who works on every project except SSD.

Queries:

T1 := project E#, P# (Workson);

T2 := project P# (select Name != 'SSD' (Projects));

T3 := T1 divideby T2;

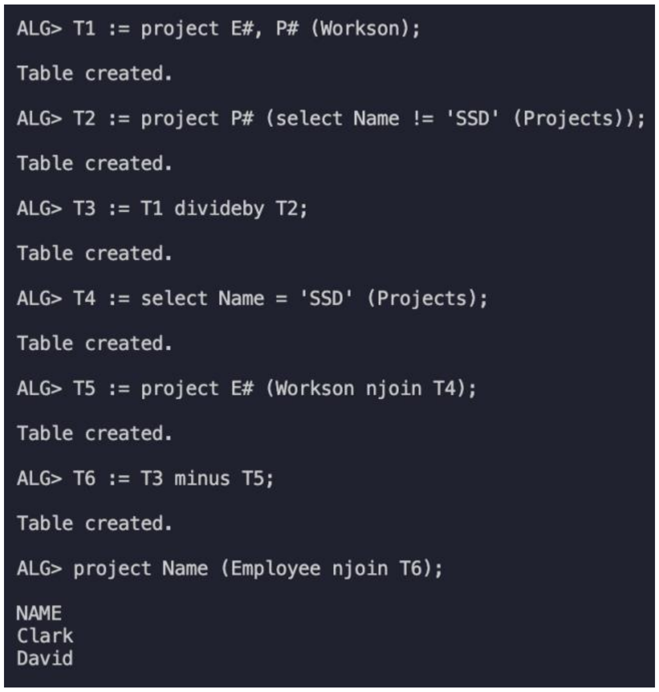
T4 := select Name = 'SSD' (Projects);

T5 := project E# (Workson njoin T4);

T6 := T3 minus T5;

project Name (Employee njoin T6);

Result:



```
ALG> T1 := project E#, P# (Workson);

Table created.

ALG> T2 := project P# (select Name != 'SSD' (Projects));

Table created.

ALG> T3 := T1 divideby T2;

Table created.

ALG> T4 := select Name = 'SSD' (Projects);

Table created.

ALG> T5 := project E# (Workson njoin T4);

Table created.

ALG> T6 := T3 minus T5;

Table created.

ALG> project Name (Employee njoin T6);

NAME
Clark
David
```

8) Get the name of the employee who works on every project that Clark works on.

Queries:

T1 := project E#, P# (Workson);

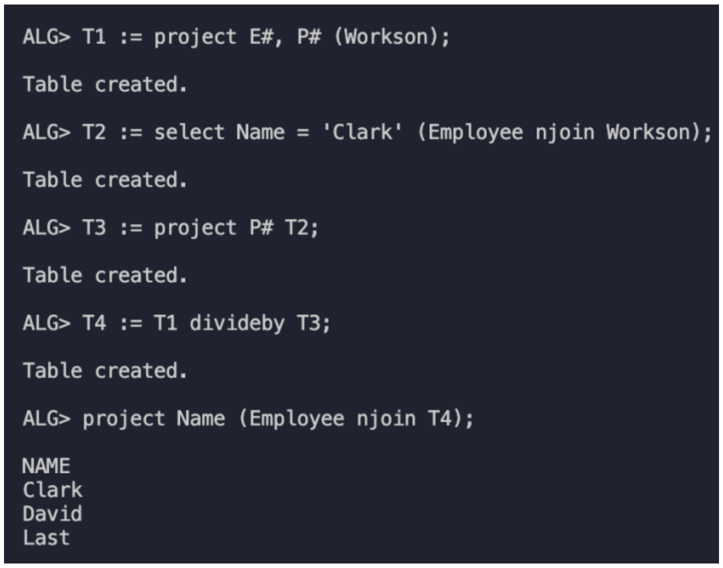
T2 := select Name = 'Clark' (Employee njoin Workson);

T3 := project P# T2;

T4 := T1 divideby T3;

project Name (Employee njoin T4);

Result:



```
ALG> T1 := project E#, P# (Workson);

Table created.

ALG> T2 := select Name = 'Clark' (Employee njoin Workson);

Table created.

ALG> T3 := project P# T2;

Table created.

ALG> T4 := T1 divideby T3;

Table created.

ALG> project Name (Employee njoin T4);

NAME
Clark
David
Last
```

9) Get the name of the employee who works on the same projects that Clark works on.

Queries:

First four are the same as prev question:

T1 := project E#, P# (Workson);

T2 := select Name = 'Clark' (Employee njoin Workson);

T3 := project P# T2;

T4 := T1 divideby T3; //this was all the employees that work on every project Clark works on

Now we add:

T5 := project P# (Workson); //all projects

T6 := project P# (select Name = 'Clark' (Employee njoin Workson)); //grab the project #’s that clark works on

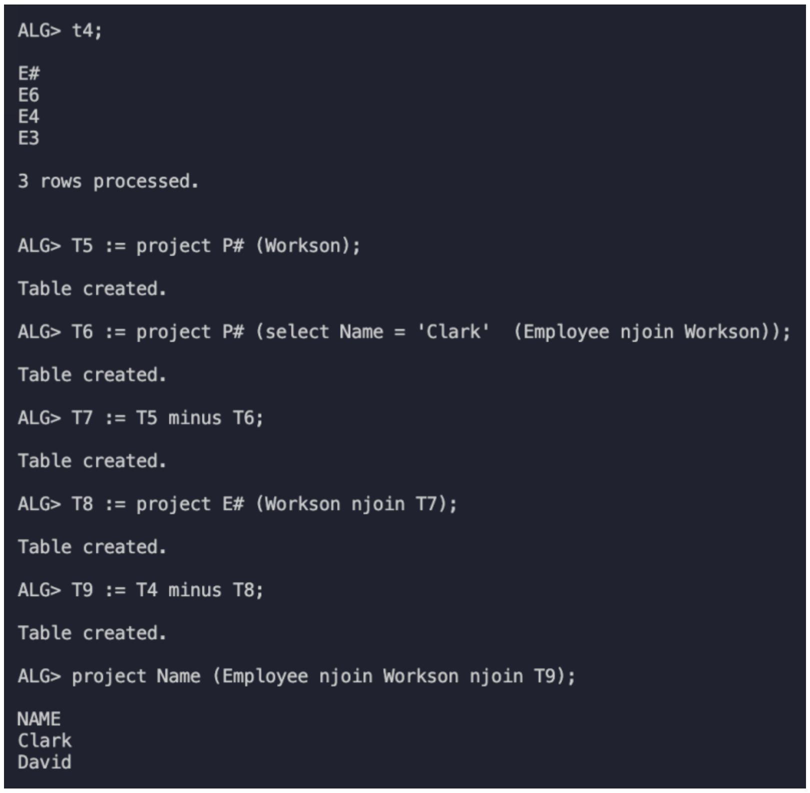
T7 := T5 minus T6; //get the project nums that clark does not work on

T8 := project E# (Workson njoin T7); //get the employee #’s of the employees that work on the ones that clark does not

T9 := T4 minus T8; //subtract the employees that work on the projects that clark works on with the employees that also work on other projects

project Name (Employee njoin Workson njoin T9);

Result:



```
ALG> t4;

E#
E6
E4
E3

3 rows processed.


ALG> T5 := project P# (Workson);

Table created.

ALG> T6 := project P# (select Name = 'Clark'  (Employee njoin Workson));

Table created.

ALG> T7 := T5 minus T6;

Table created.

ALG> T8 := project E# (Workson njoin T7);

Table created.

ALG> T9 := T4 minus T8;

Table created.

ALG> project Name (Employee njoin Workson njoin T9);

NAME
Clark
David
```

10) Get the name of the employee who works on more than two projects.

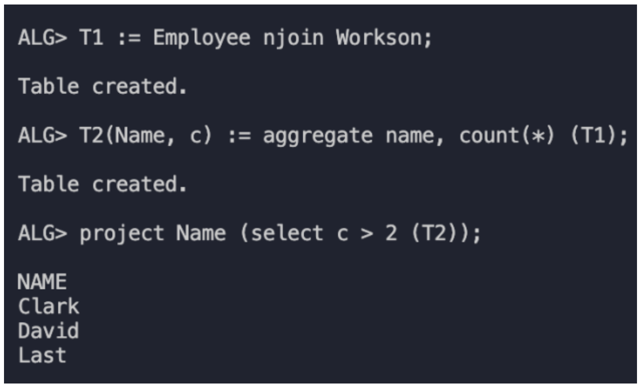
Queries:

T1 := Employee njoin Workson;

T2(Name, c) := aggregate name, count(*) (T1);

project Name (select c > 2 (T2));

Result:



```
ALG> T1 := Employee njoin Workson;

Table created.

ALG> T2(Name, c) := aggregate name, count(*) (T1);

Table created.

ALG> project Name (select c > 2 (T2));

NAME
Clark
David
Last
```