**COMP 3005**  
**Assignment #3**  
**Due: Oct. 19 @11:59PM**

**Instruction**

1. Do the assignments independently. Copying is not allowed.
2. The database for this assignment is the same as in Assignment #1. Do the assignment directly on this document and rename it with your last name + first name and submit to **brightspace**. Make sure your uploaded file can be opened.
3. You can use any version of Oracle VM or download and install latest version of Oracle from its official website.

**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.

1. Type compatibility (2)

This is used to make sure that data types are compatible with each other in order to complete an operation or comparison

1. Relationally complete (2)

This refers to the ability for a query language to express all types of operations and queries using the relational model to its fullest capabilities.

1. Basic relational operators (2)

These are the operators that are used to manipulate and query the data in a relational database. These include: Selection, Projection, Union, Intersection, Difference, Cartesian Product, Join. The provide the foundation for more complex queries to be constructed.

1. Free variable (2)

Free variables are variables in tuple relations that do not appear in the query result.

1. Bound variable (2)

Bounded variables are variables that are used to specify the result, they are bounded if they are quantified (used in an existensal or universal quantification)

1. Grouping (2)

Grouping is the process of categorizing or using aggregate functions to group data base on a condition(s). in SQL this is shown by using the GROUPBY clause that is typically used with an aggregate function to parse the data

1. Aggregate functions (2)

Aggregate functions are functions that operate on data and result in a sub-section of data, sometimes filtered. These aggregate functions include: Sum, Count, Avg, Max, Min.

1. Use relational algebra to represent R1(S#, C#) divideby R2 (S#) (6)

R1 R2

**Part 2 (80 marks)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Employees** | | | |
| **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 |

|  |  |  |
| --- | --- | --- |
| **Workon** | | |
| **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 |

|  |  |  |
| --- | --- | --- |
| **Projects** | | |
| **P#** | **Name** | **Location** |
| P1 | CPU | B1 |
| P2 | GPU | B2 |
| P3 | GPU | B2 |
| P4 | SSD | B3 |

Given the employees and projects databases the same as in Assignment #1. Use Query By Example (QBE) to express queries 1,2,3,4,5,6 and use SQL to express all 10 queries. (80)  
Each QBE query is 5 marks. Each SQL query is 4 marks and the result is 1 mark. Screenshot of both SQL query and running results are needed in order to get the 5 marks.

1. Get the age of Last.

QBE Query:

Employees

|  |  |  |  |
| --- | --- | --- | --- |
| E# | Name | Age | Manager |
|  | Last | P. |  |

SQL Query:

select Age from Employees where Name = 'Last';

SQL Result:

A black background with white text

Description automatically generated

1. Get the name of Last’s manager

QBE Query:

Employees Employees

|  |  |  |  |
| --- | --- | --- | --- |
| E# | Name | Age | Manager |
| \_E1 | P. |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| E# | Name | Age | Manager |
|  | Last |  | \_M2 |

Condition:

\_E1 = \_M2

SQL Query:

select Name from Employees where E# = (select Manager from Employees where Name = 'Last');

SQL Result:

A screen shot of a computer

Description automatically generated

1. Get the name of the employee who works on GPU project.

QBE Query:

Employees Workson Projects

|  |  |  |  |
| --- | --- | --- | --- |
| E# | Name | Age | Manager |
| \_E | P. |  |  |

|  |  |  |
| --- | --- | --- |
| P# | Name | Location |
| \_P | GPU |  |

|  |  |  |
| --- | --- | --- |
| E# | #P | Hours |
| \_E | \_P |  |

SQL Query:

select distinct E.Name from Employees E, Workson W, Projects P where E.E# = W.E# and W.P# = P.P# and P.Name = 'GPU';

SQL Result:

A blue screen with white text

Description automatically generated

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

QBE Query: (not on the Projects table)

Employees Workson Projects

|  |  |  |  |
| --- | --- | --- | --- |
| E# | Name | Age | Manager |
| \_E | P. |  |  |

|  |  |  |
| --- | --- | --- |
| P# | Name | Location |
| \_P |  |  |

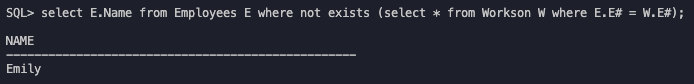
|  |  |  |
| --- | --- | --- |
| E# | #P | Hours |
| \_E | \_P |  |

¬

SQL Query:

select E.Name from Employees E where not exists (select \* from Workson W where E.E# = W.E#);

SQL Result:



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

QBE Query:

Employees Workson Projects

|  |  |  |  |
| --- | --- | --- | --- |
| E# | Name | Age | Manager |
| \_E | P. |  |  |

|  |  |  |
| --- | --- | --- |
| P# | Name | Location |
| \_P | P. |  |

|  |  |  |
| --- | --- | --- |
| E# | #P | Hours |
| \_E | \_P | <300 |

SQL Query:

select distinct E.Name, P.Name from Employees E, Projects P, Workson W where (E.E# = W.E# and W.P# = P.P# and W.Hours < 300);

SQL Result:

A screenshot of a computer

Description automatically generated

1. Get the name of the employee who works on every project

QBE Query: (not on the Workson and Projects table)

Employees Workson Projects

|  |  |  |  |
| --- | --- | --- | --- |
| E# | Name | Age | Manager |
| \_E | P. |  |  |

|  |  |  |
| --- | --- | --- |
| P# | Name | Location |
| \_P |  |  |

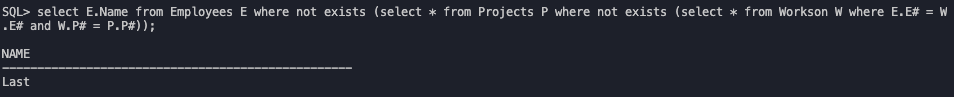
|  |  |  |
| --- | --- | --- |
| E# | #P | Hours |
| \_E | \_P |  |

¬ ¬

SQL Query:

select E.Name from Employees E where not exists (select \* from Projects P where not exists (select \* from Workson W where E.E# = W.E# and W.P# = P.P#));

SQL Result:

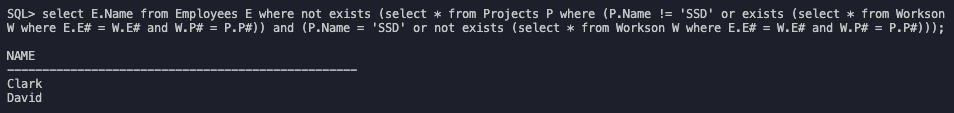


1. Get the name of the employee who works on every project except SSD.

SQL Query:

select E.Name from Employees E where not exists (select \* from Projects P where (P.Name != 'SSD' or exists (select \* from Workson W where E.E# = W.E# and W.P# = P.P#)) and (P.Name = 'SSD' or not exists (select \* from Workson W where E.E# = W.E# and W.P# = P.P#)));

SQL Result:

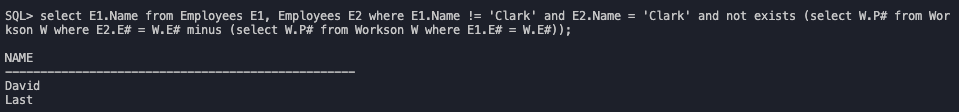


1. Get the name of the employee who works on every project that Clark works on.

SQL Query:

select E1.Name from Employees E1, Employees E2 where E1.Name != 'Clark' and E2.Name = 'Clark' and not exists (select W.P# from Workson W where E2.E# = W.E# minus (select W.P# from Workson W where E1.E# = W.E#));

SQL Result:

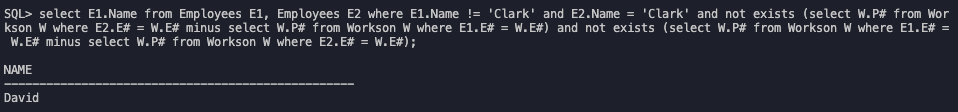


1. Get the name of the employee who works on the same projects that Clark works on.

SQL Query:

select E1.Name from Employees E1, Employees E2 where E1.Name != 'Clark' and E2.Name = 'Clark' and not exists (select W.P# from Workson W where E2.E# = W.E# minus select W.P# from Workson W where E1.E# = W.E#) and not exists (select W.P# from Workson W where E1.E# = W.E# minus select W.P# from Workson W where E2.E# = W.E#);

SQL Result:



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

SQL Query:

select E.Name from Employees E, Workson W where E.E# = W.E# group by E.Name having count(\*) > 2;

SQL Result:

A screen shot of a computer

Description automatically generated