

Consider the following schema:  
Supplier(sid: integer, sname: string, address: string)

Part(pid: integer, pname: string, colour: string)

Catalog(sid: integer, pid: integer, cost: real)

The relation Supplier stores suppliers and the key of that relation is sid. The relation Part stores parts, and pid is the key of that relation. Finally, Catalogue stores which supplier supplies which part at which cost. The key is the combination of the two attributes sid and pid.

**1. Write queries in relational algebra 3%**

|  |
| --- |
| Supplier |
| sid |
| sname |
| address |

|  |
| --- |
| Catalog |
| sid |
| pid |
| cost |

|  |
| --- |
| Part |
| pid |
| pname |
| colour |

Write the following queries in relational algebra.

1. Find the names of suppliers who supply some red part.

Answer:

∏sname(𝜎colour=”red”(Supplier⨝Part⨝Catalog))

Duplicate rows are automatically eliminated, as relation is a set.

Or the queries in the relational algebra could be

∏ sname(∏sid((((𝜎colour=”red” Part)⨝ Catalog) ⨝Supplier))

1. Find the IDs of suppliers who supply some red or green part.

|  |
| --- |
| (𝜎colour=”red”v colour=”green”Part) |
| Pid |
| Pname |
| Colour |

|  |
| --- |
| ((𝜎colour=”red”v colour=”green”Part)⨝ Catalog) |
| Pid |
| sid |
| Pname |
| Colour |
| Cost |

|  |
| --- |
| Catalog |
| sid |
| pid |
| cost |

Answer: ∏ sid((𝜎colour=”red”v colour=”green”Part)⨝ Catalog)

1. Find the IDs of suppliers who supply some red part or are based at 21 George Street.

|  |
| --- |
| ∏ pid (𝜎colour=”red”Part) |
| Pid |

|  |
| --- |
| ∏ sid(∏ pid (𝜎colour=”red”Part)⨝ Catalog) |
| sid |

|  |
| --- |
| ∏ sid(𝜎address=”21 George Street”Supplier) |
| sid |

Answer:

∏sid(∏pid(𝜎colour=”red”Part)⨝Catalog)U(∏sid(𝜎address=”21 George Street”Supplier))

1. Find the names of suppliers who supply some red part or are based at 21 George Street.

|  |
| --- |
| ∏ pid (𝜎colour=”red”Part) |
| Pid |

|  |
| --- |
| ∏ sid(𝜎colour=”red”Part)⨝ Catalog) |
| sid |

|  |
| --- |
| ∏sname(∏sid(𝜎colour=”red”Part)⨝Catalog))⨝Supllier) |
| sname |

|  |
| --- |
| ∏ sname(𝜎address=”21 George Street”Supplier) |
| sname |

Answer:

∏sname(∏sid(𝜎colour=”red”Part)⨝Catalog))⨝Supllier)U∏sname(𝜎address=”21George Street”Supplier)

1. Find the IDs of suppliers who supply some red part and some green part.

(Hint: use intersection of relations or join the same relation several times)

|  |
| --- |
| ∏ sid((𝜎colour=”red”Part)⨝ Catalog) |
| sid |

|  |
| --- |
| ∏ sid((𝜎colour=”green”Part)⨝ Catalog) |
| sid |

Answer:

∏sid((𝜎colour=”red”Part)⨝ Catalog) ∩ ∏ sid((𝜎colour=”green”Part)⨝ Catalog)

1. Find pairs of IDs such that the supplier with the first ID charges more for some part than the supplier with the secondID. (Hint: you may want to create temporary relations to get two copies of Catalog)
   1. Create two copies of Catalog and use Cartesian Product to compare each row of first table to each row of table 2

(Catalog 1 X Catalog 2)

* 1. Only compare the same product in each tuple and different sid (𝜎Catalog1.pid=Catalog2.pid^Catalog1.sid!=Catalog2.sid(Catalog 1 X Catalog 2))
  2. Compare cost (𝜎Catalog1.pid=Catalog2.pid^Catalog1.sid!=Catalog2.sid^Catalog1.cost>Catalog2.cost(Catalog 1 X Catalog 2))
  3. Project the sid
  4. Answer:

∏Catalog1.sid, Catalog2.sid(𝜎Catalog1.pid=Catalog2.pid^Catalog1.sid!=Catalog2.sid^Catalog1.cost>Catalog2.cost(Catalog 1 X Catalog 2))

1. Find the IDs of suppliers who supply only red parts. (Hint: A supplier supplies only red parts if it is not the case that the supplier offers a part that is not red. This question is a challenge!)
   1. Find the sid supply part which is not red :

∏sid((𝜎colour!=”red”Part) ⨝Catalog)

* 1. A supplier supplies only red parts if it is not the case that the supplier offers a part that is not red.
  2. Answer:

∏sidSupplier-∏sid((𝜎colour!=”red”Part) ⨝Catalog)

1. Find the IDs of suppliers who supply every part. (Hint: A supplier supplies every part if it is not the case that there is some part which they do not supply. Use set difference and cartesian product. This question is a challenge, too!)
   1. Create a relation indicate each supplier ID supply all the part:

∏sid,Supplier X ∏pidPart

* 1. Use division operator to find who actually supply every part:

∏sid, pid(Catalog) / ∏sid,Supplier X ∏pidPart

* 1. Find IDs of suppliers who supply every part:

Answer:

∏sid(∏sid, pid(Catalog) /∏sid, pid(Supplier X Part))

**2. Queries in relational algebra, what do they mean? 3%**

For each of the following relational algebra queries, say in English what they mean.

1. πsname(σcolour=”red” (Part) x σcost<100 (Catalog) x Supplier)

Answer: Find the name of supplier who supply some red part less than cost 100

2. πsname(πsid(σcolour=”red” (Part) x σcost<100 (Catalog)) x Supplier)

Answer: Find the name of supplier who supply some red part less than cost 100

3. πsname(σcolour=”red” (Part) x σcost<100 (Catalog) x Supplier) ∩ πsname(σcolour=”green” (Part) x σcost<100 (Catalog) x Supplier)

πsname(σcolour=”red” (Part) x σcost<100 (Catalog) x Supplier):

set of name in suppliers supplying some red part less than cost 100

πsname(σcolour=”green” (Part) x σcost<100 (Catalog) x Supplier):

set of name in suppliers supplying some green part less than cost 100

Answer: Find the name of supplier who supply some red part and the supplier who supply green part less than cost 100

Note: two distinct suppliers with the same name may be returned by this query even if the first doesn’t supply a green part, and the second doesn’t supply a red part.

4. πsid(σcolour=”red” (Part) x σcost<100 (Catalog) x Supplier) ∩πsid(σcolour=”green” (Part) x σcost<100 (Catalog) x Supplier)

Answer: Find the ID of supplier who supply some red part and green part both less than cost 100

5. πsname(πsid,sname(σcolour=”red” (Part) x σcost<100 (Catalog) x Supplier) ∩ πsid,sname(σcolour=”green” (Part) x σcost<100 (Catalog) x Supplier))

πsname(πsid,sname(σcolour=”red” (Part) x σcost<100 (Catalog) x Supplier):

Project the name of suppliers from the table contains two attributes: sid and sname, indicate from which red part less than cost 100 supplied from.

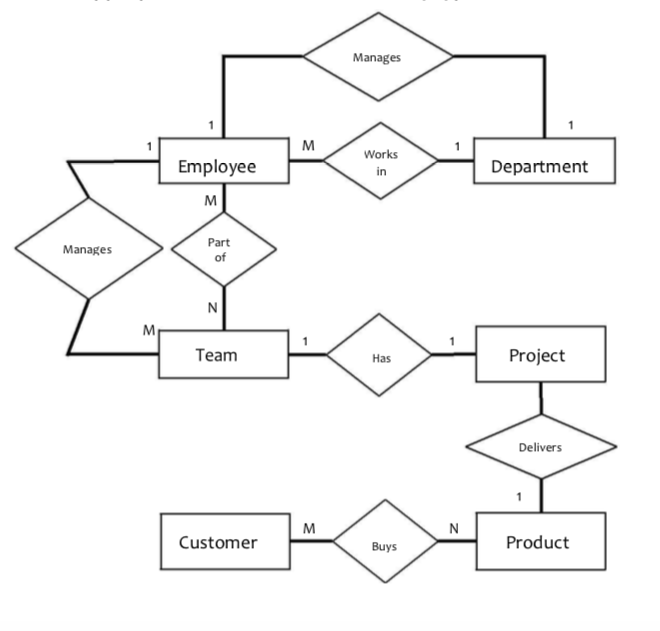
πsid,sname(σcolour=”green” (Part) x σcost<100 (Catalog) x Supplier)): Project the name and ID of suppliers from the joined table indicates from which green part less than cost 100 supplied from.

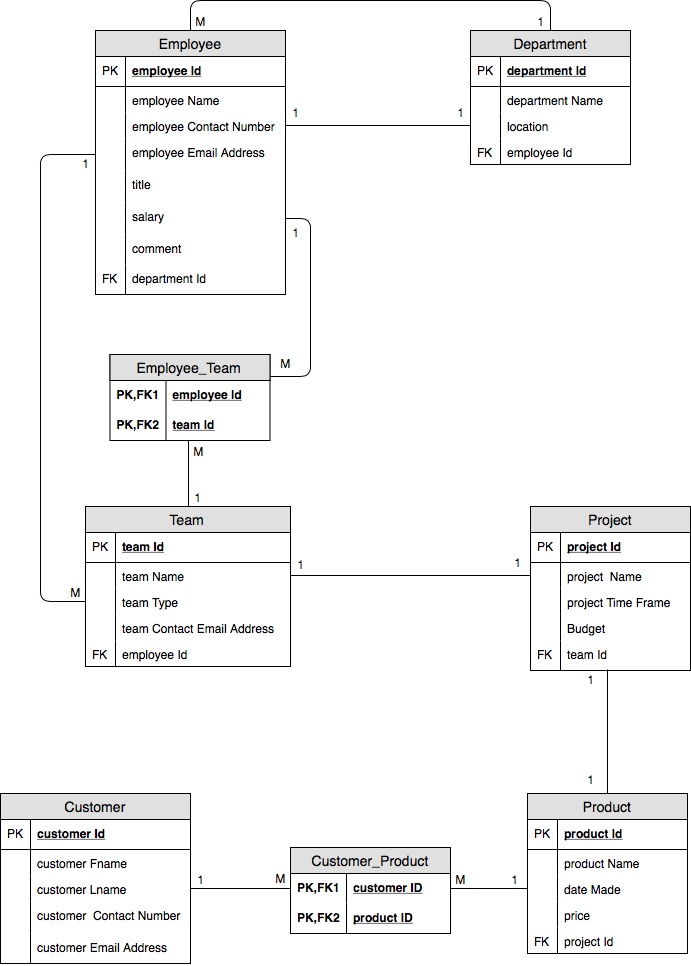
Answer:

Find the name of suppliers who supply some red part and some green part for less than 100.

**3. Mapping & Normalisation 9%**

a) Mapping the following Entity Relationship Diagram into relational schema. Choose appropriate attributes for each entity type. 5%





b) Database normalization – consider the following relation: 5%

Employee (emp-id, frst-name, last-name, address, phone, role-id, role-name)

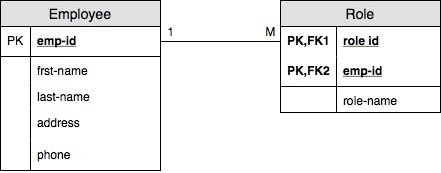
Let’s assume each employee has one address and one phone number but each employee within the company is given more than one roles.

For example an employee may take responsibility of providing telephone support and face to face support.

• State whether the given table structure is in 1NF. If not, make appropriate changes so that it is in 1NF.

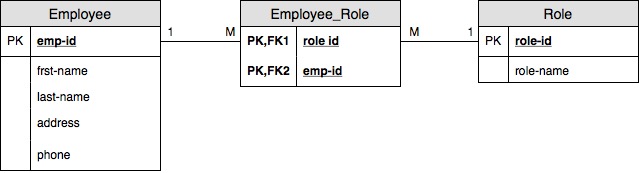
No, the given table structure is not in 1NF. A relation is in 1NF if, and only if, it contains no repeating attributes or groups of attributes. In other words each of the column in a table should contain one value and just one value.

Decomposing:



• State whether the given table structure is in 2NF. If not, make appropriate changes so that it is in 2NF.

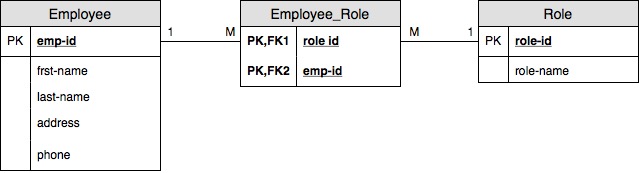
No, the given table structure is not in 2NF. No non-key fields may depend on a portion of the primary key in 2NF. Check all of the non-key attributes against each part of the key attributes to ensure they are functionally dependent on it.



• State whether the given table structure is in 3NF. If not, make appropriate changes so that it is in 3NF.

No, the given table structure is not in 3NF.

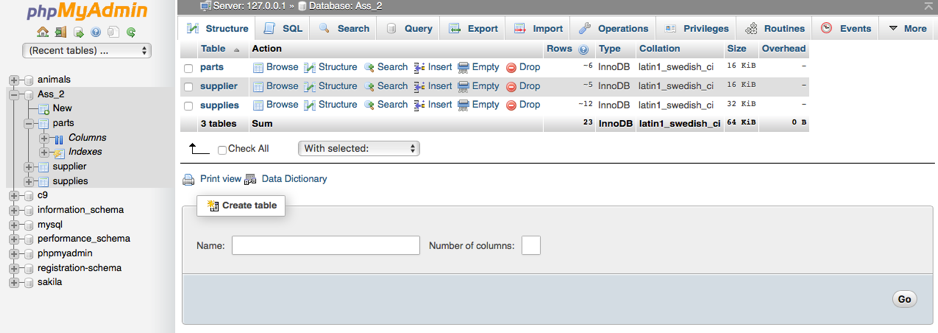
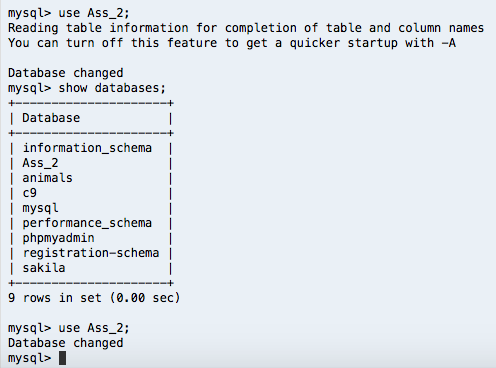
Third Normal From (3FN): No fields may depend on other non- key fields. In other words, each field in a record should contain information about the entity that is defined by the primary key.



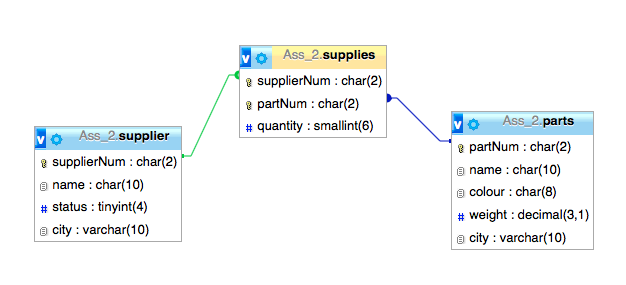
**4. SQL 5%**

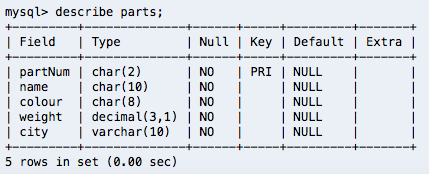
Run the SQL – DDL and DML statements provided in assignment2\_\_DDL\_DML.sql file (on Moodle). This will create 3 tables and populate them with data.

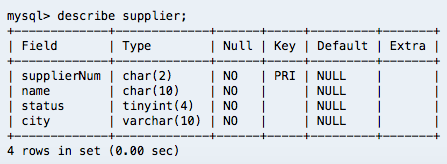
As the file is in .sql format, to save time you can copy and paste when executing these statements.

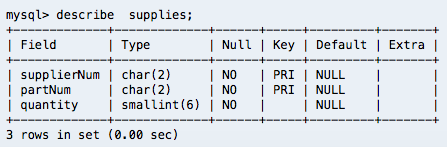
a) Read the statements when you execute them so you understand the tables, columns, and their relationships. Once you have run all the statements successfully, now create and execute the following queries:

b) How many relationships exist between these tables? Specify their type & cardinality.

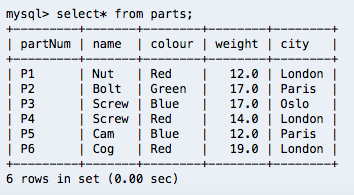


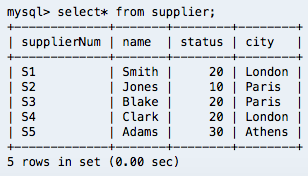


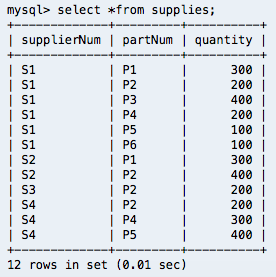




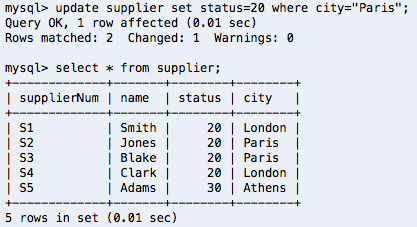
c) List all the records in supplier, parts, & supplies tables. One table at a time.



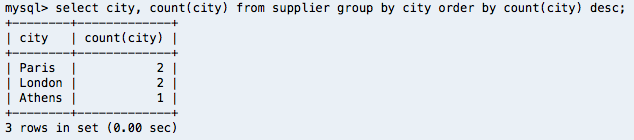




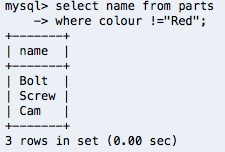
d) Update the supplier table to reflect a change in supplier’s status to the value 20 for all suppliers from Paris.



e) Show the number of suppliers in each city ordered from highest to lowest.

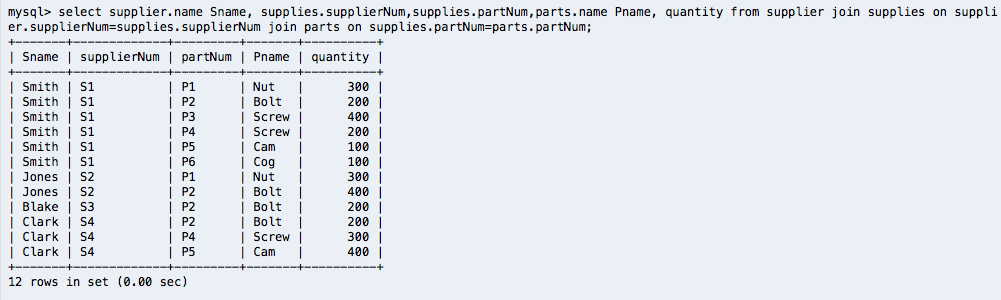


f) List only the name of all the parts except the Red parts.

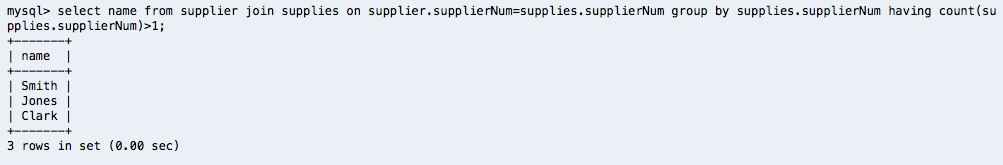


g) Show all entries from the supplies table with their corresponding part names and

supplier names. Rename the columns to appropriate ones.



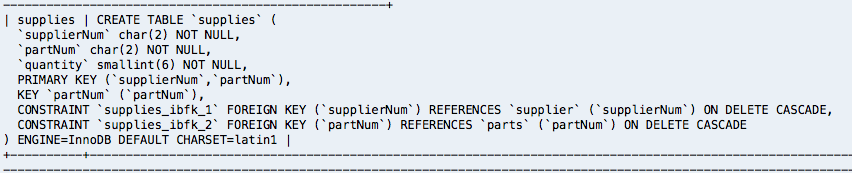
h) Show the names of all suppliers that appear more than once in the supplies table.



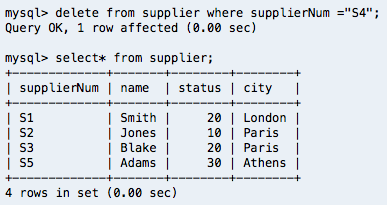
i) Supplier with supplierNum = S4 has closed down his business. Delete all the records related to this supplier from all relevant tables.

Note: MySQL provides a more effective way called ON DELETE CASCADE referential action for a [foreign key](http://mysqltutorial.org/mysql-foreign-key/) that allows you to delete data from child tables automatically when you delete the data from the parent table. For this assignment the original database file is altered in .txt and imported to myphpadmin. In addition, the action of this query will affect the following two queries due to deletion of supplierNum = S4.

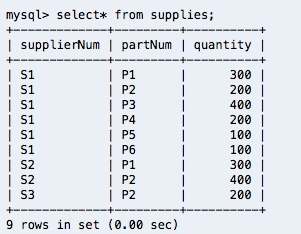
Shows the CREATE TABLE statement that creates the named table to make sure  ON DELETE CASCADE  clause at the end of the foreign key constraint definition.



Delete:



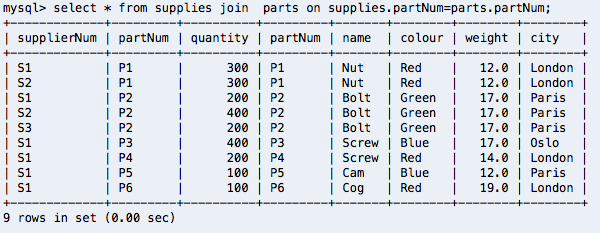
Check



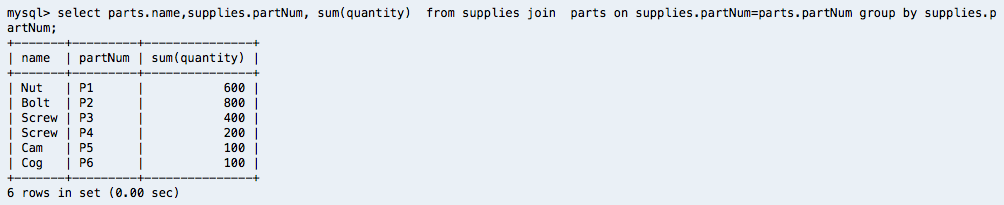
j) List all the parts except those with a quantity of 200.

Note: assuming exclude the total quantity of 200 of each parts

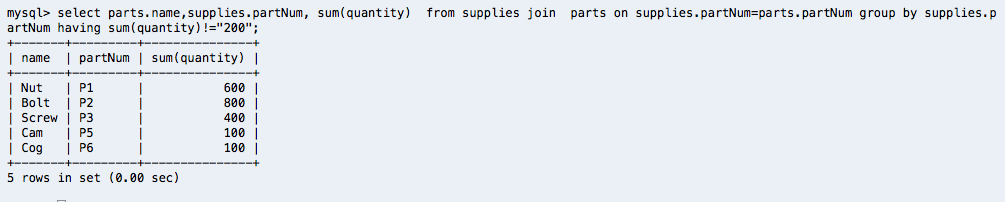
1. Join two tables



1. Sum and group

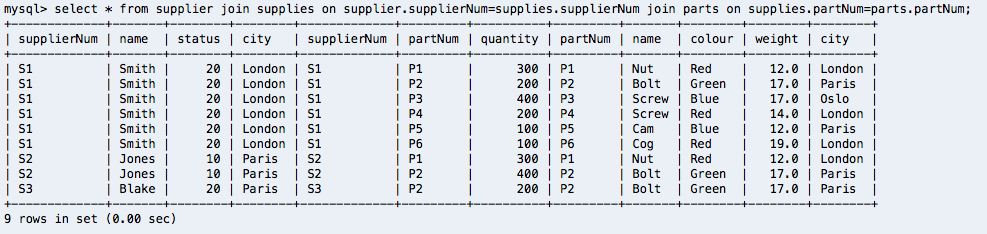


1. List all the parts except those with a quantity of 200.



k) List part names, their colour, and the supplier(s) who supply them.

1. Join three tables required



1. List columns

