Relational Algebra Solution of Tutorial

First Schema

Suppliers(<u>sID</u>, sName, address) Parts(<u>pID</u>,
pName, colour) Catalog(<u>sID, pID</u>, price)

Catalog[sID] ⊆ Suppliers[sID] Catalog[pID] ⊆
Parts[pID]

1. Answer: Just because it is a key in one relation doesn't mean it is in another; being a key is relative to the relation. But *is* it a key for Catalog? No. We almost surely want to be able to list multiple parts by one supplier in our catalog.
2. Answer: $\Pi_{pName}(\sigma_{colour="red"} Parts)$
3. Answer: $\Pi_{price}((\sigma_{colour="red" \vee colour="green"} Parts) \ da \ Catalog)$
4. Answer: $\Pi_{sID}((\sigma_{colour="red" \vee colour="green"} Parts) \ da \ Catalog)$
5. Answer: Trick question. Each tuple has only one colour, and each part has only one tuple (since pID is a key), so no part can be recorded as both red and green.
6. Answer: $\Pi_{sName}((\Pi_{sID}((\sigma_{colour="red" \vee colour="green"} Parts) \ da \ Catalog)) \ da \ Supplievs)$

Second Schema

Employees(<u>number</u>, name, age, salary)

Supervises(boss, <u>employee</u>)

Supervises[boss] ⊆ Employees[number] Supervises[employee]
⊆ Employees[number]

7. Answer: Every employee has one boss.
8. Answer: Yes
9. Answer: It would mean that every boss could have at most one employee. Not very sensible!
10. Answer: This would imply that neither alone is a key, since keys are minimal. Thus, bosses could have multliple employees (sensible) and employees could have multiple bosses (possibly sensible).
11. Answer: $\Pi_{name,salary}\sigma boss=numbev((\Pi_{boss}\sigma_{number=employee}((\Pi_{number}\sigma_{salary>100} Employee) \times Supevvises)) \times Employee)$

Third Schema

This schema is for a salon. Services could be things like "haircut" or "manicure".

Clients(<u>CID</u>, name, phone)

Staff(SID, name)

Appointments(CID, date, time, service, SID)

Appointments[CID] $\subseteq$ Clients[CID]

Appointments[SID] $\subseteq$ Staff[SID]

12. Answer: $\Pi_{name,time}((\Pi_{CID,SID,time}\sigma_{date="Feb14"}Appointments)\ da\ (\Pi_{SID}\sigma_{name="Giuliano"}Staff)\ da\ Clients)$
13. Answer: This time, we mustn't use natural join or we'll force the client name and staff names to match, which would be very inappropriate! So we use Cartesian product and are stuck enforcing all the things that do need to match, like SID when we combine Staff and Appointments.
14. Answer:The first is more "efficient" because it produces smaller intermediate relations. But since this is all just math, it doesn't matter! (And in a DMBS, where queries are actually executed and can therefore be more or less efficient, the DBMS optimizes our queries).