Question 1

a)

i) "Find the names of suppliers who supply 'PPE' products that cost less than $6."

ii) *Projecting 'sname' on a relation that only has projected column 'sid' will result in either an error or no relation.*

iii) "Find the names of suppliers who supply both 'PPE' products that cost less than $6 and 'SuperTech' products that cost less than $6."

iv) "Find the sids of suppliers who supply either 'PPE' products that cost less than $6 or 'SuperTech' products that cost less than $6 or both."

v) "Find the names of suppliers who supply both 'PPE' products that cost less than $6 and 'SuperTech' products that cost less than $6."
This query differs from query iii by finding unique (sid, sname) pairs first, before projecting unique supplier names. However, RA removes all duplicate rows, so this operation makes no difference in the relation produced compared to query iii.

b)

i) $\pi_{\text{sname}}\left((\sigma_{\text{tagname}='\text{PPE}'}\text{ProductTag}) \bowtie \text{Catalog} \bowtie \text{Suppliers}\right) \cup$
$\pi_{\text{sname}}\left((\sigma_{\text{tagname}='\text{Testing}'}\text{ProductTag}) \bowtie \text{Catalog} \bowtie \text{Suppliers}\right)$

ii) $\pi_{\text{sid}}\left((\sigma_{\text{tagname}='\text{PPE}'}\text{ProductTag}) \bowtie (\sigma_{\text{cost}<10}\,\text{Catalog}) \bowtie \text{Suppliers}\right) \cap$
$\pi_{\text{sid}}\left((\sigma_{\text{tagname}='\text{PPE}'}\text{ProductTag}) \bowtie (\sigma_{\text{cost}>420}\text{Catalog}) \bowtie \text{Suppliers}\right)$

iii) $\pi_{\text{sid}}\left((\sigma_{\text{tagname}='\text{PPE}'}\text{ProductTag}) \bowtie \text{Catalog} \bowtie \text{Suppliers}\right) - \pi_{\text{sid}}\left((\sigma_{\text{tagname}='\text{PPE}'}\text{ProductTag}) \bowtie (\sigma_{\text{cost}<10 \vee \text{cost}>1337}\text{Catalog}) \bowtie \text{Suppliers}\right)$

iv)

$$AllPossibleCleaning := \pi_{pid}(\sigma_{tagname='Cleaning'}ProductTag) \times \pi_{sid}(Supplier)$$

$$ExistingCleaning := \pi_{\text{pid,sid}}((\sigma_{tagname='Cleaning'}ProductTag) \bowtie \text{Catalog})$$
$$CleaningNotSold := AllPossibleCleaning - ExistingCleaning$$
$$\pi_{sid}(Supplier) - \pi_{sid}(CleaningNotSold)$$

v)

$$CatCopy1 := \rho_A(Catalog)$$
$$CatCopy2 := \rho_B(Catalog)$$
$$DoubleCatalog := (CatCopy1 \bowtie_{A.pid=B.pid} CatCopy2)$$
$$\pi_{A.sid,B.sid}(\sigma_{A.cost>=1.2*B.cost}DoubleCatalog)$$

vi)

$$catalogRename := \rho_{cat2}(\text{Catalog})$$
$$select := (\text{Catalog}) \bowtie_{\text{cat2.pid = catalog.pid and cat2.sid} \neq \text{catalog.sid}} (catalogRename)$$
$$\pi_{\text{catalog.pid}}\ (select)$$

vii)

$$superTechUSA :=$$
$$((\pi_{pid}\,(\sigma_{\text{tagname = 'Super Tech'}}\,(ProductTag))) \bowtie (\text{Catalog})) \bowtie (\pi_{sid}\,(\sigma_{\text{scountry = 'USA'}}\,(Suppliers)))$$

$$\pi_{sid}(superTechUSA) - \pi_{y.sid}(\sigma_{z.cost > y.cost}((\pi_{cost, sid}(\rho_z(superTechUSA)) \times \pi_{cost, sid}(\rho_y(superTechUSA)))))$$

viii)

$$superTechUSA :=$$
$$\pi_{catalog.sid, catalog.cost}((\pi_{pid}(\sigma_{tagname = 'Super\ Tech'}(ProductTag))) \bowtie (Catalog)) \bowtie (\pi_{sid}(\sigma_{scountry = 'USA'}(Suppliers)))$$

$$r2 := \pi_{sid, cost}(superTechUSA) - \pi_{y.sid, y.cost}(\sigma_{z.cost > y.cost}((\pi_{cost, sid}(\rho_z(superTechUSA)) \times \pi_{cost, sid}(\rho_y(superTechUSA)))))$$

$$max1 := superTechUSA - r2$$

$$(\pi_{\text{catalog.sid}}(max1)) - (\pi_{\text{rmax1y.sid}}(\sigma_{\text{rmax1z.cost} > \text{rmax1y.cost}}((\rho_{\text{rmax1z}}(max1)) \times (\rho_{\text{rmax1y}}(max1)))))$$

ix)

$$all := \pi_{suppliers.sid, product.pid}(Suppliers \times Product)$$

$$suppliedbyall := (\pi_{\text{pid}}(Product)) - (\pi_{\text{product.pid}}((all) - (\pi_{\text{sid, pid}}(Catalog))))$$

$$more := \pi_{\text{catalog.pid}}(\sigma_{\text{product.pid = catalog.pid and catalog.cost > 69}}(Catalog \times Product))$$

$$(suppliedbyall - more)$$

x)

$$productPids := \pi_{pid}(Product)$$
$$inventoryPids := \pi_{\text{pid}}(\sigma_{quantity \neq 0} Inventory)$$
$$\pi_{\text{pid}}(productPids - inventoryPids)$$

c)

i)

/* Find the sids of all suppliers with a business relationship. This can produce duplicate entries for reciprocal subsuppliers */

$$AllBusSuppliers := \varrho_{\text{AllBusSuppliers(sid1,sid2)}}(\pi_{\text{subid,sid}}(Subsuppliers))$$

/* Find the sids of all unique pairs of reciprocal subsuppliers. This is necessary for the removal of duplicates in AllBusSuppliers */

$$recip :=$$
$$\varrho_{\text{Recip(sid1,sid2)}}(\pi_{RS1.sid, RS2.sid}(\rho_{RS1}(Subsuppliers)$$
$$\bowtie_{RS1.sid = RS2.subid \wedge RS1.subid = RS2.sid \wedge RS1.sid > RS2.sid} \rho_{RS2}(Subsuppliers)))$$

/* Find all unique business suppliers as all business suppliers minus all unique reciprocal subsuppliers */

$$BusSuppliers := AllBusSuppliers - Recip$$

/* Find the products offered by both business suppliers */

$$BusProducts \coloneqq \varrho_{BusProducts(pid,sid1,sid2,cost1,cost2)}(\pi_{C1.pid,BusSuppliers.sid1,BusSuppliers.sid2,C1.cost,C2.cost}$$

$$(\sigma_{C1.pid=C2.pid}\ BusSuppliers \bowtie_{BusSuppliers.sid1=C1.sid} \rho_{C1}Catalog \bowtie_{BusSuppliers.sid2=C2.sid \wedge C1.pid=C2.pid}$$

$$\rho_{C2}Catalog))$$

/*Find products the university has stock of*/

$$InventoryPids \coloneqq \pi_{pid}\left(\sigma_{quantity \neq 0} Inventory\right)$$

/* Find products offered by both suppliers that the university has no stock of, and return columns as pid, sid1, sid2, cost1, cost2. */

$$\varrho_{Return(pid,sid1,sid2,cost1,cost2)}(\pi_{BusProducts.pid,BusProducts.sid1,BusProducts.sid2,BusProducts.cost1,BusProducts.cost2}$$

$$(BusProducts - (BusProducts \bowtie InventoryPids)))$$

ii)

/* Cross suppliers with themselves. There will be tuples with the same 2 suppliers in a different order, but this is okay since we only return one of the two suppliers in the final relation. */

$$SupDoubles \coloneqq \sigma_{s1.sid \neq s2.sid}\left((\rho_{s1}Suppliers) \times (\rho_{s2}Suppliers)\right)$$

/* Find the suppliers that have one product listed at the exact same price */

$$SamePriceSups \coloneqq \left((SupDoubles) \bowtie_{s1.sid=cat1.sid} \left(\rho_{cat1}(Catalog)\right)\right)$$

$$\bowtie_{cat2.sid=s2.sid \wedge cat1.pid=cat2.pid \wedge cat1.cost=cat2.cost} \left(\rho_{cat2}(Catalog)\right)$$

/* Return cat1.pid, s1.sid, and cat1.cost as pid, sid, and cost */

$$\varrho_{Return(pid,\ sid,\ cost)}\ (\pi_{cat1.pid,s1.sid,cat1.cost}(SamePriceSups))$$

iii)

/* Find all non-“SuperTech” ProductTag entries */

$$NonSuper \coloneqq \left(\pi_{pid}(ProductTag) - \pi_{pid}\left(\sigma_{tagname="SuperTech"} ProductTag\right)\right) \bowtie ProductTag$$

/* Join non-‘SuperTech’ products and their tags with the same pid together three times */

$$TripleJoin = (\rho_A NonSuper) \bowtie_{A.pid=B.pid} (\rho_B NonSuper) \bowtie_{A.pid=C.pid} (\rho_C NonSuper)$$

/* Find the pids of all products that have at least 3 different tag names where one of them is “PPE” */

$$TagsProducts = \pi_{A.pid}\left(\sigma_{A.tagname=PPE \wedge A.tagname \neq B.tagname \wedge B.tagname \neq C.tagname} TripleJoin\right)$$

/* Find pid, pname, cost of products that satisfy tagname criteria */

$$\pi_{A.pid,pname,cost}\left(TagsProducts \bowtie_{A.pid=pid} Product \bowtie_{A.pid=pid} Catalog\right)$$

iv)

/* Find the sids of unique pairs of Reciprocal Subsuppliers as RS1.sid and RS2.sid. Note RS1.sid > RS2.sid is compared in the join predicate to ensure uniqueness of supplier relations in Recip */

$$Recip = \pi_{RS1.sid,RS2.sid}\left(\rho_{RS1}(Subsuppliers) \bowtie_{RS1.sid=RS2.subid \wedge RS1.subid=RS2.sid \wedge RS1.sid > RS2.sid} \rho_{RS2}(Subsuppliers)\right)$$

/* Find all subsuppliers of reciprocal subsupplier 1 */

$$RecipSubRS1 := \pi_{RS1.sid,RS2.sid,subid,subname,subaddress}\left(Recip \bowtie_{RS1.sid=Subsuppliers.sid} Subsuppliers\right)$$

/* Find all subsuppliers of reciprocal subsupplier 2 */

$$RecipSubRS2 := \pi_{RS1.sid,RS2.sid,subid,subname,subaddress}\left(Recip \bowtie_{RS2.sid=Subsuppliers.sid} Subsuppliers\right)$$

/* Find all rows that are not in both the sets of subsuppliers of RS1 and subsuppliers of RS2. */

$$(RecipSubRS1 \cup RecipSubRS2) - (RecipSubRS1 \cap RecipSubRS2)$$

d) Revised Schema:

Subsuppliers (<u>sid: integer</u>, <u>subid: integer</u>, sname: text, subname: text)

Suppliers (<u>sid: integer</u>, sname: text, saddress: text, scountry: text)

ProductTag (<u>tid: integer</u>, <u>pid: integer</u>, tagname: text)

Catalog (<u>sid: integer</u>, <u>pid: integer</u>, cost: real)

Inventory (<u>pid: integer</u>, pname: text, quantity: integer)

In order to improve the database schema, the following changes were made:

- Columns from the Subsuppliers relation that are retrievable from the Suppliers relation were removed to reduce memory costs and improve performance when working with the Subsuppliers relation.
- The Product relation was merged into the Inventory relation as both have the same primary key and both give information about products currently in the inventory of the university or otherwise available to the university. All products not in the inventory can be listed with a quantity of 0.

e)

Question 2

a)

i)

*/*subtract approved from all to get not approved*/*

$$\pi_{utorid}(\text{Student}) - \pi_{utorid}(\sigma_{roomid='IC404'}(\text{Approved}))$$

ii)

*/*rename for comparisons*/*

$$\text{ApprEmployee1} := \rho_{\text{ApprEmployee1}(utorid1,\ roomid1)}(\text{Employee} \bowtie \text{Approved})$$

$$\text{ApprEmployee2} := \rho_{\text{ApprEmployee2}(utorid2,\ roomid2)}(\text{ApprEmployee1})$$

$$\text{ApprEmployee3} := \rho_{\text{ApprEmployee3}(utorid3,\ roomid3)}(\text{ApprEmployee1})$$

*/*at least 2 different*/*

$$\text{TwoDiff} := \text{ApprEmployee1} \bowtie_{utorid1=utorid2 \wedge roomid1<>roomid2} (\text{ApprEmployee2})$$

*/*at least 3 different*/*

$$\text{ThreeDiff} := \text{TwoDiff} \bowtie_{utorid1=utorid3 \wedge roomid1<>roomid3 \wedge roomid2<>roomid3} (\text{ApprEmployee3})$$

*/*project just the utorid*/*

$$\pi_{utorid1}(ThreeDiff)$$

iii)

*/*rename for comparisons*/*

$$ApprEmployee1 := \rho_{ApprEmployee1(utorid1,\ roomid1)}(Employee \bowtie Approved)$$

$$ApprEmployee2 := \rho_{ApprEmployee2(utorid2,\ roomid2)}(ApprEmployee1)$$

$$ApprEmployee3 := \rho_{ApprEmployee3(utorid3,\ roomid3)}(ApprEmployee1)$$

$$ApprEmployee4 := \rho_{ApprEmployee4(utorid4,\ roomid4)}(ApprEmployee1)$$

*/*at least 2 different*/*

$$TwoDiff := ApprEmployee1 \rhd\lhd_{utorid1=utorid2 \wedge roomid1<>roomid2}(ApprEmployee2)$$

*/*at least 3 different*/*

$$ThreeDiff := TwoDiff \rhd\lhd_{utorid1=utorid3 \wedge roomid1<>roomid3 \wedge roomid2<>roomid3}(ApprEmployee3)$$

*/*make sure there is no 4th one*/*

$$NoFourth := ThreeDiff \rhd\lhd_{utorid1=utorid4 \wedge (roomid4=roomid1 \vee roomid4=roomid2 \vee roomid4=roomid3)}(ApprEmployee4)$$

*/*project just the utorid*/*

$$\pi_{utorid1}(NoFourth)$$

iv)

*/*rename for comparisons*/*

$$ApprEmployee1 := \rho_{ApprEmployee1(utorid1,\ roomid1)}(Employee \bowtie Approved)$$

$$ApprEmployee2 := \rho_{ApprEmployee2(utorid2,\ roomid2)}(ApprEmployee1)$$

$$ApprEmployee3 := \rho_{ApprEmployee3(utorid3,\ roomid3)}(ApprEmployee1)$$

$$ApprEmployee4 := \rho_{ApprEmployee4(utorid4,\ roomid4)}(ApprEmployee1)$$

*/*at least 2 diffferent*/*

$$TwoDiff := ApprEmployee1 \rhd\lhd_{utorid1=utorid2 \wedge roomid1<>roomid2}(ApprEmployee2)$$

*/*at least 3 different*/*

$$ThreeDiff := TwoDiff \rhd\lhd_{utorid1=utorid3 \wedge roomid1<>roomid3 \wedge roomid2<>roomid3}(ApprEmployee3)$$

*/*at least 4 different*/*

$$FourDiff := ThreeDiff \rhd\lhd_{utorid1=utorid4 \wedge roomid1<>roomid3 \wedge roomid2<>roomid4 \wedge roomid3<>roomid4}(ApprEmployee4)$$

*/*project utorids for at least 4*/*

$$\text{FourUtor} := \pi_{\text{utorid1}}(\text{FourDiff})$$

*/*at most 3 = all – at least 4*/*

$$\pi_{\text{utorid}}(\text{Employee}) - \text{FourUtor}$$

v)

*/*join to make alert level/threshold comparisons*/*

$$\text{AlertThreshold} := \text{Occupancy} \bowtie \text{Room}$$

*/*filter by alert and date*/*

$$\text{FilterDateAlert} := \sigma_{\text{alertlevel>alertthreshold} \wedge (\text{date}>\text{'2021-09-01'} \wedge \text{date}<\text{'2021-12-31'})}(\text{AlertThreshold})$$

*/*find utorid of 'Oscar Lin'*/*

$$\text{OscarRecords} := \rho_{\text{oscar(studentid)}}(\pi_{\text{utorid}}(\sigma_{\text{name='Oscar Lin'}}(\text{Student} \bowtie \text{Member})))$$

*/*join on Oscar's utorid*/*

$$\text{OscarDateAlert} := \text{OscarRecords} \bowtie_{\text{studentid=utorid}} (\text{FilterDateAlert})$$

*/*project roomid*/*

$$\pi_{\text{roomid}}(\text{OscarDateAlert})$$

vi)

*/*get utorids of all entries between the desired dates*/*

$$\text{Dates} := \pi_{\text{utorid}}(\sigma_{\text{date}>\text{'2020-03-17'} \wedge \text{date}<\text{'2021-12-31'}}(\text{Occupancy}))$$

*/*get the utorids of all members that are not allowed in at least one room*/*

$$\text{Rooms} := \pi_{\text{utorid}}(\pi_{\text{utorid, roomid}}(\text{Room} \times \text{Member}) - \text{Approved})$$

*/*project utorids of members that satisfy both conditions*/*

$$\text{Dates} \cap \text{Rooms}$$

vii)

Cannot be done in RA – there is no relational algebra operator that sums up tuple entries.

viii)

*/*get students with vaxstatus 0*/*

$$\text{VaxZeroStud} := \sigma_{\text{vaxstatus}=0}(\text{Student} \bowtie \text{Member})$$

*/*get occasions where alertlevel>threshold*/*

$$\text{Alert} := \sigma_{\text{alertlevel}>\text{alertthreshold}}(\text{Occupancy} \bowtie \text{Room})$$

*/*project utorid and email*/*

$$\pi_{\text{utorid, email}}(\text{VaxZeroStud} \bowtie \text{Alert})$$