(1) Schema 1: Convert RA to English

(a) $\pi_{sname}(\pi_{sid}((\sigma_{tagname='PPE'}ProductTag) \bowtie (\sigma_{cost<6}Catalog)) \bowtie Suppliers)$
Names of the suppliers who have a product tagged as 'PPE' and the cost is less than $6

(b) $\pi_{sname}(\pi_{sid}((\sigma_{tagname='PPE'}ProductTag) \bowtie (\sigma_{cost<6}Catalog) \bowtie Suppliers))$
Invalid query. The query simplifies to $\pi_{sname}\pi_{sid}(Stuff)$.
Cannot project attribute sname when there is no sname in $\pi_{sid}Stuff$

(c) $\pi_{sname}((\sigma_{tagname='PPE'}ProductTag) \bowtie (\sigma_{cost<6}Catalog) \bowtie Suppliers) \cap$
$\pi_{sname}((\sigma_{tagname='SuperTech'}ProductTag) \bowtie (\sigma_{cost<6}Catalog) \bowtie Suppliers)$
Names of suppliers who supply a product tagged as PPE and a product tagged as SuperTech that costs less than $6. In the case that there are duplicate supplier names, this RA only guarantees that at least one of those suppliers with duplicate names satisfies the sentence above.

(d) $\pi_{sid}((\sigma_{tagname='PPE'}ProductTag) \bowtie (\sigma_{cost<6}Catalog) \bowtie Suppliers) \cup$
$\pi_{sid}((\sigma_{tagname='SuperTech'}ProductTag) \bowtie (\sigma_{cost<50}Catalog) \bowtie Suppliers)$
SIDs of suppliers supplying a product tagged as PPE that costs less than $6, or a product tagged as SuperTech that costs less than $6

(e) $\pi_{sname}((\pi_{sid,sname}((\sigma_{tagname='PPE'}ProductTag) \bowtie (\sigma_{cost<6}Catalog) \bowtie Suppliers)) \cap$
$(\pi_{sid,sname}((\sigma_{tagname='SuperTech'}ProductTag) \bowtie (\sigma_{cost<6}Catalog) \bowtie Suppliers)))$
Names of suppliers who supply a product tagged as PPE that costs less than $6 and a product tagged as SuperTech that costs less than $6. See part c for the difference.

(2) Schema 1: Write RA

(i) Find the names of the suppliers who supply some 'PPE' or 'Testing' product
$\pi_{sname}(\pi_{sid}((\sigma_{tagname='PPE' \vee tagname='Testing'}ProductTag) \bowtie Catalog) \bowtie Suppliers)$

Commentary: The projection of sid is not strictly needed here, but is a nice touch. You can also do two separate queries and then perform $\cup$ on the sets. Be careful to use set operators like $\cup$ outside of the set, not inside. Also do not write words like 'OR' in replacement for symbols.

(ii) Find the sids of the suppliers who supply 'PPE' lower than $10 and greater than $420
$PPESuppliers(sid, pid) := \sigma_{tagname='PPE'}(ProductTag \bowtie Catalog)$
$\pi_{sid}(\sigma_{cost<10}PPESuppliers) \cap \pi_{sid}(\sigma_{cost>420}PPESuppliers)$

Commentary: Need to use set operators, specifically intersection here. $\wedge$ will not work as it cannot reference more than one row at a time.

(iii) Find the sids of the suppliers who do not supply 'PPE' lower than $10 or greater than $1337
$PPE(sid, pid) := \pi_{sid,pid}\sigma_{tagname='PPE'}(ProductTag)$
$PPELowerThan10(sid, pid) := \pi_{sid,pid}\sigma_{cost<10}(PPE \bowtie Catalog)$
$PPEGreaterThanOrEqualTo10(sid, pid) := \pi_{sid,pid}(Catalog) - PPEMoreThan10$
$PPEMoreThan1337(sid, pid) := \pi_{sid,pid}((\sigma_{cost>1337}ProductTag) \bowtie Catalog)$
$Result(sid) := \pi_{sid}(PPEGreaterThanOrEqualTo10 \cup PPEMoreThan1337)$

Commentary: Many of you assumed this was $\neg(A \vee B)$. However, we do not use the word 'nor' here. So, it is $\neg A \vee B$. Someone had asked about this on piazza and I sent a few links on the difference of OR and NOR their way. It seems like I was not on the same page with the vast majority of people, so no one was penalized.

(v) Find pairs of sids such that the supplier with the first sid charges 20% or more for some product than the supplier with the second sid.
$SameProduct20PercentMore(sid1, sid2) :=$
$\pi_{a.sid,b.sid}\sigma_{a.sid<b.sid,a.pid=b.pid,a.cost*1.2>=b.cost}(\rho_a Supplier) \times (\rho_b Supplier)$

Commentary: Many people did not use $a.sid < b.sid$ and used $a.sid \neq b.sid$ instead. This will result in duplicates like the following table.

a.pid	b.pid	a.cost	b.cost
1	2	1.20	1
2	1	1	1.20

In Relational Algebra, we want to remove duplicates by default unless specified otherwise. You were only penalized once for this, and was not docked marks on other questions. You should also rename the final result attributes from a.sid, b.sid to sid1 and sid2 for consistency.

(vii) Find the sid selling the most expensive 'Super Tech' located in 'USA'
$SuperTechUSA(sid, pid) := \pi_{sid,pid}\sigma_{tagname='SuperTech' \wedge scountry='USA'}(Supplier \bowtie ProductTag)$

$NotMostExpensive(sid, pid) := \pi_{sid,pid}\sigma_{a.cost<b.cost}(\rho_a SuperTechUSA) \times (\rho_b SuperTechUSA)$

$MostExpensive(sid) := \pi_{sid}(SuperTechUSA - NotMostExpensive)$

Commentary: We should project the composite key (sid, tid) instead of just sid. This will ensure the unique entries remain unique. The difference is highlighted in the table below. Using only sid will result in the same answer, but you should still write it this way for best practice. You may not get so lucky in other queries.

sid	pid
1	1
1	2

sid
1

(viii) Find the sid selling the second most expensive 'Super Tech' located in 'USA'
$SuperTechUSA(sid, pid) := \pi_{sid,pid}\sigma_{tagname='SuperTech' \wedge scountry='USA'}(Supplier \bowtie ProductTag)$

$NotMostExpensive(sid, pid) := \pi_{sid,pid}\sigma_{a.cost<b.cost}(\rho_a SuperTechUSA) \times (\rho_b SuperTechUSA)$

$MostExpensive(sid) := \pi_{sid}(SuperTechUSA - NotMostExpensive)$

$SuperTechUSAExceptMostExpensive(sid, pid) := \pi_{sid,pid}(SuperTechUSA - MostExpensive)$

$NotMostOrSecondMostExpensive(sid, pid) :=$
$\pi_{sid,pid}\sigma_{a.cost<b.cost}(\rho_a SuperTechUSAExceptMostExpensive) \times (\rho_b SuperTechUSAExceptMostExpensive)$

$SecondMostExpensive(sid) := \pi_{sid}(SuperTechUSA - NotMostOrSecondMostExpensive)$

(ix) Find the pids of products supplied by every supplier at less than \$69
$SuppliedByEverySupplierLessThan69(pid) := \pi_{pid}(\pi_{pid,sid}\sigma_{cost<69}Catalog)/\pi_{sid}Suppliers)$

Commentary: It is necessary to specify the attributes when doing division here. You can also replace the Supplier Table with Catalog since all we need is sid. This assumes that every supplier has a catalog, and vice versa. People also did this the manual way, which is cool too. But on a test just write the division operator to save time.

(x) Find the pids of products of which we do not have any in our inventory
$(\pi_{pid}Catalog - \pi_{pid}Inventory) \cup \pi_{pid}\sigma_{quantity=0}Inventory$

Commentary: Need to take into account the items we have in our inventory with quantity 0 and items that we have never bought from the catalog before. Someone asked about this on piazza and I laid out the conditions. The ambiguity that exists here is by design. Writing queries is not only about the technical parts, but also spotting the potential communication edge cases.

(3) Schema 1: Write Harder RA

(iii) Find the pids that have been listed as at least 3 different tags. However, one of the tags must be 'PPE', and one of them must not be 'Super Tech'. Return columns containing the pid, uname, cost.

$ProductTagPPE(pid) := \pi_{pid}\sigma_{tagname='PPE'}ProductTag$

$ProductTagSuperTech(pid) := \pi_{pid}\sigma_{tagname='SuperTech'}ProductTag$

$ProductWithAtLeastThreeTags(pid) :=$
$\pi_{a.pid}\sigma_{a.pid=b.pid=c.pid,a.tid<b.tid<c.tid}(\rho_a ProductTag) \times (\rho_b ProductTag) \times (\rho_b ProductTag)$

$AtLeastThreeTagsWithPPEWithoutSuperTech(pid, pname, cost) :=$
$\pi_{pid,pname,cost}(ProductWithAtLeastThreeTags \cap ProductTagPPE) - ProductTagSuperTech$

Commentary: We must use set operators to do this question. Most of you tried to force the tagname conditions here doing things like
$\sigma_{a.tagname="PPE" \vee b.tagname="PPE" \vee c.tagname="PPE"}(A \times B \times C)$
$\sigma_{a.tagname\neq"SuperTech" \vee b.tagname\neq"SuperTech" \vee c.tagname="SuperTech"}(A \times B \times C)$

Here is an example of where that will not work. We have 1 pid but 4 different tagnames. One of the tags is 'PPE', and we also have a 'SuperTech'. This violates our conditions. So, if our query is correct, the returning result should have no entries.

pid	tid	tagname
1	1	PPE
1	2	SuperTech
1	3	Vaccine
1	4	Testing

We start by the question by applying the cartesian product
$\sigma_{a.pid=b.pid=c.pid,a.tid<b.tid<c.tid}(\rho_a ProductTag) \times (\rho_b ProductTag) \times (\rho_b ProductTag)$

We get the following table

a.pid	b.pid	c.pid	a.tid	b.tid	c.tid	a.tagname	b.tagname	c.tagname
1	1	1	1	2	3	PPE	SuperTech	Vaccine
1	1	1	1	2	4	PPE	SuperTech	Testing
1	1	1	1	3	4	PPE	Vaccine	Testing
1	1	1	2	3	4	SuperTech	Vaccine	Testing

Then, apply the conditions many of you did

$\sigma_{a.tagname="PPE" \vee b.tagname="PPE" \vee c.tagname="PPE"}(A \times B \times C)$

$\sigma_{a.tagname \neq "SuperTech" \vee b.tagname \neq "SuperTech" \vee c.tagname="SuperTech"}(A \times B \times C)$

We get the following table

a.pid	b.pid	c.pid	a.tid	b.tid	c.tid	a.tagname	b.tagname	c.tagname
1	1	1	1	3	4	PPE	Vaccine	Testing

Bam. Somehow this cheeky guy made it through even though our result should have no leftover entries. The way to do this question is to use set operators. The difference operator here will make sure any pid containing 'SuperTech' is ultimately removed.

(4) Schema 2: Write RA

(i) Find the utorids of students not approved to be in room 'IC404'

$StudentsApprovedInIC404(utorid) := \pi_{utorid}\sigma_{roomname='IC404'}(Approved \bowtie Room)$

$StudentsNotApprovedInIC404(utorid) := \pi_{utorid}(Student) - StudentsApprovedInIC404$

Commentary: Must use the difference operator here. It is possible to have entries in the Approved table where the same student is not approved for IC404, but is approved for another room. In that case, something like $(\sigma_{room \neq 'IC404'}(Approved) \bowtie Student)$ will allow the latter entry to pass through, which would break our query.

(ii) Find the utorids of employees that are approved for at least 3 rooms

$EAR = EmployeesApprovedRooms(utorid, room) := \pi_{utorid,room}(Approved \bowtie Employee)$

$EmployeesApprovedAtLeastThreeRooms(utorid) :=$
$\pi_{utorid}\sigma_{a.roomid<b.roomid<c.roomid, a.utorid=b.utorid=c.utorid}(\rho_a EAR \times \rho_b EAR \times \rho_b EAR)$

(iii) Find the utorids of employees that are approved for exactly 3 rooms

$EAR = EmployeesApprovedRooms(utorid, room) := \pi_{utorid,room}(Approved \bowtie Employee)$

$EmployeesApprovedAtLeastThreeRooms(utorid) :=$
$\pi_{utorid}\sigma_{a.roomid<b.roomid<c.roomid, a.utorid=b.utorid=c.utorid}(\rho_a EAR \times \rho_b EAR \times \rho_b EAR)$

$EmployeesApprovedAtLeastFourRooms(utorid) :=$
$\pi_{utorid}\sigma_{a.roomid<b.roomid<c.roomid<d.roomid, a.utorid=b.utorid=c.utorid=d.utorid}(\rho_a EAR \times \rho_b EAR \times \rho_c EAR \times$

$\rho_d EAR)$

$ExactlyThrice(utorid) := EmployeesApprovedAtLeastThreeRooms - EmployeesApprovedAtLeastFourRooms$

Commentary: For reasons similar to the conflicting tagnames question, we cannot try to "manually" code this out.

(iv) Find the utorids of employees that are approved for at most 3 rooms

$EAR = EmployeesApprovedRooms(utorid, room) := \pi_{utorid,room}(Approved \bowtie Employee)$

$EmployeesApprovedAtLeastFourRooms(utorid) :=$
$\pi_{utorid}\sigma_{a.roomid<b.roomid<c.roomid<d.roomid,a.utorid=b.utorid=c.utorid=d.utorid}(\rho_a EAR \times \rho_b EAR \times \rho_c EAR \times \rho_d EAR)$

$ExactlyThrice(utorid) := \pi_{utorid} Employee - EmployeesApprovedAtLeastFourRooms$

(vi) Find utorids of members that traveled between 2020-03-17 and 2021-12-31 but not approved to be in at least one room

$NotApprovedAtLeastOneRoom(utorid) := \pi_{utorid} Member - \pi_{utorid} Approved$

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

$IllegalTravellers(utorid) := NotApprovedAtLeastOneRoom \bowtie TravelBetweenGivenTime$

Commentary: There are some disputes on if 'between' includes the endpoints or not. From an algebraic perspective, between does not include the endpoints. However, when people use between for dates, it's generally acknowledged to include the endpoints. For this reason, both options got the mark. Again, another example of why spotting potential miscommunication or edge cases are important.