i

select name

from renteree

where memberno in (select memberno from dvd\_rental)

order by name asc

ii

(select storename

from dvd\_store

order by storename asc)

Except nm

(select storename

from dvd\_store

where storeid in (select distinct storeid

from dvd\_rental join movie on dr.movieid = m.movieid

where producername <> “DreamWorks”)

order by storename asc)

I got

SELECT DISTINCT StoreName

FROM DVD\_STORE

WHERE NOT EXISTS (SELECT \*

FROM (DVDs NATURAL JOIN Movie) AS m

WHERE

m.StoreID = StoreID

AND

m.ProducerName <> “DreamWorks”)

ORDER BY StoreName ASC;

*Idea: (all stores) minus (stores that have at least 1 non-DreamWorks movie)*

SELECT StoreName

FROM DVD\_STORE

WHERE “DreamWorks” = ALL (SELECT ProducerName

FROM DVDs NATURAL JOIN MOVIE

WHERE DVD\_STORE.StoreID = StoreID)

ORDER BY StoreName

iii

(select DISTINCT storename

from dvd\_store

order by storename asc)

except

(select storename

from dvd\_store

where storeid in (select distinct storeid

from dvd\_rental dr join movie m on dr.movieid = m.movieid

where producername = “DreamWorks”)

order by storename asc)

*Idea: (all stores) minus (stores that have at least 1 DreamWorks movie)*

iv

select storename

from dvd\_store

where storeid in (

select storeid, count(memberno) as mem\_count

from (

select distinct storeid, memberno

from dvd\_rental)

group by storeid

having mem\_count > 500)

order by storename asc

Cant you just use count(distinct memberno) and remove need for inner subquery?

SELECT ds.StoreName FROM DVD\_STORE ds WHERE

Ds.StoreID IN

(SELECT dr.StoreID, COUNT(DISTINCT ds.MemberNo) num\_members from DVD\_RENTAL

GROUP BY dr.StoreID

HAVING num\_members > 500)

v

select memberno from

(select memberno, count(storeid) as store\_count

from (select distinct storeid, memberno

from dvd\_rental)

group by memberno

having store\_count = 1)

order by memberno asc

SELECT MemberNo

FROM DVD\_RENTAL

GROUP BY MemberNo

HAVING COUNT(DISTINCT StoreID) = 1

ORDER BY MemberNo

**Patient** (pat\_id (PK), Fname, Lname)

**Medical\_Examination** (unique\_num (PK), pat\_id (FK), Name, doc\_id (FK))

pat\_id references Patient.pat\_id

doc\_id references Doctor.doc\_id

**Diagnostic\_Test** (Name (PK), unique\_num(PK, FK))

unique\_num references Medical\_Examination.unique\_num on delete cascade

**Decides** (unique\_num(PK, FK), diagnosticTestName(PK, FK), DiagnosticTestMedicalExaminationUniqueNum (PK, FK), Doc \_id(FK))

unique\_num references Medical\_Examination.unique\_num on delete cascade

diagnosticTestName references Diagnostic\_Test.Name on delete cascade

DiagnosticTestMedicalExaminationUniqueNum references DiagnosticTest.unique\_num on delete cascade

Doc\_id references Doctor.doc\_id on delete cascade

**Conducts** (unique\_num (PK, FK), Name (PK, FK), doc\_id(PK, FK))

unique\_num references Diagnostic\_Test.unique\_num on delete cascade

Name references Diagnostic\_Test.Name on delete cascade

doc\_id references Doctor.doc\_id on delete cascade

**Doctor** (doc\_id(PK), Fname, Lname)

**Hospitalize** (doc\_id(PK, FK), pat\_id (PK, FK), admission date)

doc\_id references Doctor.doc\_id on delete cascade

pat\_id references Patient.pat\_id on delete cascade



BD -> AC

C -> E

B -> CE

D -> C

B -> A

AB -> E

ABD -> E

C -> AB

Rewrite the mf with only one attribute on RHS:

BD -> A(A B ➝ C)

BD -> C

C -> A

C -> B

C -> E

B -> A

B -> C

B -> E

D -> C

AB -> E

ABD -> E

Now remove all the extraneous shit on the LHS:

D -> A (B removed since we have D -> C and C -> A)

D -> C (B removed since we have D -> C)

C -> A

C -> B

C -> E

B -> A

B -> C

B -> E

D -> C

B -> E (A removed since we have B -> E)

D -> E (A and B removed since we have D -> C -> B -> E)

Get rid of redundant FDs using some transitivity 𝓂𝒶𝑔𝒾𝒸 and you get a canonical cover:

D -> C

C -> B

B -> ACE



E -> BD

D -> E

E -> B

Unique RHS:

E -> B

E -> D

D -> E

E -> B

Remove redundant stuff:

E -> BD

D -> E

* 1. D = {R(ABCD), R(DE), R(BCE)}, no relation is a subset of another relation and the key (BD) is included in the first relation so it’s all gucci.

$(

Bit nicer if u do:

R0(BCE) :B -> CE

R1(DE): D -> E

R2(ABD): BD -> A (BD) is a superkey of R

D -> BE

DE -> AB

A -> B

DF -> BE

C -> F

F -> BE

F -> C

A -> EF

CE -> A

To canonical cover

D -> A

A -> F

C -> F

F -> BCE

C -> A

Key: D

R0(AD)

R1(BCEF)

R2(AF)

E.

DF ->BE

BF -> CE

E -> F

AD ->EF

Canonical cover:

DF ->B

BF ->CE

E ->F

AD ->E

Candidate key AD:

Decomposing on E->F

R1(EF)