# Question 1

## Part a

No tuples are definitely in S.

The following tuples (X,Y,Z) may be in S:

* (5,1,8)
* (5,1,9)
* (6,1,8)
* (6,1,9)

S must have between 2 and 4 tuples inclusive.

Y is definitely not a key for S. In S1, it can be seen that for tuples where the Y value is 1, there exists two tuples with different X values, and hence there must be at least two tuples in S with the same Y value (wof 1).

## Part b

B+ = {A,B,C,D,E}

## Part c

1. A → B C D
2. B C → D E
3. B → D
4. D → A
5. A → B C decomposition (1)
6. A → D E transitivity (5,2)
7. A → A B C D E augmentation (1,6) and set union for A
8. A F → A B C D E F set union for F (Isn’t this “augmentation with F”?)

By (8), we have shown that {A,F}+ = {A,B,C,D,E,F}, which is the set of all attributes contained in R and therefore, {A,F} is a key.

1. A → B C D
2. B C → D E
3. B C → B C D E augmentation(2)
4. B C D → B C D E augmentation(3)
5. A → B C D E transitivity (1,4)
6. A → A B C D E augmentation(5)
7. A F → A B C D E F augmentation(6)

## Part d

***If you are knowledgeable, please comment on which approach is better/correct Tto use.***

### Approach 1

A → B C ~~D~~ extraneous D

B ~~C~~ → .~~D~~ E extraneous D

B → D

D → A

Combine second and third FDs to get {A → B C; B → D E; D → A}

### Approach 2 - Splitting RHS

Split RHS of all FDs to form set {A→B, A→C, A→D, BC→D, BC→E, B→D, D→A}

Foreach FD with more than one LHS attribute, check if each attribute is extraneous

C extraneous in BC→D

C extraneous in BC→E

Remaining FDs: {A→B, A→C, A→D, B→E, B→D, D→A}

Foreach FD check if RHS contained in set of FDs without this FD

If so, it is not necessary and remove it

Leaves: {A→B, A→C, B→E, B→D, D→A}

Apply union of same LHS: {A→BC, B→DE, D→A}

## Part e

(A,B,C) (A,D) (A,F) (B,D,E)

## Part f

(A,B,C,D) (A,E) (A,F)



# Question 2

## Custom views used

CREATE VIEW xplanepilot AS

SELECT \* FROM pilot natural join plane;

CREATE VIEW xpilotemp AS

SELECT \* FROM pilot natural join employee;

CREATE VIEW xplanepilotemp AS

SELECT \* FROM xplanepilot natural join xpilotemp;

## Part a

SELECT eID, name

FROM xpilotemp

WHERE eID NOT IN

( SELECT eID

FROM xplanepilot

WHERE make <> ‘Airbus’ OR model <> ‘A380’

)

;

## Part b

SELECT fID

FROM flight

WHERE distance <=

( SELECT min(range)

FROM xplanepilotemp

WHERE salary > 150000

)

;

## Part c

SELECT eID, name

FROM employee

WHERE eID IN

( SELECT eID FROM xplanepilot WHERE range > 5000 )

AND eID NOT IΝ

( SELECT eID FROM xplanepilot WHERE make = ‘Airbus’ )

;

## Part d

CREATE VIEW mostplanesperpilot AS

SELECT count(pID) as c

FROM xplanepilot

GROUP BY eID

ORDER BY c DESC

LIMIT 1;

SELECT eID, name, count(pID) as c

FROM xplanepilotemp

WHERE c = mostplanesperpilot;

## Part e

It is easy to determine whether there is a route consisting of just a single flight. The SQL query for this is trivial:

SELECT fID

FROM flight

WHERE fromCity = 'London' AND toCity = 'Sydney';

However, if there is no single-flight solution, we must use an inner join between two flight tables to determine whether there exists a route made up of two flights. However, this query would not show single-flight routes. The SQL query would be:

SELECT f1.fID, f2.fID

FROM flight f1 JOIN flight f2 ON f1.toCity = f2.fromCity

WHERE f1.fromCity = ‘London’ AND f2.toCity= ‘Sydney’;

Subsequently, we must join together three flight tables in the same way to show routes using three flights. Again, this would not show routes using less than three flights.

In conclusion, using a SQL query to determine whether a route exists between London and Sydney is possible in these simple cases I have shown, but definitely not scalable to show routes using multiple flights, as separate queries must be used to find routes with a certain number of flights involved. An SQL query is therefore feasible but not practical or exhaustive.