Part 2:- Table design

1.

Users(u_id, user_type, registration_date, email, first_name, last_name, country, zip)
u_id not null

u_id int
user_type varchar(20)
registration_date timestamp
email varchar(20)
first_name varchar(10)
last_name varchar(10)
country varchar(10)
zip number(5)

2.

Group1(group_name, g_id, abbrev)
g_id not null

group_name varchar(255)
g_id int
abbrev varchar(20)

3.

GroupJoin(group_name, g_id, datetime, u_id)
u_id is a Foreign Key referencing Users (u_id)
g_id not null

group_name varchar(255)
g_id int not null
datetime timestamp
u_id int

4.

Company(comp_id, company_name)
comp_id not null

comp_id int
company_name varchar(20)

5.

CompanyFollow(comp_id, datetime, u_id)
u_id is a Foreign Key referencing Users(u_id)
comp_id not null

comp_id int
datetime timestamp
u_id int

6.

PostShare(p_id, post_type, p_content, share_type, like_count, comment_count, datetime, sender_id, reciever_id)
p_id not null

p_id int
post_type varchar(20)
p_content varchar(255)
share_type varchar(20)
like_count number(3)
comment_count number(3)
datetime TIMESTAMP
sender_id int
reciever_id int

7.

CommentHas(c_id, c_content, is_like, is_shared, p_id, sender_id)
p_id is a Foreign Key referencing PostShare(p_id)
c_id not null

c_id int
c_content varchar(255)
is_like int
is_shared int
p_id int
sender_id int

8.

ResourceAttach(r_id, r_link, r_type, p_id)
p_id is a Foreign Key referencing PostShare(p_id)
r_id not null

r_id int
r_link varchar(20)
r_type varchar(20)
p_id int

9.

ResourceOwn(r_id, u_id)
u_id is a Foreign Key referencing Users(u_id)
r_id not null

r_id int
u_id int

10.

Connections(fromuser, touser, conn_type)
fromuser is a Foreign Key referencing Users(u_id)
fromuser not null
touser not null

fromuser int
touser int
conn_type varchar(20)

Part 4: Table design question:-

When we need to connect several tables and get specific data both IN and/or INNER JOIN can be used. IN gets only those fields which are required in a tuple i.e. some fields in a row whereas INNER JOIN gets the entire row. INNER JOIN is much faster as compared to IN as it gets the entire row itself. If a large amount of data is to be invoked, INNER JOIN proves to be more efficient than that of IN as it will directly get all the required rows instead of getting some fields from those rows and wasting enough time.
If we want to further improve its design, we combine the tables in such a way that it is possible to get all the values when required. The concept of foreign keys come here wherein we combine 2 tables using its relation and hence reduce redundancy. Also, we can further split them into tables converting them into 2NF or 3NF.i.e. normalizing the tables as possible as we can. Using these techniques we can surely eliminate redundancy and also achieve better performance than that of the individual tables.