Q1

```
----------------------------
id -> user_id
id -> date
id inv_id -> inv_confirmed
product -> p_price
id product -> p_amount
```

Q2

```
----------------------------
multivalued: id user_id date ->> inv_id inv_confirmed
inclusion:
join:      \join{id user_id date,
            id inv_id inv_confirmed,
            id product p_price p_amount
            }
```

Q3

```
----------------------------
```

apply the algorithm

```
I -> St, Si, Ss, Sd
I -> Fi, Fl, Fs, Ri, Rs
Si -> Si, Ss, Sd
Ss -> Sd
Sd -> Ss
St, Ri -> Fi
Fi -> Fl
Fi, Si -> Fs
Ri -> Rs


1.result = \emptyset
2. cover := a minimal cover

{
I -> St, Si, Ss, Sd
I -> Fi, Fl, Fs, Ri, Rs
Si -> Si, Ss, Sd
Ss -> Sd
Sd -> Ss
St, Ri -> Fi
Fi -> Fl
Fi, Si -> Fs
Ri -> Rs
}

{
I -> St
I -> Si
I -> Ss
I -> Sd
I -> Fi
I -> Fl
I -> Fs
I -> Ri
I -> Rs
Si -> Si
Si -> Ss
Si -> Sd
Ss -> Sd
Sd -> Ss
St, Ri -> Fi
Fi -> Fl
Fi, Si -> Fs
Ri -> Rs
}
```

delete Si -> Si

```
{
I -> St
I -> Si
I -> Ss
I -> Sd
I -> Fi
I -> Fl
I -> Fs
I -> Ri
I -> Rs
Si -> Ss
Si -> Sd
Ss -> Sd
Sd -> Ss
St, Ri -> Fi
Fi -> Fl
Fi, Si -> Fs
Ri -> Rs
}
```

delete I -> Ss, I -> Sd, I -> Fl, I -> Rs by transitivity

```
{
I -> St
I -> Si
I -> Fi
I -> Fs
I -> Ri
Si -> Ss
Si -> Sd
Ss -> Sd
Sd -> Ss
St, Ri -> Fi
Fi -> Fl
Fi, Si -> Fs
Ri -> Rs
}
```

delete Si -> Sd by transitivity

```
{
I -> St
I -> Si
I -> Fi
I -> Fs
I -> Ri
Si -> Ss
Ss -> Sd
Sd -> Ss
St, Ri -> Fi
Fi -> Fl
Fi, Si -> Fs
Ri -> Rs
}
```

delect I -> Fi by Union I -> St and I -> Ri and then transitivity

```
{
I -> St
I -> Si
I -> Fs
I -> Ri
Si -> Ss
Ss -> Sd
Sd -> Ss
St, Ri -> Fi
Fi -> Fl
Fi, Si -> Fs
Ri -> Rs
}
```

delete I -> Fs similar as above

```
{
I -> St
I -> Si
I -> Ri
Si -> Ss
Ss -> Sd
Sd -> Ss
St, Ri -> Fi
Fi -> Fl
Fi, Si -> Fs
Ri -> Rs
}
```

```
3: for attributes A of R such that (A,àí,ÜíX) ,àà cover do
4:    Let B = {Y | (A ,àí,Üí Y) ,àà cover}.
```

```
5:    Add relational schema with attributes A ,à™ B to result.
A = I
B = St Si Ri
result = {I St Si Ri}

...
result = {I St Si Ri, Si Ss}

...
result = {I St Si Ri, Si Ss, Ss Sd}

...
result = {I St Si Ri, Si Ss, Ss Sd}

...
result = {I St Si Ri, Si Ss, Ss Sd, St Ri Fi}

...
result = {I St Si Ri, Si Ss, Ss Sd, St Ri Fi, Fi Fl}

...
result = {I St Si Ri, Si Ss, Ss Sd, St Ri Fi, Fi Fl, Fi Si Fs}

...
result = {I St Si Ri, Si Ss, Ss Sd, St Ri Fi, Fi Fl, Fi Si Fs, Ri Rs}

6: if none of the schemas in result contain a key for R then
7: Let key be the attributes of a key of R.
8: Add relational schema with attributes key to result.
result = {I St Si Ri, Si Ss, Ss Sd, St Ri Fi, Fi Fl, Fi Si Fs, Ri Rs, I P Rp}

9: while the attributes of R,Ä≤ ,àà result are a subset of another schema in result do
10:   Remove R,Ä≤ from result.
NOP

11: return result.
result = {I St Si Ri, Si Ss, Ss Sd, St Ri Fi, Fi Fl, Fi Si Fs, Ri Rs, I P Rp}
```

holding functional dependencies

```
For I St Si Ri : {I -> St, I -> Si, I -> Ri}
For Si Ss : {Si -> Ss}
For Ss Sd : {Ss -> Sd}
For St Ri Fi : {St Ri -> Fi}
For Fi Fl : {Fi -> Fl}
For Fi Si Fs: {Fi Si -> Fs}
For Ri Rs : {Ri -> Rs}
For I P Rp : {}
```

lossless-join? Yes, all possible combinations appear in the original table.

dependency-preserving? Yes, each dependency from the original minimal cover appears at least once in the holding functional dependencies.

Decomposition

I	St	Si	Ri
1	Nov.1,1pm	1	7
2	Nov.1,1pm	2	7
3	Nov.7,2pm	2	3

Si	Ss
1	Oct.1
2	Oct.3

Ss	Sd
Oct.1	31
Oct.3	29

St	Ri	Fi
Nov.1,1pm	7	5
Nov.7,2pm	3	9

Fi	Fl
5	120
9	99

Fi	Si	Fs
5	1	great
5	2	awful
9	2	not-scored

Ri	Rs
7	medium
3	large

I	P	Rp
1	ticket	3D
1	ticket	Dolby
1	3D	3D
1	3D	Dolby
2	ticket	3D
2	ticket	Dolby
2	3D	3D
2	3D	Dolby
3	ticket	IMAX
3	IMAX	IMAX
3	ticket	4D

```
3 IMAX 4D
```

Q4

```
----------------------------
```

apply the algorithm

```
R = (I, St, P, Si, Ss, Sd, Fi, Fl, Fs, Ri, Rs, Rp)
{
I -> St Si Ss Sd,
I -> Fi Fl Fs Ri Rs,
Si -> Si Ss Sd,
Ss -> Sd,
Sd -> Ss,
St Ri -> Fi,
Fi -> Fl,
Fi Si -> Fs,
Ri -> Rs
}

violation Ss -> Sd
R1 = {Ss, Sd}
R2 = {I, St, P, Si, Ss, Fi, Fl, Fs, Ri, Rs, Rp}

violation Fi -> Fl
R2,1 = {Fi, Fl}
R2,2 = {I, St, P, Si, Ss, Fi, Fs, Ri, Rs, Rp}

violation Ri -> Rs
R2,2,1 = {Ri, Rs}
R2,2,2 = {I, St, P, Si, Ss, Fi, Fs, Ri, Rp}

violation St Ri -> Fi
R2,2,2,1 = {St, Ri, Fi}
R2,2,2,2 = {I, St, P, Si, Ss, Fs, Ri, Rp}

violation I -> St Si Ss Fs Ri
R2,2,2,2,1 = {I, St, Si, Ss, Fs, Ri}
R2,2,2,2,2 = {I, P, Rp}

violation Si -> Ss
R2,2,2,2,1,1 = {Si, Ss}
R2,2,2,2,1,2 = {I, St, Si, Fs, Ri}

violation St, Si, Ri -> Fs
R2,2,2,2,1,2,1 = {St, Si, Ri, Fs}
R2,2,2,2,1,2,2 = {I, St, Si, Ri}
```

```
return R1 U R2,1 U R2,2,1 U R2,2,2,1 U R2,2,2,2,2 U R2,2,2,2,1,1 U
R2,2,2,2,1,2,1 U R2,2,2,2,1,2,2
```

holding functional dependencies

```
For R1 : {Ss -> Sd, Sd -> Ss}
For R2,1 : {Fi -> Fl}
For R2,2,1 : {Ri -> Rs}
For R2,2,2,1 : {St Ri -> Fi}
For R2,2,2,2,2 : {}
For R2,2,2,2,1,1 : {Si -> Ss}
For R2,2,2,2,1,2,1 : {St, Si, Ri -> Fs}
For R2,2,2,2,1,2,2 : {I -> St, I -> Si, I -> Ri}
```

lossless-join?

Yes, all possible combinations appear in the original table.

dependency-preserving?

Yes, each dependency from the original minimal cover appears at least once in the holding functional dependencies.

Decomposition

Ss	Sd
Oct.1	31
Oct.3	29

Fi	Fl
5	120
9	99

Ri	Rs
7	medium
3	large

St	Ri	Fi
Nov.1	7	5
Nov.7	3	9

I	P	Rp
1	ticket	3D
1	ticket	Dolby
1	3D	3D
1	3D	Dolby
2	ticket	3D
2	ticket	Dolby
2	3D	3D
2	3D	Dolby
3	ticket	IMAX
3	IMAX	IMAX
3	ticket	4D

```
3 IMAX 4D

Si Ss
-------
1 Oct.1
2 Oct.3

St, Si, Ri, Fs
----------------
Nov.1 1 7 great
Nov.1 2 7 awful
Nov.7 2 3 not-scored

I, St,      Si, Ri
------------------
1 Nov.1,1pm 1 7
2 Nov.1,1pm 2 7
3 Nov.7,2pm 2 3
```

Q5

```
----------------------------
```

apply the algorithm

```
R = (I, St, P, Si, Ss, Sd, Fi, Fl, Fs, Ri, Rs, Rp)
{
I ->> St Si Ss Sd,
I ->> Fi Fl Fs Ri Rs,
Si ->> Si Ss Sd,
Ss ->> Sd,
Sd ->> Ss,
St Ri ->> Fi,
Fi ->> Fl,
Fi Si ->> Fs,
Ri ->> Rs,
ID ->> P,
ID ->> Rp
}
where ID = {I, St, Si, Ss, Sd, Fi, Fl, Fs, Ri, Rs}

violation Ss ->> Sd
R1 = {Ss, Sd}
R2 = {I, St, P, Si, Ss, Fi, Fl, Fs, Ri, Rs, Rp}

violation Fi ->> Fl
R2,1 = {Fi, Fl}
R2,2 = {I, St, P, Si, Ss, Fi, Fs, Ri, Rs, Rp}

violation Ri ->> Rs
R2,2,1 = {Ri, Rs}
```

```
R2,2,2 = {I, St, P, Si, Ss, Fi, Fs, Ri, Rp}

violation St Ri ->> Fi
R2,2,2,1 = {St, Ri, Fi}
R2,2,2,2 = {I, St, P, Si, Ss, Fs, Ri, Rp}

violation Si ->> Ss
R2,2,2,2,1 = {Si Ss}
R2,2,2,2,2 = {I, St, P, Si, Fs, Ri, Rp}

violation St Si Ri ->> Fs
R2,2,2,2,2,1 = {St, Si, Ri, Fs}
R2,2,2,2,2,2 = {I, St, P, Si, Ri, Rp}

return R1 U R2,1 U R2,2,1 U R2,2,2,1 U R2,2,2,2,1 U R2,2,2,2,2,1 U
R2,2,2,2,2,2
```

holding functional dependencies

```
For R1 : {Ss ->> Sd, Sd ->> Ss}
For R2,1 : {Fi ->> Fl}
For R2,2,1 : {Ri ->> Rs}
For R2,2,2,1 : {St Ri ->> Fi}
For R2,2,2,2,1 : {Si Ss}
For R2,2,2,2,2,1 : {St Si Ri ->> Fs}
For R2,2,2,2,2,2 : {I ->> P, I ->> Rp, I ->> St, I ->> Si, I ->> Ri}
```

lossless-join? Yes, all possible combinations appear in the original table.

dependency-preserving? Yes, each dependency from the original minimal cover appears at least once in the holding functional dependencies.

Decomposition

```
R1 = {Ss, Sd}
R2,1 = {Fi, Fl}
R2,2,1 = {Ri, Rs}
R2,2,2,1 = {St, Ri, Fi}
R2,2,2,2,1 = {Si Ss}
R2,2,2,2,2,1 = {St, Si, Ri, Fs}
R2,2,2,2,2,2 = {I, St, P, Si, Ri, Rp}

Ss     Sd
--------
Oct.1  31
Oct.3  29

Fi Fl
--------
5 120
9 99

Ri Rs
--------
```

7	medium
3	large

St	Ri	Fi
Nov.1,1pm	7	5
Nov.7,2pm	3	9

Si	Ss
1	Oct.1
2	Oct.3

St	Si	Ri	Fs
Nov.1,1pm	1	7	5
Nov.1,1pm	2	7	5
Nov.7,2pm	2	3	9

I	St	P	Si	Ri	Rp
1	Nov.1,1pm	ticket	1	7	3D
1	Nov.1,1pm	ticket	1	7	3D
1	Nov.1,1pm	ticket	1	7	Dolby
1	Nov.1,1pm	3D	1	7	3D
1	Nov.1,1pm	3D	1	7	Dolby
2	Nov.1,1pm	ticket	2	7	3D
2	Nov.1,1pm	ticket	2	7	Dolby
2	Nov.1,1pm	3D	2	7	3D
2	Nov.1,1pm	3D	2	7	Dolby
3	Nov.7,2pm	ticket	2	3	IMAX
3	Nov.7,2pm	IMAX	2	3	IMAX
3	Nov.7,2pm	ticket	2	3	4D
3	Nov.7,2pm	IMAX	2	3	4D

Q6

No, none of them solves all the design issues. There are always some duplicates in this example:

I	P	Rp
1	ticket	3D
1	ticket	Dolby
1	3D	3D
1	3D	Dolby
2	ticket	3D
2	ticket	Dolby
2	3D	3D

2	3D	Dolby
3	ticket	IMAX
3	IMAX	IMAX
3	ticket	4D
3	IMAX	4D

This could be avoided by further splitted off to (I, P) and (I, Rp). However, in 4NF, this cannot be done because I is a key in 4NF; in BCNF, this cannot be done, because I does not have some relation with P Rp (i.e. no violation in G+). For 3NF, it cannot all the problems that BCNF cannot solve. Therefore, all of these methods are not perfect by this analysis.