Answer 1

Artist (AName, Main Style, Epoch, Country, Born, Died)

Art Object (Id No., Artist_Name, Year, Title, Description)

Sculpture (S_id [FK to Art Object(Id No)] , Paint type, Material, Height, Weight, Styles)

Statue (Statue_id [FK to Art Object(Id No)], Honoree)

Painting (P_id [FK to Art Object(Id No)] , Material, Style)

Other (O_id [FK to Art Object(Id No)] , Type, Style)

Permanent (Perm_objId [FK to Art Object(Id No)], Cost, Date Acquired, On Display(y/n), Source)

Borrowed (Brro_id [FK to Art Object(Id No)] , Collection_Name [FK to Collection(ColName)], Date Borrowed, Date to Return)

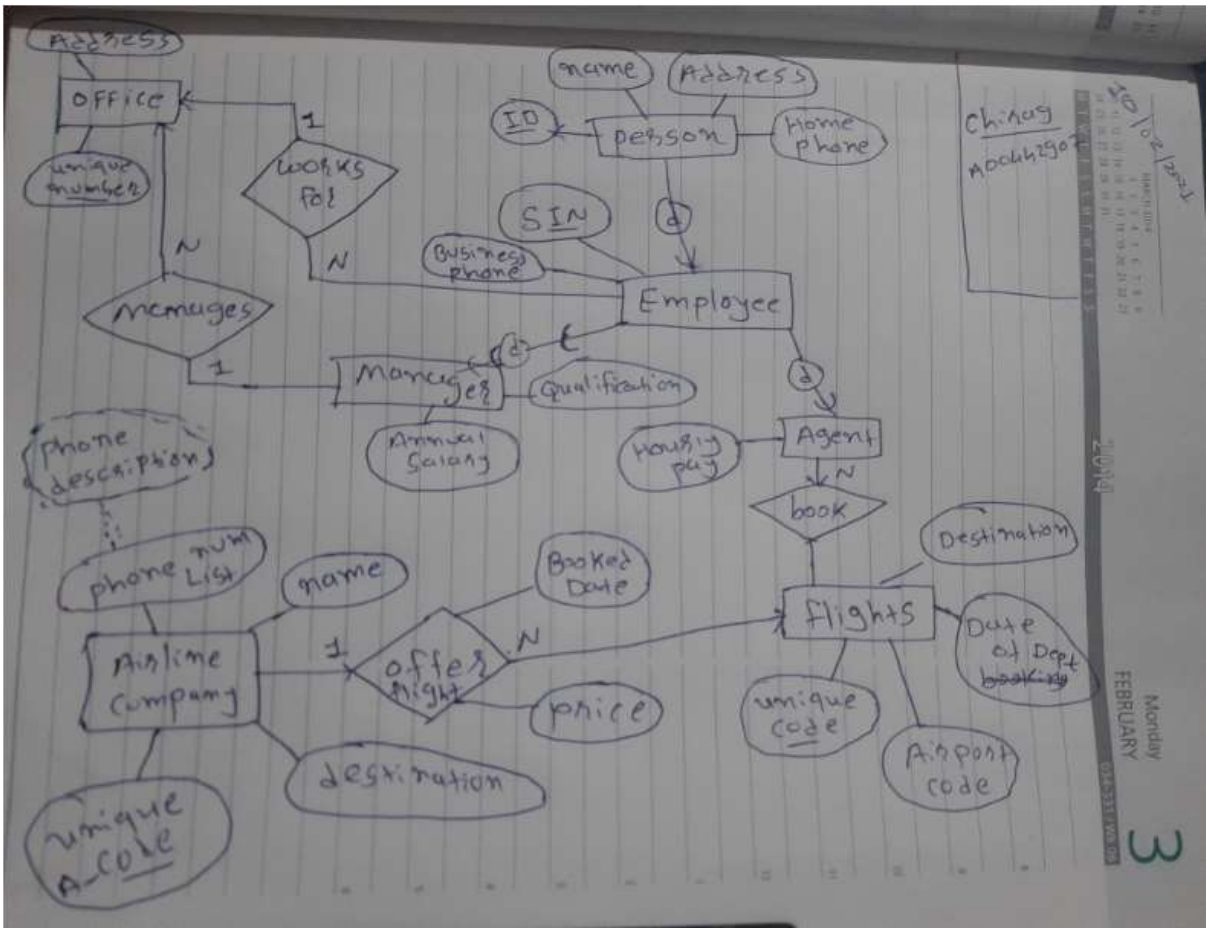
Collection (ColName, Description, Contact Person, Type, Phone Number)

Exhibition (EName, Start Date, End Date)

Shown At (Art_Id [FK to Art Object(Id No)] , Exihibition_Name [FK to Exhibition (EName)])

Answer 2

(Enhanced) Entity Relationship diagram



Answer 3 (a)

Given

1. AB -> C
2. CD -> E
3. DE -> B

My solution

4. A -> A (Reflexivity)
5. AB -> AB (Augmentation – added B)
6. AB -> ABC (1 U 2 – Union rule)
7. ABD -> ABCD (Augmentation – added D)
8. ABD -> CD (Decomposition rule on 7)
9. ABD -> E (Transitive rule on 2 and 8)
10. **ABD -> ABCDE** (Union rule on 7 and 9)

Answer 3 (b)

AB is not a super key of R ?

Because B is partially dependant on keys D and E (DE -> B)

So AB together can not be super key with given dependency.

Answer 4

BOOK (title (A), author (B), type (C), list_price (D), affiliation (E), publisher (F))
Here I have taken A,B,C,D,E,F to describe dependency from above Book table

A -> BC

C -> D

E -> F

Ans (a)

Candidate key {title,publisher}

Ans (b)

Not in 2NF -> Because list_price is depending on publisher and type. That is partial dependency

Ans (c)

Not in 3NF -> Because it's not in 2NF also list price is depends on non prime attributes publisher and type. publisher and type is not prime key

Ans (d)

Here at least one decomposed table must satisfy every dependency, Natural join should be original table Book

B1 = ABCDF, B2=BE

B1 ∩ B2 = B didn't derives B1 or B2. So this design **is additive**

A -> BC and C -> D is part of B1, but E -> F is not part of any table = which is **not dependency preserving**

2NF -> Not in 2NF, Because list_price is depending on publisher and type. That is partial dependency

3NF -> Not in 3NF, Because it's not in 2NF also list price is depends on non prime attributes publisher and type. And publisher and type is not prime key, Also type is not the Super key

Ans (e) – assigned ABCDEF to each attributes

B1 = { title (A), author (B), type (C), list_price (D)} = {A,B,C,D}

B2 = { title (A), list_price (D), affiliation (E), publisher (F) } = {A,D,E,F}

B1 ∩ B2 = AD

So AD -> ABCD because A -> BC (title -> publisher,type) Given, B1 ∩ B2 is part of B1 schema

So these tables are non additive.

All dependency A -> BC, C-> D, E->F is part of either B1 or B2 so It's dependency preserving.

Answer 5

whether the query is legal in SQL99 (with the optional feature T301)

select B, count(*) from GBy1 natural join GBy2 group by B; -->

This query doesn't have any nonaggregate column in select list which not part of group by clause. Summarizing, Query is legal because it has only B column as nonaggregate column which is part of GROUP By clause. Therefore it is an executable query as per SQL99.

select B, D, count(*) from GBy1 natural join GBy2 group by B; -->

MySQL **rejects** queries for which the select list refer to nonaggregate columns that are neither named in the GROUP BY clause (Column D is not listed) nor are functionally dependent on them (Column D is not functionally dependent only on b as (B, C) makes unique key). Therefore, it is not an executable query as per SQL99.

select B, E, count(*) from GBy1 natural join GBy2 group by B; -->

MySQL **rejects** queries for which the select list refer to nonaggregate columns that are neither named in the GROUP BY clause (Column E is not listed) nor are functionally dependent on them (Column E is not functionally dependent only on B as (B, B1) makes unique key). Therefore it is not an executable query as per SQL99.

select B, D, count(*) from GBy1 natural join GBy2 group by B, C; -->

Although the select list refer to nonaggregate columns that are not named in the GROUP BY clause , SQL99 allows if they are functionally dependent on GROUP BY columns. Here Column D is functionally dependent on unique key (B, C). This provision makes it legal in SQL99