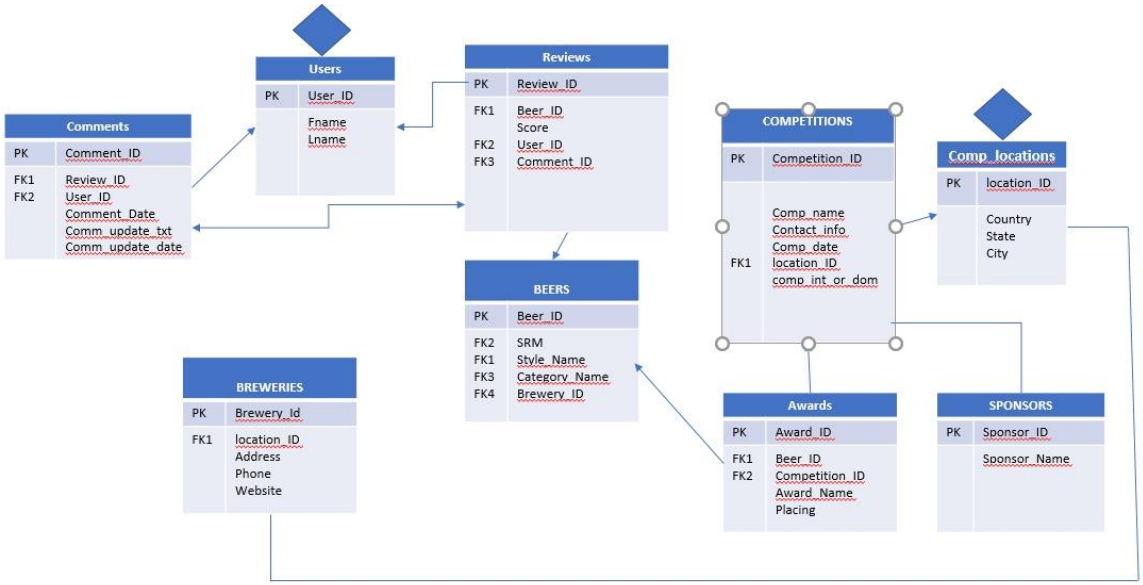
Advanced Database Management
ISM 6218
Group 2 – Sahil Shah, Alphonse Aloia, Krishna Basetty, Shruthi Athikam, Malavika Byju

1. **Update the design to fulfill the requirements specified in the scenario below. This should include developing an extended ERD.**

ERD NEW:



2. Correct any errors / deficiencies in the existing design. That is, reduce data redundancy.

   We resolved the redundancy errors and came up with the above ER diagram. We just though of an improvement In the pre existing design and that is described below.

   There is a many to one relation between beers and style, beers and catogery. Both these tables contain only one column, hence we write these 2 in the beers table itself without violating any normalization.

3. Normalize the entire design to 3rd normal form.
   1. First Normal Form: In our design we made to sure there are only single valued attributes, and as primary key constraint is enforced on each table making the records unique.

2. Second Normal Form: The design is already in 1NF, and we tried to update the design to make sure each column is only dependent on the primary key of respective table. Therefore, we created separate tables for each new requirement and formed appropriate relations. Each column of the new table that is created is entirely dependent only on the primary key.
3. Third Normal Form: The design is already in 2NF, on top of it we made sure that each non key attribute is independent of each other. As part of this we created a new table called comp_locations, that will track the location ID and store the city, state and country. For all other tables, non key attributes are independent f each other.

4. **For each synthetic primary key created either: 1) list a candidate key or, 2) explicitly state no viable candidate key exists**

    1. **User_ID** – No viable candidate key exists
    2. **Review_ID** – No single column viable candidate key. Could create composite PK of the following columns (user_id, comment_date, beer_ID)
    3. **Competition_ID** - No single column viable candidate key. Could create composite PK of the following columns (comp_date, comp_name, location_id)
    4. **Comment_ID** – No viable candidate key exists
    5. **Location_ID** – SNo viable candidate key exists. Could create composite PK of the following columns (country, state, city). This can also be similar to zipcode.
    6. **Award_ID** - No viable candidate key exists. Could create composite PK of the following columns (Competition_ID, Placing)
    7. **Sponsor_ID** - No viable candidate key exists

5. **List all critical assumptions made.**

Competitions:

1. One Location can have many different competitions
2. Many beers can be in many competitions
3. Locations has been made into its own table as its fields would not reference the PK in competitions and thus not be in accordance with 3rd Normal Form
4. Every competition must have at least one sponsor

Awards

1. Each award can only belong to one competition
2. Each award can only go to one beer
3. Each beer can have many awards
4. Each competition can have many awards

Reviews

1. Each user can submit one review per beer
2. All users in user table must have at least one review
3. Each user can make multiple comments (but only one per beer)
4. Each beer can have many reviews (by different users)
5. Each review corresponds to one beer

6. A comment update text and date would be filled in for an existing review (no new review created)
7. Users can add on to an existing comment update but not delete the original text

Sponsors

1. Many Sponsors can sponsor a single competition
2. Each sponsor can provide sponsorship for multiple competitions
3. Sponsor in the table will be added only after they have confirmed sponsorship for at least one competition

6. Assume you implemented your complete design. Write two interesting queries that can now be run using your design. Include the SQL statement for each query, along with an description.

1. Pull a list of all the international competitions

```
SELECT
        Competition_ID, comp_name,comp_date,location_id
FROM
        competitions
        LEFT JOIN comp_location on comp_location.location_id = competitions.location_id
WHERE
        County NOT LIKE 'United States'
```

2. Count how many times each beer has won first place (placing = 1) in a competition and arrange in descending order by total count, and then in alphabetical order of beer name where winning count is the same for multiple beers

```
SELECT
        COUNT(placing), beer_id,beer_name,competition_id
FROM
        Awards
        RIGHT JOIN BEERS on beers.beer_id = award.beer_id
WHERE
        placing = '1'
        ORDER BY placing DESC
        ORDER BY beer_name ASC
```