Entities

Users: This is an entity since we need to capture important personal information about each user like their authentication details as well as their preferences on fragrances.
Perfumes: The perfume entity will capture attributes of each unique fragrance. Each entry in Perfumes is unique and independent of each other. Therefore this is an entity.
Scent: The scent is an entity as it captures important information about the qualities involved in a user's preferences. Each scent encapsulates specific feelings and notes that can be match to certain preferences.
Reviews: The reviews are an entity since each review is independent of the others. Each review captures notes and feelings which mimic the format of preferences from users.
Dupes: The dupes entity details each available dupe for different perfumes and each dupe is independent of others. There is an independent amount of dupes for perfumes so this is used as a separate entity.

Relationships

Reviewed: Each perfume has 1 to many reviews (1…*). We will guarantee that each perfume application has been reviewed. This way, we know we can extract the scent details (notes, feelings) from each perfume. Each review has exactly one perfume (1…1) just as any product review applies to one product in question on sites like Amazon or Ebay.
Attributes: Each perfume will have its own unique set of scent details (1…1) since each perfume makes up a unique combination of notes and feelings that separate the perfume from others. Likewise, each set of scent details corresponds to one perfume.
HasDupe: Each perfume has from 0 to many dupes (0…*). This is because not all perfumes will have a dupe, but other more popular perfumes may have many. In the other direction, each dupe is relating to one specific perfume (1…1); a dupe can't replicate multiple perfumes at the same time.
UserPreferences: A user has from one to many scent preferences (1…*). Each user must have at least one scent preference since their input in the scent sense quiz will be required. After this, there may be more than one scent that matches their preferences. In the other direction, each scent preference will have one to many users associated with it (1…*). This is because a single scent can be preferred by multiple humans.

Functional Dependencies for Each Entity:

1. **User**
    - Primary Key: UserID
    - Functional Dependencies:
        - UserID → FirstName, LastName, Password, Notes, Feelings
2. **Perfumes**
    - Primary Key: PerfumeID
    - Functional Dependencies:
        - PerfumeID → Name, Price, Image, Brand

3. **Reviews**
    - Primary Key: ReviewID
    - Functional Dependencies:
        - ReviewID → PerfumeID, ReviewerName, Notes, Feelings
        - PerfumeID → Perfume (foreign key)
4. **Scent**
    - Primary Key: ScentID
    - Functional Dependencies:
        - ScentID → PerfumeID, Notes, Feelings
        - PerfumeID → Perfume (foreign key)
5. **Dupes**
    - Primary Key: DupeID
    - Functional Dependencies:
        - DupeID → OriginalID, ogBrand, dBrand
        - OriginalID → Perfume (foreign key)

Entity	Left Side Only	Middle	Right Side Only
User	UserID		FirstName, LastName, Password, Notes, Feelings
Perfumes	PerfumeID		Name, Price, Image, Brand
Reviews	ReviewID	PerfumeID	ReviewerName, Notes, Feelings
Scent	ScentID	PerfumeID	Notes, Feelings
Dupes	DupeID	OriginalID	ogBrand, dBrand

BCNF Proof for each Entity:

1. User (UserID, FirstName, LastName, Password, Notes, Feelings)

- **Candidate Key**: UserID
- No partial or transitive dependencies exist here. All attributes depend on UserID, the primary key.
- This is in **BCNF**.

2. Perfumes (PerfumeID, Name, Price, Image, Brand)

- **Candidate Key**: PerfumeID
- Again, no partial or transitive dependencies. All non-key attributes (like Name, Price, etc.) depend only on PerfumeID.

- This is in **BCNF**.

3. Reviews (ReviewID, PerfumeID, ReviewerName, Notes, Feelings)

- **Candidate Key**: ReviewID
- ReviewID is the primary key, and it uniquely identifies a review.
- PerfumeID is a foreign key, but no non-key attribute depends on anything other than ReviewID.
- This is in **BCNF**.

4. Scent (ScentID, Type, PerfumeID, Notes, Feelings)

- **Candidate Key**: ScentID
- All non-key attributes depend only on the ScentID.
- PerfumeID is a foreign key, and there is no partial or transitive dependency in the table.
- This is in **BCNF**.

5. Dupes (DupeID, OriginalID, oBrand, dBrand)

- **Candidate Key**: DupeID
- All attributes depend only on the DupeID, which uniquely identifies each dupe.
- OriginalID is a foreign key pointing to the original perfume, and there are no partial or transitive dependencies.
- This is in **BCNF**.

Relational Schema:

1. Table- User(UserID:VARCHAR(8) [PK], FirstName:VARCHAR(255), LastName:VARCHAR(255), Password:VARCHAR(12), Notes:VARCHAR(255), Feelings:VARCHAR(255)**)**

2. **Table- Perfumes(**PerfumeID:VARCHAR(8) [PK], Name:VARCHAR(255), Price:VARCHAR(255), Image:VARCHAR(255), Brand:VARCHAR(255)**)**

3. **Table- Reviews(**ReviewID:VARCHAR(8) [PK], PerfumeID:VARCHAR(8) [FK to Perfumes.PerfumeID], ReviewerName:VARCHAR(255), Notes:VARCHAR(255), Feelings:VARCHAR(255)**)**

4. **Table- Scent(**ScentID:VARCHAR(8) [PK], PerfumeID:VARCHAR(8) [FK to Perfumes.PerfumeID], Notes:VARCHAR(255), Feelings:VARCHAR(255)**)**

5. **Table- Dupes(**DupeID:VARCHAR(8) [PK], OriginalID:VARCHAR(8) [FK to Perfumes.PerfumeID], ogBrand:VARCHAR(255), dBrand:VARCHAR(255)**)**