Worksheet 00 – DB Concepts

1. What tables and fields might need to be included in the database?
   - Album: ID, title, genre, release date, copies
   - Media: ID, type (physical or digital)
   - Artist: ID, name
   - Track: title, track number, length
   - Checkout: ID, checkout date, return date, media id, card number (of patron who checked out)
   - Person: first name, last name, card number, activation date, email
     - Employee (Subclass of “Person”): position, salary, start date

2. What informal queries and update operations that might users expect?
   Queries:
     - List of music by artist/genre
     - List of available music
     - Currently checked out by a patron
     - Feedback given in one category
     - Reviews for a given album/artist/genre
   - Update Operations:
     - Return an album (requires update of Checkout tuple)
     - Updating an album name that was entered incorrectly
     - Updating a patron’s email address
     - Increasing an employee’s salary
     - Updating a track name that was entered incorrectly

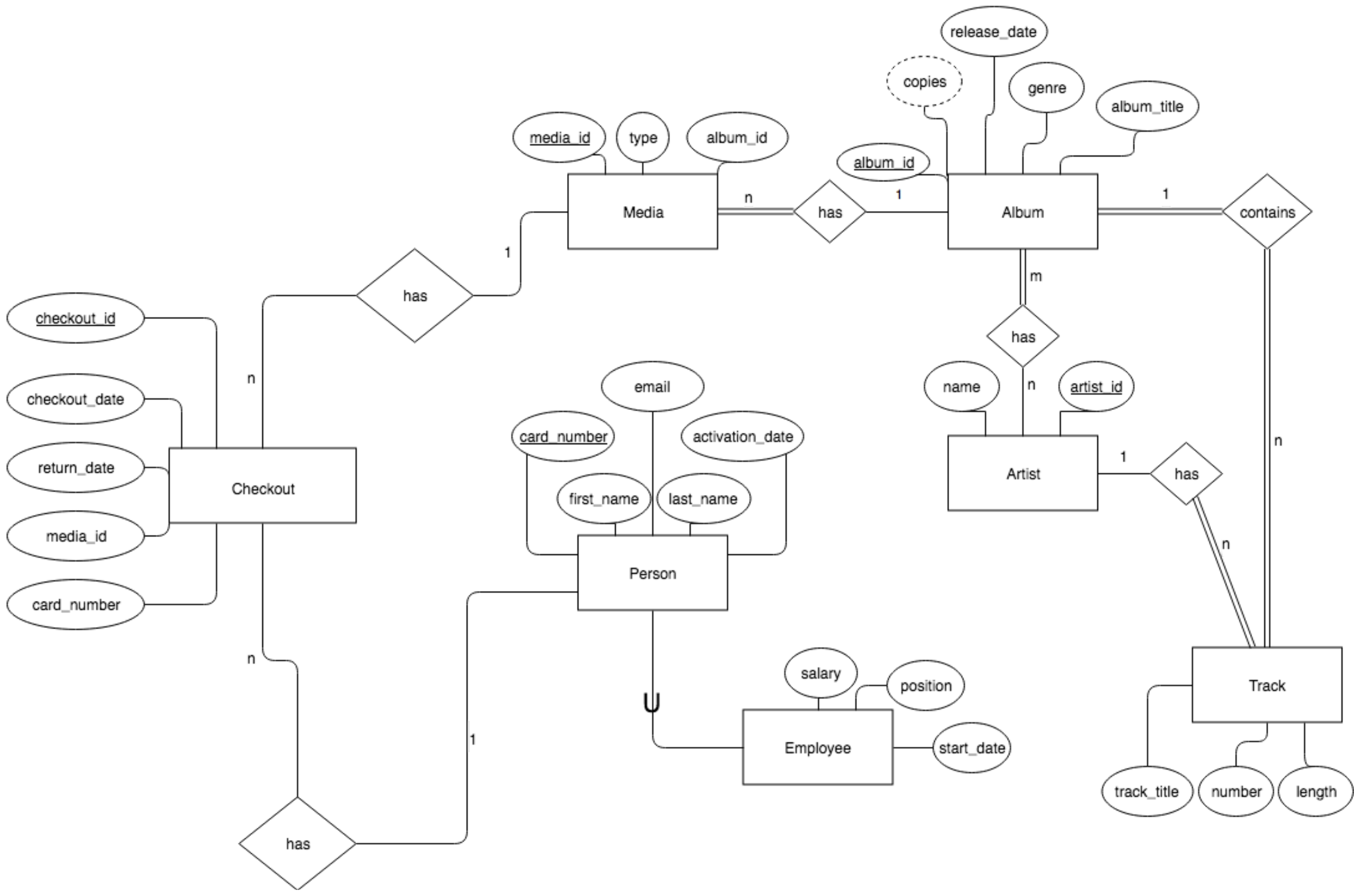
3. What kind of reports might be desired by the users?
   - All historical checkout data for a given subset of patrons
   - Album popularity over time based on review data and checkout data
   - Projected available stock for future dates based on currently available stock and return dates

4. Give some examples of integrity constraints that you think can apply to the database.
   - If an artist is deleted, corresponding albums would also need to be deleted due to foreign key constraints. This could be remedied with cascading, but it would be preferable to restrict the operation to admins only, and then cascade if the deletion is absolutely needed.
   - If a patron is deleted, the corresponding historical checkout data and corresponding reviews would also need to be deleted. The same remedy as above would also apply here.

5. If a patron changes their address, what would need to be updated? What other changes could affect the data and relationships between the tables?
   - Only the person relation would need to be updated, since nothing is linked to email address (nothing uses it as a foreign key
   - Any update the primary keys being referenced as foreign keys would affect the relationships between the tables. But, primary keys should be values that don’t really change, and thus integrity should be maintained.

Worksheet – 01 Entity-Relationship (ER)

Provide an ER diagram for your database



Worksheet – 02 Enhanced ER (EER)

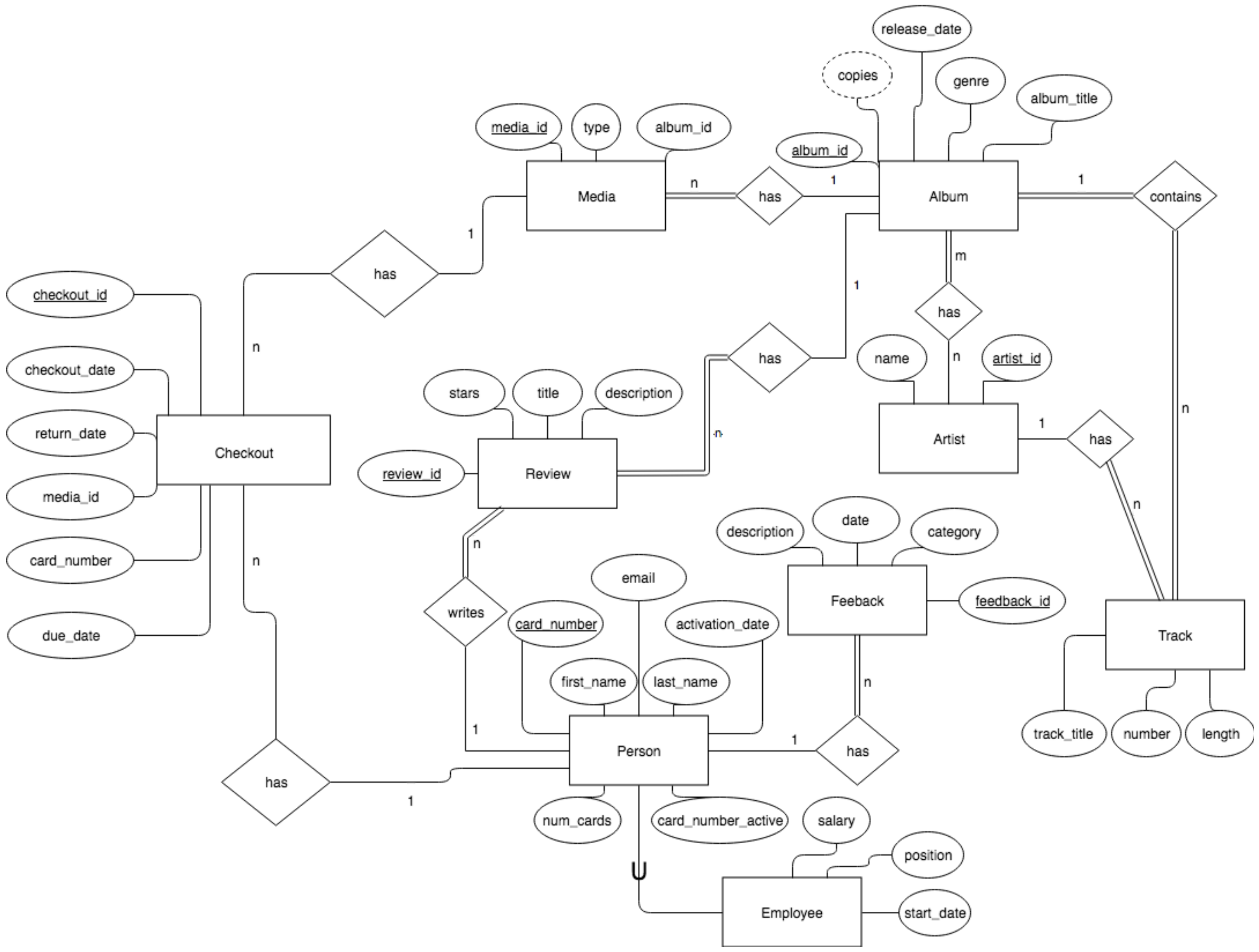
Are there any hierarchical relationships in the database? If so, what are they? Think of at least two hierarchical relationships you can add to the database.

All 1-n relationships in the EER diagram can be thought of as hierarchical. So, in our schema, we have the following hierarchical relationships:

- An album (parent) can have many media (children)
- An album (parent) contains many tracks (children)
- An artist (parent) authors many tracks (children)
- A media (parent) has many historical checkouts (children)
- A person (parent) has many historical checkouts (children)
- An album (parent) has many reviews (children)

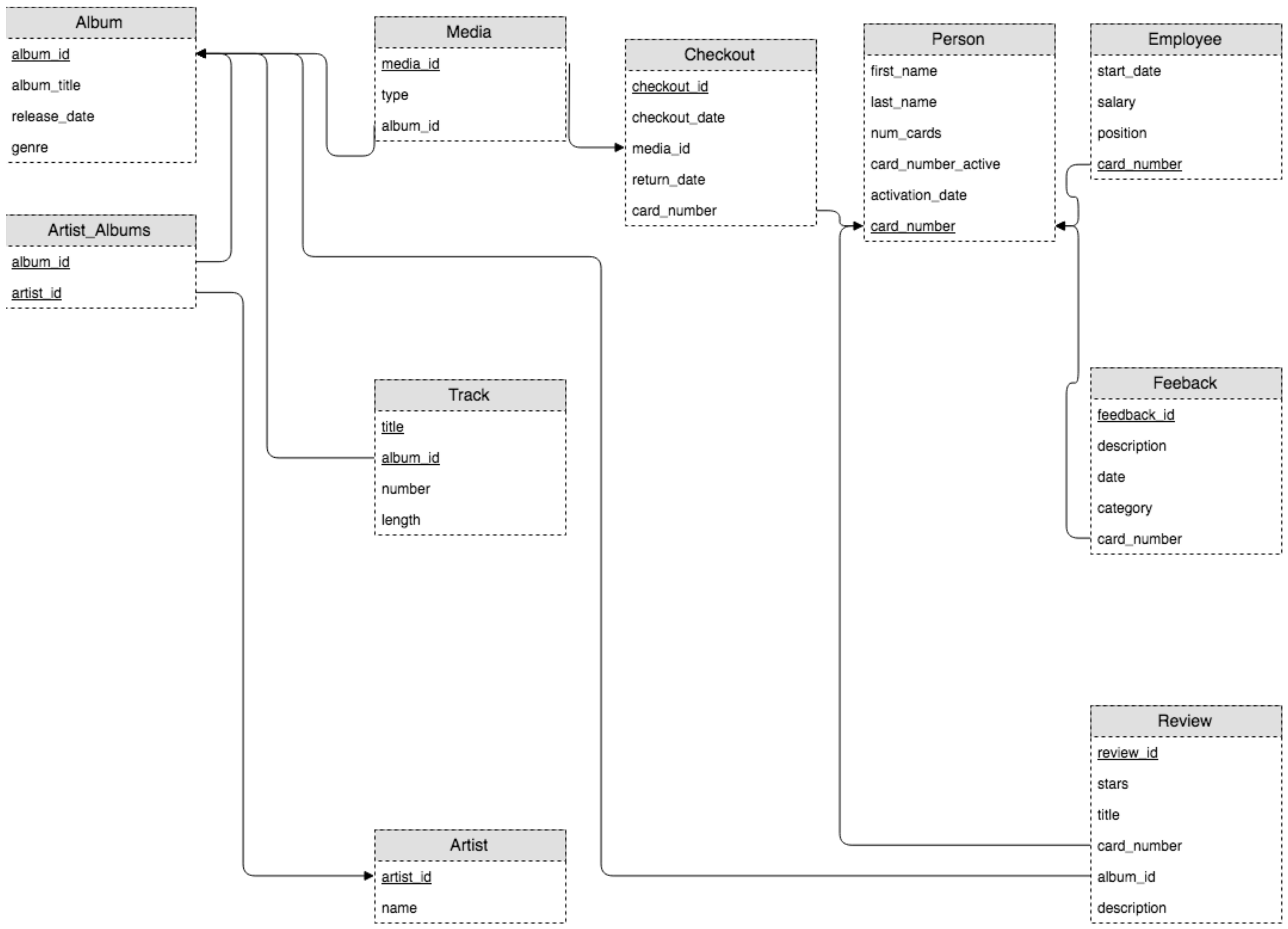
Reviews and Feedbacks are generated by a Person, and are two new hierarchical relationships added to the database (person is the parent, and reviews and feedback are the children in the relationship).

Refine your ERD to show these new relationships.



Worksheet – 03 EER to Relational Mapping

Map your EER to a relational schema and show all primary and foreign keys.



Worksheet – 04 Relational Data Model

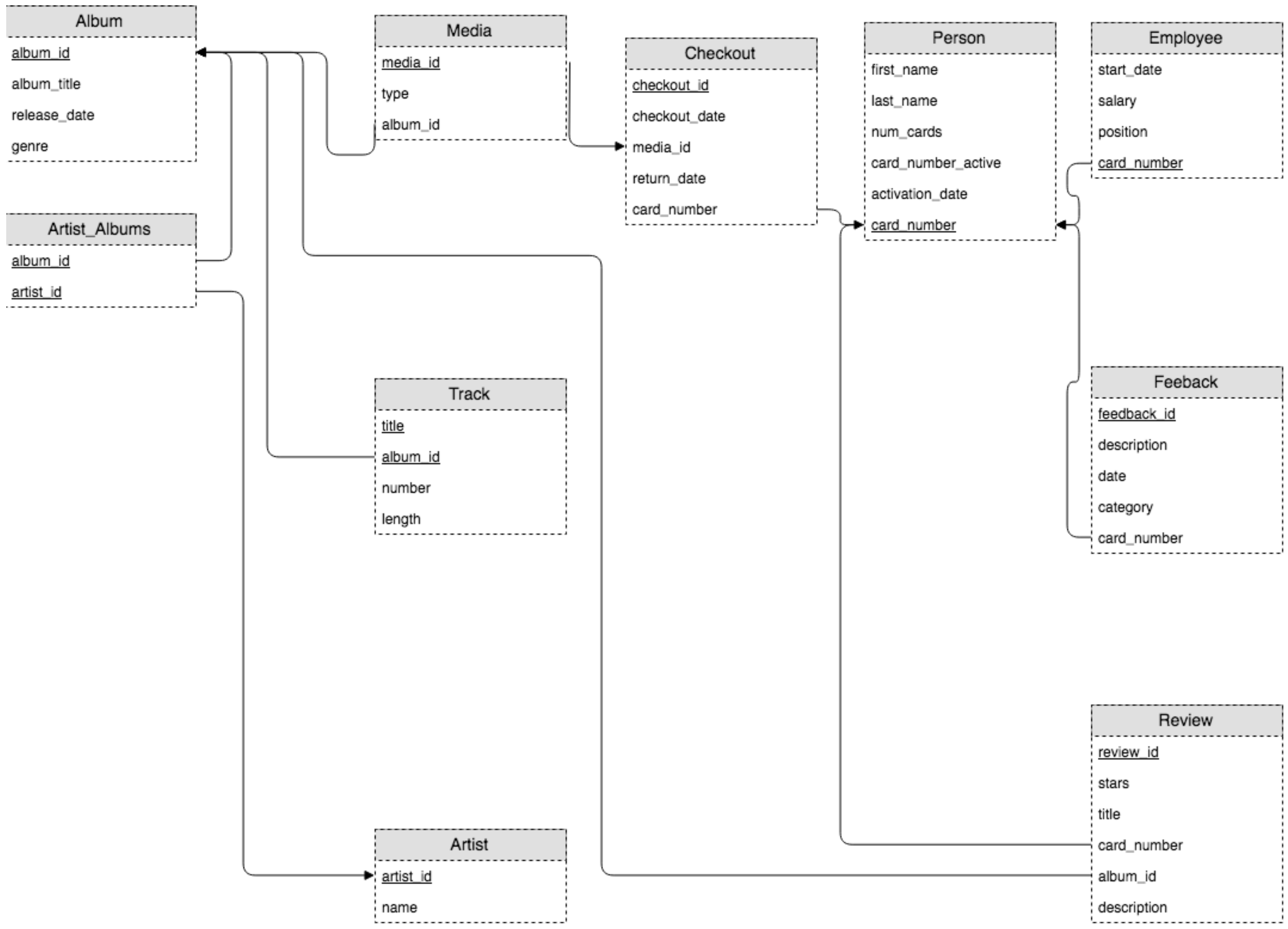
Propose ten Update operations that could be applied directly to the database. For each discuss *all* integrity constraints violated by each operation, if any, and the different ways of enforcing these constraints:

Do create operations count as updates?

	Operation	Integrity constraints
1.	Update an artist's name	None; name is not referenced as a foreign key
2.	Update a person's email	None; email is not referenced as a foreign key
3.	Update the return date of a checkout (album was returned)	None; return date is not referenced as a foreign key
4.	Update the size in bytes of tracks on an album (digital copy acquired)	None; byte size is not referenced as a foreign key
5.	Update an album's genre	None; genre is not referenced as a foreign key
6.	Update a reviews star rating	None; rating is not referenced as a foreign key
7.	Update the artist who authored an album (to correct a mistake)	Must ensure the referenced artist exists to avoid violating foreign key constraint
8.	Update a track with the correct containing album (to correct a mistake)	Must ensure the reference album exists to avoid violating foreign key constraint
9.	Update employee salary (give a raise)	None; salary is not referenced as a foreign key
10.	Update a person's number of cards (new card issued)	None; number of cards is not referenced as a foreign key

Worksheet – 05 Relational Algebra

Show schema for your database that you have so far. Indicate relationships, and foreign keys using the "arrow" notation from the book



Specify the following queries in relational algebra.

1. Find all the albums by ARTIST that were released before YEAR.

   $\pi_{\text{album\_title}}(\sigma_{\text{release\_date < YEAR}}(\sigma_{\text{name = ARTIST}}$ ARTIST*ARTIST_ALBUMS*ALBUM*TRACK))

2. Give all the albums and dates for check outs made by a particular patron.

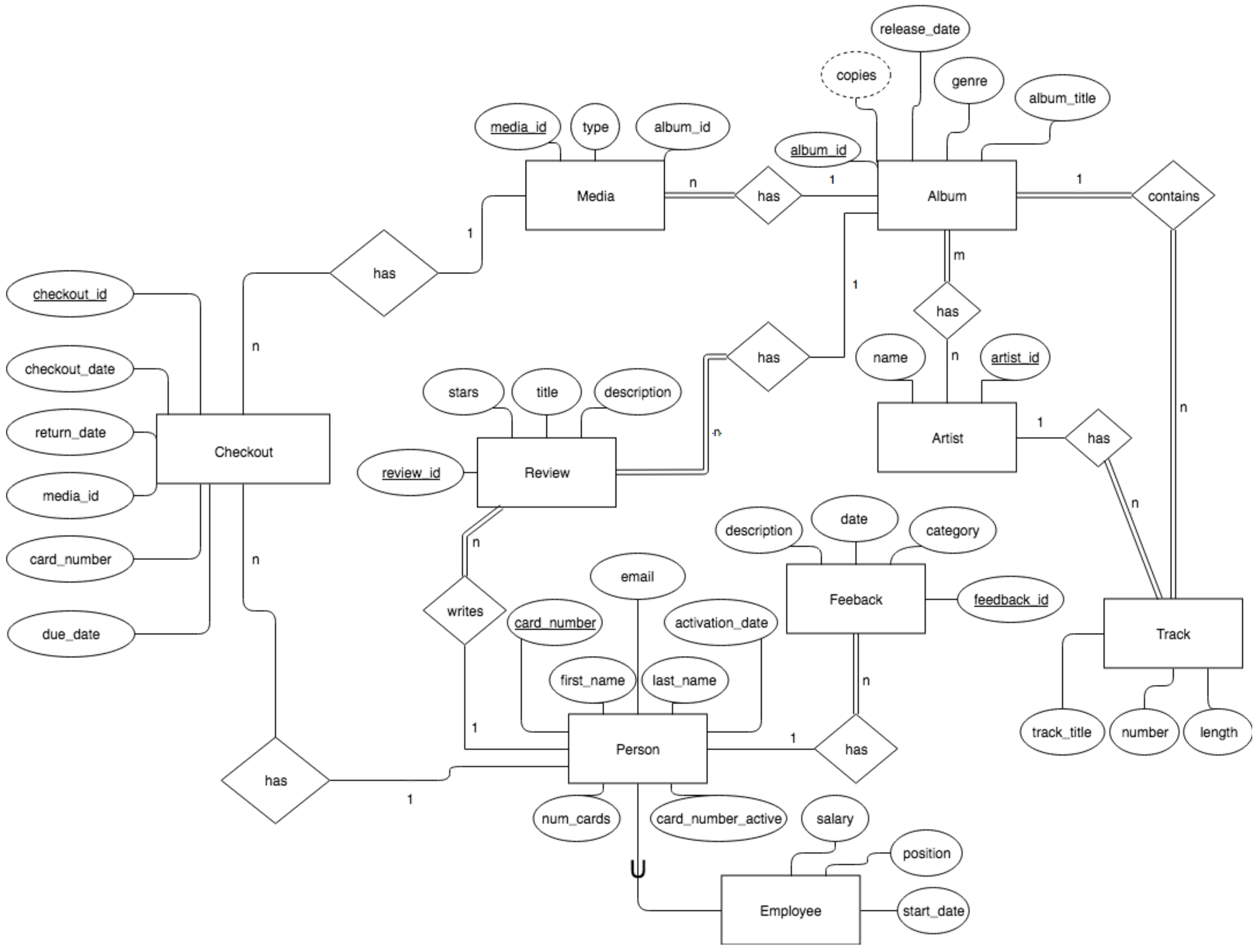
   $\pi_{\text{album\_title, checkout\_date}}\ \sigma_{\text{card\_number = CARD NUMBER}}$(PERSON*MEDIA*ALBUM)

3. List all the albums with less than 5 copies available to check out.

   $\pi_{\text{album\_id, title}}(\sigma_{\text{count < 5}}({}_{\text{album\_id}}F_{\text{COUNT album\_id}}$(ALBUM*MEDIA)))

Worksheet – 06 SQL

1. What are the referential integrity constraints that should hold on the schema?
   - Upon creating a new album, a tuple must be added to artist_albums to link the album to an artist. Must also create the artist if it does not already exist.
   - If an album is deleted, must also delete the corresponding entry in the artist_album table. Also, must remove any media instances and its tracks as well.
   - If a person is removed, their feedbacks, reviews, and checkouts must be removed as well. Must also remove the associated employee, if a subclass instance exists.
   - If a media is removed, must also remove checkouts referencing it.
2. What tables do you think should be included? What fields? Provide a copy of your relational diagram from Worksheet 04 (or an updated version if you have changed it since then).



3. Write appropriate SQL DDL statements to define the database.

```
CREATE TABLE album (

        album_id INTEGER PRIMARY KEY,

        album_title VARCHAR(155) NOT NULL,
       release_date DATETIME NOT NULL,

       genre VARCHAR(155) NOT NULL

);

CREATE TABLE person (

   card_number CHAR(36) PRIMARY KEY, -- UUID field

   email VARCHAR(155) UNIQUE,

   first_name VARCHAR(155) NOT NULL,

   last_name VARCHAR(155) NOT NULL,

   activation_date TIMESTAMP NOT NULL,

   num_cards INTEGER NOT NULL DEFAULT (1),

   card_number_active BOOLEAN NOT NULL DEFAULT 1

);

CREATE TABLE artist (

       artist_id INTEGER PRIMARY KEY,

       name VARCHAR(155) NOT NULL

);

CREATE TABLE track (

       title VARCHAR(155) NOT NULL,

       album_id REFERENCES album(album_id) NOT NULL,

       number SMALLINT,

       length REAL NOT NULL,

       artist_id REFERENCES artist(artist_id) NOT NULL,

       size_bytes BIGINT, -- library may not have a digital copy of a track, so may be NULL

       PRIMARY KEY(title, album_id)

);

CREATE TABLE media (

       media_id INTEGER PRIMARY KEY,

       type VARCHAR(8) NOT NULL,

       album_id REFERENCES album(album_id) NOT NULL

);

CREATE TABLE checkout (

   checkout_id INTEGER PRIMARY KEY,
   checkout_date DATETIME NOT NULL,
   return_date DATETIME, -- `NULL` until returned
   due_date DATETIME NOT NULL,
   media_id INTEGER NOT NULL,
   card_number CHAR(36) NOT NULL
);
CREATE TABLE artist_albums (
       artist_id REFERENCES artist(artist_id) NOT NULL,
       album_id REFERENCES album(album_id) NOT NULL,
       PRIMARY KEY (artist_id, album_id)
);
CREATE TABLE employee (
       start_date DATETIME NOT NULL,
       salary REAL NOT NULL,
       position VARCHAR(155),
       card_number REFERENCES person(card_number) PRIMARY KEY
);
CREATE TABLE feedback (
       feedback_id INTEGER PRIMARY KEY,
       description VARCHAR(500) NOT NULL,
       date DATETIME NOT NULL,
       category VARCHAR(155) NOT NULL,
       card_number REFERENCES person(card_number) -- can be anonymous
);
CREATE TABLE review (
       review_id INTEGER PRIMARY KEY,
       stars SMALLINT NOT NULL,
       title VARCHAR(155), -- doesn't require a title
       description VARCHAR(500), -- doesn't require a description
       album_id REFERENCES album(album_id) NOT NULL,
       card_number REFERENCES person(card_number) -- can be anonymous
);
```

Worksheet – 07 More SQL

Specify the following queries and show a sample results for the database.

1. Find all the albums by an arbitrary artist that were released before an arbitrary year.
   a) SQL

```
SELECT a.album_title

    FROM album a

         JOIN artist_albums aa ON aa.album_id = a.album_id

         JOIN artist art ON art.artist_id = aa.artist_id

    WHERE name = 'AC/DC' AND

          release_date < 1982;
```

   b) Sample Result

album_title
For Those About To Rock We Salute You
Let There Be Rock

2. Give all the albums and checkout dates for check outs made by a particular patron.
   a) SQL

```
SELECT album_title, checkout_date

   FROM Album AS a

    JOIN Media AS m ON a.album_id = m.album_id

    JOIN Checkout AS c ON m.media_id = c.media_id

    JOIN Person AS p ON c.card_number = p.card_number

   WHERE p.card_number = '18108a24-4eac-498d-a2d3-4dc4cccf405a';
```

   b) Sample Result

album_title	checkout_date
Stormbringer	2017-09-08 06:24:56

3. List all the albums with less than 5 copies available to check out.
   a) SQL

```
SELECT a.album_title

    FROM album a
```

```
            JOIN media m ON a.album_id = m.album_id
        GROUP BY a.album_id
        HAVING COUNT(*) < 5;
```

b) Sample Result

album_title
For Those About To Rock We Salute You
Let There Be Rock
Balls to the Wall
...
UB40 The Best Of - Volume Two [UK]
Van Halen
Van Halen III

4. List all the patrons with an overdue album checked out, ordered by most overdue.
   a. SQL

```
SELECT p.first_name || ' ' || p.last_name AS name,
       due_date,
       julianday('now') - julianday(due_date) AS days_overdue
   FROM checkout AS c
        JOIN person p ON p.card_number = c.card_number
   WHERE return_date IS NULL AND
         due_date < date('now')
   ORDER BY days_overdue;
```

   b. Sample Result

name	due_date	days_overdue
Aaron Gregory	2017-12-09 13:12:44	128.34627396985888
Adam Pope	2017-12-09 13:12:44	128.34627396985888
...	...	...
Tyler Lane	2017-12-09 13:12:44	128.34627396985888
Veronica Johnson	2017-12-09 13:12:44	128.34627396985888

Sample results vary because query is dependent on current date.

5. List all tracks that have a file size larger than the average file size.
   a. SQL

```
SELECT title, size_bytes
       FROM track
       WHERE size_bytes > (
              SELECT AVG(size_bytes)
                     FROM track
);
```

   b. Sample Result

title	size_bytes
For Those About To Rock (We Salute You)	11170334
Go Down	10847611
...	...
One I Want	10743970
Year to the Day	16621333

6. List all patrons who have been issued multiple library cards.
   a. SQL

```
SELECT first_name || ' ' || last_name AS full_name,
       num_cards
       FROM person
       WHERE num_cards > 1;
```

   b. Sample Result

full_name	num_cards
Audrey Robinson	2
Connie Benjamin	3
Jamie Fitzgerald	5

7. List all patrons with one or more albums checked out with a library card that is now inactive.
   a. SQL

```
SELECT first_name || ' ' || last_name AS full_name
       FROM person p
       JOIN checkout c ON p.card_number = c.card_number
```

```
                    WHERE card_number_active = 0;
```

b. Sample Result

full_name
Derek Stokes
Tyler Lane

8. List the top 10 albums (physical or digital), ordered by number of checkouts.
   a. SQL

```
SELECT album_title, COUNT(*) AS checkouts
    FROM album a
        JOIN media m ON a.album_id = m.album_id
        JOIN checkout c ON c.media_id = m.media_id
    WHERE return_date IS NULL
    GROUP BY album_title
    ORDER BY COUNT(*) DESC
    LIMIT 10;
```

   b. Sample Result

album_title	checkouts
Rock In Rio [CD2]	2
Afrociberdelia	1
American Idiot	1
Balls to the Wall	1
Bark at the Moon (Remastered)	1
Big Ones	1
Black Sabbath	1
Blue Moods	1
Deep Purple In Rock	1
For Those About To Rock We Salute You	1

Worksheet –08 Normalization

Is your database fully normalized? Why or why not?

The database was not quite fully normalized at first. The `track` relation had an erroneous `artist_id` column, which created a transitive dependency since `album_id` maps directly to the artist information. This column was not actually being used in any queries either, so it was removed for the final submission.

Other than that, all relations feature no redundancies that would create additional 1-1 maps between themselves. Thus, every non-key attribute in each relation of the database is non-transitively dependent on that relation’s key. Since every non-key in each relation is also fully dependent on the relation’s key, in addition to all domain values in each relation being atomic, the database is fully normalized into 3NF.

If not, restate your schema in 3NF.

The schema remains identical, other than the track relation:

```
CREATE TABLE track (

        title VARCHAR(155) NOT NULL,

        album_id REFERENCES album(album_id) NOT NULL,

        number SMALLINT,

        length REAL NOT NULL,

        size_bytes BIGINT, -- library may not have a digital copy of a track, so may be NULL

        PRIMARY KEY(title, album_id)

);
```