Name: _____

CS109 SQL - Test 3

Short Answer (5pts each)

For the following questions, provide a short written answer of around 3-5 sentences. A simple 'yes' or 'no' will not suffice.

1. Define and explain union compatibility. How does the database handle union compatibility between columns of differing data types?

   Similar data types. Exact same number of columns.

   To handle different data types you cast to the item with the lowest complexity.

2. Name the 4 set operations and describe how they modify the resultset when used.

   UNION - Combine all rows from query A and query B
   UNION ALL - Perform a UNION but don't remove duplicates
   INTERSECT - Display only rows that are in both query A and query B
   IN - Perform an INTERSECT based on a single column
   NOT IN - Display all rows that would not be filtered by an IN
   EXCEPT - Remove all rows from query A that appear in query B (Subtract)

3. Explain what would happen if a table with an auto-incrementing primary key had 10 rows and a new row was added to the table.

   A new row with id of 11 would be added. The auto_increment value would increase by 1

4. What is the DEFAULT constraint often seen with? Why is this pairing common?

NOT NULL. Common because if NULL is inserted in a NOT NULL column, then the DEFAULT value would be used instead.

Table Creation (15pts each)

On the attached page are two database tables `actor` & `movie` containing information related to actors in James Bond movies and the associated films. If you wanted to track which of the actors were in each movie you would need to create a Many:Many relationship between them.

1. Write the CREATE TABLE statements for the two database tables. All information must be represented including the gender_id code table foreign key, primary keys, and any NOT NULL constraints that should be set. The statements should be able to be run in a single transaction.

```
CREATE TABLE actor (
        id INTEGER PRIMARY KEY,
        first VARCHAR(100) NOT NULL,
        last VARCHAR(100) NOT NULL,
        gender_id CHAR(1) NOT NULL,
        dob DATE,
        height FLOAT,
        hometown VARCHAR(200),
        FOREIGN KEY gender_id REFERENCES gender(id)
);

CREATE TABLE movie (
        id INTEGER PRIMARY KEY,
        title VARCHAR(100) NOT NULL,
        release INT NOT NULL,
        budget DECIMAL(11, 2) NOT NULL
);
```

2. Write the CREATE TABLE statement for a M:M table `cast` that connects `actor` & `movie`.

   This `cast` table must have at minimum the following:
   - Required foreign keys pointing to the two tables, creating the M:M relationship
   - A column to contain the name of the character that the actor portrays. The name is required and, if not provided, should automatically be set to 'Uncredited'
   - A column stating whether the actor was in the leading role or not
   - A constraint allowing an actor to play a character role only once in each movie. For example, this constraint would not allow Sean Connery to play James Bond twice in the same movie, but *would* allow Eddie Murphy to play as each member of the Klump family in the movie *The Nutty Professor*.

```
CREATE TABLE cast (
        id INTEGER PRIMARY KEY,
        actor_id INT NOT NULL,
        movie_id INT NOT NULL,
        character VARCHAR(100) NOT NULL DEFAULT 'Uncredited',
        leading BOOL,
        FOREIGN KEY actor_id REFERENCES actor(id),
        FOREIGN KEY movie_id REFERENCES movie(id),
        CONSTRAINT unq_char UNIQUE(actor_id, movie_id, character)
);
```

Set Operations (10pts each)

The following queries reference data that is in the example database tables provided on the attached page. Using the database, write down a single query using set operations that performs the requested operations. If the question specifies the results to be ordered or specific columns to be selected it is expected that this be part of the query.

Note -- Since Joins & AND/OR operators may often produce the same functionality that set operations provide, they are explicitly disallowed for these queries.

1. Display only the gender_id for all actors whose last name starts with 'B' and all actors whose last name starts with 'C'.
   The number of times each gender_id occurs should still be equal to the number of actors. For example, there should be 4 rows with 'M' as the only value.

   ```
   SELECT gender_id
   FROM actor
   WHERE last LIKE 'B%'
   UNION ALL
   SELECT gender_id
   FROM actor
   WHERE last LIKE 'C%'
   ```

2. A hypothetical biographical film named *Connery* was made covering Sean Connery's life in which he briefly reprised his role as James Bond. It would be nice to have this film included in the movie list.

   Using Sean Connery's row in the `actor` table and the entries in the `movie` table, list all movies including *Connery*. You must keep the movie release dates & budgets intact even if *Connery* is missing that information (remember union compatibility).

   ```
   SELECT last, NULL, NULL
   FROM actor
   WHERE last = 'Connery'
   UNION
   SELECT title, release, budget
   FROM movie
   ```

3. Display only the names of all actors who have not yet retired their character roles.

```
SELECT first, last
FROM actor
WHERE id NOT IN (SELECT actor_id FROM actor_retired)
```

4. List all movies with a budget lower than $10 million that also have a budget greater than $5 million. If you decide to use IN rather than INTERSECT to answer this you may use the AND operator in the WHERE clause when necessary.

```
SELECT *
FROM movie
WHERE budget < 10000000
INTERSECT
SELECT *
FROM movie
WHERE budget > 5000000

OR

SELECT *
FROM movie
WHERE budget < 10000000 AND id IN (SELECT id FROM movie WHERE budget > 5000000)
```

5. List all movies that were released in 1995, 1997, 1999, or 2012.

```
SELECT *
FROM movie
WHERE release IN (1995, 1997, 1999, 2012);
```

Extra Credit (10pts)

Design and write the creation script for a database centered around a table that is in a 1:M relationship with itself. To get full extra credit points you must:

- Create at least 1 table, one of which has a foreign key referencing itself.
- Declare all proper constraints & keys.
- Choose valid names & data types.
- Write all statements as a single transaction.

If your Individual SQL project contains a similar layout I would advise you work with that. Points will be given according to both creativity and correct syntax; if you are having trouble with the technical requirements then a little humor goes a long way!

Any table set up would work as long as it has the following basic relationship:

```
CREATE TABLE table (
        id INTEGER PRIMARY KEY,
        table_id INT,
        FOREIGN KEY table_id REFERENCES table(id)
);
```

The possible points are determined by the verbosity of your answer and the creativity of the database layout.