CS2102: Database System

Midterm Assessment — AY2023/24 Sem 2

Instructions

1. Please read the **Instructions** carefully.
2. This assessment is using **Examplify**: this is a **NON-SECURE BLOCK INTERNET** assessment.
3. There are **10 (TEN)** questions, the total mark for the assessment is **60 Points**, you are to **answer ALL questions**.

MRQ : Q1	**FITB** : Q5 - Q6
MCQ : Q2 - Q4	**SQL** : Q7 - Q10

4. This is an **OPEN-BOOK** assessment.
5. You are **NOT** allowed to have additional devices (*e.g.*, second monitor, smart-watch, earpiece, *etc*).
6. The assessment starts at **12:30** and ends at **13:30**.
   - You are to start the assessment **IMMEDIATELY** once the password is released.
   - The timer will start once you see the first question.
   - No additional time will be given to submit except for technical issues.
7. **SQL Instructions**
   - Use text-editor to write your queries.
   - Use **psql** (*or other applications*) running on PostgreSQL 16 to test your queries.
     - Answers that cannot be run **psql** may receive 0 marks, **even for minor error**.
   - Your answer should consists only of a single statement.
   - Allocate sufficient time to prepare and **COPY-PASTE** your answer into **Examplify**.
   - Read the **SQL Queries** section for more information.
     - This is similar to *Assignment 01* with added emphasis that "you are **NOT** allowed to use concepts not taught in class".
8. MCQ/MRQ options may be **randomized** on **Examplify**.
   - Select "*None of the Above*" only if other options are incorrect.
   - For MRQ, selecting "*None of the Above*" with other options will be marked as wrong.
   - Partial marks may be given for MRQ depending on the options chosen as well as the actual number of correct options.
9. Failure to follow instructions above may result in deductions of your marks.

Good Luck!

1. ER Model 35 Points

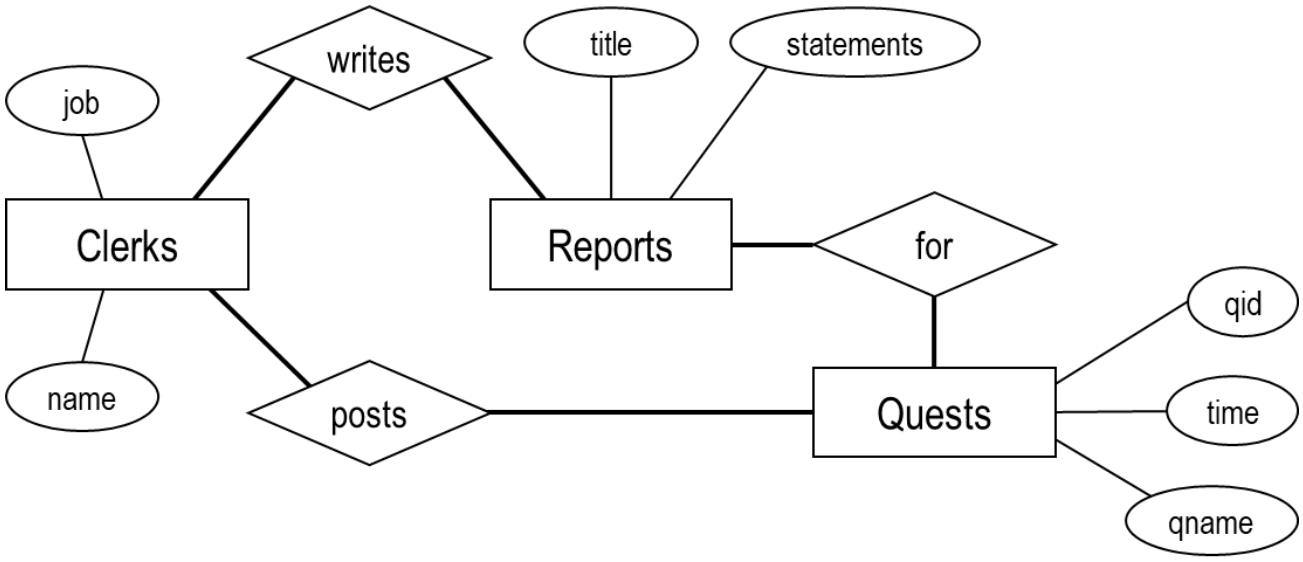
Story Line *can be safely skipped*

Oh no, you did not watch when you were crossing the road on Central Library. You have been hit by a truck! Fortunately, you are being isekai-ed into another world where adventurers are common place and Demon King has been reborn.

During your transport, you were given a random cheat-skill. The cheat-skill you were blessed with is called the "Master of Data". You can process any data in a very efficient manner. You are promised a safe return to NUS if you help defeating the Demon King.

As you wake up, you arrive at the Kingdom of TasaBade. To help defeating the Demon King, you work for the most famous guild called Guild ErgPost in the City of Esquel. There, you work as a clerk to help other adventurers to defeat the Demon King. Given your cheat-skill, you are tasked with managing the clerks instead. The job of a clerk is to post quests as well as writes reports for quests.

Given your duty at Guild ErgPost, you decide to make an ER diagram to capture the data and constraints needed for the daily working of the guild. You come up with the following basic ER diagram.



The ER diagram is still incomplete, but you also come up with the following constraints below. Your task is to complete the ER diagram.

1. Each quest can be uniquely identified by either the quest id (`qid`) or by the *combination* of quest name (`qname`) and the time the quest is posted (`time`).
2. Each quest must be posted by at most one clerk. However, we also accept quest posted by other adventurers for which no clerks are posting the quest.
3. Each quest may have multiple reports written but each report is for exactly one quest. In addition, when a quest is posted, it must immediately have a report indicating that the quest is posted. A potential scenario showing the sequence is shown below:
   (1) A quest is posted, then a report is immediately made about the quest creation.
   (2) A quest is taken but ended in a failure, then a report is made about the failure.
   (3) A quest is taken and ended in a success, then a report is made about the success.
4. Each report must be written by exactly one clerk and the title of the reports *uniquely* identifies the report from among the report written by the clerks. The same title may be used by other clerks.

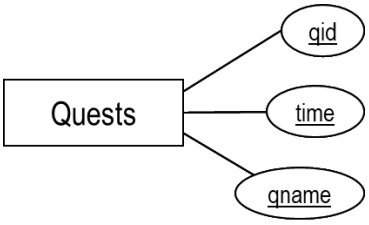
You are advised to work on the ER diagram to captures the 4 constraints before attempting the next 4 questions.

1. (5 points) **[MRQ]** Consider Constraint #1 reproduced below:

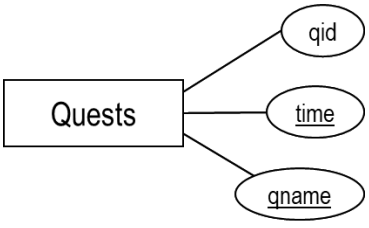
   > Each quest can be uniquely identified by either the quest id (`qid`) or by the *combination* of quest name (`qname`) and the time the quest is posted (`time`).

   Select **ALL** the ER diagram that captures this constraint.

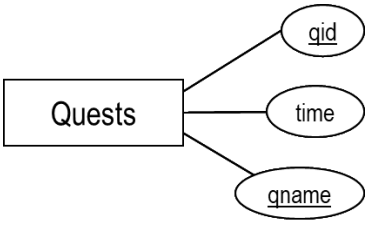
(A)



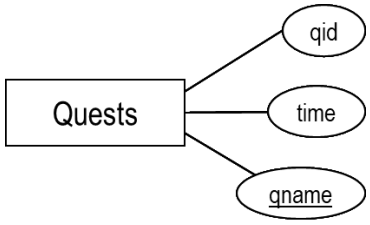
(✓)



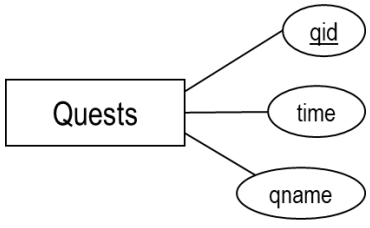
(C)



(D)



(✓)

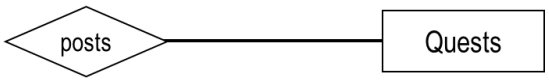


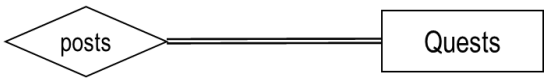
(F) *None of the above*

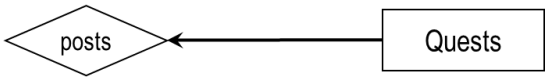
2. (4 points) **[MCQ]** Consider Constraint #2 reproduced below:

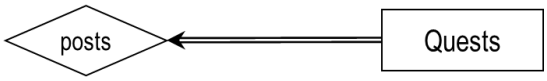
> Each quest must be posted by at most one clerk. However, we also accept quest posted by other adventurers for which no clerks are posting the quest.

Select the ER diagram that captures this constraint.

(A) 

(C) 

(✓) 

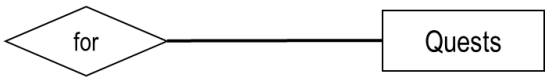
(D) 

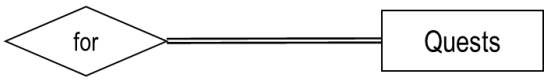
(E) *None of the above*

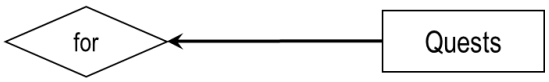
3. (4 points) **[MCQ]** Consider Constraint #3 reproduced below:

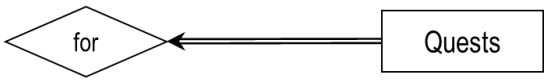
> Each quest may have multiple reports written but each report is for exactly one quest. In addition, when a quest is posted, it must immediately have a report indicating that the quest is posted. A potential scenario showing the sequence is shown below:
>
> (1) A quest is posted, then a report is immediately made about the quest creation.
>
> (2) A quest is taken but ended in a failure, then a report is made about the failure.
>
> (3) A quest is taken and ended in a success, then a report is made about the success.

Select the ER diagram that captures this constraint.

(A) 

(✓) 

(B) 

(D) 

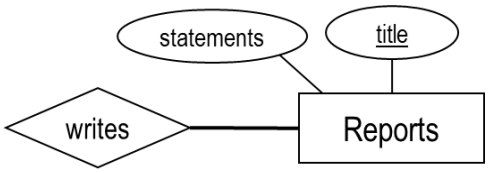
(E) *None of the above*

4. (4 points) **[MCQ]** Consider Constraint #4 reproduced below:

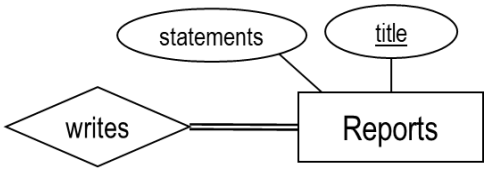
> Each report must be written by exactly one clerk and the title of the reports *uniquely* identifies the report from among the report written by the clerks. The same title may be used by other clerks.

Select the ER diagram that captures this constraint.

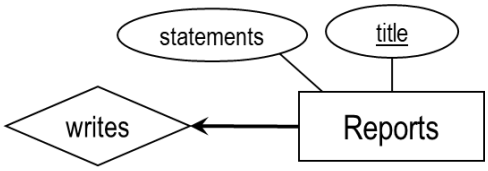
(A)



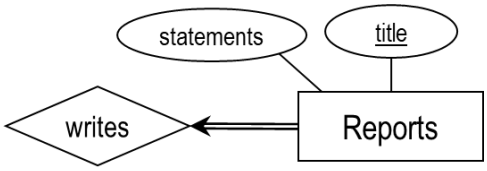
(C)



(B)



(D)



(✓) *None of the above*

Comments:

Q1: This is similar to our Quiz, you can only indicate one primary key (*i.e.*, set of attributes) in ER diagram. So either {`qid`} is a primary key (*i.e.*, Option E), or {`qname`, `time`} is the primary key (*i.e.*, Option B).

Q2: Based on "*Each quest must be posted by at most one clerks.*", we have upper bound 1 for quest with respect to posts (*i.e.*, Option B or D).

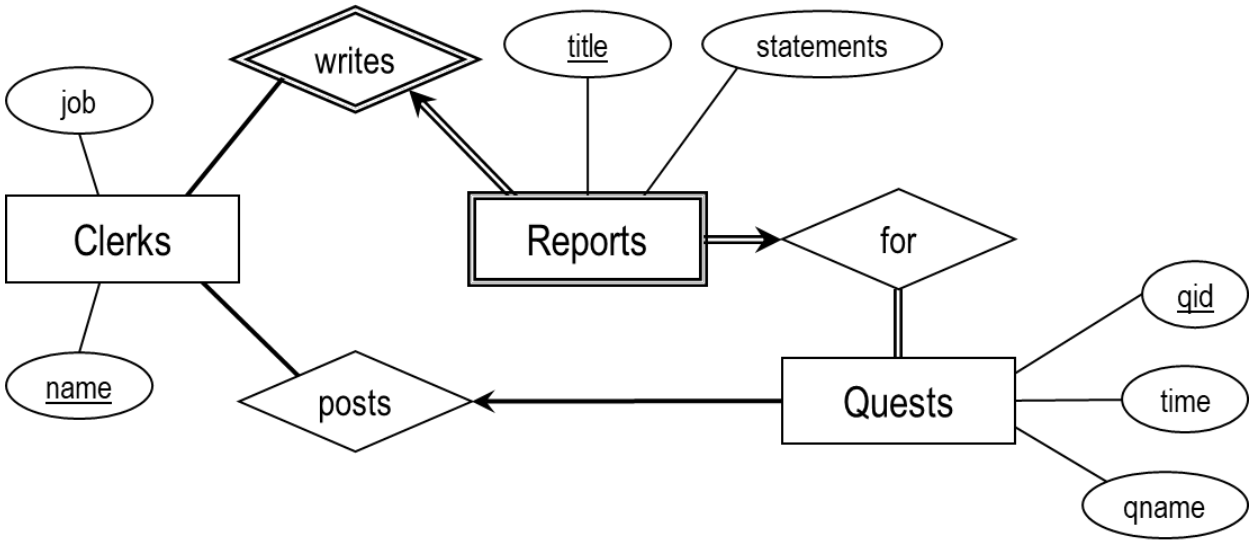
Since we accept quest posted by other adventurers, this quest has no associated clerks. This indicates lower bound of 0 for quest with respect to posts. Therefore, the answer is Option B and not Option D (*because the lower bound is 1 here*).

Q3: Based on "*Each quest may have multiple reports written ...*", we have upper bound greater than 1 (*i.e.*, Option A or C).

Based on "*In addition, when a quest is posted, it must immediately have a report indicating that the quest is posted.*", we have lower bound of 1 instead of 0. Therefore, the answer is Option C and not option A.

Q4: Based on "*title of the reports **uniquely** identifies the report from among the report written by the clerks.*", reports must be a *weak entity set*. Therefore, there is no answer (*i.e.*, Option E, *None of the above*).

This gives us the following ERD:



Story Line *can be safely skipped*

Now that the problem with the guild management is resolved, you decide to help adventurers to fight the Demon King. You are at the Demon King's Castle and faced with several different puzzles before you can enter the castle.

Why do these final bosses always use puzzles? Have they never heard of locks and guards? Why are these treasure chests also freely available for the invading parties? Keys in the treasure chest?! Is the Demon King really that confident in his power?

*"**Nah, I'd win.**" You thought to yourself.*

As the Master of Data, you have access to the creation of the castle and found the following schema:

```
CREATE TABLE Items (
  item_id    INT    PRIMARY KEY,
  item_name  TEXT   NOT NULL,
  kind       TEXT   NOT NULL
);

CREATE TABLE Doors (
  location   TEXT   PRIMARY KEY,
  lock_key   INT    NOT NULL
    REFERENCES Items
);

CREATE TABLE Locations (
  location   TEXT   PRIMARY KEY
);

CREATE TABLE TreasureChests (
  location   TEXT
    REFERENCES Locations,
  content    INT
    REFERENCES Items,
  PRIMARY KEY (location, content)
);
```

You know for sure that there are only three kinds of items: `'Weapon'`, `'Armor'`, and `'Key'`. Additionally, while not reflected in the schema, the `lock_key` can only be an item with some `item_id` such that the `kind` is `'Key'`. All doors are locked, and the keys are recorded in `lock_key`. As you help explore the castle, you found the following series of puzzles. Having access to the schema, you decide to simply solve them using relational algebra.

There can be multiple answers, as long as your answer produces valid result, you will be given the mark. Answers that do **NOT** follow the correct upper-/lower-case and the use of underscore (*i.e.*, `_`) may be penalized (*e.g.*, "`Location`" instead of "`location`" or "`ItemName`" instead of "`item_name`"). Additionally, your answer for each blank must be a single relation/attribute name.

5. (12 points) **Puzzle #1**

   The door to the Demon King's Throne is a locked door located at the same location as a treasure chest containing an item with id of 0.

   You came up with the following skeleton for the relational algebra expression.

$$Q_1 := \boxed{\texttt{TreasureChests}} \bowtie \boxed{\texttt{Doors}} \times \boxed{\texttt{Items}}$$

$$Q_2 := \pi_{[\ \boxed{\texttt{location}}\ ]}\left(\sigma_{[\ \boxed{\texttt{item\_id}} = 0 \wedge \boxed{\texttt{item\_id}} = \boxed{\texttt{content}}\ ]}\left(Q_1 \right) \right)$$

6. (6 points) **Puzzle #2**

> *The key to the Demon King's Throne is in the location where the door is locked and the lock is located in the treasure chest in that same location.*

You came up with the following skeleton for the relational algebra expression.

$$Q_3 := \rho_{[\ \texttt{content} \leftarrow \texttt{lock\_key}\ ]}(\ \texttt{Doors}\ ) \bowtie \texttt{TreasureChests}$$

$$Q_4 := \pi_{[\ \texttt{location}\ ]}(\ Q_3\ )$$

Comments:

Q5: Notice the two different operators: $\bowtie$ (*natural join*) and $\times$ (*cross product*). There is a limitation of cross product on our relational algebra expression, namely that you cannot produce duplicate attributes. On the other hand, natural join relies on common attributes. These facts limit our choice to the following:

- `TreasureChests` $\bowtie$ `Doors` $\times$ `Items`
- `Doors` $\bowtie$ `TreasureChests` $\times$ `Items`

Since natural join already ensures that common attributes have equal value, we already have equality on `location`. So the next thing we need to do is to equate the attributes `content` and `item_id`. Here, we need to ensure that `item_id = content = 0`. We have 4 choices which is a mix-and-match of the left and the right.

- `item_id = 0`
- `item_id = content`
- `content = 0`
- `content = item_id`

Q6: For the natural join to work, they must have the same attribute names. Since `location` already appears in both, we need to make `lock_key` equal to `content`. This can be done by renaming one of them. So we have 2 choices:

- $\rho_{[\ \texttt{content} \leftarrow \texttt{lock\_key}\ ]}(\ \texttt{Doors}\ ) \bowtie \texttt{TreasureChests}$
- $\rho_{[\ \texttt{lock\_key} \leftarrow \texttt{content}\ ]}(\ \texttt{TreasureChests}\ ) \bowtie \texttt{Doors}$

Story Line *can be safely skipped*

Congratulations!!

The Demon King has been defeated. You are transported back to NUS. As you open your eyes, you are already in MPSH 2 at NUS in the middle of answering questions for CS2102 midterm.

*You are now face-to-face with another **tyrant**, your CS2102 lecturers where you must complete the SQL query problems on the next section. Luckily, you are well-prepared with the Schema from Assignment 01.*

2. SQL Queries 25 Points

In this part of the assessment, you will formulate **4 (FOUR)** SQL queries. We use the same database introduced in *Assignment 01*. This means you can run and test your query directly on psql. We also provide the ER diagram and `CREATE TABLE` statements in Appendix.

- Submit each question into its own individual answer box on **Examplify**. Make sure that you submit only **ONE** SQL statement for each question. This statement should be an SQL query and ends with a semi-colon (*i.e.*, `;`).
- Each question has a specification on the output schema. You **MUST** follow the expected column name, otherwise there will be penalty even if you have the correct result. Consider the following template below using `AS` keyword for column renaming.

```
SELECT attr1 AS name, attr2 AS year
  query
;
```

If the code above can be run correctly, then it satisfies the following schema:

name	year

- Each answer must be a syntactically valid SQL query without any typographical errors: an SQL answer that is syntactically invalid or contains very minor typographical error will receive **ZERO (0)** marks even if the answer is semantically correct.
- For each question, your answer view must not contain any duplicate records. You may use `DISTINCT` as you see fit.
- Each question must be answered independently of other questions (*i.e.*, the answer for a question must not refer to any other view that is created for another question).
- You are **NOT** allowed to do any of the following for the SQL queries.
  - Creating `VIEW`, new table, temporary or otherwise
  - Using Common Table Expression (**CTE**)
  - Using transaction
  - **You can only use the concepts taught in the class**
- Unless a value is explicitly specified in the question (*e.g.*, `'James Dean'`), you are not allowed to hardcode any value.
  - Your code will be run against additional datasets.
- You are **NOT** allowed to import any libraries and your answers must be executable on PostgreSQL 16.
  - Your code will be run on PostgreSQL 16.

7. (5 points) Let's start with a simple question.

   Find all character name (`cname`) and the actors playing the characters (`pname`) such that the character name only has one word (*i.e.*, no space).

   Exclude characters that is not played by any actors.

   Produce the table with the schema below satisfying the above condition.

cname	pname

   There should be 5 rows in the output based on the current dataset.

8. (5 points) You now want to find talented actors. We define talented actors as actors that have played in a film as two different characters.

   Find all the actor names (`pname`) and character names (`cname`) such that the actor has played as at least two different characters in the same film.

   For instance, Kevin Spacey is both Roger Kint and (*spoiler alert*) Keyser Soze in the film called The Usual Suspect. In the Batman Trilogy by Christopher Nolan, we also have Aaron Eckhart playing both Harvey Dent and the character named Two Face.

   In our data set, we have one actor that fits this. Richard Burton has played as both "Becket" and "Thomas Becket". Although this is *likely* due to incorrect data input, we will still acknowledge the talent of Mr. Richard Burton.

   Produce the table with the schema below satisfying the above condition.

pname	cname

   There should be 1 row in the output based on the current dataset.

9. (7 points) Maybe old thriller films are more of your style.

   Find all film names (`tname`) and the starting year `syear` where the genre of the film is `'Thriller'` and the starting year is *no later than* 1960.

   Produce the table with the schema below satisfying the above condition.

tname	syear

   There should be 3 rows in the output based on the current dataset.

10. (8 points) Okay, maybe too old is not too good. You also want to find new genre. You can do this by looking at the database and find for each year, what is the most popular genre.

> For each year (**syear**) from 1960 onwards, find the most popular genre (**genre**). The most popular genre is the genre with the most number of films for the given year. Note that the most popular genre may not be unique. In which case, you output all genres that are the most popular for that particular year.
>
> Exclude the year without any films.

Some years have multiple most popular genre. For instance, in 1961, there are 2 popular genres: Drama and War.

Some years are also missing from the output. For instance, there are no films between 1966 and 1971.

Produce the table with the schema below satisfying the above condition.

syear	genre

There should be 35 rows in the output based on the current dataset.

Comments:

```sql
-- Q7
SELECT DISTINCT cname, pname
FROM   PlaysAs
WHERE  cname NOT LIKE '% %';
```

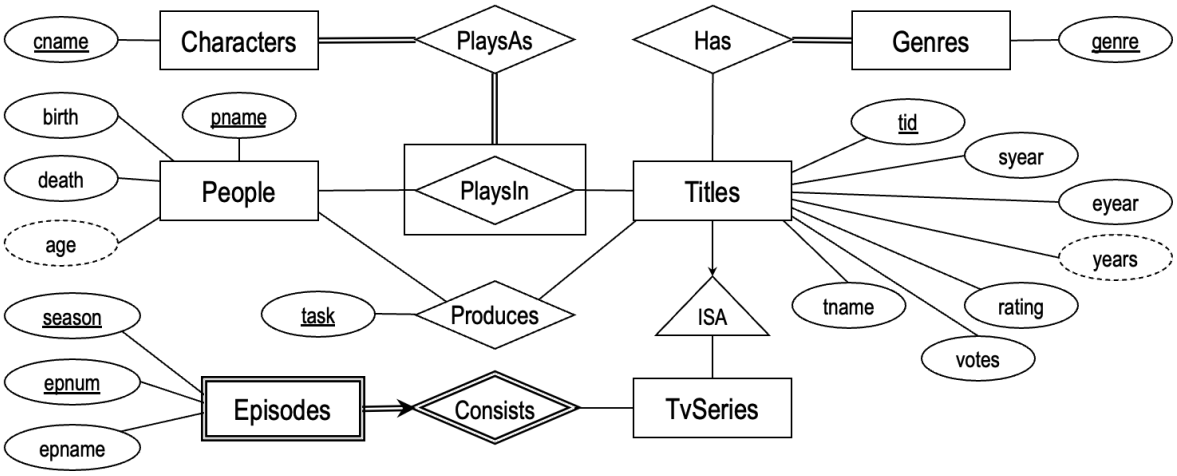
```sql
-- Q8
SELECT DISTINCT P1.pname, P1.tid, P1.cname
FROM   PlaysAs P1, PlaysAs P2
WHERE  P1.tid = P2.tid AND P1.pname = P2.pname
  AND  P1.cname <> P2.cname;
```

```sql
-- Q9
SELECT DISTINCT T.tname, T.syear
FROM   Titles T, Genres G
WHERE  G.tid = T.tid AND G.genre = 'Thriller' AND T.syear <= 1960;
```

```sql
-- Q10
SELECT DISTINCT T1.syear, G1.genre
FROM   Titles T1, Genres G1
WHERE  G1.tid = T1.tid AND T1.syear >= 1960
GROUP BY T1.syear, G1.genre
HAVING COUNT(T1.tid) >= ALL ( -- Find max count for that year
  SELECT COUNT(T2.tid)
  FROM   Titles T2, Genres G2
  WHERE  G2.tid = T2.tid AND T1.syear = T2.syear
  GROUP BY T2.syear, G2.genre
);
```

A Appendix

A.1 ER Diagram



A.2 Additional Constraints

- We refer to entries in `Titles` as "film title" even when the title may also be a TV series (*i.e.*, an entry in `TvSeries`).
- If a film title is in `TvSeries`, we refer to them as "TV title", "Series title", "TV series title", or simply "TV series".
- If a film title is in `Titles` but **NOT** in `TvSeries`, we refer to them as "movie title" or simply "movies".
- We refer to "actor" as entries in `People` who are also entries in `PlaysIn`.
- The derived attribute `age` in `People` is derived purely from `death` - `birth` regardless of the date.
- If the person (*i.e.*, entries in `People`) is still alive, the value of `death` is `NULL`. Otherwise, the value of `death` is recorded and it must be greater than `birth` (*i.e.*, no person with age less than 1 year is recorded).
- The derived attribute `years` in `Titles` is derived purely from `eyear` - `syear` regardless of the date.
- If the film title (*i.e.*, entries in `Title`) is still ongoing, the value of `eyear` is `NULL`. Otherwise, the value of `eyear` is recorded.
- The `runtime` in `Titles` is specified in minutes.
- The `rating` in `Titles` must be between 0 and 10 (*inclusive of both*).
- For simplicity, the type of `birth`, `death`, `syear` (*i.e.*, start year), `eyear` (*i.e.*, end year) are *`INT`*.
- The people who produces film title (*i.e.*, entries in `Produces`) may be `'director'`, `'composer'`, or other task (*e.g.*, `'producer'`, *etc*).

A.3 Schema

```
CREATE TABLE People (
  pname   TEXT PRIMARY KEY,
  birth   INT  NOT NULL CHECK (birth > 0),
  death   INT  CHECK (death > 0 AND death > birth)
);
```

```
CREATE TABLE Titles (
  tid       VARCHAR(10) PRIMARY KEY,
  tname     TEXT NOT NULL,
  syear     INT  NOT NULL CHECK (syear > 0),
  eyear     INT  CHECK (eyear >= syear),
  runtime   INT  NOT NULL CHECK (runtime > 0),
  rating    NUMERIC  CHECK (rating >= 0 and rating <= 10),
  votes     INT      CHECK (votes >= 0)
);
```

```
CREATE TABLE TvSeries (
  tid  VARCHAR(10) PRIMARY KEY
    REFERENCES Titles (tid)
);
```

```
CREATE TABLE Genres (
  tid    VARCHAR(10)  REFERENCES Titles (tid),
  genre  VARCHAR(200),
  PRIMARY KEY (tid, genre)
);
```

```
CREATE TABLE Episodes (
  tid     VARCHAR(10)  REFERENCES TvSeries (tid)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  season  INT CHECK (season > 0),
  epnum   INT CHECK (epnum > 0),
  epname  TEXT,
  PRIMARY KEY (tid, season, epnum)
);
```

```
CREATE TABLE Produces (
  tid    VARCHAR(10)  REFERENCES Titles (tid),
  pname  TEXT         REFERENCES People (pname),
  task   VARCHAR(100),
  PRIMARY KEY (tid, pname, task)
);
```

```
CREATE TABLE Characters (
  cname  TEXT  PRIMARY KEY
);
```

```
CREATE TABLE PlaysIn (
  pname   TEXT  REFERENCES People (pname),
  tid     VARCHAR(10)  REFERENCES Titles (tid),
  PRIMARY KEY (pname, tid)
);
```

```
CREATE TABLE PlaysAs (
  pname   TEXT,
  tid     VARCHAR(10),
  cname   TEXT  REFERENCES Characters (cname),
  FOREIGN KEY (pname, tid) REFERENCES PlaysIn (pname, tid),
  PRIMARY KEY (pname, tid, cname)
);
```

– End of Paper –