# ASSIGNMENT 1

**Problem 1 (40 points):** Consider the following relational schema for a website for fans of live music that keeps track of artists (individual musicians or groups), concerts by artists, and the times and venues (places such as bars, concert halls, etc.) where concerts take place:

Artist (aid, aname, adesc)
ArtistGenre (aid, genre)
Concert (cid, vid, cdate, cstarttime, cendtime, cdesc)
ConcertArtist (cid, aid)
Venue (vid, vname, vaddress, vcity, vphone)
TicketType (cid, ticketclass, ticketcost)

Thus, for each artist we have an aid, a name, and a short description of their music. An artist can also play one or more genres, such as Jazz, Americana, Soul, etc. For each concert, we store where it takes place (the venue), the date, the start and end time, and a short description. A concert may involve several artists. For each venue we store its name, address, and contact phone number. Finally, we store information about the prices of different categories of tickets for each event. Note that the site is not keeping track of individual tickets and who purchased them; it only provides information about upcoming concerts and their ticket prices (probably with hyperlinks to the actual sites selling tickets).

(a) (2 points) Why does TicketType and ConcertArtist have two primary keys? What would be the problem of using only one primary key? Explain.

Primary key is used to identify a tuple uniquely. The relations TicketType and ConcertArtist require two attributes to be able to identify tuples uniquely. For example, if ConcertArtist had only aid as primary key then all it wouldn't be able to identify the different Concert with different cid. Similarly, if we would take only cid as the primary key for TicketType then we would get many items with the same cid thus not uniquely identifying the tuple.

(b) (2 points) Identify suitable foreign key representing the tables.

A foreign key is a field (or collection of fields) in one table that refers to the primary key in another table.
In this schema,

- aid is the foreign key in relation ArtistGenre
- cid and aid are the foreign key in relation ConcertArtist
- vid is the foreign key in relation Concert
- cid is the foreign key in relation TicketType

(c) (12 points) Write statements in SQL for the following queries.

1. Output the names of the venues in city 'New York'.

   SELECT vname FROM Venue WHERE vcity= "New York";

2. Output all the concerts that were played in ‘Chicago’ in year 2017.

   ```
   SELECT cid FROM Concert natural join Venue WHERE vcity=”Chicago” and extract(year
   from cdate) = 2017;i
   ```

3. For each venue, output the number of artists played in year 2017.

   ```
   SELECT vid, count(aid)
   FROM Venue natural join Concert natural join ConcertArtist
   WHERE extract(year from cdate) = 2017;
   ```

4. Give the cid and ticket class where the cost of ticket exceeded $100.

   ```
   SELECT cid ,ticketclass FROM TicketType WHERE ticketcost > 100;
   ```

5. Output the aids of all artists who have appeared together with “Bruno Mars” in at least two concerts during 2017.

   ```
   SELECT aid, aname
   FROM Artist NARTURAL JOIN ConcertArtist
   WHERE ananme = "Bruno Mars" AND cid in (
   SELECT cid FROM Concert NATURAL JOIN ConcertArtist NATURAL JOIN Artist
   WHERE name = "Bruno Mars" and YEAR(cdate) = 2016)
   GROUP BY aid, aname
   HAVING COUNT(*) >= 2
   ```

6. Give the names of the artist who have done the concerts at all the venues at least once.

   ```
   SELECT aid,aname
   FROM (Venue natural join Concert natural join ConcertArtist natural join Artist)
   group by aid
   having (count(distinct(vid)) = (select count(vid) from Venue));
   ```

(d) (12 points) Write relational algebra statements for the above queries if it is possible, or give a reason why is it not possible.

RELATIONAL ALGEBRA —

d)

1. $\Pi_{vname} \left( \sigma_{vcity = \text{"New York"}} \left( Venue \right) \right)$

2. $\Pi_{cid, cname} \left( \sigma_{vcity = \text{"Chicago"} \wedge extract(year\ from\ cdate) = 2017} \left( Concert \bowtie Venue \right) \right)$

3. $\Pi_{vid, count(aid)} \left( \sigma_{extract(year\ from\ cdate) = 2017} \left( Venue \bowtie Concert \bowtie ConcertArtist \right.\right.$

4. $\Pi_{cid,\ ticketclass} \left( \sigma_{ticketcost > 100} \left( TicketType \right) \right)$

5. $temp \leftarrow \Pi_{cid} \left( \sigma_{aname = \text{'BrunoMars"} \wedge Year(cdate) = 2017} \left( ConcertArtist \bowtie Concert \bowtie Artist \right.\right.$

$temp2 \leftarrow {}_{aid, aname} G_{count(*) as\ ccount} \left( \sigma_{aname = \text{"BrunoMars"}} \left( Artist \bowtie ConcertArtist \bowtie temp \right.\right.$

[illegible] $\Pi_{aid, aname}\ \sigma_{Count > 1}\ temp2$

6. [illegible]

$temp \leftarrow \Pi_{count(vid)\ as\ vcount} \left( Venue \right)$

$temp2 \leftarrow {}_{aname, aid} G_{count(distinct(vid)) as\ ccount} \left( Venue \bowtie Concert \bowtie ConcertArtist \bowtie Artist \right)$

[illegible] $\Pi_{aid, aname} \left( \sigma_{vcount = ccount} \left( temp \bowtie temp2 \right) \right)$

(e) (12 points) Write statements in (Domain or Tuple) Relational Calculus for the above queries or explain why it is not possible

RELATIONAL CALCULUS

(e)

1. $\{ t \mid \exists A \in Venue\ (t[vname] = A[vname] \land A[vcity] = \text{"New York"}) \}$

~~2. {t | ∃ v ∈ Venue (t[vcity] =~~

2. $\{ t \mid \exists C \in Concert\ (t[cid] = C[cid] \land year[c[cdate]] = 2017 \land \exists v \in Venue\ (c[vid] = v[vid] \land v[vcity] = \text{"Chicago"})) \}$

3. Cant express it in TRC or DRC due to the aggregation function.

4. $\{ t \mid \exists a \in TicketType\ (t[cid] = a[cid] \land t[ticketclass] = a[ticketclass] \land a[ticketcost] > 100) \}$

5. Cant express it in TRC/DRC due to the aggregation function.

6. Cant express it in TRC/DRC due to aggregation function.

**Problem 2 (16 points):** In this problem you need to design a relational schema for a baseball stadium booking agency. This website is used to let various baseball teams book the stadiums on demand.

For each baseball teams, it has information like their name, no. of players, home city, coach name. You also need to store the unique ids for each of the teams. For each stadium, there would be different kinds of information like name, city, area, capacity, built in (year), manager name and would also have a unique id.

Each stadium has a schedule on the website, so that it is possible to search which stadiums are available at a particular time. For simplicity, you may assume that the schedule consists of slots lasting 3 hours. Teams may also upload stadium requests, specifying the time and date.

When a team and stadium manager comes to an agreement about the requested time, a booking is confirmed. All the necessary information regarding the booking need to be stored including the slots, booking fee, etc. A team can also rate the stadium between 1 (very bad) to 5 (very good).

(a) (8 points) Design a relational database schema for the given problem, with suitable table, attributes, primary keys, and foreign keys. Discuss any assumptions you are making in your design!

Team(tid, tname, number_players, tcity, coach_name)
Stadium(sid, sname, scity,sarea, capacity, built_year, manager_name)
StadiumRating(sid, tid,rating, rdate)
Schedule(slotid, sid, start_time, date, bookingid)
StadiumRequest(reqid, sid, tid, time, date)
Booking(bookingid, tid, sid, reqid, fee)

Primary Key- tid for relation Team, sid for relation Stadium, (sid,tid) for StadiumRating, slotid for Schedule,reqid for StadiumRequest, bookingid for relation Booking.

Foreign Key-

Assumptions: slots are 3 hours long with their start time stored in the relation Schedule. Rating ranges from 1 to 5 in StadiumRating.

(b) (8 points) Write SQL statements for the following queries. If your schema does not support these, you need to modify it appropriately.

1) Output the names of all stadiums available between 9 am to 6 pm on Feb 10th, 2018. (They should be available all the hours, not just part of the time.)

```
SELECT sid, sname
FROM Stadium
WHERE NOT EXISTS( SELECT sid, sname FROM Stadium natural join Schedule WHERE
(start_time between 7am and 5pm) and date= “Feb 10th, 2018” and bookingid=NOT NULL);
```

2) For each stadium, output their ids, and the number of distinct team they have been booked to in 2017.

```
SELECT sid, distinct(tid)
FROM Schedule
WHERE extract (year from date) = 2017
INNER JOIN Booking ON Schedule.bookingid = Booking.bookingid;
```

3) Output the stadium(s) who earned the most rating overall during 2017.

```
SELECT max (avg_rating)
FROM(SELECT sid, avg(rating) as avg_rating
FROM StadiumRating
WHERE extract (year from rdate) = 2017
INNER JOIN Stadium ON StadiumRating.sid = Stadium.sid
GROUP BY sid);
```

4) Output the names of teams who have booked at least 5 stadiums, but who have never given a rating of 4 stars or higher.

```
SELECT tid, avg_rating
FROM (
SELECT tid, sid, avg(rating) as avg_rating
FROM Booking natural join StadiumRating
GROUP BY tid
HAVING count(sid) >= 5)
WHERE avg_rating < 4;
```