CS486/586 Introduction to Databases
Spring 2022 Quarter
Assignment 2 – DDL & DML; SQL & Relational Algebra
Due: Friday, April 15th, 11:59PM on Canvas
Submitted by: - Parth Parashar and Kirtan Patel

Part I – More SQL (50 points)
Write a single SQL statement for each of the following queries. Show the first five rows of the result for each query (or fewer, if the result is smaller) and the number of rows returned.

Question-1) Different types of JOINs and SET operators

(a) Find the team name for all teams with at least one agent who is skilled at Counterintelligence.

(b) List the team name for each team that has at least one agent who can speak Bengali and at least one agent who speaks Polish.

Answer: -

a) The query for this problem statement is given below: -

Select distinct t.name from team t join teamrel tr ON t.team_id=tr.team_id join skillrel sr ON tr.agent_id=sr.agent_id JOIN skill s ON sr.skill_id=s.skill_id where s.skill='Counterintelligence';

The screenshot below gives the result: -

```
OpenSSH SSH client
spr2022adb35=> select distinct t.name from team t join teamrel tr ON t.team_id=tr.team_id
spr2022adb35-> join skillrel sr ON tr.agent_id=sr.agent_id
spr2022adb35-> JOIN skill s ON sr.skill_id=s.skill_id
spr2022adb35-> where s.skill='Counterintelligence';
      name
---------------
 Oink
 Roadkill
 Beasties
 F Sharp
 Boat Team 1
 Swing Voters
 Camaro
 Wired
 Rimspeed
 Blaster
 Cyclone
 Boat Team 6
 Blackout
 Widow Makers
 Charley Hunter
 Boat Team 3
 Blue Dagger
 Boat Team 2
 SqueakyClean
 Terminator
 FlyOnTheWall
(21 rows)
```

There are 21 rows in the result.

b) The query for this problem statement is given below: -

(SELECT DISTINCT t.name FROM team t JOIN teamrel tr ON t.team_id = tr.team_id JOIN languagerel lr ON tr.agent_id = lr.agent_id JOIN language l ON lr.lang_id = l.lang_id WHERE l.language = 'Bengali') INTERSECT (SELECT DISTINCT t.name FROM team t JOIN teamrel tr ON t.team_id = tr.team_id JOIN languagerel lr ON tr.agent_id = lr.agent_id JOIN language l ON lr.lang_id = l.lang_id WHERE l.language = 'Polish');

The result for the same is displayed below: -

```
OpenSSH SSH client
spr2022adb35=> (SELECT DISTINCT t.name FROM team t JOIN teamrel tr ON t.team_id = tr.team_id
spr2022adb35(> JOIN languagerel lr ON tr.agent_id = lr.agent_id
spr2022adb35(> JOIN language l ON lr.lang_id = l.lang_id WHERE l.language = 'Bengali')
spr2022adb35-> INTERSECT
spr2022adb35-> (SELECT DISTINCT t.name FROM team t JOIN teamrel tr ON t.team_id = tr.team_id
spr2022adb35(> JOIN languagerel lr ON tr.agent_id = lr.agent_id
spr2022adb35(> JOIN language l ON lr.lang_id = l.lang_id WHERE l.language = 'Polish');
      name
----------------
 Roadkill
 F Sharp
 Camaro
 Ghost Hunters
 Renegade
 Failsafe
 Jester
 Widow Makers
 Charley Hunter
 Boat Team 3
 Boat Team 2
 Boat Team 7
 Oink
 Timebomb
 Beasties
 Swing Voters
 SpecialForces
 Rimspeed
 Cha Cha Cha
 Cyclone
 Gypsies
 Haberdash
 Giraffe
 BumbleBee
 Blackout
 Scorpion
 Boat Team 4
 Terminator
 FlyOnTheWall
(29 rows)
```

There are 29 rows in the result

Question-2) Aggregation, Group by, Having

(a) Produce a list of the number of different skills that are had by members of each team. (Your result will be a list of teams and the number of skills had by members of each team.)

(b) For each language, list the number of different teams whose members know that language. (Your result will be a list of languages and the count of teams with <u>at least one</u> member who knows that language.)

Answer: -

a) The query for this problem statement is given below:-

select t.name, COUNT(DISTINCT s.skill_id) from team t, skill s, teamrel tr, skillrel sr where t.team_id=tr.team_id AND tr.agent_id=sr.agent_id AND s.skill_id=sr.skill_id
GROUP BY t.name;

The result for this query is given below in the screenshot: -

OpenSSH SSH client

```
spr2022adb35=> select t.name, COUNT(DISTINCT s.skill_id)
spr2022adb35-> from team t, skill s, teamrel tr, skillrel sr
spr2022adb35-> where t.team_id=tr.team_id AND tr.agent_id=sr.agent_id
spr2022adb35-> AND s.skill_id=sr.skill_id
spr2022adb35-> GROUP BY t.name;
      name       | count
----------------+-------
 Beasties        |    29
 Blackout        |    16
 Blaster         |    17
 Blue Dagger     |    21
 Blunt           |    25
 Boat Team 1     |    21
 Boat Team 2     |    20
 Boat Team 3     |    20
 Boat Team 4     |    22
 Boat Team 6     |    19
 Boat Team 7     |    19
 BumbleBee       |    21
 Camaro          |    24
 Cha Cha Cha     |    26
 Charley Hunter  |    25
 Cyclone         |    26
 Failsafe        |    22
 FlyOnTheWall    |    22
 F Sharp         |    27
 Ghost Hunters   |    22
 Giraffe         |    20
 Gypsies         |    23
 Haberdash       |    22
 Jester          |    27
 Leadphut        |    19
 Oink            |    21
 Renegade        |    27
 Rimspeed        |    24
 Roadkill        |    24
 Rolaids         |    25
 Scorpion        |    17
 ShowBiz         |    18
 SpecialForces   |    21
 Spoiler         |     2
 SqueakyClean    |    20
 Swing Voters    |    17
 Terminator      |    30
 Thunderbird     |    25
 Timebomb        |    17
 Vikings         |    29
 Widow Makers    |    24
 Wired           |     9
(42 rows)

spr2022adb35=>
```

There are 42 rows in the result.

b) The query for this problem statement along with the screenshot of the result is given below.
Select l.language, count(t.name) from language l, team t, teamrel tr, languagerel lr where l.lang_id=lr.lang_id and tr.agent_id=lr.agent_id AND tr.team_id=t.team_id GROUP BY l.language;

The screenshot given below depicts the result of the given query.

OpenSSH SSH client

```
spr2022adb35=> Select l.language, count(t.name) from language l, team t, teamrel tr, languagerel lr where
l.lang_id=lr.lang_id and tr.agent_id=lr.agent_id AND tr.team_id=t.team_id GROUP BY l.language;
  language   | count
-------------+-------
 Malay       |    62
 Turkish     |    56
 German      |    64
 Cherokee    |    52
 Chinese     |    46
 Pashtu      |    55
 Japanese    |    60
 Spanish     |    60
 Vietnamese  |    56
 Arabic      |    82
 Portuguese  |    76
 French      |    55
 Farsi       |    65
 Hindi       |    74
 Hebrew      |    57
 English     |     6
 Korean      |    65
 Polish      |    69
 Russian     |    68
 Bengali     |    48
(20 rows)
```

There are 20 rows in this.

Part II Table Creation, Population, and Constraints (50 pts)

For the following exercises, you will be creating, modifying, and querying SQL tables. For each item, show the SQL you used and the resulting state (*all rows*) of your table(s) (or the error message that SQL returns). Do all these tasks using SQL statements (not a GUI). You will be using the data from the PDF file linked here: CS486-586_HW2_MusicSrc.pdfand posted in Week 3 in the class folder (adapted from wikipedia and bigfooty.com).

Question-3) Create Table commands (various point values)

(a) Create a table called **Musicians** with columns for artist name, birthday, birth town, country of origin, Albums sold, studio albums, live albums, and gender

a. With artist name as the primary key
b. Birthday is a date, and it should not allow null values.
c. Gender limited to "Male, Female, Non-binary"

Answer: - There are two ways in which we can solve this problem

1) By using Enum

The query used for getting the required result is given below: -

CREATE TYPE gender_enum AS ENUM(‘Male’, ‘Female’, ‘Non-binary’);

CREATE TABLE musicians
(artist_name TEXT PRIMARY KEY, birthday DATE NOT NULL, birthtown TEXT, country TEXT, albums_sold INTEGER, studio_albums INTEGER, live_albums INTEGER, gender gender_enum);

2) The second way to solve this problem is to use the CHECK constraint checker. The query used for solving is given below: -

CREATE TABLE Musicians
(artist_name TEXT PRIMARY KEY, birthday DATE NOT NULL, birthtown TEXT, country TEXT, albums_sold INTEGER, studio_albums INTEGER, live_albums INTEGER, gender TEXT CHECK(gender IN('Male','Female','Non-binary')));

The screenshot depicting the successful creation of this table is given below: -

OpenSSH SSH client

```
spr2022adb35=> CREATE TABLE Musicians
spr2022adb35-> (artist_name TEXT PRIMARY KEY, birthday DATE NOT NULL,
spr2022adb35(> birthtown TEXT, country TEXT,
spr2022adb35(> albums_sold INTEGER, studio_albums INTEGER,
spr2022adb35(> live_albums INTEGER,
spr2022adb35(> gender TEXT CHECK(gender IN('Male','Female','Non-binary')));
CREATE TABLE
spr2022adb35=>
```

(b) Insert rows for all musicians with <u>10 or more Studio Albums</u>

Answer: - The query along with the results is given below: -

Insert into musicians values

('David Gilmore', '3/9/1946', 'Cambridge', 'England', 230, 19, 5, 'Male'),

('Jimmy Page', '1/9/1944', 'Middlesex', 'England', 201, 14, 6, 'Male'),

('Beyonce', '9/4/1981', 'Houston, Tx', 'USA', 121, 10, 4, 'Female'),

('Freddy Mercury', '9/5/1946', 'Stone Town', 'Zanzibar', 238, 15, 10, 'Male'),

('Neil Young', '11/12/1945', 'Toronto, Ontario', 'Canada', 101, 45, 9, 'Male');

The screenshot depicting the successful insertion of the records is given below: -

OpenSSH SSH client

```
spr2022adb35=> Insert into musicians values
('David Gilmore', '3/9/1946', 'Cambridge', 'England', 230, 19, 5, 'Male'),
('Jimmy Page', '1/9/1944', 'Middlesex', 'England', 201, 14, 6, 'Male'),
('Beyonce', '9/4/1981', 'Houston, Tx', 'USA', 121, 10, 4, 'Female'),
('Freddy Mercury', '9/5/1946', 'Stone Town', 'Zanzibar', 238, 15, 10, 'Male'),
('Neil Young', '11/12/1945', 'Toronto, Ontario', 'Canada', 101, 45, 9, 'Male');
INSERT 0 5
```

(c) Modify your table to add columns for **Full Name**.

Answer: - The query along with the result in the screenshot is given below: -

ALTER TABLE musicians ADD full_name TEXT;

```
spr2022adb35=>
spr2022adb35=>
spr2022adb35=> ALTER TABLE musicians ADD full_name TEXT;
ALTER TABLE
spr2022adb35=> \d musicians;
                Table "spr2022adb35.musicians"
    Column      |  Type   | Collation | Nullable | Default
----------------+---------+-----------+----------+---------
 artist_name    | text    |           | not null |
 birthday       | date    |           | not null |
 birthtown      | text    |           |          |
 country        | text    |           |          |
 albums_sold    | integer |           |          |
 studio_albums  | integer |           |          |
 live_albums    | integer |           |          |
 gender         | text    |           |          |
 full_name      | text    |           |          |
Indexes:
    "musicians_pkey" PRIMARY KEY, btree (artist_name)
Check constraints:
    "musicians_gender_check" CHECK (gender = ANY (ARRAY['Male'::text, 'Female'::text, 'Non-binary'::text]))

spr2022adb35=>
```

(d) Update the existing rows in the table to add **Full Name** information.

Answer: - The queries along with the result in the screenshots is given below: -

→ UPDATE musicians SET full_name = 'David Jon Gilmore' WHERE artist_name='David Gilmore';

→UPDATE musicians SET full_name = 'James Patrick Page' WHERE artist_name='Jimmy Page';

→UPDATE musicians SET full_name = 'Beyonce Giselle Knowles' WHERE artist_name='Beyonce';

→UPDATE musicians SET full_name = 'Farrokh Bulsara' WHERE artist_name='Freddy Mercury';

→UPDATE musicians SET full_name = 'Neil Percival Young' WHERE artist_name='Neil Young';

```
OpenSSH SSH client
spr2022adb35=> UPDATE musicians SET full_name = 'David Jon Gilmore' WHERE artist_name='David Gilmore';
UPDATE 1
spr2022adb35=> UPDATE musicians SET full_name = 'James Patrick Page' WHERE artist_name='Jimmy Page';
UPDATE 1
spr2022adb35=> UPDATE musicians SET full_name = 'Beyonce Giselle Knowles' WHERE artist_name='Beyonce';
UPDATE 1
spr2022adb35=> UPDATE musicians SET full_name = 'Farrokh Bulsara' WHERE artist_name='Freddy Mercury';
UPDATE 1
spr2022adb35=> UPDATE musicians SET full_name = 'Neil Percival Young' WHERE artist_name='Neil Young';
UPDATE 1
spr2022adb35=>
```

(e) What happens if you try to insert **Jimmy Page** a second time?

Answer: - When we try to insert Jimmy Page tuple again, we will receive an error message conveying that we are trying to enter a duplicate primary key.

This can be verified by the screenshot given below: -

```
OpenSSH SSH client
spr2022adb35=> INSERT INTO musicians values ('Jimmy Page', '1/9/1944', 'Middlesex', 'England', 201, 14, 6, 'Male');
ERROR:  duplicate key value violates unique constraint "musicians_pkey"
DETAIL:  Key (artist_name)=(Jimmy Page) already exists.
spr2022adb35=>
spr2022adb35=>
```

(f) Create a second table called **genre** to hold the list of available genres, with a single column, called genre, where genre is unique.

Answer: - The query for creating a table genre is given below: -

create table genre(genre TEXT UNIQUE);

```
OpenSSH SSH client
spr2022adb35=> create table genre(genre TEXT UNIQUE);
CREATE TABLE
spr2022adb35=> \d genre
           Table "spr2022adb35.genre"
 Column | Type | Collation | Nullable | Default
--------+------+-----------+----------+---------
 genre  | text |           |          |
Indexes:
    "genre_genre_key" UNIQUE CONSTRAINT, btree (genre)

spr2022adb35=> CREATE TABLE genre (genre TEXT UNIQUE);
```

(g) Insert rows in the second table corresponding to all possible genres.

Answer: -

The query for inserting multiple rows in the genre table along with the screenshot for the insertion is given below: -

INSERT INTO genre values ('Psychedelic Rock'), ('Blues'), ('Rock'), ('Folk'), ('Hard Rock'), ('R&B'), ('Pop'), ('Hip Hop'), ('Country Rock'), ('Reggae');

OpenSSH SSH client

```
spr2022adb35=> INSERT INTO genre values
spr2022adb35-> ('Psychedelic Rock'),
spr2022adb35-> ('Blues'),
spr2022adb35-> ('Rock'),
spr2022adb35-> ('Folk'),
spr2022adb35-> ('Hard Rock'),
spr2022adb35-> ('R&B'),
spr2022adb35-> ('Pop'),
spr2022adb35-> ('Hip Hop'),
spr2022adb35-> ('Country Rock'),
spr2022adb35-> ('Reggae');
INSERT 0 10
spr2022adb35=>
```

(h) Create a third table called **genrerel** to hold each musician's genres, with columns for musician name and genre, with musician name as a foreign key to the first table, and genre a foreign key to the second table

Answer: -

The query for the insertion as well as screenshot depicting the successful insertion is given below: -

CREATE TABLE genrerel (artist_name TEXT, genre TEXT, CONSTRAINT fk_musician FOREIGN KEY(artist_name) REFERENCES musicians(artist_name), CONSTRAINT fk_genre FOREIGN KEY(genre) REFERENCES genre(genre));

OpenSSH SSH client

```
spr2022adb35=> CREATE TABLE genrerel
spr2022adb35-> (artist_name TEXT, genre TEXT,
spr2022adb35(> CONSTRAINT fk_musician FOREIGN KEY(artist_name) REFERENCES musicians(artist_name),
spr2022adb35(> CONSTRAINT fk_genre FOREIGN KEY(genre) REFERENCES genre(genre));
CREATE TABLE
spr2022adb35=>
spr2022adb35=> \d genrerel
              Table "spr2022adb35.genrerel"
   Column    | Type | Collation | Nullable | Default
-------------+------+-----------+----------+---------
 artist_name | text |           |          |
 genre       | text |           |          |
Foreign-key constraints:
    "fk_genre" FOREIGN KEY (genre) REFERENCES genre(genre)
    "fk_musician" FOREIGN KEY (artist_name) REFERENCES musicians(artist_name)

spr2022adb35=>
```

(i) Insert rows in the third table corresponding to all musicians in the first table. For musicians with multiple genres, each genre should be listed separately.

Answer: - The queries for the insertions as well as the screenshot depicting the successful insertion is given below: -

→INSERT INTO genrerel values ('David Gilmore', 'Psychedelic Rock'), ('David Gilmore', 'Blues'), ('David Gilmore', 'Rock');

→INSERT INTO genrerel values ('Jimmy Page', 'Rock'), ('Jimmy Page', 'Blues'), ('Jimmy Page', 'Folk'), ('Jimmy Page', 'Hard Rock');

→INSERT INTO genrerel values ('Beyonce', 'R&B'), ('Beyonce', 'Pop'), ('Beyonce', 'Hip Hop');

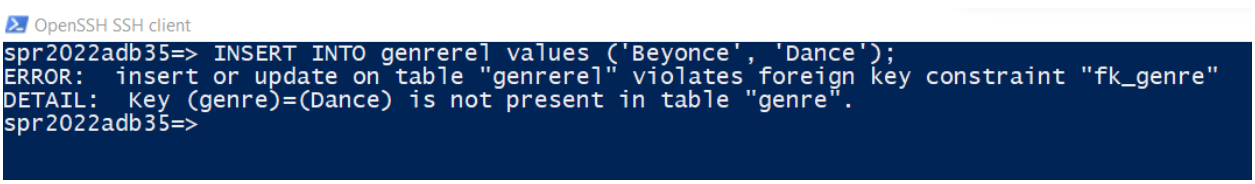
→INSERT INTO genrerel values ('Freddy Mercury', 'Rock');

→INSERT INTO genrerel values ('Neil Young', 'Rock'), ('Neil Young', 'Folk'), ('Neil Young', 'Hard Rock'), ('Neil Young', 'Country Rock');

(j) What happens if you try to insert **Dance** as a genre for **Beyonce?**

Answer: - When we try to insert dance as a genre for Beyonce, it gives an error. This is because it is violating the foreign key constraint. Hence the insertion would be unsuccessful. This can also be seen from the screenshot below: -

INSERT INTO genrerel values ('Beyonce', 'Dance');



(k) What happens if you try to delete the row in the first table for **David Gilmore**?

Answer: - When we try to delete the tuple David Gilmore, then we will receive the error for constraint violation. This is because the referenced value from another table cannot be deleted until and unless reference is also deleted.

The query and the associated result (in the form of a screenshot) is given below: -

DELETE FROM musicians WHERE artist_name='David Gilmore';

OpenSSH SSH client

```
spr2022adb35=> DELETE FROM musicians WHERE artist_name='David Gilmore';
ERROR:  update or delete on table "musicians" violates foreign key constraint "fk_musician" on table "genrerel"
DETAIL:  Key (artist_name)=(David Gilmore) is still referenced from table "genrerel".
spr2022adb35=>
spr2022adb35=>
spr2022adb35=>
```

(l) Write a query to find the total sales amount for all Folk musicians from England and Canada.

Answer: - The query along with the screenshot of the result is attached below: -

SELECT SUM(m.albums_sold) FROM musicians m, genrerel gr WHERE gr.genre = 'Folk' AND m.artist_name = gr.artist_name AND (m.country = 'England' OR m.country = 'Canada');

OpenSSH SSH client

```
spr2022adb35=> SELECT SUM(m.albums_sold) FROM musicians m, genrerel gr
spr2022adb35-> WHERE gr.genre = 'Folk' AND m.artist_name = gr.artist_name
spr2022adb35-> AND (m.country = 'England' OR m.country = 'Canada');
 sum
-----
 302
(1 row)

spr2022adb35=>
```