CS486/586 Introduction to Databases
Spring 2021 Quarter

Assignment 2 – DDL & DML; SQL & Relational Algebra
Due: Friday, October 15th, 11:59PM on Canvas

Submitted By: Agnita, Raghavendra (Team 3)

Updates:

- 3k - Pink Floyd should be David Gilmore
- 3m - question deleted (material not covered yet)
- Modified Part III to be Part II.
- My apologies for the series of errors. I am modifying homeworks from the in-person class to make them work for the online class and the transition has not been perfectly smooth. Sorry!

Instructions & Notes:

- Do this assignment in groups of 2.
- Ensure that each group member's name is listed on the assignment, and in the notes field of Canvas to ensure credit.
- Submit your assignment in PDF format.
- Submit your completed assignment on Canvas, one submission per group.
- 100 points total.

Part I – More SQL (45 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.

1) Different types of JOINs and SET operators (10 pts each)
   (a) Find the team name for all teams with at least one agent who is skilled at Biologist.

Solution:

```
SELECT DISTINCT team.name FROM team
INNER JOIN teamrel  ON team.team_id=teamrel.team_id
INNER JOIN skillrel ON teamrel.agent_id=skillrel.agent_id
INNER JOIN skill ON skillrel.skill_id=skill.skill_id
WHERE skill.skill='Biologist';
```

```
fall2021db78=> SELECT DISTINCT team.name FROM team
fall2021db78-> INNER JOIN teamrel ON team.team_id=teamrel.team_id
fall2021db78-> INNER JOIN skillrel ON teamrel.agent_id=skillrel.agent_id
fall2021db78-> INNER JOIN skill ON skillrel.skill_id=skill.skill_id
fall2021db78-> WHERE skill.skill='Biologist';
     name
---------------
 Leadphut
 Oink
 Camaro
 Ghost Hunters
 Vikings
 Cha Cha Cha
 Thunderbird
 Cyclone
 Renegade
 Jester
 Failsafe
 Giraffe
 Blackout
 BumbleBee
 Widow Makers
 Boat Team 3
 Blue Dagger
 SqueakyClean
 Terminator
(19 rows)
```

The Number of rows returned: 19 (DISTINCT)

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

Solution:

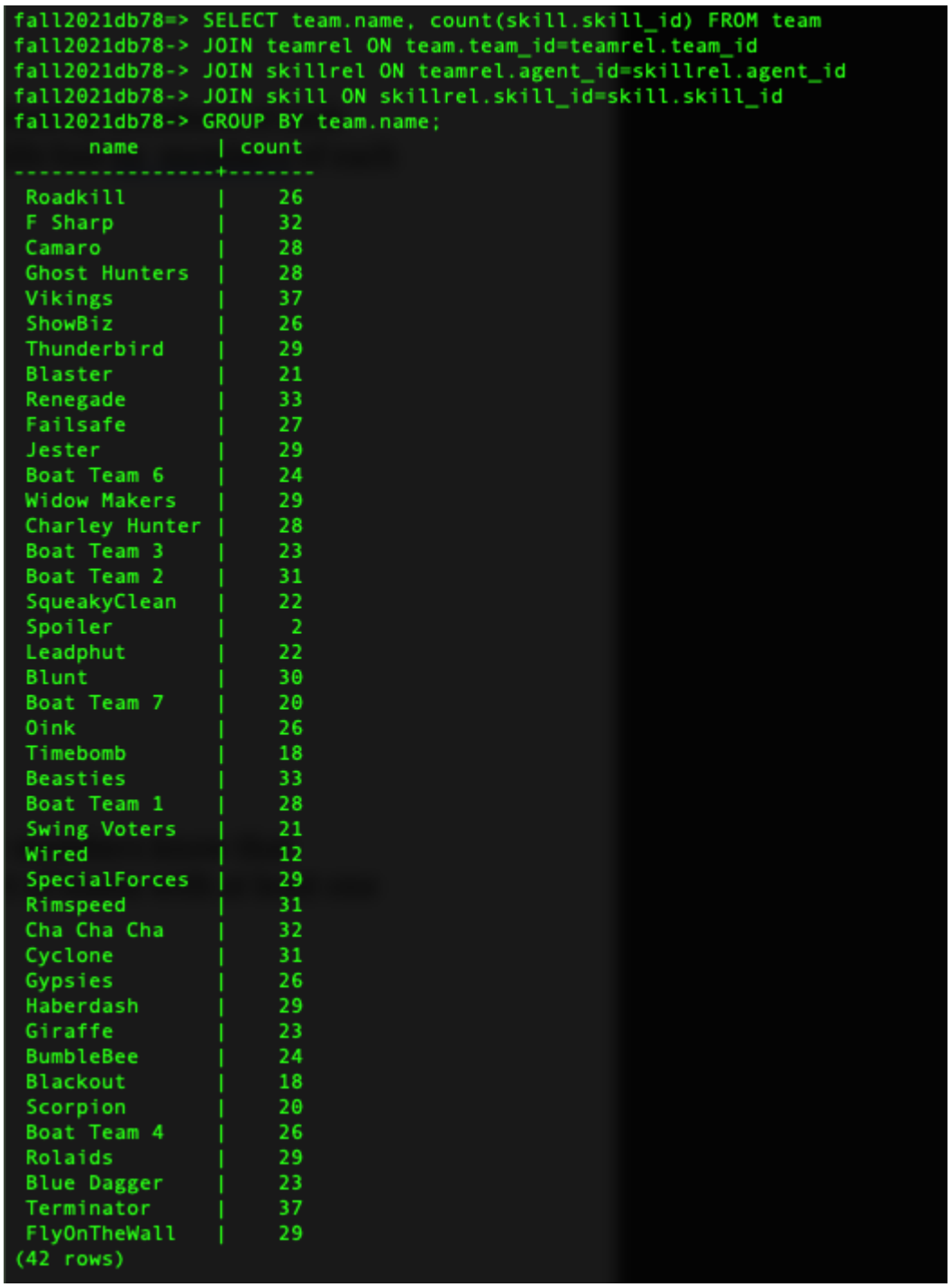
```
SELECT DISTINCT team.name FROM team
INNER JOIN teamrel ON team.team_id = teamrel.team_id
INNER JOIN languagerel ON teamrel.agent_id= languagerel.agent_id
INNER JOIN language ON languagerel.lang_id=language.lang_id
WHERE language.language in('German')
INTERSECT
SELECT DISTINCT team.name FROM team
INNER JOIN teamrel ON team.team_id = teamrel.team_id
INNER JOIN languagerel ON teamrel.agent_id=languagerel.agent_id
INNER JOIN language ON languagerel.lang_id=language.lang_id
WHERE language.language in('Arabic');
```

```
fall2021db78=> SELECT DISTINCT team.name FROM team
fall2021db78-> INNER JOIN teamrel ON team.team_id = teamrel.team_id
fall2021db78-> INNER JOIN languagerel ON teamrel.agent_id= languagerel.agent_id
fall2021db78-> INNER JOIN language ON languagerel.lang_id=language.lang_id
fall2021db78-> WHERE language.language in('German')
fall2021db78-> INTERSECT
fall2021db78-> SELECT DISTINCT team.name FROM team
fall2021db78-> INNER JOIN teamrel ON team.team_id = teamrel.team_id
fall2021db78-> INNER JOIN languagerel ON teamrel.agent_id=languagerel.agent_id
fall2021db78-> INNER JOIN language ON languagerel.lang_id=language.lang_id
fall2021db78-> WHERE language.language in('Arabic');
      name
----------------
 F Sharp
 Camaro
 Ghost Hunters
 Vikings
 ShowBiz
 Thunderbird
 Renegade
 Jester
 Boat Team 6
 Widow Makers
 Charley Hunter
 Boat Team 3
 Leadphut
 Blunt
 Boat Team 7
 Oink
 Timebomb
 Beasties
 Boat Team 1
 SpecialForces
 Wired
 Rimspeed
 Cha Cha Cha
 Gypsies
 Haberdash
 Giraffe
 BumbleBee
 Scorpion
 Rolaids
 Boat Team 4
 Blue Dagger
(31 rows)
```

2) Aggregation, Group by, Having (10 pts each)
   (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.)

Solution:

```
SELECT team.name, count(skill.skill_id) FROM team
JOIN teamrel ON team.team_id=teamrel.team_id
JOIN skillrel ON teamrel.agent_id=skillrel.agent_id
JOIN skill ON skillrel.skill_id=skill.skill_id
GROUP BY team.name;
```



```
fall2021db78=> SELECT team.name, count(skill.skill_id) FROM team
fall2021db78-> JOIN teamrel ON team.team_id=teamrel.team_id
fall2021db78-> JOIN skillrel ON teamrel.agent_id=skillrel.agent_id
fall2021db78-> JOIN skill ON skillrel.skill_id=skill.skill_id
fall2021db78-> GROUP BY team.name;
      name       | count
-----------------+-------
 Roadkill        |    26
 F Sharp         |    32
 Camaro          |    28
 Ghost Hunters   |    28
 Vikings         |    37
 ShowBiz         |    26
 Thunderbird     |    29
 Blaster         |    21
 Renegade        |    33
 Failsafe        |    27
 Jester          |    29
 Boat Team 6     |    24
 Widow Makers    |    29
 Charley Hunter  |    28
 Boat Team 3     |    23
 Boat Team 2     |    31
 SqueakyClean    |    22
 Spoiler         |     2
 Leadphut        |    22
 Blunt           |    30
 Boat Team 7     |    20
 Oink            |    26
 Timebomb        |    18
 Beasties        |    33
 Boat Team 1     |    28
 Swing Voters    |    21
 Wired           |    12
 SpecialForces   |    29
 Rimspeed        |    31
 Cha Cha Cha     |    32
 Cyclone         |    31
 Gypsies         |    26
 Haberdash       |    29
 Giraffe         |    23
 BumbleBee       |    24
 Blackout        |    18
 Scorpion        |    20
 Boat Team 4     |    26
 Rolaids         |    29
 Blue Dagger     |    23
 Terminator      |    37
 FlyOnTheWall    |    29
(42 rows)
```

(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 at least one member who knows that language.)

Solution:

```
SELECT language.language, count(team.name) FROM language
JOIN languagerel ON language.lang_id=languagerel.lang_id
JOIN teamrel ON languagerel.agent_id=teamrel.agent_id
JOIN team ON teamrel.team_id=team.team_id
GROUP BY language.language;
```

```
fall2021db78=> SELECT language.language, count(team.name) FROM language
fall2021db78-> JOIN languagerel ON language.lang_id=languagerel.lang_id
fall2021db78-> JOIN teamrel ON languagerel.agent_id=teamrel.agent_id
fall2021db78-> JOIN team ON teamrel.team_id=team.team_id
fall2021db78-> GROUP BY language.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)
```

Part II Table Creation, Population, and Constraints (55 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.pdf and posted in Week 3 in the class folder (adapted from wikipedia and bigfooty.com).

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 (3 pts)

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"

Solution:

```
CREATE TABLE Musicians(
Artist_Name varchar(255),
Birthday DATE NOT NULL,
Birth_Town varchar(255),
Country_Of_Origin varchar(255),
Albums_Sold INT,
```

```
Studio_Albums INT,
Live_Album INT,
Gender varchar(255),
PRIMARY KEY(Artist_name),
CONSTRAINT CHK_Gender CHECK(Gender IN('Male', 'Female', 'Non-binary')));
```

```
fall2021db78=> SELECT * FROM musicians ;
 artist_name | birthday | birth_town | country_of_origin | albums_sold | studio_albums | live_album | gender
-------------+----------+------------+-------------------+-------------+---------------+------------+--------
(0 rows)
```

(b) Insert rows for all musicians with <u>10 or more Studio Albums</u> (3 pts)

Solution:

```
INSERT INTO Musicians(Artist_Name, Birthday, Birth_Town, Country_Of_Origin, Albums_Sold, Studio_Albums, Live_Album, Gender)
VALUES ('David Gilmore', '3/6/1946', 'Cambridge', 'England', 230, 19, 5, 'Male');
INSERT INTO Musicians(Artist_Name, Birthday, Birth_Town, Country_Of_Origin, Albums_Sold, Studio_Albums, Live_Album, Gender)
VALUES ('Jimmy Page', '1/9/1944', 'Middlesex', 'England', 201, 14, 6, 'Male');
INSERT INTO Musicians(Artist_Name, Birthday, Birth_Town, Country_Of_Origin, Albums_Sold, Studio_Albums, Live_Album, Gender)
VALUES ('Beyonce', '9/4/1981', 'Houston, Tx', 'USA', 121, 10, 4, 'Female');
INSERT INTO Musicians(Artist_Name, Birthday, Birth_Town, Country_Of_Origin, Albums_Sold, Studio_Albums, Live_Album, Gender)
VALUES ('Freddy Mercury', '9/5/1946', 'Stone Town', 'Zanzibar', 238, 15, 10, 'Male');
INSERT INTO Musicians(Artist_Name, Birthday, Birth_Town, Country_Of_Origin, Albums_Sold, Studio_Albums, Live_Album, Gender)
VALUES ('Neil Young', '11/12/1945', 'Toronto, Ontario', 'Canada', 101, 45, 9, 'Male');
```

```
fall2021db78=> select * from musicians ;
  artist_name   |  birthday  |    birth_town    | country_of_origin | albums_sold | studio_albums | live_album | gender
----------------+------------+------------------+-------------------+-------------+---------------+------------+--------
 David Gilmore  | 1946-03-06 | Cambridge        | England           |         230 |            19 |          5 | Male
 Jimmy Page     | 1944-01-09 | Middlesex        | England           |         201 |            14 |          6 | Male
 Beyonce        | 1981-09-04 | Houston, Tx      | USA               |         121 |            10 |          4 | Female
 Freddy Mercury | 1946-09-05 | Stone Town       | Zanzibar          |         238 |            15 |         10 | Male
 Neil Young     | 1945-11-12 | Toronto, Ontario | Canada            |         101 |            45 |          9 | Male
(5 rows)
```

(c) Modify your table to add columns for **Full Name**. (3 pts)

Solution:

```
ALTER table musicians ADD Full_name varchar(255);
```

```
  artist_name   |  birthday  |    birth_town    | country_of_origin | albums_sold | studio_albums | live_album | gender | full_name
----------------+------------+------------------+-------------------+-------------+---------------+------------+--------+-----------
 David Gilmore  | 1946-03-06 | Cambridge        | England           |         230 |            19 |          5 | Male   |
 Jimmy Page     | 1944-01-09 | Middlesex        | England           |         201 |            14 |          6 | Male   |
 Beyonce        | 1981-09-04 | Houston, Tx      | USA               |         121 |            10 |          4 | Female |
 Freddy Mercury | 1946-09-05 | Stone Town       | Zanzibar          |         238 |            15 |         10 | Male   |
 Neil Young     | 1945-11-12 | Toronto, Ontario | Canada            |         101 |            45 |          9 | Male   |
(5 rows)
```

(d) Update the existing rows in the table to add **Full Name** information. (3 pts)

Solution:

```
UPDATE musicians SET full_name = 'David Jon Gilmour' 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';
```

```
fall2021db78=> select * from musicians;
  artist_name   |  birthday  |     birth_town     | country_of_origin | albums_sold | studio_albums | live_album | gender |        full_name
----------------+------------+--------------------+-------------------+-------------+---------------+------------+--------+-------------------------
 David Gilmore  | 1946-03-06 | Cambridge          | England           |         230 |            19 |          5 | Male   | David Jon Gilmour
 Jimmy Page     | 1944-01-09 | Middlesex          | England           |         201 |            14 |          6 | Male   | James Patrick Page
 Beyonce        | 1981-09-04 | Houston, Tx        | USA               |         121 |            10 |          4 | Female | Beyonce Giselle Knowles
 Freddy Mercury | 1946-09-05 | Stone Town         | Zanzibar          |         238 |            15 |         10 | Male   | Farrokh Bulsara
 Neil Young     | 1945-11-12 | Toronto, Ontario | Canada            |         101 |            45 |          9 | Male   | Neil Percival Young
(5 rows)
```

(e) What happens if you try to insert **Jimmy Page** a second time? (3 pts)

Solution:

```
INSERT INTO Musicians(Artist_Name, Birthday, Birth_Town,
Country_Of_Origin, Albums_Sold, Studio_Albums, Live_Album, Gender,
Full_Name)
VALUES ('Jimmy Page', '9/5/1946', 'Stone Town', 'Zanzibar', 238, 15, 10,
'Male', 'James Patrick Page');
```

We will get an error because we are trying to enter a duplicate key twice in the primary key column

```
fall2021db78=> INSERT INTO Musicians(Artist_Name, Birthday, Birth_Town, Country_Of_Origin, Albums_Sold, Studio_Albums, Live_Album, Gender, Full_Name)
fall2021db78-> VALUES ('Jimmy Page', '9/5/1946', 'Stone Town', 'Zanzibar', 238, 15, 10, 'Male', 'James Patrick Page');
ERROR:  duplicate key value violates unique constraint "musicians_pkey"
DETAIL:  Key (artist_name)=(Jimmy Page) already exists.
fall2021db78=>
```

(f) Create a second table called **genre** to hold the list of available genres, with a single column, called genre, where genre is unique. (3 pts)

Solution:

```
CREATE TABLE genre (genre varchar(255) UNIQUE);
```

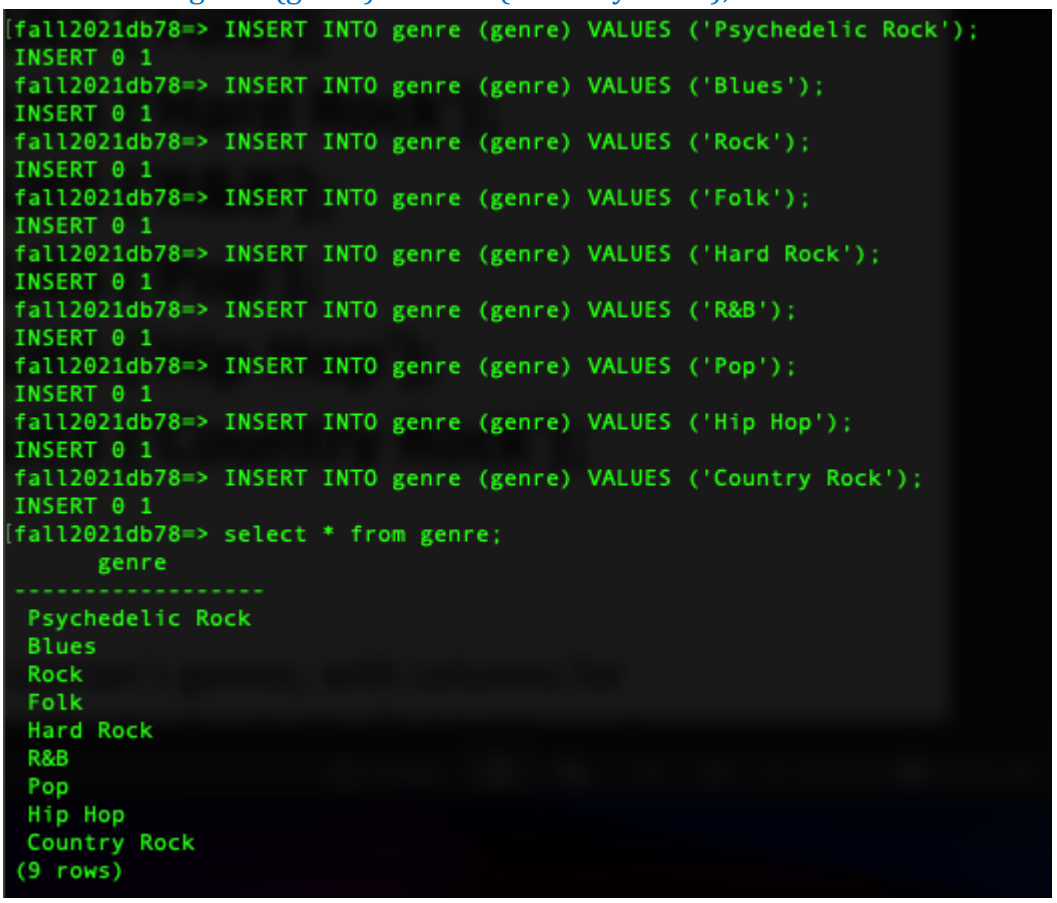
```
fall2021db78=> CREATE TABLE genre (genre varchar(255) UNIQUE);
CREATE TABLE
fall2021db78=> select * from genre;
 genre
-------
(0 rows)
```

(g) Insert rows in the second table corresponding to all possible genres. (3 pts)

Solution:

```
INSERT INTO genre (genre) VALUES ('Psychedelic Rock');
INSERT INTO genre (genre) VALUES ('Blues');
INSERT INTO genre (genre) VALUES ('Rock');
INSERT INTO genre (genre) VALUES ('Folk');
INSERT INTO genre (genre) VALUES ('Hard Rock');
INSERT INTO genre (genre) VALUES ('R&B');
```

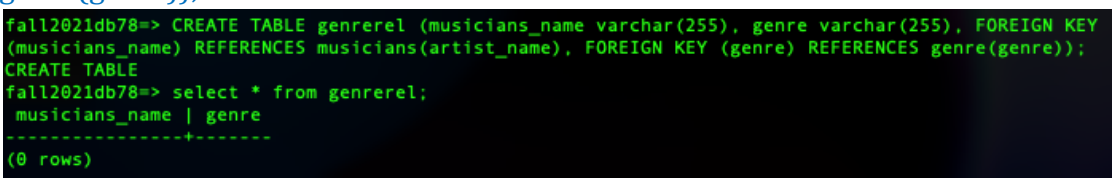
INSERT INTO genre (genre) VALUES ('Pop');
INSERT INTO genre (genre) VALUES ('Hip Hop');
INSERT INTO genre (genre) VALUES ('Country Rock');



(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 (3 pts)

Solution:

CREATE TABLE genrerel (musicians_name varchar(255), genre varchar(255), FOREIGN KEY (musicians_name) REFERENCES musicians(artist_name), FOREIGN KEY (genre) REFERENCES genre(genre));

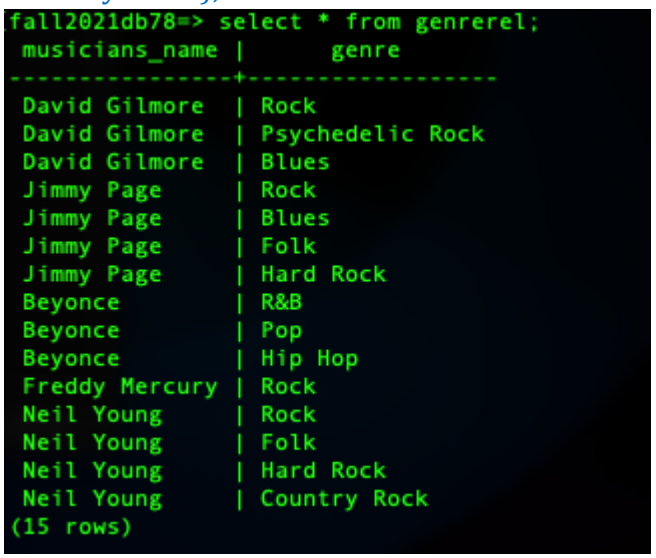


(i) Insert rows in the third table corresponding to all musicians in the first table. For musicians with multiple genres, each genre should be listed separately. (3 pts)

Solution:

INSERT INTO genrerel (musicians_name, genre) VALUES ('David Gilmore', 'Rock');
INSERT INTO genrerel (musicians_name, genre) VALUES ('David Gilmore', 'Psychedelic Rock');

```
INSERT INTO genrerel (musicians_name, genre) VALUES ('David Gilmore', 'Blues');
INSERT INTO genrerel (musicians_name, genre) VALUES ('Jimmy Page', 'Rock');
INSERT INTO genrerel (musicians_name, genre) VALUES ('Jimmy Page', 'Blues');
INSERT INTO genrerel (musicians_name, genre) VALUES ('Jimmy Page', 'Folk');
INSERT INTO genrerel (musicians_name, genre) VALUES ('Jimmy Page', 'Hard Rock');
INSERT INTO genrerel (musicians_name, genre) VALUES ('Beyonce', 'R&B');
INSERT INTO genrerel (musicians_name, genre) VALUES ('Beyonce', 'Pop');
INSERT INTO genrerel (musicians_name, genre) VALUES ('Beyonce', 'Hip Hop');
INSERT INTO genrerel (musicians_name, genre) VALUES ('Freddy Mercury', 'Rock');
INSERT INTO genrerel (musicians_name, genre) VALUES ('Neil Young', 'Rock');
INSERT INTO genrerel (musicians_name, genre) VALUES ('Neil Young', 'Folk');
INSERT INTO genrerel (musicians_name, genre) VALUES ('Neil Young', 'Hard Rock');
INSERT INTO genrerel (musicians_name, genre) VALUES ('Neil Young', 'Country Rock');
```



```
fall2021db78=> select * from genrerel;
 musicians_name |      genre
----------------+------------------
 David Gilmore  | Rock
 David Gilmore  | Psychedelic Rock
 David Gilmore  | Blues
 Jimmy Page     | Rock
 Jimmy Page     | Blues
 Jimmy Page     | Folk
 Jimmy Page     | Hard Rock
 Beyonce        | R&B
 Beyonce        | Pop
 Beyonce        | Hip Hop
 Freddy Mercury | Rock
 Neil Young     | Rock
 Neil Young     | Folk
 Neil Young     | Hard Rock
 Neil Young     | Country Rock
(15 rows)
```

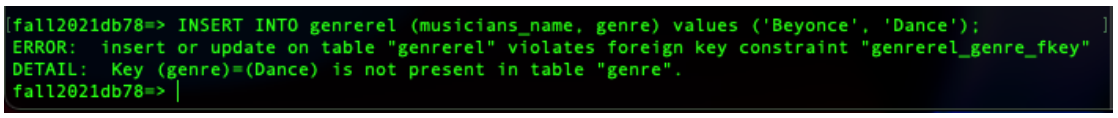
(j) What happens if you try to insert **Dance** as a genre for **Beyonce?** (3 pts)

Solution:

```
INSERT INTO genrerel (musicians_name, genre) values ('Beyonce', 'Dance');
```

We will get an error because we are trying to enter a foreign key that is

not present in the referenced table i.e., genre

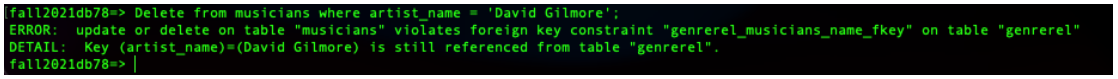


(k) What happens if you try to delete the row in the first table for **David Gilmore** ~~**Pink Floyd**~~? (3 pts)

Solution:

Delete from musicians where artist_name = 'David Gilmore';

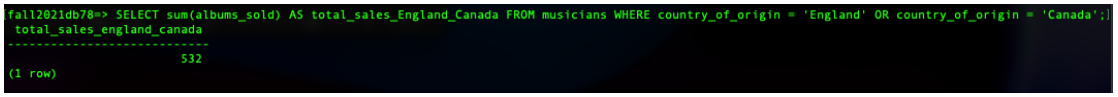
We will get an error because we are deleting a foreign key from musicians table that is already mapped from general table and is still being used.



(l) Write a query to find the total sales amount for all Folk musicians from England and Canada. (6 pts)

Solution:

SELECT sum(albums_sold) AS total_sales_England_Canada FROM musicians WHERE country_of_origin = 'England' OR country_of_origin = 'Canada';



~~(m) Write a query to find the musicians with the highest combination of **studio** and **live albums**. (6 pts)~~