Advanced Queries for Checkpoint 5

a.

```
SELECT Person.Name, SUM(Movie.Length) AS TotalMovieRuntime
FROM Person, Patron, Loanable, Movie
WHERE Person.PersonID = Patron.PersonID AND Loanable.PersonID = Person.PersonID
AND Movie.ID = Loanable.ID
GROUP BY Person.Name
```

Expected:

Name	TotalMovieRuntime
Emma Fuerst	1
Joe Schmoe	1
Michael Not	3
Michael Pot	2
Michael Squat	2
Michael Taught	2
Tyler Fuerst	1

b.

```
SELECT Person.Name, LibraryPerson.Email
FROM Person, Patron, LibraryPerson, Album, Loanable
WHERE Person.PersonID = Patron.PersonID AND Person.PersonID =
LibraryPerson.PersonID AND Loanable.PersonID = Person.PersonID AND Loanable.ID =
Album.ID
GROUP BY(Person.Name)
HAVING COUNT(Loanable.Name) >
(
        SELECT AVG(NumAlbums) AS AvgNumAlbums
        FROM
        (
                SELECT Name, COUNT(*) AS NumAlbums
                FROM
                (
                        SELECT *
                        FROM Person, Patron, Album, Loanable
                        WHERE Person.PersonID = Patron.PersonID AND Person.PersonID
                        = Loanable.PersonID AND Loanable.ID = Album.ID
                ) AS CountOfAlbums
                GROUP BY CountOfAlbums.Name
        )
);
```

Expected:

Name	Email
Michael Not	e1234@gmail.com
Michael Pot	f1234@gmail.com
Tyler Fuerst	p1234@gmail.com

c.

```
SELECT Loanable.Name, COUNT(*) AS NumLentOut
FROM Loanable, Movie
WHERE Loanable.ID = Movie.ID AND Loanable.CheckoutDate IS NOT NULL
GROUP BY Loanable.Name
ORDER BY NumLentOut DESC
```

Expected:

Name	NumLentOut
Forrest Gump	1
Lord of the Rings: The Fellowship of the Ring	1
Lord of the Rings: The Return of the King	1
Lord of the Rings: The Two Towers	1
Star Wars Episode I: The Phantom Menace	1
Star Wars Episode II: The Attack of the Clones	1
Star Wars Episode III: The Revenge of the Sith	1
Star Wars Episode IV: A New Hope	1

d.

```
SELECT Loanable.Name, COUNT(*) AS NumLentOut
FROM Loanable, Album
WHERE Loanable.ID = Album.ID AND Loanable.PersonID IS NOT NULL
GROUP BY Loanable.Name
ORDER BY NumLentOut DESC
```

Expected:

Name	NumLentOut
808s and heartbreak	1
Graduation	1
Jesus is King	1
Kamikaze	1
Late Registration	1
Music to be murdered by	1
My beautiful dark twisted fantasy	1
Recovery	1
Relapse	1
Revival	1
The Best Album EVER	1
The Eminem show	1
The Marshall Mathers LP	1
The Marshall Mathers LP 2	1
The life of Pablo	1
Yeezus	1

e.

```
SELECT Name
FROM
(
        SELECT Person.Name, COUNT(*) AS NumMovies
        FROM Person, Actor, Loanable, Movie, MovieActor
                WHERE Person.PersonID = Actor.PersonID AND Movie.ID = Loanable.ID
                AND MovieActor.ActorID = Actor.PersonID AND MovieActor.MovieID =
                Movie.ID AND Loanable.PersonID IS NOT NULL
        GROUP BY Person.Name
        ORDER BY NumMovies DESC
        LIMIT 1
) AS TopMovieCount
```

Expected:

Name
Dwight Fruit

f.

```
SELECT Name
FROM
(
	SELECT Person.Name, (COUNT(*) * Track.Length) AS TotalTimeForArtist
	FROM Person, Artist, Loanable, Album, ArtistAlbum  AS AA, Track
	WHERE Person.PersonID = Artist.PersonID AND Loanable.ID  = Album.ID AND
	AA.AlbumID = Album.ID AND AA.ArtistID = Artist.PersonID  AND Album.ID = Track.ID
	AND Loanable.PersonID IS NOT NULL
	GROUP BY Person.Name
	ORDER BY TotalTimeForArtist DESC
	LIMIT 1
);
```

Expected:

Name

John Kennedy

g.

```
SELECT P.Name, LP.Email, LP.Address
FROM Person AS P, LibraryPerson AS LP, Actor, Movie,  Loanable, MovieActor, Person AS
A,
(
	SELECT Name
	FROM
	(
		SELECT Person.Name, COUNT(*) AS NumMovies
		FROM Person, Actor, Loanable, Movie, MovieActor
		WHERE Person.PersonID = Actor.PersonID AND Movie.ID  = Loanable.ID
		AND MovieActor.ActorID = Actor.PersonID AND MovieActor.MovieID  =
		Movie.ID AND Loanable.PersonID IS NOT NULL
		GROUP BY Person.Name
		ORDER BY NumMovies DESC
		LIMIT 1
	)
) AS TopMovieActor
WHERE P.PersonID = LP.PersonID AND Loanable.ID = Movie.ID AND Loanable.PersonID =
P.PersonID AND MovieActor.MovieID = Movie.ID AND MovieActor.ActorID  = Actor.PersonID
AND A.PersonID = Actor.PersonID AND A.Name = TopMovieActor.Name
```

Expected:

Name	Email	Address
Michael Taught	b1234@gmail.com	8222 Tree Street, Columbus OH
Michael Not	e1234@gmail.com	7222 Bight Street, Columbus OH
Michael Squat	j1234@gmail.com	1892 Candy Street, Columbus OH

h.

```
SELECT A.Name
FROM Person AS A, Person AS P, Album, ArtistAlbum,  Loanable,
(
	-- taken and slightly modified from Part b
	SELECT Person.PersonID
	FROM Person, Patron, LibraryPerson, Album, Loanable
	WHERE Person.PersonID = Patron.PersonID AND Person.PersonID =
	LibraryPerson.PersonID AND Loanable.PersonID = Person.PersonID AND
	Loanable.ID = Album.ID
	GROUP BY(Person.Name)
	HAVING COUNT(Loanable.Name) >
	(
		SELECT AVG(NumAlbums) AS AvgNumAlbums
		FROM
		(
			SELECT Name, COUNT(*) AS NumAlbums
			FROM
			(
				SELECT *
				FROM Person, Patron, Album, Loanable
				WHERE Person.PersonID = Patron.PersonID AND
				Person.PersonID = Loanable.PersonID AND Loanable.ID =
				Album.ID
			) AS CountOfAlbums
			GROUP BY CountOfAlbums.Name
		)
	)
) AS CheckoutPatron
WHERE CheckoutPatron.PersonID = P.PersonID AND Loanable.ID = Album.ID AND
Loanable.PersonID = P.PersonID AND Album.ID = ArtistAlbum.AlbumID  AND A.PersonID =
ArtistAlbum.ArtistID
```

Expected:
Name
Tyler Fuerst
Emma Fuerst
Michael Scott
Ash Ketchum
Jim Halpert
Pamela Halpert

Note: We were not instructed to change anything from Checkpoint 4, so we will not be attaching those files. Furthermore, we were unable to get part 5 working because we had issues with JDBC. Although the group followed the directions provided, we were unable to

get past the .jar file, as it caused problems despite trying different versions (it was unable to be opened, even via terminal).