Podcast Database Part 6

Nicholas Guzman | Kristijan Hornung | Victor Cardenas
nguzman9@csustan.edu | khornung@csustan.edu | vcardenas4@csustan.edu

Section 1 - SQL Schemas:

```
        CREATE TABLE Person (
        PersonID        VARCHAR(11)       PRIMARY KEY,
        Firstname       VARCHAR(100)      NOT NULL,
        Lastname        VARCHAR(100)      NOT NULL,
        Nickname        VARCHAR(50),
        Title           VARCHAR(30),
        Age             INTEGER(3)        NOT NULL,
        Bio             TEXT(30,000),
        ProfilePic      BLOB(65,535),
        Linkedin        VARCHAR(100),
        Twitter                 VARCHAR(100),
        Facebook        VARCHAR(100),
        Wiki            VARCHAR(100),
        Youtube         VARCHAR(100),
        Instagram       VARCHAR(100));

CREATE TABLE Podcaster (
        PersonID                VARCHAR(11)       REFERENCES Person (PersonID),
        ExpertiseLevel          VARCHAR(30),
        NativeLanguage          VARCHAR(30));

Create Table Guest (
        PersonID                VARCHAR(11)       REFERENCES Person (PersonID),
        Profession              VARCHAR(30));

CREATE TABLE Podcast (
        PodID                           VARCHAR(11)       PRIMARY KEY,
        PodcastFormat                   VARCHAR(30),
        Genre                           VARCHAR(40),
        Title                           VARCHAR(40),
        NumberOfEpisodes                INTEGER(5)        NOT NULL,
        NumberOfSubscribers                     INTEGER(15),
        AgeRating                       VARCHAR(5),
        Description                     TEXT(30,000));
```

```
CREATE TABLE Episode (
        EpID                VARCHAR(11)        PRIMARY KEY,
        EpDate                      DATETIME,
        EpLength            TIME               NOT NULL,
        EpisodeNumber       INTEGER(5),
        ViewerCount         INTEGER(8),
        DownloadSize                INTEGER(7)         NOT NULL,
        Blurb               TEXT(30,000));

CREATE TABLE CuratedCollection (
        CurCollectionID     VARCHAR(11)                PRIMARY KEY,
        Name                VARCHAR(50),
        TrackSize           INTEGER(5),
        DateStarted         DATE,
        DateFinished        DATE,
        MaturityRating      VARCHAR(30),
        Description         TEXT(30,000));

CREATE TABLE Artwork (
        ArtworkID           VARCHAR(11)                PRIMARY KEY,
        Artist              VARCHAR(100),
        Theme                       VARCHAR(50),
        Title               VARCHAR(100),
        DateCreated         DATE,
        SubjectMatter               VARCHAR(100),
        MeaningDescription TEXT(30,000));

CREATE TABLE Platform (
        PlatformID          VARCHAR(11)        PRIMARY KEY,
        Name                VARCHAR(50),
        WebsiteURL          VARCHAR(100),
        Device                      VARCHAR(50),
        DownloadLink        VARCHAR(100),
        Cost                FLOAT(3,2));

CREATE TABLE Rating    (
        EpID                  VARCHAR(11),
        Review Comment      VARCHAR(200),
        NumberOfStars       VARCHAR(1),
        TimeAndDate                 DATETIME,
        Username            VARCHAR(50),
        NumberOfLikes       INTEGER(11),
        CONSTRAINT Rating_TimeAndDate_Username_EpID_pk PRIMARY KEY
(TimeAndDate, Username, EpID),
        CONSTRAINT Rating_EpID_fk FOREIGN KEY (EpID) REFERENCES Episode
(EpID) ON DELETE CASCADE);

CREATE TABLE  RaterProfile (
        Username            VARCHAR(50)      PRIMARY KEY,
        RaterPic            BLOB(65535));
        CONSTRAINT RaterProfile_Username_pk PRIMARY KEY (Username),
        CONSTRAINT RaterProfile_Username_fk FOREIGN KEY (Username)
REFERENCES Rating (Username) ON DELETE CASCADE);

CREATE TABLE Hosts (
        PersonID            VARCHAR(11),
        PodID               VARCHAR(11),
        CONSTRAINT Hosts_PersonID_PodID_ PRIMARY KEY (PersonID, PodID),
        CONSTRAINT Hosts_PersonID_fk FOREIGN KEY (PersonID) REFERENCES
Podcaster (PersonID) ON DELETE CASCADE,
        CONSTRAINT Hosts_PodID_fk FOREIGN KEY (PodID) REFERENCES
Podcast
(PodID) ON DELETE CASCADE);

CREATE TABLE AppearsIn (
        PersonID      VARCHAR(11),
        EpID          VARCHAR(11),
        CONSTRAINT AppearsIn_PersonID_EpID_pk PRIMARY KEY  (PersonID,
EpID),
        CONSTRAINT AppearsIn_fk FOREIGN KEY (PersonID) REFERENCES Guest
(PersonID) ON DELETE CASCADE,
        CONSTRAINT AppearsIn_fk FOREIGN KEY (EpID) REFERENCES Episode
(EpID) ON DELETE CASCADE);

CREATE TABLE RunOn (
        PlatformID          VARCHAR(11),
        PodID               VARCHAR(11),
        CONSTRAINT          RunOn_PlatformID_PodID_pk PRIMARY KEY
(PlatformID, PodID),
        CONSTRAINT RunOn_PlatformID_fk FOREIGN KEY (PlatformID)
REFERENCES Platform (PlatformID),
        CONSTRAINT RunOn_PodID_fk FOREIGN KEY (PodID) REFERENCES
Podcast (PodID));

CREATE TABLE Offers     (
        PodID                   VARCHAR(11),
        CurCollectionID         VARCHAR(11),
        CONSTRAINT Offers_PodID_CurCollectionID_pk PRIMARY KEY(PodID,
CurCollectionID),
        CONSTRAINT Offers_PodID_fk FOREIGN KEY (PodID) REFERENCES
Podcast (PodID) ON DELETE CASCADE,
        CONSTRAINT Offers_CurCollectionID_fk FOREIGN KEY (CurCollectionID)
REFERENCES Curated Collection (CurCollectionID) ON DELETE CASCADE);

CREATE TABLE Debuts     (
        PodID                           VARCHAR(11),
        EpID                    VARCHAR(11),
        CONSTRAINT Debuts_EpID_pk PRIMARY KEY (EpID),
        CONSTRAINT Debuts_PodID_fk FOREIGN KEY (PodID) REFERENCES
Podcast (PodID) ON DELETE CASCADE,
        CONSTRAINT Debuts_EpID_fk FOREIGN KEY (EpID) REFERENCES
Episode (EpID) ON DELETE CASCADE);

CREATE TABLE IsListedUnder      (
        CurCollectionID         VARCHAR(11),
        EpID                    VARCHAR(11),
        CONSTRAINT IsListedUnder_CurCollectionID_EpID_pk PRIMARY KEY
(CurCollectionID, EpID),
        CONSTRAINT IsListedUnder_CurCollectionID_fk FOREIGN
KEY(CurCollectionID) REFERENCES Curated Collection (CurCollectionID) ON
DELETE CASCADE,
        CONSTRAINT IsListedUnder_EpID_fk FOREIGN KEY(EpID) REFERENCES
Episode (EpID) ON DELETE CASCADE);

CREATE TABLE Presents (
        EpID VARCHAR(11),
        ArtworkID VARCHAR(11),
        CONSTRAINT Presents_ArtworkID_pk PRIMARY KEY (ArtworkID),
```

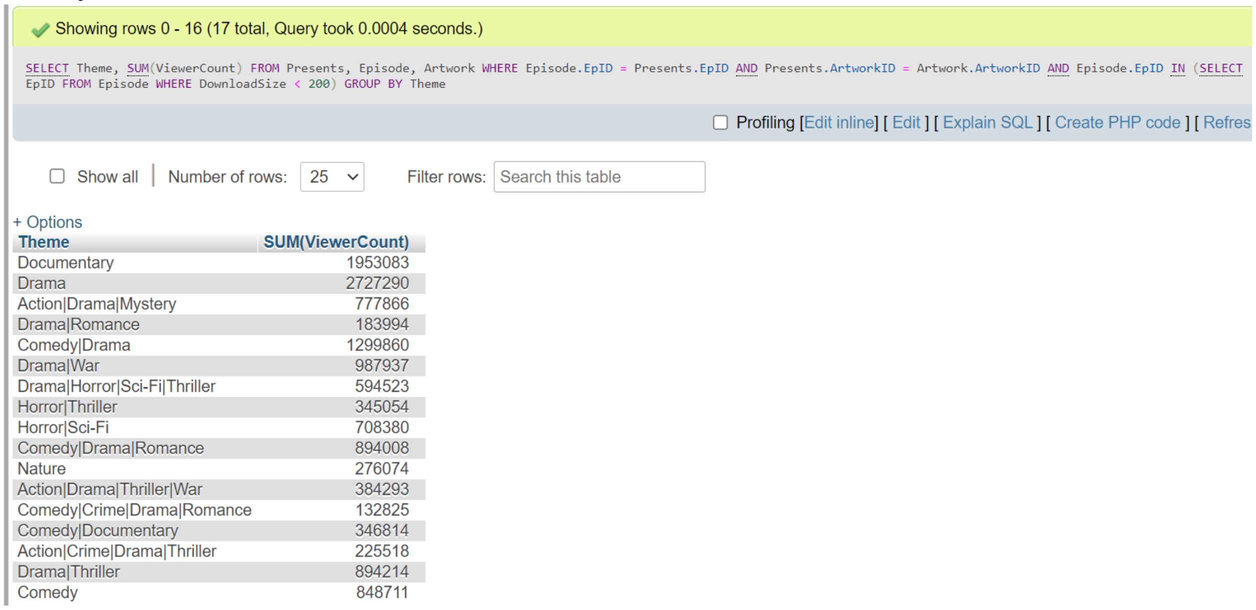
    CONSTRAINT Presents_ArtworkID_fk FOREIGN KEY (ArtworkID) REFERENCES Artwork (ArtworkID) ON DELETE CASCADE,
    CONSTRAINT Presents_EpID_fk FOREIGN KEY (EpID) REFERENCES Episode (EpID) ON DELETE CASCADE);

Section 2 Queries and Explanation:

Query 1)
SELECT Theme, SUM(ViewerCount)
FROM Presents, Episode, Artwork
WHERE Episode.EpID = Presents.EpID AND Presents.ArtworkID = Artwork.ArtworkID AND Episode.EpID IN (SELECT EpID FROM Episode WHERE DownloadSize < 200)
GROUP BY Theme

Description: This query shows a total number of viewers of podcast episodes per each artwork theme (taking into account only those episodes that contain some artwork), with a condition that episodes under consideration are below 200 MB in terms of download size. Episode artwork in our database comes from movies, so the theme is designated by movie genre.

Query result:



Showing rows 0 - 16 (17 total, Query took 0.0004 seconds.)

```
SELECT Theme, SUM(ViewerCount) FROM Presents, Episode, Artwork WHERE Episode.EpID = Presents.EpID AND Presents.ArtworkID = Artwork.ArtworkID AND Episode.EpID IN (SELECT
EpID FROM Episode WHERE DownloadSize < 200) GROUP BY Theme
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refres

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

Theme	SUM(ViewerCount)
Documentary	1953083
Drama	2727290
Action|Drama|Mystery	777866
Drama|Romance	183994
Comedy|Drama	1299860
Drama|War	987937
Drama|Horror|Sci-Fi|Thriller	594523
Horror|Thriller	345054
Horror|Sci-Fi	708380
Comedy|Drama|Romance	894008
Nature	276074
Action|Drama|Thriller|War	384293
Comedy|Crime|Drama|Romance	132825
Comedy|Documentary	346814
Action|Crime|Drama|Thriller	225518
Drama|Thriller	894214
Comedy	848711

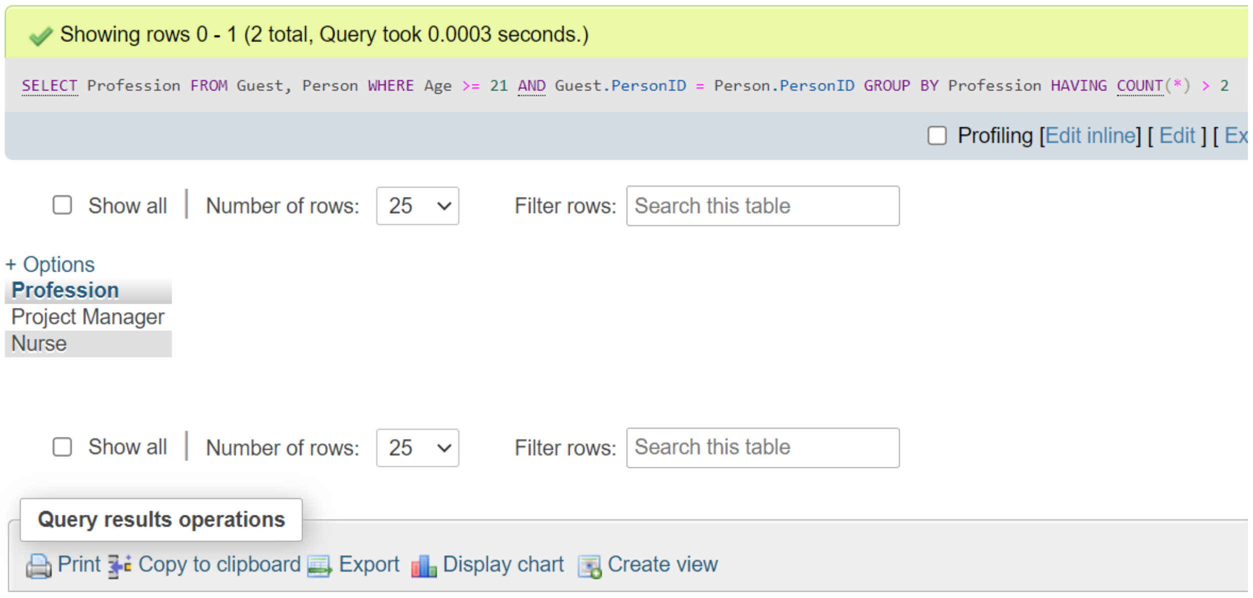
Similar example that did not work as intended:
SELECT Theme, SUM(ViewerCount)
FROM Presents, Episode, Artwork
WHERE Episode.EpID = Presents.EpID
GROUP BY Theme

Explanation: The above query was wrong in that it did not join all three tables properly (it did not join Artwork with Presents through foreign key, since it was missing the following part after WHERE: Presents.ArtworkID = Artwork.ArtworkID).

Query 2)
SELECT Profession
FROM Guest, Person
WHERE Age >= 21 AND Guest.PersonID = Person.PersonID
GROUP BY Profession
HAVING COUNT(*) > 2

Description: This query displays those guest professions that appeared in podcasts in our database more than twice, considering only guests who are over 21 years old. As the query result below shows, only nurses and project managers appeared more than two times.

Query result:



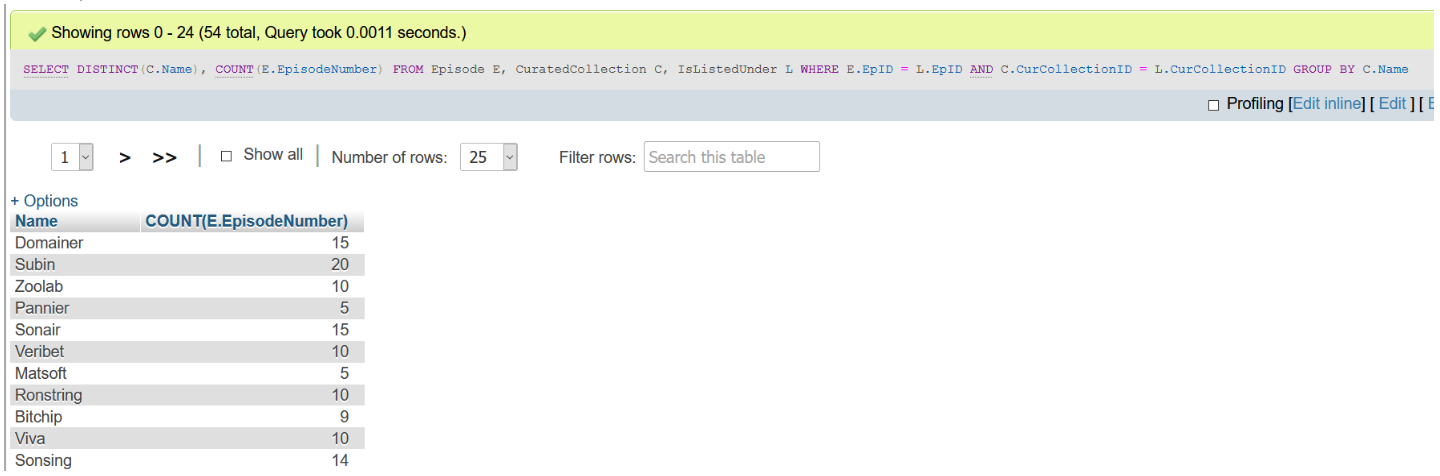
```
Showing rows 0 - 1 (2 total, Query took 0.0003 seconds.)
SELECT Profession FROM Guest, Person WHERE Age >= 21 AND Guest.PersonID = Person.PersonID GROUP BY Profession HAVING COUNT(*) > 2
```

Profession
Project Manager
Nurse

Query 3)
SELECT C.Name, COUNT(E.EpisodeNumber)
FROM Episode E, CuratedCollection C, IsListedUnder L
WHERE E.EpID = L.EpID AND C.CurCollectionID = L.CurCollectionID
GROUP BY C.Name

Description: This query looks for the names of Curated Collections and seeks the number of episodes that each specific Collection possesses. Grouping the query with the Name of the Collection enables the results to be illustrated distinctively by the Name with each episode count; throwing out any duplicates in the process.

Query results:


Showing rows 0 - 24 (54 total, Query took 0.0011 seconds.)

```
SELECT DISTINCT(C.Name), COUNT(E.EpisodeNumber) FROM Episode E, CuratedCollection C, IsListedUnder L WHERE E.EpID = L.EpID AND C.CurCollectionID = L.CurCollectionID GROUP BY C.Name
```

☐ Profiling [Edit inline] [Edit] [E
1
>
>>
☐ Show all
Number of rows: 25
Filter rows: Search this table
+ Options

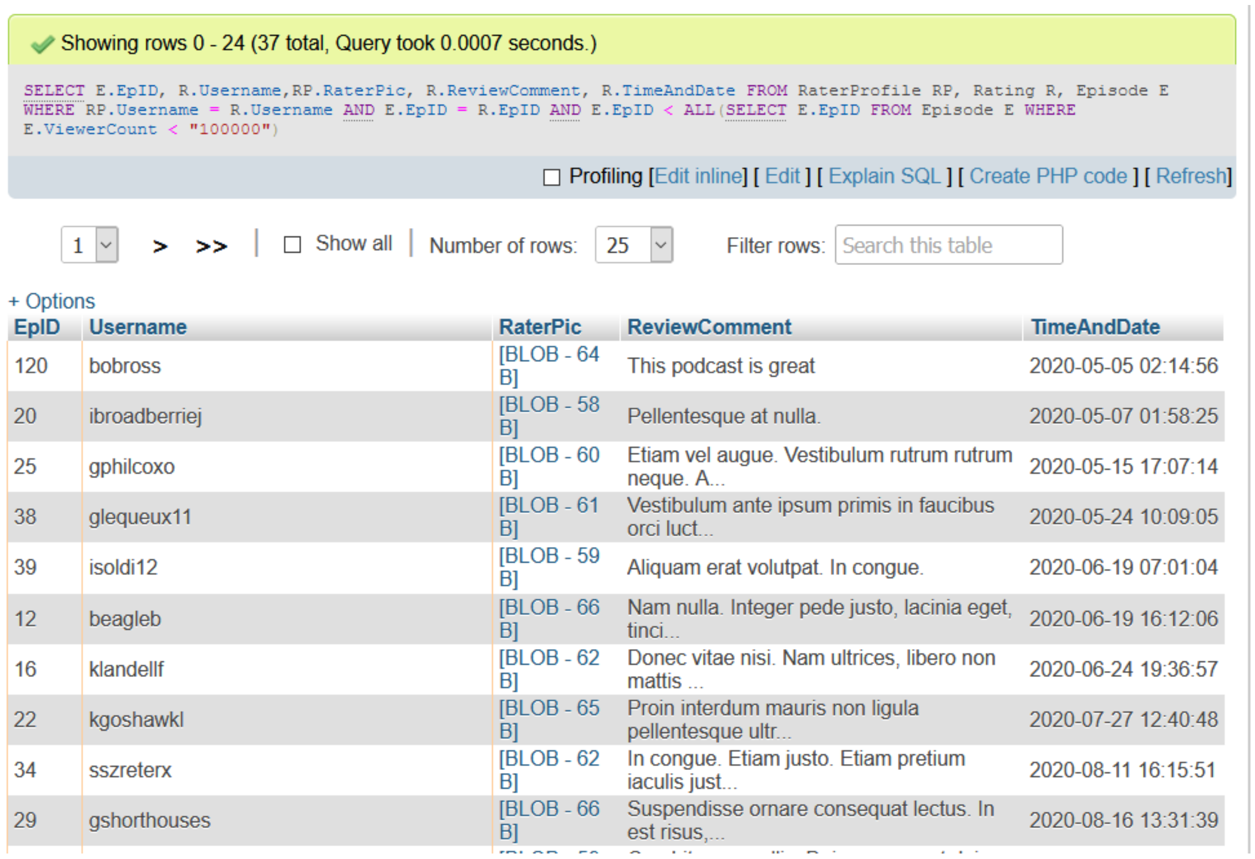
Name	COUNT(E.EpisodeNumber)
Domainer	15
Subin	20
Zoolab	10
Pannier	5
Sonair	15
Veribet	10
Matsoft	5
Ronstring	10
Bitchip	9
Viva	10
Sonsing	14

*The final output was truncated (it shows the first 11 rows out of 54 total)

Query 4)
SELECT E.EpID, R.Username,RP.RaterPic, R.ReviewComment, R.TimeAndDate
FROM RaterProfile RP, Rating R, Episode E
WHERE RP.Username = R.Username AND E.EpID = R.EpID AND E.EpID < ALL(SELECT E.EpID FROM Episode E WHERE E.ViewerCount < "100000")

Description: The query in this example looks for the Episode ID, Username, the Usernames profile picture, User comment, and the TimeAndDate of the user comment. By joining the three relations Episode, RaterProfile, and Rating we are then able to gather the information that is being sought. In addition, a subquery is in the works that looks through all Episode IDs that are less than ALL the selected EpisodeIDs whose ViewerCount is less than “100000”. This query accurately illustrates the real world application that this podcast database offers, as the query seeks to understand User input from Podcast Episodes that may not be having much exposure.

Query Results:



✓ Showing rows 0 - 24 (37 total, Query took 0.0007 seconds.)

```
SELECT E.EpID, R.Username,RP.RaterPic, R.ReviewComment, R.TimeAndDate FROM RaterProfile RP, Rating R, Episode E
WHERE RP.Username = R.Username AND E.EpID = R.EpID AND E.EpID < ALL(SELECT E.EpID FROM Episode E WHERE
E.ViewerCount < "100000")
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
1 > >> | ☐ Show all | Number of rows: 25 Filter rows: Search this table
+ Options

EpID	Username	RaterPic	ReviewComment	TimeAndDate
120	bobross	[BLOB - 64 B]	This podcast is great	2020-05-05 02:14:56
20	ibroadberriej	[BLOB - 58 B]	Pellentesque at nulla.	2020-05-07 01:58:25
25	gphilcoxo	[BLOB - 60 B]	Etiam vel augue. Vestibulum rutrum rutrum neque. A...	2020-05-15 17:07:14
38	glequeux11	[BLOB - 61 B]	Vestibulum ante ipsum primis in faucibus orci luct...	2020-05-24 10:09:05
39	isoldi12	[BLOB - 59 B]	Aliquam erat volutpat. In congue.	2020-06-19 07:01:04
12	beagleb	[BLOB - 66 B]	Nam nulla. Integer pede justo, lacinia eget, tinci...	2020-06-19 16:12:06
16	klandellf	[BLOB - 62 B]	Donec vitae nisi. Nam ultrices, libero non mattis ...	2020-06-24 19:36:57
22	kgoshawkl	[BLOB - 65 B]	Proin interdum mauris non ligula pellentesque ultr...	2020-07-27 12:40:48
34	sszreterx	[BLOB - 62 B]	In congue. Etiam justo. Etiam pretium iaculis just...	2020-08-11 16:15:51
29	gshorthouses	[BLOB - 66 B]	Suspendisse ornare consequat lectus. In est risus,...	2020-08-16 13:31:39

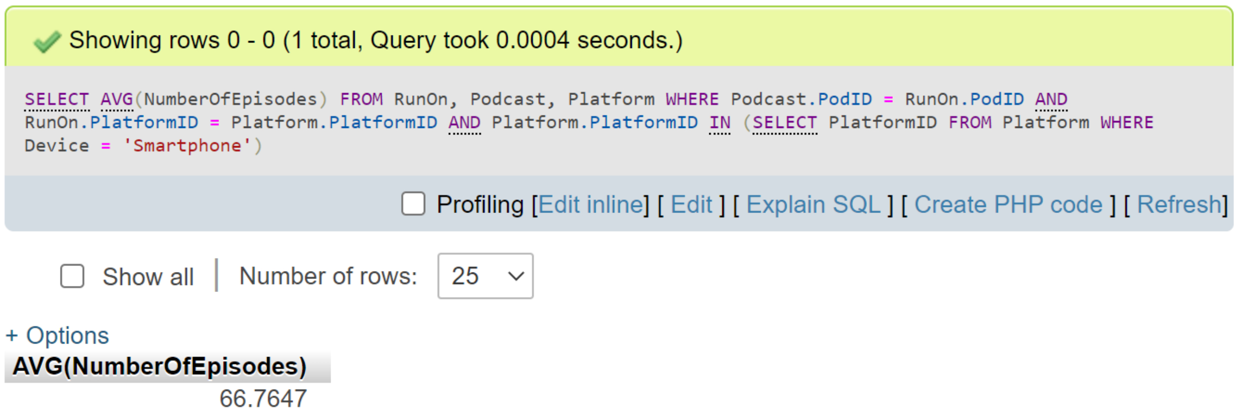
*The final output was truncated (it shows the first 10 rows out of 37 total)

Query 5)

```
SELECT AVG(NumberOfEpisodes)
FROM RunOn, Podcast, Platform
WHERE Podcast.PodID = RunOn.PodID AND RunOn.PlatformID =
Platform.PlatformID AND Platform.PlatformID
IN (SELECT PlatformID
    FROM Platform
    WHERE Device = 'Smartphone')
```

Description: This query calculates the average number of episodes in podcasts that can be listened to on smartphones. The query first selects the PlatformIDs of platforms that are specifically on smartphone devices. It then uses the PlatformIDs to link to the Podcasts table where the query selects podcasts that were just collected from the platforms. The NumberOfEpisodes values are selected from these and are then averaged. The result was that podcasts available on smartphones tend to have around 67 episodes.

Query Results:



Showing rows 0 - 0 (1 total, Query took 0.0004 seconds.)

```
SELECT AVG(NumberOfEpisodes) FROM RunOn, Podcast, Platform WHERE Podcast.PodID = RunOn.PodID AND
RunOn.PlatformID = Platform.PlatformID AND Platform.PlatformID IN (SELECT PlatformID FROM Platform WHERE
Device = 'Smartphone')
```

```
AVG(NumberOfEpisodes)
              66.7647
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows: 25
+ Options

Section 3 Contribution:

Victor Cardenas (email: vcardenas4@csustan.edu); Help with creating queries that worked in the database.

Nicholas Guzman (email: nguzman9@csustan.edu); Helped create queries for the database.

Kristijan Hornung (email: khornung@csustan.edu); Provided queries 1) and 2) together with descriptions and query result screenshots. Provided one example of a query that did not work correctly.