# Ryan Timbrook

## IST 659 Data Admin Concepts &Db Mgmt

## Date: 8/27/18

## Lab Assignment: Lab 7, Advanced Querying

## Description / Learning Objective

* Demonstrate data manipulation language (DML) proficiency
* Perform basic data analysis using descriptive statistics provided by SQL aggregate functions

## Responses

### Part 1, Exploratory Data Analysis

Our database is alive and it’s time to start mining the data for some insights. We will use SQL SELECT statements to get some descriptive statistics from our data.

## Questions our data should answer

## Basic summaries should answer:

How the users are using the system

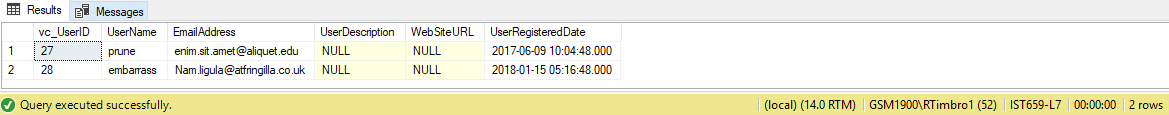
* How many videos each user has made
* Who our least prolific users are, which users have less than 10 vidcasts in the database

## Advanced summaries should provide:

Descriptive statistics on the duration of finished VidCasts

* Number of minutes between StartDateTime and EndDateTime for all VidCasts with a Finished status

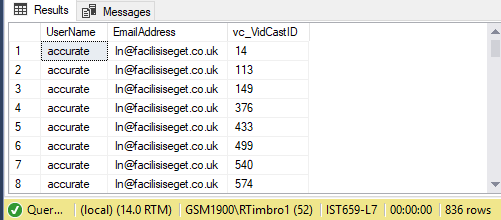
P1-TODO-1: *Look for users who have not yet made any VidCasts*



#### SQL Query

|  |
| --- |
| -- Look for users who have not yet made any VidCasts  SELECT \*  FROM vc\_User  WHERE vc\_UserID NOT IN(SELECT vc\_UserID FROM vc\_VidCast) |

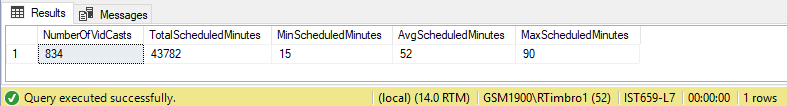
P1-TODO-2: *Need to tell SQL Server to include all users, even if they have no VidCasts in the database*



#### SQL Query

|  |
| --- |
| -- Be sure to include all vc\_User records  SELECT vc\_User.UserName, vc\_User.EmailAddress, vc\_VidCast.vc\_VidCastID  FROM vc\_VidCast  RIGHT JOIN vc\_User ON vc\_User.vc\_UserID = vc\_VidCast.vc\_UserID  ORDER BY vc\_User.UserName |

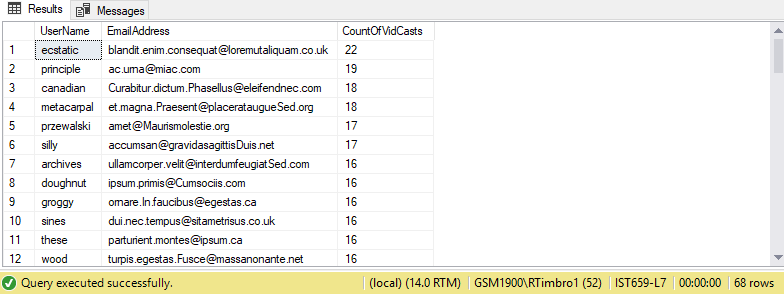
P1-TODO-3: *High-level descriptive statistics for vc\_VidCast*



#### SQL Query

|  |
| --- |
| SELECT  COUNT(vc\_VidCastID) AS NumberOfVidCasts,  SUM(ScheduleDurationMinutes) AS TotalScheduledMinutes,  MIN(ScheduleDurationMinutes) AS MinScheduledMinutes,  AVG(ScheduleDurationMinutes) AS AvgScheduledMinutes,  MAX(ScheduleDurationMinutes) AS MaxScheduledMinutes  FROM vc\_VidCast |

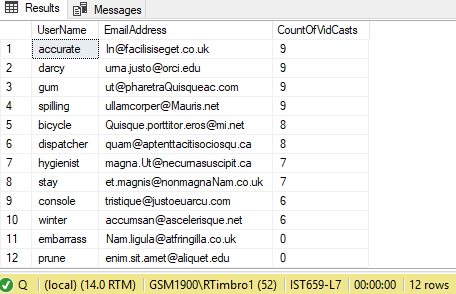
#### P1-TODO-4: Amend query with error due to unaggregated columns in the SELECT list along with a column that has been aggregated.



#### SQL Query

|  |
| --- |
| -- Amend prior query to include GROUP BY clause  SELECT  vc\_User.UserName,  vc\_User.EmailAddress,  COUNT(vc\_VidCast.vc\_VidCastID) AS CountOfVidCasts  FROM  vc\_VidCast  RIGHT JOIN  vc\_User ON vc\_User.vc\_UserID = vc\_VidCast.vc\_UserID  GROUP BY  vc\_User.UserName,  vc\_User.EmailAddress  ORDER BY  CountOfVidCasts DESC,  vc\_User.UserName |

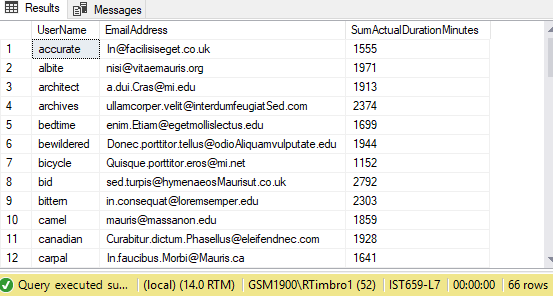
#### P1-TODO-5: Filter our result set by the result of one or more aggregate functions. Answer the question, who our least prolific users are. Which users have less than 10 vidcasts in the database.



#### SQL Query

|  |
| --- |
| -- HAVING Clause, Amend prior query to filter result set by the result of one or more aggregate functions  SELECT  vc\_User.UserName,  vc\_User.EmailAddress,  COUNT(vc\_VidCast.vc\_VidCastID) AS CountOfVidCasts  FROM  vc\_VidCast  RIGHT JOIN  vc\_User ON vc\_User.vc\_UserID = vc\_VidCast.vc\_UserID  GROUP BY  vc\_User.UserName,  vc\_User.EmailAddress  HAVING COUNT(vc\_VidCast.vc\_VidCastID) < 10  ORDER BY  CountOfVidCasts DESC,  vc\_User.UserName |

#### P1-TODO-6: Perform descriptive statistics on the actual duration of finished VidCasts. Calculate the number of minutes between StartDateTime and EndDateTime for all VidCasts with a Finished status.



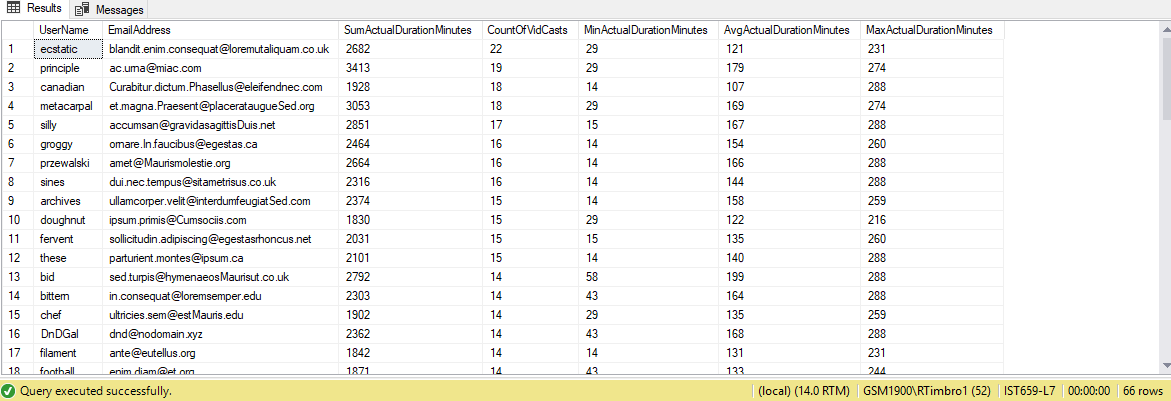
#### SQL Query

|  |
| --- |
| /\*  Advanced Summaries: Descriptive statistics on the duration of finished VidCasts  -Number of minutes between StartDateTime and EndDateTime for all VidCasts with a Finished status  -NOTE: Since we're only interested in VidCasts that are Finished, we don't need vc\_User records with no VidCasts, we do not  need a LEFT or RIGHT JOIN in the FROM clause  \*/  SELECT  vc\_User.UserName,  vc\_User.EmailAddress,  SUM(DateDiff(n,StartDateTime,EndDateTime)) AS SumActualDurationMinutes  FROM  vc\_VidCast  JOIN  vc\_User ON vc\_User.vc\_UserID = vc\_VidCast.vc\_UserID  JOIN  vc\_Status ON vc\_Status.vc\_StatusID = vc\_VidCast.vc\_StatusID  WHERE  vc\_Status.StatusText = 'Finished'  GROUP BY  vc\_User.UserName,  vc\_User.EmailAddress  ORDER BY  vc\_User.UserName |

## Part 2, Putting it All Together

In this part, you’ll amend the previous query to show some more descriptive statistics for the VidCast actual duration.

P2-TODO-1: *Amend the query from the end of part one, adding the count of VidCasts, minimum, average, and maximum actual durations for each vc\_User record. Sort the results in descending order by the count of videos, then by the UserName.*



#### SQL Query

|  |
| --- |
| /\*  Part 2, Putting it All Together  -Amend prior queries to show more descriptive statistics for the VidCast actual duration  TODO: Amend the query from the end of part one, adding the count of VidCasts, minimum, average, and maximum actual durations for each vc\_User record.  Sort the results in descending order by the count of videos, then by the UserName.  \*/  SELECT  vc\_User.UserName,  vc\_User.EmailAddress,  SUM(DateDiff(n,StartDateTime,EndDateTime)) AS SumActualDurationMinutes,  COUNT(vc\_VidCast.vc\_VidCastID) AS CountOfVidCasts,  MIN(DateDiff(n,StartDateTime,EndDateTime)) AS MinActualDurationMinutes,  AVG(DateDiff(n,StartDateTime,EndDateTime)) AS AvgActualDurationMinutes,  MAX(DateDiff(n,StartDateTime,EndDateTime)) AS MaxActualDurationMinutes  FROM  vc\_VidCast  JOIN  vc\_User ON vc\_User.vc\_UserID = vc\_VidCast.vc\_UserID  JOIN  vc\_Status ON vc\_Status.vc\_StatusID = vc\_VidCast.vc\_StatusID  WHERE  vc\_Status.StatusText = 'Finished'  GROUP BY  vc\_User.UserName,  vc\_User.EmailAddress  ORDER BY  CountOfVidCasts DESC,  vc\_User.UserName |

#### SQL Query File

|  |
| --- |
| /\*  Author: Ryan Timbrook  Course: IST659 Data Admin Concepts & Db Mgmt  Term: Summer 2018  Lab: 7, Advanced Querying  \*/  /\*  Part 1, Exploratory Data Analysis  \*/  -- Validate the database is setup correctly by running the below SELECT statement  SELECT vc\_User.UserName, vc\_User.EmailAddress, vc\_VidCast.vc\_VidCastID  FROM vc\_VidCast  JOIN vc\_User ON vc\_User.vc\_UserID = vc\_VidCast.vc\_UserID  ORDER BY vc\_User.UserName  -- Look for users who have not yet made any VidCasts  SELECT \*  FROM vc\_User  WHERE vc\_UserID NOT IN(SELECT vc\_UserID FROM vc\_VidCast)  -- Be sure to include all vc\_User records  SELECT vc\_User.UserName, vc\_User.EmailAddress, vc\_VidCast.vc\_VidCastID  FROM vc\_VidCast  RIGHT JOIN vc\_User ON vc\_User.vc\_UserID = vc\_VidCast.vc\_UserID  ORDER BY vc\_User.UserName  -- High-level descriptive statistics for vc\_VidCast  SELECT  COUNT(vc\_VidCastID) AS NumberOfVidCasts,  SUM(ScheduleDurationMinutes) AS TotalScheduledMinutes,  MIN(ScheduleDurationMinutes) AS MinScheduledMinutes,  AVG(ScheduleDurationMinutes) AS AvgScheduledMinutes,  MAX(ScheduleDurationMinutes) AS MaxScheduledMinutes  FROM vc\_VidCast  -- Amending prior SELECT statement to include GROUP BY clause  /\*  Expect to get this ERROR message when running the below statement. This is intentional to highlight how we  added unaggregated columns in the SELECT list along with a column that has been aggregated.  Msg 8127, Level 16, State 1, Line 43  Column "vc\_User.UserName" is invalid in the ORDER BY clause because it is not contained in either an aggregate function or the GROUP BY clause.  \*/  SELECT  COUNT(vc\_VidCastID) AS NumberOfVidCasts,  SUM(ScheduleDurationMinutes) AS TotalScheduledMinutes,  MIN(ScheduleDurationMinutes) AS MinScheduledMinutes,  AVG(ScheduleDurationMinutes) AS AvgScheduledMinutes,  MAX(ScheduleDurationMinutes) AS MaxScheduledMinutes  FROM vc\_VidCast  RIGHT JOIN vc\_User ON vc\_User.vc\_UserID = vc\_VidCast.vc\_UserID  ORDER BY vc\_User.UserName  -- Amend prior query to include GROUP BY clause  SELECT  vc\_User.UserName,  vc\_User.EmailAddress,  COUNT(vc\_VidCast.vc\_VidCastID) AS CountOfVidCasts  FROM  vc\_VidCast  RIGHT JOIN  vc\_User ON vc\_User.vc\_UserID = vc\_VidCast.vc\_UserID  GROUP BY  vc\_User.UserName,  vc\_User.EmailAddress  ORDER BY  CountOfVidCasts DESC,  vc\_User.UserName  -- HAVING Clause, Amend prior query to filter result set by the result of one or more aggregate functions  SELECT  vc\_User.UserName,  vc\_User.EmailAddress,  COUNT(vc\_VidCast.vc\_VidCastID) AS CountOfVidCasts  FROM  vc\_VidCast  RIGHT JOIN  vc\_User ON vc\_User.vc\_UserID = vc\_VidCast.vc\_UserID  GROUP BY  vc\_User.UserName,  vc\_User.EmailAddress  HAVING COUNT(vc\_VidCast.vc\_VidCastID) < 10  ORDER BY  CountOfVidCasts DESC,  vc\_User.UserName  /\*  Advanced Summaries: Descriptive statistics on the duration of finished VidCasts  -Number of minutes between StartDateTime and EndDateTime for all VidCasts with a Finished status  -NOTE: Since we're only interested in VidCasts that are Finished, we don't need vc\_User records with no VidCasts, we do not  need a LEFT or RIGHT JOIN in the FROM clause  \*/  SELECT  vc\_User.UserName,  vc\_User.EmailAddress,  SUM(DateDiff(n,StartDateTime,EndDateTime)) AS SumActualDurationMinutes  FROM  vc\_VidCast  JOIN  vc\_User ON vc\_User.vc\_UserID = vc\_VidCast.vc\_UserID  JOIN  vc\_Status ON vc\_Status.vc\_StatusID = vc\_VidCast.vc\_StatusID  WHERE  vc\_Status.StatusText = 'Finished'  GROUP BY  vc\_User.UserName,  vc\_User.EmailAddress  ORDER BY  vc\_User.UserName  /\*  Part 2, Putting it All Together  -Amend prior queries to show more descriptive statistics for the VidCast actual duration  TODO: Amend the query from the end of part one, adding the count of VidCasts, minimum, average, and maximum actual durations for each vc\_User record.  Sort the results in descending order by the count of videos, then by the UserName.  \*/  SELECT  vc\_User.UserName,  vc\_User.EmailAddress,  SUM(DateDiff(n,StartDateTime,EndDateTime)) AS SumActualDurationMinutes,  COUNT(vc\_VidCast.vc\_VidCastID) AS CountOfVidCasts,  MIN(DateDiff(n,StartDateTime,EndDateTime)) AS MinActualDurationMinutes,  AVG(DateDiff(n,StartDateTime,EndDateTime)) AS AvgActualDurationMinutes,  MAX(DateDiff(n,StartDateTime,EndDateTime)) AS MaxActualDurationMinutes  FROM  vc\_VidCast  JOIN  vc\_User ON vc\_User.vc\_UserID = vc\_VidCast.vc\_UserID  JOIN  vc\_Status ON vc\_Status.vc\_StatusID = vc\_VidCast.vc\_StatusID  WHERE  vc\_Status.StatusText = 'Finished'  GROUP BY  vc\_User.UserName,  vc\_User.EmailAddress  ORDER BY  CountOfVidCasts DESC,  vc\_User.UserName |