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

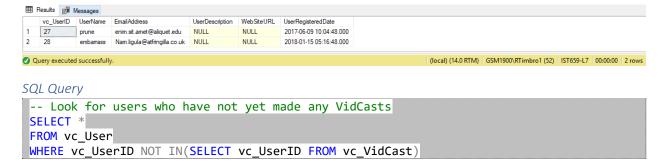
- How many videos each user has made
- Who our least prolific users are, which users have less than 10 vidcasts in the database

Advanced summaries should provide:

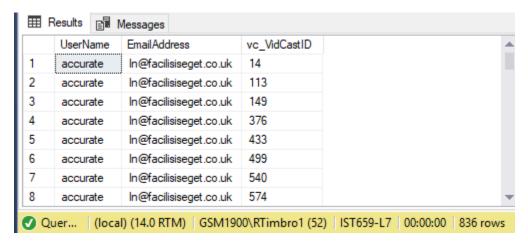
Descriptive statistics on the duration of finished VidCasts

• 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



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

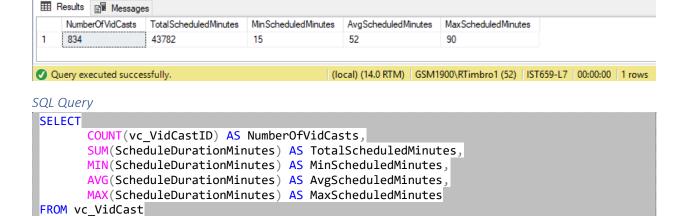
SELECT vc_User.UserName, vc_User.EmailAddress, vc_VidCast.vc_VidCastID

FROM vc_VidCast

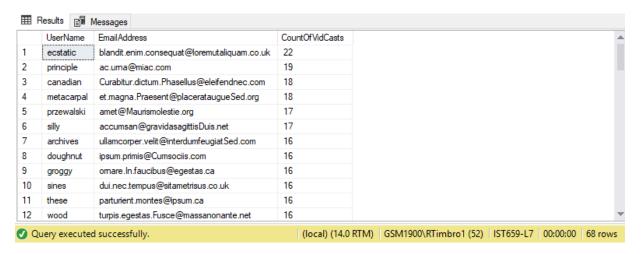
RIGHT JOIN vc_User ON vc_User.vc_UserID = vc_VidCast.vc_UserID

ORDER BY vc_User.UserName
```

P1-TODO-3: High-level descriptive statistics for vc_VidCast



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

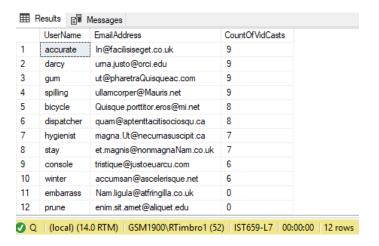
GROUP BY

vc_User.UserName,
vc_User.EmailAddress

ORDER BY

CountOfVidCasts DESC,
vc_User.UserName
```

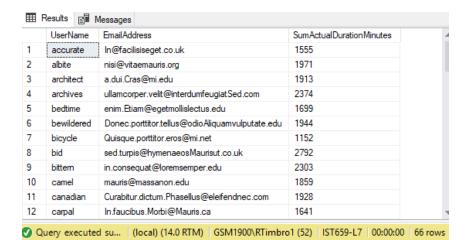
P1-TODO-5: Filter our result set by the result of one or more aggregate functions. Answer the question, who our least prolific users are. Which users have less than 10 vidcasts in the database.



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</pre>
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.

	UserName	EmailAddress	SumActual Duration Minutes	CountOfVidCasts	MinActual Duration Minutes	AvgActual Duration Minutes	MaxActualDurationMinutes	
1	ecstatic	blandit.enim.consequat@loremutaliquam.co.uk	2682	22	29	121	231	
2	principle	ac.uma@miac.com	3413	19	29	179	274	
3	canadian	Curabitur.dictum.Phasellus@eleifendnec.com	1928	18	14	107	288	
4	metacarpal	et.magna.Praesent@placerataugueSed.org	3053	18	29	169	274	
5	silly	accumsan@gravidasagittisDuis.net	2851	17	15	167	288	
6	groggy	omare.ln.faucibus@egestas.ca	2464	16	14	154	260	
7	przewalski	amet@Maurismolestie.org	2664	16	14	166	288	
8	sines	dui.nec.tempus@sitametrisus.co.uk	2316	16	14	144	288	
9	archives	ullamcorper.velit@interdumfeugiatSed.com	2374	15	14	158	259	
10	doughnut	ipsum.primis@Cumsociis.com	1830	15	29	122	216	
11	fervent	sollicitudin.adipiscing@egestasrhoncus.net	2031	15	15	135	260	
12	these	parturient.montes@ipsum.ca	2101	15	14	140	288	
13	bid	sed.turpis@hymenaeosMaurisut.co.uk	2792	14	58	199	288	
14	bittem	in.consequat@loremsemper.edu	2303	14	43	164	288	
15	chef	ultricies.sem@estMauris.edu	1902	14	29	135	259	
16	DnDGal	dnd@nodomain.xyz	2362	14	43	168	288	
17	filament	ante@eutellus.org	1842	14	14	131	231	
18	football	enim diam@et om	1871	14	∆ 3	133	244	

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
```

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</pre>
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,
       {\color{red} SUM(DateDiff(n,StartDateTime,EndDateTime))} \  \  {\color{red} 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
```