

Assignment 2 – SQL

CSCI 585 20172

In this assignment, we will use Google Cloud SQL to work with SQL queries. This will help us learn how to use cloud services as well as run code on SQL. The document is divided into several parts. Parts 1 through 3 go through the basics of setting up the platform. Part 4 is the assignment. Parts 5 and 6 are optional resources that you can use. Good luck.

Contents

Part I: Setting up Google Cloud Platform.....	2
Part 2: Setting up Cloud SQL	3
Create a Cloud SQL Instance	4
Part 3: Working with SQL (Optional).....	7
Part 4: Programming Assignment	8
Submission Guidelines	8
Set 1: GoodReads database	9
Set 2: GitHub Database	10
Set 3: Ad-targeting on YouTube using LinkedIn.....	11
Part 5: Starter Code (Optional)	12
Set 1: GoodReads database	12
Set 2: GitHub database	12
Part 6: Sample Tables (Optional)	13
Set 1: GoodReads database	13
Set 2: GitHub database	14

Part I: Setting up Google Cloud Platform

Google Cloud Platform helps you to run your work on Google Compute Engine (GCE) and to use its core infrastructure, data analytics and machine learning.

To set up GCP, follow the steps below.

1. Go to <https://console.cloud.google.com/freetrial> using your **personal** Google account. (Do not use the USC account.)
This gives you a \$300 credit to spend on the GCP for the next 12 months. Agree to the acceptances and click on 'Agree and Continue', as in Figure 1.1.
2. In the Customer info screen, select **Account Type: Individual** and fill in all the details. (You will be required to enter credit/debit card details, but you will not be charged as long as you don't exceed the \$300, which is sufficient for all the database assignments.)
3. Click on 'Start my Free Trial' and wait for Google Cloud Platform to set up.

Congratulations! You just finished the first part of the assignment.

Google Cloud Platform

Try Cloud Platform for free Google

Country

United States

Acceptances

Please email me updates regarding feature announcements, performance suggestions, feedback surveys and special offers.

☐ Yes ☒ No

I agree that my use of any services and related APIs is subject to my compliance with the applicable Terms of Service. I have also read and agree to the Google Cloud Platform Free Trial Terms of Service.

Required to continue

☐ Yes ☒ No

Agree and continue

Privacy policy

Access to all Cloud Platform Products
Get everything you need to build and run your apps, websites, and services, including Firebase and the Google Maps API.

\$300 credit for free
Sign up and get \$300 to spend on Google Cloud Platform over the next 12 months.

No autocharge after free trial ends
We ask you for your credit card to make sure you are not a robot. You won't be charged unless you manually upgrade to a paid account.

Figure 1.1: Google Cloud Platform Sign-up page

Part 2: Setting up Cloud SQL

Cloud SQL is a part of the GCE to run PostgreSQL and MySQL scripts.


Go to <https://cloud.google.com/sql/>.

If you prefer to use MySQL for this assignment, you can find the Quick Start guide at: <https://cloud.google.com/sql/docs/mysql/quickstart>.

If you prefer to use PostgreSQL instead, visit <https://cloud.google.com/sql/docs/postgres/quickstart>.

The pages are self-explanatory, and in case you do not face a problem, setting it up, feel free to skip the rest of Part 2. Below are detailed steps from the same page.

Before you begin

1. Select or create a Cloud platform project.
Go to: <https://console.cloud.google.com/start>. At the top, click on 'Select a project', and click on the  sign.
2. In the next screen, (as in Figure 2.1), enter a project name.
3. Enable billing for your project.
In the navigation drawer on the left, click on Billing and add a billing account. Follow the steps and make a billing profile. The details would already be pre-filled as you had entered card details before. Click on 'Submit and enable billing'.
4. Enable the Cloud SQL Administration API. (Select the project that you made earlier in the screen like Figure 2.2.) Wait for API to be enabled and then click 'Continue'. You will be redirected to the dashboard

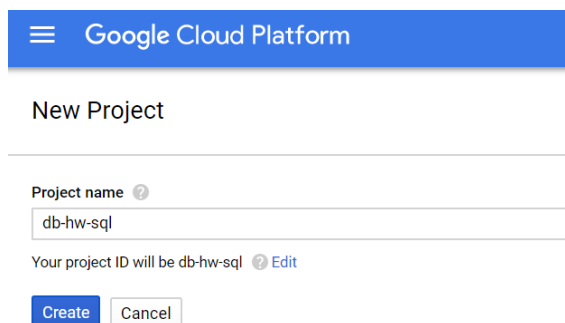


Figure 2.1: New Project Screen

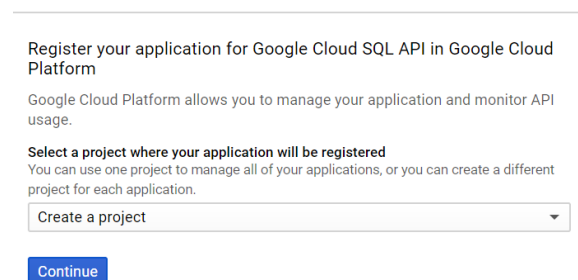


Figure 2.2: Enable the Cloud API.

Create a Cloud SQL Instance

1. Click on the 'Go to the Cloud SQL instances page' button on the Quick-start page. You will get a screen like Fig. 2.3 (a). Click on 'Select', select the project and then click 'Open'. (Fig. 2.3 (b)).
2. Click on 'Create Instance' in the cloud Instances page. (Fig. 2.4).
3. Select one of MySQL or PostgreSQL and click 'Next'. Note that PostgreSQL is in beta and might undergo changes which will not be backward compatible.

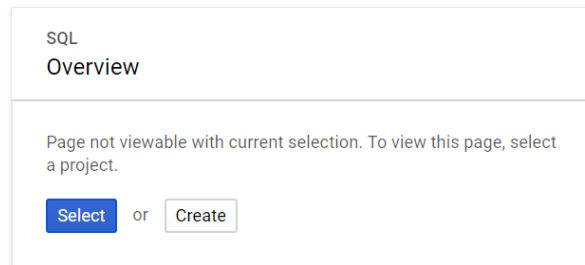


Figure 2.3 (a): SQL Overview Page.

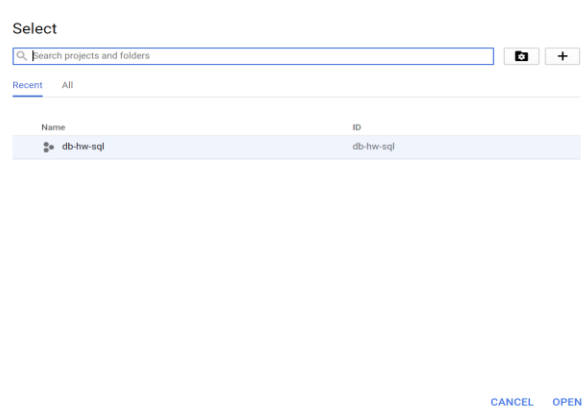


Figure 2.3 (b): Select the project

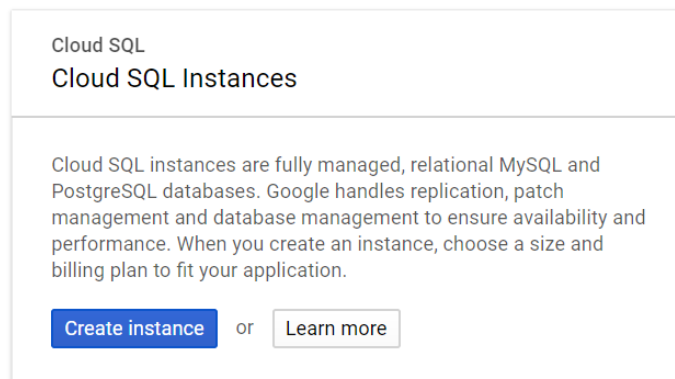


Fig 2.4: Create Instance

There are two types of Cloud SQL MySQL instances. [Learn more](#)

MySQL Second Generation (Recommended)

High performance, high storage capacity, low cost.

- Up to 7X throughput and 20X storage capacity of First Generation
- Less expensive than First Generation for most use cases
- Option to add High Availability failover and read replication
- Configurable backup period and maintenance window
- Supports only MySQL 5.6 and 5.7

Choose Second Generation

MySQL First Generation (Legacy)

Older version of Cloud SQL providing basic performance and storage capacity. Does not support MySQL 5.7.

Choose First Generation



Figure 2.5: MySQL Second Generation page

The next steps are explained with MySQL.

4. Click on 'Choose second generation' in case you get the next screen as Figure 2.5.
5. In the Instance details page, provide an Instance ID name and a root password. Leave the rest as they are. Update this:

Instance ID
Cannot be changed later. Use lowercase letters, numbers, and hyphens. Start with a letter.

Click on 'Create'. You will see 'Instance is being created'. Wait until the left most wheel turns into a green tick.

Instance ID [?]	Type	IP address	Failover	Storage used [?]	Location
 sql-db-1	MySQL Second Generation		—	—	us-central1
Instance ID [?]	Type	IP address	Failover	Storage used [?]	Location
 sql-db-1	MySQL Second Generation		Add Failover	—	us-central1

Note: On the right-hand side, the three-dot menu has a "Delete" option. Be sure to delete this instance once you are done with the homework to avoid extra charges on the instance.

- Click on the instance ID name to open the 'Instance details' page, and then click on the console button on the top-right corner.

At the Cloud Shell prompt, connect to your Cloud SQL instance. When the Cloud shell finishes initializing you should see:

```
Welcome to Cloud Shell! Type "help" to get started.  
anindya_dutta22@db-hw-sql:~$
```

db-hw-sql/ would be replaced with the name of your project.

- At the Cloud Shell prompt, connect to your cloud SQL instance.

```
gcloud beta sql connect myinstance --user=root
```

Replace **myinstance** with the name of your instance, (in this example, sql-db-1.)

```
Welcome to Cloud Shell! Type "help" to get started.  
anindya_dutta22@db-hw-sql:~$ gcloud beta sql connect sql-db-1 --user=root
```

Enter your password (it is a linux terminal so you won't see it being typed). You should now be able to see the mysql prompt.

```
anindya_dutta22@db-hw-sql:~$ gcloud beta sql connect sql-db-1 --user=root  
Whitelisting your IP for incoming connection for 5 minutes...done.  
Enter password:  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 101  
Server version: 5.7.14-google-log (Google)  
  
Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql>
```

Congratulations! You just finished the second part of the assignment.

Part 3: Working with SQL (Optional)

In this part of the assignment, we will build a database with one table and run queries to see if MySQL works as expected.

1. Create a SQL database on your Cloud SQL instance.

```
CREATE DATABASE test;
```

2. Insert sample data into the guestbook database:

```
USE test;
CREATE TABLE entries (guestName VARCHAR(255), content VARCHAR(255),
  entryID int not null AUTO_INCREMENT, PRIMARY KEY(entryID));
INSERT into entries (guestName, content) values ("first guest", "I got
here!");
INSERT into entries (guestName, content) values ("second guest", "Me
too!");
```

3. Retrieve the data.

```
SELECT * FROM entries;
```

You should see:

```
+-----+-----+-----+
| guestName | content | entryID |
+-----+-----+-----+
| first guest | I got here! | 1 |
| second guest | Me too! | 2 |
+-----+-----+-----+
2 rows in set (0.00 sec)
mysql>
```

Congratulations! You are now ready to solve the assignment.

Part 4: Programming Assignment

This assignment is worth 10 points.

There are three sets of questions. The first two sets are worth 5 points each. Solving two sets would result in successful completion of the assignment, but it is recommended to solve all three sets.

You get a **bonus 1 point** if you submit the third set.

Submission Guidelines:

Once you have run all the queries on the Cloud SQL, you are ready to submit the assignment. To submit the assignment, do the following.

1. For each query, take a screenshot of the output. Submit the query, the output screenshot and an explanation of why the query works the way it does. (The explanation can be a one-liner.)
2. At the beginning of the assignment, mention your student ID, full name, and whether you've used PostgreSQL or MySQL.
3. If you're submitting the bonus question, please submit the DDL commands as well (creation of the table, insertion commands and any other details you would like to mention).
4. Please make sure to mention all assumptions you make.
5. For all questions that have X, Y, or Z mentioned as variables, you can substitute them with hard-coded names. However, the code should be generic and should run for any test case.
6. The name of the submitted PDF file should be studentid_HW2.pdf. For example, a valid name is 1234567890_HW2.pdf.
7. Queries will **not be tested on bad/malformed/invalid** data. While making the table, do not worry about corner cases, null values, or tricky test cases (such as negative entries into columns, invalid strings, etc.) The aim of the homework is to help you learn to write queries. You can safely assume that the table has valid data.
8. If you have questions, please make use of HW 2 discussion forum on DEN site. Please double-check if your question has already been asked. You can ask and reply to others as long as your questions/answers do not contain SQL queries or textual explanations of SQL queries.
9. If you need help or have any questions, please visit TAs or graders in the office hours. **Please do not post SQL queries on the discussion forum.**
10. You can submit multiple times. The most recent submission will be graded.
11. The deadline is **Monday, June 12, 2017 11:59 PM**. No submissions will be accepted past the deadline.

Set 1: GoodReads database

GoodReads is a website that allows users to add books that they have read, review them and socialize with other people. They have hired you as their new database admin and provided you with access to their tables. They have some questions for you. Formulate queries for each of the questions.

List of tables:

Book: ISBN, Title, AuthorID, NumPages, AverageRating.

Users: UID, Name, Age, Sex, Location, Birthday, ReadCount, CurrentlyReadingCount, ToReadCount

Shelf: UID, ISBN, Rating, ShelfName, DateRead, DateAdded

Friends: UID, FriendID

Author: AuthorID, Name.

Questions:

1. User adds a new book to his shelf with a rating. Update the average rating of that book.
 2. Find the names of the common books that were read by any two users X and Y.
 3. For all books that user X has in his *to-read* list, find the number of friends that have already read each book and list them by book names.
 4. Find the names of all books where user X and any of his/her friends X gave the book a rating of 5 stars. (e.g., if X and Y gave 5 stars to book 1, and X and Z gave 5 stars to book 2, both books 1 and 2 should be included in the output.)
 5. Find users who have only read books that have more than 300 pages.
-

Set 2: GitHub Database

Your start-up wants to make a Version Control System, and you are taking inspiration from GitHub to organize your work. You have identified some tables, and now you want to test your VCS by running a few queries.

List of tables:

Repository: RepoID, UserID, IssueCount, PullRequestCount, ProjectCount, Wiki (yes/no)

Issues: IssueID, CreatorID, RaiseDate, ResolverID, ResolveDate.

Code: RepoID, CommitCount, BranchCount, ReleaseCount, ContributorCount

Commit: CommitID, BranchID, Timestamp, NumFiles, Additions, Deletions

Branches: UserID, BranchID, RepoID

User: UserID, NumRepos, Location, Email, Website, ContributionCount

Questions:

1. Find the commits made by user X in all branches of a repository between 1st May and 10th May.
 2. Display the top ten users who made the most contributions.
 3. Find the users who made branches of either of repositories X or Y but not of a repository Z.
 4. Find the top commit with the highest lines of code reduced. (Hint: We need to find the maximized value of: number of deletions - number of additions in each commit).
 5. List the users who solved more issues than they raised. (i.e. number of issues in which they were the resolver is greater than the number of issues where they were the creator.)
-

Set 3: Ad-targeting on YouTube using LinkedIn

You want to analyze what kinds of jobs users see on LinkedIn to show them better job advertisements on YouTube. You have been given a list of some of the tables from both LinkedIn and YouTube databases that the admin team thinks might be useful to carry out this task.

Assumption: The same e-mail ID is used to log in to YouTube and LinkedIn accounts.

LinkedIn database snippet:

List of tables:

User: UserID, Email, Handle, Name, Headline, Location.

Organization: OrganizationID, Name.

Experience: UserID, ExperienceID, Title, OrganizationID, From, To, Location, Description

Jobs: JobID, DatePosted, OrganizationID, Genre.

Application: UserID, JobID

Network: UserID, FriendID, DateAdded, Note.

YouTube database snippet:

List of tables:

User: UserID, Email, Phone, Handle, Name, FavoriteGenre

Subscription: ChannelID, UserID.

Channel: ChannelID, Genre, OwnerID, VideoCount, SubscriberCount.

Video: VideoID, ChannelID.

Questions:

1. List all channels that have more than 50 videos and have the same genres as jobs searched on LinkedIn by user X.
 2. Delete all channels that have less than 50 subscribers but move their videos to the channel with highest number of subscribers in the same genre.
 3. Find all jobs on LinkedIn whose genre matches Owner-ID X's channel's genre.
 4. Find friends of friends of User X on LinkedIn who have subscribed to channels with the genre that matches user X's favorite genre on YouTube. (This can be used to suggest friends.)
 5. Add a column "location" to YouTube's User table based on the location associated with their LinkedIn account.
-

Part 5: Starter Code (Optional)

This starter code is to be used only as a reference to remove ambiguities in the problem statement. It contains the setup code for the first two databases. However, feel free to write this code on your own if you want to practice setting up databases as well. The starter code for **Set 3** will not be provided. The aim is to let you design and develop the entire database from scratch.

Set 1: GoodReads database

```
mysql> create database goodreads;
mysql> use goodreads;
mysql> create table author(authorId int, name varchar(20) not null, primary key
    (authorId));
mysql> create table book (isbn varchar(255), title varchar(20), authorId int,
    numpages int not null, avgrating decimal(3,2), primary key(isbn),
    constraint fk1 foreign key(authorId) references author(authorId));
mysql> create table users(uid int, name varchar(20), age int, sex char(1),
    location varchar(20), birthday date, readCt int, toReadCt int,
    currentlyReadCt int, primary key(uid));
mysql> create table shelf(uid int, isbn varchar(255), name varchar(20), rating
    decimal(3,2), dateRead date, dateAdded date, primary key(uid,
    isbn), constraint fk2 foreign key(uid) references users(uid),
    constraint fk3 foreign key(isbn) references book(isbn));
mysql> create table friends(uid int, fid int, primary key(uid, fid),
    constraint fk4 foreign key(uid) references users(uid),
    constraint fk5 foreign key(fid) references users(uid));
```

Set 2: GitHub database

```
mysql> create database github;
mysql> use github;
mysql> create table users (userId int, noOfRepos int, location varchar(50),
    email varchar(50), website varchar(50), contributions int,
    primary key(userId));
mysql> create table repository (repoId int, userId int not null,
    issueCount int, pullCount int, projectsCount int, wiki boolean
    primary key (repoId), constraint fk1 foreign key(userId)
    references users(userId));
mysql> create table issue (issueId int, creatorId int not null,
    raiseDate date, resolverId int, resolveDate date, primary key (issueId),
    constraint fk2 foreign key(creatorId) references users(userId),
    constraint fk3 foreign key(resolverId) references users(userId));
mysql> create table codes (repoId int, commits int not null, branches int not
    null, releases int, contributors int, primary key(repoId),
    constraint fk4 foreign key(repoId) references repository(repoId));
mysql> create table branch (branchId int, repoId int not null, userId int not
    null, primary key(branchId), constraint fk5 foreign key(repoId)
    references repository(repoId), constraint fk6 foreign key(userId)
    references users(userId));
mysql> create table commits (commitId int, branchId int not null, commitTime
    datetime, noOfFiles int, additions int, deletions int, primary key
    (commitId), constraint fk7 foreign key(branchId) references
    branch(branchId));
```

Part 6: Sample Tables (Optional)

Below are some sample tables that you can refer for testing. However we will be using different tables to test the code, so be sure to check additional test cases.

Set 1: GoodReads database

```
mysql> select * from author;
```

authorId	name
1	Joe Sacco
2	Tolkien
3	George Martin

```
mysql> select * from book;
```

isbn	title	authorId	numpages	avgrating
9730618260320	ASOS	3	150	2.50
9770618260320	ACOK	3	150	3.47
9780618260300	The Hobbit	2	366	3.40
9780618260301	LOTR 1	2	350	3.40
9780618260320	LOTR 2	2	150	2.50
9781560974321	Palestine	1	288	4.20
9880618260320	AGOT	3	150	2.50

```
mysql> select * from users;
```

uid	name	age	sex	location	birthday	readCt	toReadCt	currentlyReadCt
1	User 1	21	M	India	1992-01-14	10	5	1
2	User 2	21	M	USA	1992-02-14	8	10	2
3	User 3	21	M	London	1992-03-14	10	20	5
4	user4	20	m	jampit	2000-01-01	10	3	12

```
mysql> select * from shelf;
```

uid	isbn	name	rating	dateRead	dateAdded
1	9770618260320	Read	5.00	2000-01-01	2000-02-02
1	9780618260301	Read	5.00	2000-01-01	2000-01-01
1	9780618260320	Read	5.00	2000-01-01	2000-02-02
1	9781560974321	Read	5.00	2000-01-01	2000-02-02
2	9770618260320	To-Read	2.70	2000-01-01	2000-02-02
2	9781560974321	To-Read	2.70	2000-01-01	2000-02-02
3	9770618260320	Read	5.00	1999-12-11	2000-01-01
4	9780618260300	Currently-Reading	5.00	2000-01-01	2000-01-01
4	9780618260301	Currently-Reading	5.00	2000-01-01	2000-01-01

```
mysql> select * from friends;
```

uid	fid
2	1
3	1
1	2
1	3

Set 2: GitHub database

```
mysql> select * from users;
+-----+-----+-----+-----+-----+-----+
| userId | noOfRepos | location | email   | website | contributions |
+-----+-----+-----+-----+-----+-----+
| 1      | 15       | jampot   | a@x.com | a.com    | 100           |
| 2      | 20       | jampot   | a@x.edu | b.com    | 240           |
+-----+-----+-----+-----+-----+-----+
```

```
mysql> select * from repository;
+-----+-----+-----+-----+-----+-----+
| repoId | userId | issueCount | pullCount | projectsCount | wiki |
+-----+-----+-----+-----+-----+-----+
| 1      | 1      | 10         | 10        | 10            | 1    |
| 2      | 1      | 10         | 10        | 10            | 1    |
+-----+-----+-----+-----+-----+-----+
```

```
mysql> select * from issue;
+-----+-----+-----+-----+-----+-----+
| issueId | creatorId | raiseDate | resolverId | resolveDate |
+-----+-----+-----+-----+-----+-----+
| 1       | 1         | 2000-01-01 | 2          | 2000-02-02 |
| 2       | 1         | 2000-01-01 | 2          | 2000-02-02 |
| 3       | 2         | 2000-01-01 | 2          | 2000-02-02 |
| 4       | 2         | 2000-01-01 | 1          | 2000-02-02 |
+-----+-----+-----+-----+-----+-----+
```

```
mysql> select * from codes;
+-----+-----+-----+-----+-----+-----+
| repoId | commits | branches | releases | contributors |
+-----+-----+-----+-----+-----+-----+
| 1      | 2       | 1         | 1         | 2            |
| 2      | 2       | 1         | 1         | 2            |
+-----+-----+-----+-----+-----+-----+
```

```
mysql> select * from commits;
+-----+-----+-----+-----+-----+-----+
| commitId | branchId | commitTime | noOfFiles | additions | deletions |
+-----+-----+-----+-----+-----+-----+
| 1        | 1        | 2000-01-01 11:00:00 | 2         | 1000      | 2000      |
| 2        | 1        | 2000-01-01 11:00:00 | 2         | 100       | 20000     |
+-----+-----+-----+-----+-----+-----+
```

Good luck with your assignment! Remember to submit it before **Monday, June 12, 2017 11:59 PM**.