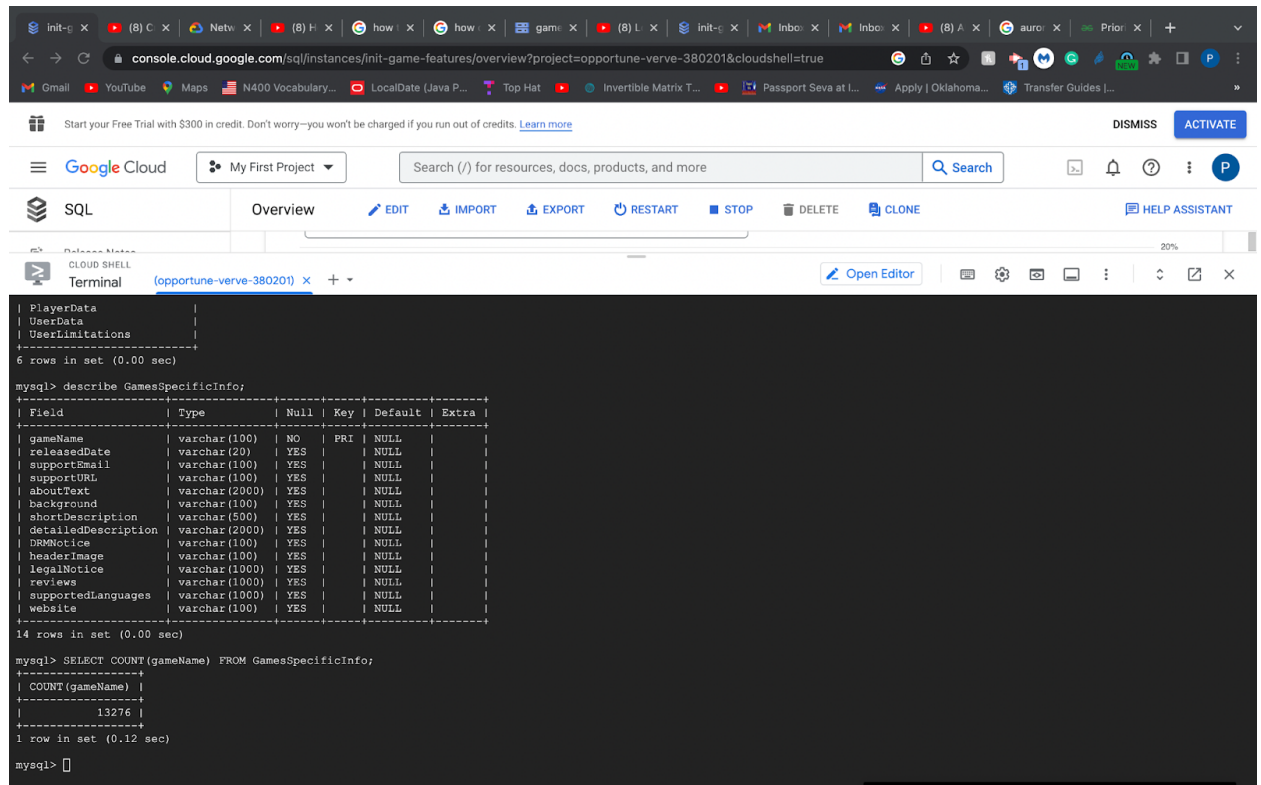


1. Implement at least four main tables (i.e., tables that include core application information, not auxiliary information, such as user profiles and login information). → **DONE**



The screenshot shows the Google Cloud SQL console interface. The top navigation bar includes the Google Cloud logo, a search bar, and various icons. The main content area is divided into two tabs: 'SQL' and 'Overview'. The 'SQL' tab is active, displaying a terminal window with the following content:

```
PlayerData
UserData
UserLimitations
6 rows in set (0.00 sec)

mysql> describe GamesSpecificInfo;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| gameName | varchar(100) | NO | PRI | NULL | |
| releaseDate | varchar(20) | YES | | NULL | |
| supportEmail | varchar(100) | YES | | NULL | |
| supportURL | varchar(100) | YES | | NULL | |
| aboutText | varchar(2000) | YES | | NULL | |
| background | varchar(100) | YES | | NULL | |
| shortDescription | varchar(500) | YES | | NULL | |
| detailedDescription | varchar(2000) | YES | | NULL | |
| DRMNotice | varchar(100) | YES | | NULL | |
| headerImage | varchar(100) | YES | | NULL | |
| legalNotice | varchar(1000) | YES | | NULL | |
| reviews | varchar(1000) | YES | | NULL | |
| supportedLanguages | varchar(1000) | YES | | NULL | |
| website | varchar(100) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
14 rows in set (0.00 sec)

mysql> SELECT COUNT(gameName) FROM GamesSpecificInfo;
+-----+
| COUNT(gameName) |
+-----+
| 13276 |
+-----+
1 row in set (0.12 sec)

mysql>
```

Database created → **DummyGameFeat**

```
Database changed
mysql> show databases;
+-----+
| Database |
+-----+
| DummyGameFeat |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.00 sec)

mysql>
```

Tables in our database DummyGameFeat

```
mysql> use DummyGameFeat;
Database changed
mysql> show tables;
+-----+
| Tables_in_DummyGameFeat |
+-----+
| GameCategories           |
| GamesOwned               |
| GamesSpecificInfo        |
| PlayerData               |
| UserData                 |
| UserLimitations          |
+-----+
6 rows in set (0.01 sec)
```

2. In the Database Design markdown or pdf, provide the Data Definition Language (DDL) commands you all used to create each of these tables in the database. Here's the syntax of the CREATE TABLE DDL command:
CREATE TABLE table_name (column1 datatype, column2 datatype, column3 datatype,...); → **DONE** → **uploaded last time on GitHub but copy pasted on the last page for reference.**
3. Insert data into these tables. You should insert at least 1000 rows **each in** three of the tables. Try to use real data, but if you cannot find a good dataset for a particular table, you may use auto-generated data. → **DONE** → **Andrew Zhao working**

Table GamesSpecificInfo:

```
mysql> describe GamesSpecificInfo;
```

Field	Type	Null	Key	Default	Extra
gameName	varchar(100)	NO	PRI	NULL	
releasedDate	varchar(20)	YES		NULL	
supportEmail	varchar(100)	YES		NULL	
supportURL	varchar(100)	YES		NULL	
aboutText	varchar(2000)	YES		NULL	
background	varchar(100)	YES		NULL	
shortDescription	varchar(500)	YES		NULL	
detailedDescription	varchar(2000)	YES		NULL	
DRMNotice	varchar(100)	YES		NULL	
headerImage	varchar(100)	YES		NULL	
legalNotice	varchar(1000)	YES		NULL	
reviews	varchar(1000)	YES		NULL	
supportedLanguages	varchar(1000)	YES		NULL	
website	varchar(100)	YES		NULL	

```
mysql> SELECT COUNT(gameName) FROM GamesSpecificInfo;
+-----+
| COUNT(gameName) |
+-----+
|          13276 |
+-----+
1 row in set (0.12 sec)

mysql> █
```

Table: UserData

```
mysql> describe UserData;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| playerId  | int           | NO   | PRI | NULL    |       |
| password  | varchar(100)  | YES  |     | NULL    |       |
| userName  | varchar(100)  | YES  |     | NULL    |       |
| index_    | int           | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> SELECT COUNT(playerId) FROM UserData;
+-----+
| COUNT(playerId) |
+-----+
|          1000 |
+-----+
1 row in set (0.05 sec)
```

```
mysql> SELECT COUNT(playerId) FROM UserData;
+-----+
| COUNT(playerId) |
+-----+
|          1000 |
+-----+
1 row in set (0.05 sec)

mysql> describe GamesOwned;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| index_         | int           | NO   | PRI | NULL    |       |
| gameName       | varchar(100)  | YES  |     | NULL    |       |
| played        | tinyint(1)    | YES  |     | NULL    |       |
| hoursPlayed    | int           | YES  |     | NULL    |       |
| playerId       | int           | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)

mysql> SELECT COUNT(hoursPlayed) FROM GamesOwned;
+-----+
| COUNT(hoursPlayed) |
+-----+
|          32793 |
+-----+
1 row in set (0.01 sec)
```

4. As a group, develop two advanced SQL queries related to the project that are different from one another. The two advance queries are expected to be part of your final application. The queries should **each** involve **at least two** of the following SQL concepts:
- Join of multiple relations
 - Set operations
 - Aggregation via GROUP BY
 - Subqueries

→ Query 1: Finding the most popular games based on Critics // popular games based on played

```
SELECT DISTINCT g.gameName
FROM GamesSpecificInfo g NATURAL JOIN PlayerData p NATURAL
JOIN GameCategories c
WHERE g.GameNp.metaCritic > 75 AND p.playersTotal >= 4400
AND
GROUP BY g.gameName;
ORDER BY p.playersTotal;
LIMIT 15;

SELECT gameName
FROM GameSpecificInfo g NATURAL JOIN PlayerData p
NATURAL JOIN GameCategories c
WHERE p.metaCritic > 75 AND p.playersTotal >= 4400
GROUP BY p.playersTotal;
ORDER BY p.metaCritic;
LIMIT 500;
```

```

+-----+
| gameName
+-----+
| 1001 Spikes
| 140
| 1979 Revolution: Black Friday
| 80 Days
| AaAaA!!! - A Reckless Disregard for Gravity
| ABZU
| ACE COMBAT™ ASSAULT HORIZON Enhanced Edition
| Act of War: Direct Action
| Actual Sunlight
| Age of Empires® III: Complete Collection
| Age of Wonders 2
| Age of Wonders III
| Age of Wonders: Shadow Magic
| AI War: Fleet Command
| Al Emmo and the Lost Dutchman's Mine
| Alan Wake
| Alien Swarm
| Alien: Isolation
| Altitude
| American Conquest
| American Truck Simulator
| Amnesia: The Dark Descent
| Anachronox
| Anarcute
| Angry Video Game Nerd Adventures
| Anno 2070
| Anomaly 2
| Anomaly Warzone Earth
| Antichamber
| Apotheon
| Aquaria
| Arcanum

```

→ Query 2: Complex Query- Friend finder

```

SELECT ud.userName, SUM(go.hoursPlayed) AS totalHoursPlayed
FROM UserData AS ud
INNER JOIN GamesOwned AS go ON ud.PlayerID = go.PlayerID
WHERE go.userName IN (
  SELECT uda.userName
  FROM UserData AS uda
  INNER JOIN GamesOwned AS goa ON uda.PlayerID = goa.PlayerID
  WHERE goa.gameName="Arcanum"
)
AND ud.PlayerID != 0
GROUP BY ud.userName
ORDER BY totalHoursPlayed DESC;

```

5. Execute your advanced SQL queries and provide a screenshot of the top 15 rows of each query result (you can use the LIMIT clause to select the top 15 rows). If your output is less than 15 rows, say that in your output. → **IN PROGRESS**

//uploaded screenshot

Indexing: As a team, for each advanced query:

1. Use the EXPLAIN ANALYZE command to measure your advanced query performance before adding indexes.

Running Query 1 w/o any indices

```
-----+-----
| -> Table scan on <temporary> (cost=0.01..20.07 rows=1406) (actual time=0.002..0.072 rows=880 loops=1)
|   -> Temporary table with deduplication (cost=2415.37..2435.43 rows=1406) (actual time=10.778..10.945 rows=880 loops=1)
|     -> Nested loop inner join (cost=2274.72 rows=1406) (actual time=0.173..10.208 rows=880 loops=1)
|       -> Nested loop inner join (cost=1782.48 rows=1406) (actual time=0.165..8.733 rows=880 loops=1)
|         -> Filter: ((p.metaCritic > 75) and (p.playersTotal >= 4400)) (cost=1290.25 rows=1406) (actual time=0.082..4.968 rows=880 loops=1)
|           -> Table scan on p (cost=1290.25 rows=12660) (actual time=0.074..4.038 rows=13276 loops=1)
|             -> Single-row index lookup on g using PRIMARY (gameName=p.gameName) (cost=0.25 rows=1) (actual time=0.004..0.004 rows=1 loops=880)
|               -> Single-row index lookup on c using PRIMARY (gameName=p.gameName) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=880)
|
|-----+-----
```

- 2.
3. Explore adding different indices to different attributes on the advanced query. For each indexing design you try, use the EXPLAIN ANALYZE command to measure the query performance after adding the indices.

```
-----+-----
| -> Table scan on <temporary> (cost=0.01..20.07 rows=1406) (actual time=0.002..0.067 rows=880 loops=1)
|   -> Temporary table with deduplication (cost=2415.37..2435.43 rows=1406) (actual time=11.797..11.912 rows=880 loops=1)
|     -> Nested loop inner join (cost=2274.72 rows=1406) (actual time=0.105..11.218 rows=880 loops=1)
|       -> Nested loop inner join (cost=1782.48 rows=1406) (actual time=0.088..9.637 rows=880 loops=1)
|         -> Filter: ((p.metaCritic > 75) and (p.playersTotal >= 4400)) (cost=1290.25 rows=1406) (actual time=0.058..5.252 rows=880 loops=1)
|           -> Table scan on p (cost=1290.25 rows=12660) (actual time=0.050..4.297 rows=13276 loops=1)
|             -> Single-row index lookup on g using PRIMARY (gameName=p.gameName) (cost=0.25 rows=1) (actual time=0.005..0.005 rows=1 loops=880)
|               -> Single-row index lookup on c using PRIMARY (gameName=p.gameName) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=880)
|
|-----+-----
```

4. Report on the index design you all select and explain why you chose it, referencing the analysis you performed in (b).
5. Note that if you did not find any difference in your results, report that as well. Explain why you think this change in indexing did not bring a better effect to your query.

Steps to indexing analysis:

1. Show databases;

2. Use dummy-game-feat
3. Use <whatever table u want to use>

DDL COMMANDS:

```
CREATE TABLE UserData(\n  playerId INT PRIMARY KEY,\n  password VARCHAR(100),\n  userName VARCHAR(100),\n  index_ int);
```

```
CREATE TABLE GamesOwned(\n  index_ INT PRIMARY KEY,\n  gameName VARCHAR(100),\n  played BOOL,\n  hoursPlayed INT,\n  playerId INT);
```

```
CREATE TABLE GamesSpecificInfo(\n  gameName VARCHAR(100) PRIMARY KEY,\n  releasedDate date,\n  supportEmail VARCHAR(100),\n  supportURL VARCHAR(100),\n  aboutText VARCHAR(2000),\n  background VARCHAR(100),\n  shortDescription VARCHAR(500),\n  detailedDescription VARCHAR(2000),\n  DRMNotice VARCHAR(100),\n  headerImage VARCHAR(100),\n  legalNotice VARCHAR(1000),\n  reviews VARCHAR(1000),\n  supportedLanguages VARCHAR(100),\n
```

```
website VARCHAR(100)\  
);
```

```
CREATE TABLE PlayerData(  
    gameName VARCHAR(100) PRIMARY KEY,\  
    metaCritic INT,\  
    recommendations INT,\  
    ownersTotal INT,\  
    ownersVariance INT,\  
    playersTotal INT,\  
    playersVariance INT\  
);
```

```
CREATE TABLE GameCategories(  
    gameName VARCHAR(100) PRIMARY KEY,\  
    requiredAge INT,\  
    singlePlayer bool,\  
    multiPlayer bool,\  
    CoOp bool,\  
    MMO bool,\  
    inAppPurchases bool,\  
    includeSrcSDK bool,\  
    includeLevelEditor bool,\  
    VRSupport bool,\  
    nonGame bool,\  
    indie bool,\  
    action_ bool,\  
    adventure bool,\  
    casual bool,\  
    strategy BOOL,\
```

```
RPG BOOL,\  
simulation BOOL,\  
earlyAccess BOOL,\  
freeToPlay BOOL,\  
sports BOOL,\  
racing BOOL,\  
massivelyMultiplayer BOOL\  
);
```

```
CREATE TABLE UserLimitations(  
gameName VARCHAR(100) PRIMARY KEY,\  
isFree BOOL,\  
purchaseAvailable BOOL,\  
subscriptionAvailable BOOL,\  
priceInitial FLOAT,\  
priceFinal FLOAT,\  
controllerSupport BOOL,\  
platformWindows BOOL,\  
platformLinux BOOL,\  
platformMac BOOL,\  
hasMiPCReqs BOOL,\  
hasRecommendedPCReqs BOOL,\  
hasMinLinuxReqs BOOL,\  
hasRecommendedLinuxReqs BOOL,\  
hasMinMacReqs BOOL,\  
hasRecommendedMacReqs BOOL,\  
PCMinReqsText VARCHAR(1000),\  
PCRecReqsText VARCHAR(1000),\  
LinuxMinReqsText VARCHAR(1000),\  
LinuxRecReqsText VARCHAR(1000),\  
MacMinReqsText VARCHAR(1000),\  
MacRecReqsText VARCHAR(1000),
```

```
MacRecReqsText VARCHAR(1000)\  
);
```