Project Title: GameUniverse

Team: Team008-CIF0027

Stage 3

SQL Statement used to create table and insert data:

DDLs to create table

```
CREATE TABLE Userinfo (
    userid INT PRIMARY KEY,
    username VARCHAR(16),
    password VARCHAR(16),
    emailaddress VARCHAR(32),
    phonenum VARCHAR(11)
);

CREATE TABLE Gameinfo (
    queryid INT PRIMARY KEY,
    responseid INT,
    queryname VARCHAR(255),
    responsename VARCHAR(255),
    releasedate VARCHAR(255),
    requiredage INT,
    democount INT,
    dlccount INT,
    metacritic INT,
    moviecount INT,
    packagecount INT,
    recommendationcount INT,
    screenshotcount INT,
```

```
steamspyowners INT,
steamspyownersvariance INT,
steamspyplayersestimate INT,
steamspyplayersvariance INT,
achievementcount INT,
achievementhighlightedcount INT,
pricecurrency VARCHAR(10),
priceinitial DECIMAL,
pricefinal DECIMAL,
supportemail VARCHAR(32),
supporturl VARCHAR(255),
abouttext TEXT,
background VARCHAR(255),
shortdescrip TEXT,
detaileddescrip TEXT,
drmnotice VARCHAR(255),
extuseracctnotice VARCHAR(255),
headerimage VARCHAR(255),
legalnotice TEXT,
supportedlanguages VARCHAR(255),
website VARCHAR(255),
pcminreqstext TEXT,
pcrecreqstext TEXT,
linuxminreqstext TEXT,
linuxrecreqstext TEXT,
macminreqstext TEXT,
macrecreqstext TEXT,
genreisnongame BOOLEAN,
genreisindie BOOLEAN,
genreisaction BOOLEAN,
genreisadventure BOOLEAN,
genreiscasual BOOLEAN,
genreisstrategy BOOLEAN,
genreisrpg BOOLEAN,
genreissimulation BOOLEAN,
genreissports BOOLEAN,
genreisracing BOOLEAN,
categorysingleplayer BOOLEAN,
```

```
    categorymultiplayer BOOLEAN,
    categorycoop BOOLEAN,
    categorymmo BOOLEAN,
    categoryinapppurchase BOOLEAN,
    categoryincludesrcsdk BOOLEAN,
    categoryincludeleveleditor BOOLEAN,
    categoryvrsupport BOOLEAN,
    controllersupport BOOLEAN,
    genreisearlyaccess BOOLEAN,
    genreisfreetoplay BOOLEAN,
    genreismassivelymultiplayer BOOLEAN,
    freeveravail BOOLEAN,
    purchaseavail BOOLEAN,
    subscriptionavail BOOLEAN,
    platformwindows BOOLEAN,
    platformlinux BOOLEAN,
    platformmac BOOLEAN,
    pcreqshavemin BOOLEAN,
    pcreqshaverec BOOLEAN,
    linuxreqshavemin BOOLEAN,
    linuxreqshaverec BOOLEAN,
    macreqshavemin BOOLEAN,
    macreqshaverec BOOLEAN
);

CREATE TABLE Gamereview (
    userid INT,
    gameid INT,
    review VARCHAR(255),
    PRIMARY KEY (userid, gameid),
    FOREIGN KEY (userid) REFERENCES Userinfo(userid),
    FOREIGN KEY (gameid) REFERENCES Gameinfo(queryid)
);

CREATE TABLE Userfavorite (
    userid INT,
    gameid INT,
    PRIMARY KEY (userid, gameid),
```

```
    FOREIGN KEY (userid) REFERENCES Userinfo(userid),
    FOREIGN KEY (gameid) REFERENCES Gameinfo(queryid)
);

CREATE TABLE Publisher (
    publishername VARCHAR(255) PRIMARY KEY,
    gamecount INT,
    avgmetacritic DECIMAL
);

CREATE TABLE Developer (
    developername VARCHAR(255) PRIMARY KEY,
    gamecount INT,
    avgmetacritic DECIMAL
);

CREATE TABLE Publish (
    publishername VARCHAR(255),
    gameid INT,
    PRIMARY KEY (publishername, gameid),
    FOREIGN KEY (publishername) REFERENCES Publisher(publishername),
    FOREIGN KEY (gameid) REFERENCES Gameinfo(queryid)
);

CREATE TABLE Develop (
    developername VARCHAR(255),
    gameid INT,
    PRIMARY KEY (developername, gameid),
    FOREIGN KEY (developername) REFERENCES Developer(developername),
    FOREIGN KEY (gameid) REFERENCES Gameinfo(queryid)
);

CREATE TABLE Xchgrate (
    currency VARCHAR(10) PRIMARY KEY,
    rate DEMICAL
);
```

DDLs to Insert data:

```
INSERT INTO Gameinfo (
      SELECT
            *
      FROM
            game_features
      WHERE
            queryid IN ( SELECT DISTINCT queryid FROM game_features gf
            JOIN steam_games sg ON gf.queryname = sg.`name` ));

INSERT INTO Developer ( developername ) (
      SELECT
            *
      FROM
            (
            SELECT
                  DISTINCT
SUBSTRING_INDEX( SUBSTRING_INDEX( sg.developer, ',', help_topic_id +
1 ), ',',- 1 ) AS depname
            FROM
                  ( SELECT developer FROM steam_games ) sg
                  JOIN mysql.help_topic b
            WHERE
                  b.help_topic_id < LENGTH( sg.developer )- LENGTH(
                  REPLACE ( sg.developer, ',', '' ))+ 1
            ) t
      WHERE
            depname NOT LIKE ' %' AND depname NOT LIKE '. %');

INSERT INTO Publisher ( publishername ) (
      SELECT
            *
      FROM
            (
            SELECT
```

```
                DISTINCT
SUBSTRING_INDEX( SUBSTRING_INDEX( sg.publisher, ',', help_topic_id +
1 ), ',',- 1 ) AS pubname
            FROM
                ( SELECT publisher FROM steam_games ) sg
                JOIN mysql.help_topic b
            WHERE
                b.help_topic_id < LENGTH( sg.publisher )- LENGTH(
                REPLACE ( sg.publisher, ',', '' ))+ 1
            ) t
      WHERE
            pubname NOT LIKE ' %' AND pubname NOT LIKE '. %');

INSERT INTO Develop ( developername, gameid ) (
      SELECT
            t.depname,
            queryid
      FROM
            (
            SELECT DISTINCT
                sg.gamename,
                SUBSTRING_INDEX( SUBSTRING_INDEX( sg.developer, ',',
help_topic_id + 1 ), ',',- 1 ) AS depname
            FROM
                ( SELECT gamename, developer FROM steam_games ) sg
                JOIN mysql.help_topic b
            WHERE
                b.help_topic_id < LENGTH( sg.developer )- LENGTH(
                REPLACE ( sg.developer, ',', '' ))+ 1
            ) t
            JOIN Gameinfo gi ON t.gamename = gi.queryname
      WHERE
            t.depname NOT LIKE ' %' AND t.depname NOT LIKE '. %'
      );

      INSERT INTO Publish ( publishername, gameid ) (
      SELECT
            t.pubname,
```

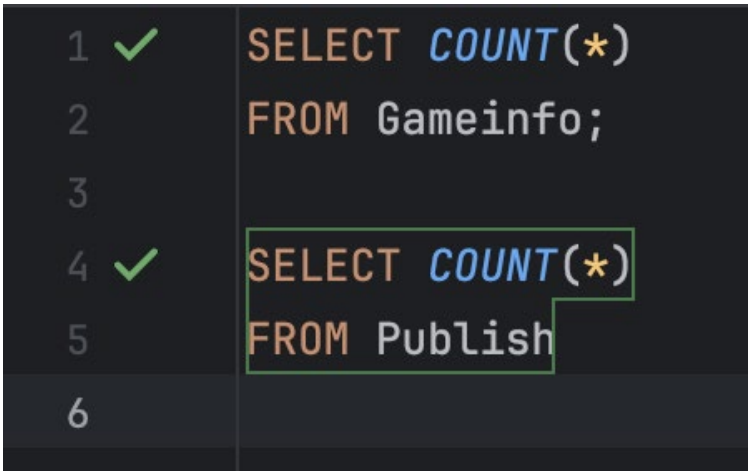
```
            queryid
      FROM
            (
            SELECT DISTINCT
                  sg.gamename,
                  SUBSTRING_INDEX( SUBSTRING_INDEX( sg.publisher, ',',
help_topic_id + 1 ), ',',- 1 ) AS pubname
            FROM
                  ( SELECT gamename, publisher FROM steam_games ) sg
                  JOIN mysql.help_topic b
            WHERE
                  b.help_topic_id < LENGTH( sg.publisher )- LENGTH(
                  REPLACE ( sg.publisher, ',', '' ))+ 1
            ) t
            JOIN Gameinfo gi ON t.gamename = gi.queryname
      WHERE
            t.pubname NOT LIKE ' %' AND t.pubname NOT LIKE '. %'
      );

INSERT INTO Xchgrate (
      SELECT
            ert.currency, ert.xch_value
      FROM
            exchange_rates ert
      WHERE
            ert.xch_date = (
                  SELECT MAX( er.xch_date )
                  FROM exchange_rates er
                  WHERE ert.currency = er.currency GROUP BY currency ));
```

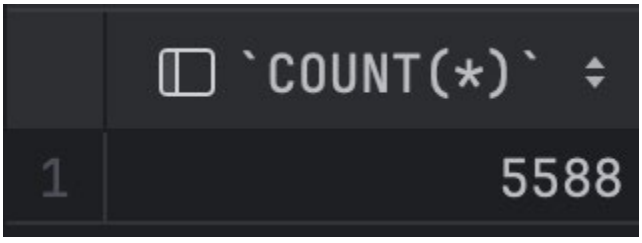
DDLs to update data:

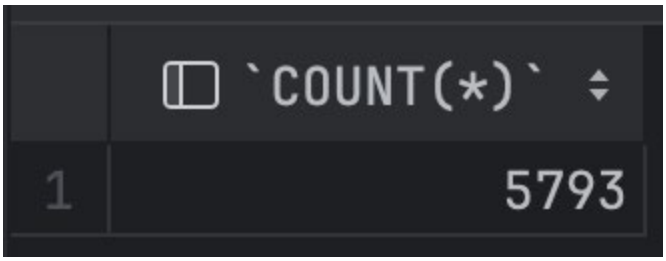
```
UPDATE Developer
SET gamecount = IFNULL((
      SELECT COUNT(*)
      FROM Develop d
      WHERE d.developername = Developer.developername),0);

UPDATE Developer
```

```
SET avgmetacritic = IFNULL((
      SELECT AVG(metacritic)
      FROM Develop d JOIN Gameinfo gi ON d.gameid = gi.queryid
      WHERE d.developername = Developer.developername),0);

UPDATE Publisher
SET gamecount = IFNULL((
      SELECT COUNT(*)
      FROM Publish p
      WHERE p.publishername = Publisher.publishername),0);

UPDATE Publisher
SET avgmetacritic = IFNULL((
      SELECT AVG(metacritic)
      FROM Publish p JOIN Gameinfo gi ON p.gameid = gi.queryid
      WHERE p.publishername = Publisher.publishername),0);
```

Inserting at least 1000 rows in the tables and a count query







Advanced queries and indexing analysis:

Query 1

- **Function:** According to the game that the user adds to his/her favorite list, infer which company the user favors most.

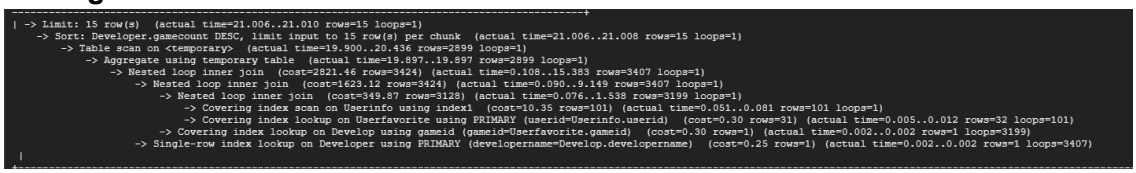
- **SQL Statement:**

```
SELECT username, developername, COUNT(*) as favoritecnt,
gamecount, avgmetacritic
FROM Userinfo NATURAL JOIN Userfavorite NATURAL JOIN Develop
NATURAL JOIN Developer
GROUP BY userid, developername
ORDER BY favoritecnt DESC
LIMIT 15;
```

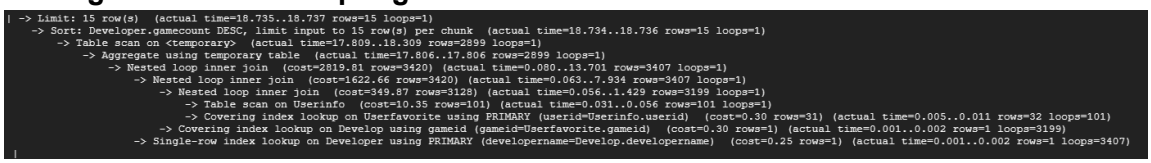
- **Screenshot:**

username	developername	favoritecnt	gamecount	avgmetacritic
liyi4	SEGA	13	14	0.000
liyi3	SEGA	13	14	0.000
liyi3	Feral Interactive (Mac)	13	21	57.905
liyi5	SEGA	13	14	0.000
liyi3	Daedalic Entertainment	12	18	52.944
liyi3	Feral Interactive (Linux)	9	12	66.750
liyi4	Spiderweb Software	8	8	18.875
liyi3	Spiderweb Software	8	8	18.875
liyi3	Team17 Digital Ltd	8	12	40.833
liyi4	Team17 Digital Ltd	7	12	40.833
liyi4	Feral Interactive (Mac)	7	21	57.905
liyi3	Bohemia Interactive	7	7	30.000
liyi3	Jackbox Games	7	11	0.000
liyi3	Double Fine Productions	6	10	65.200
liyi5	Team17 Digital Ltd	6	12	40.833

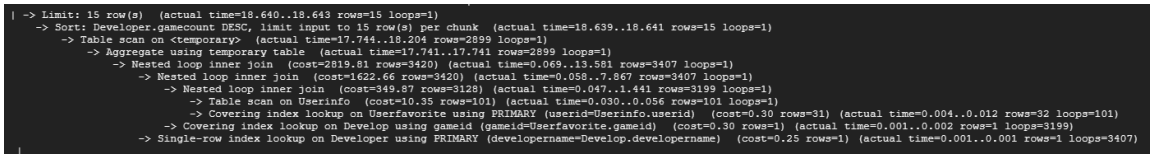
- **Indexing analysis:**
  - **Adding index on Userinfo.username:**



  - **Adding index on Developer.gamecount:**



  - **Adding index on Developer.avgmetacritic:**



    Above all, among the three outputs analyzed, the third output appears to be the most efficient, showing the shortest total execution time of approximately 18.6 seconds, despite performing a full table scan on the `Userinfo` table. Despite the fact that it led to a full table scan on the `Userinfo` table, this indexing strategy offered the quickest query performance for the specific SQL query.

Query 2

- **Function**: According to games that the users add to their favorite lists, infer the top 15 popular games whose required age is greater than 0 and has at least one demo

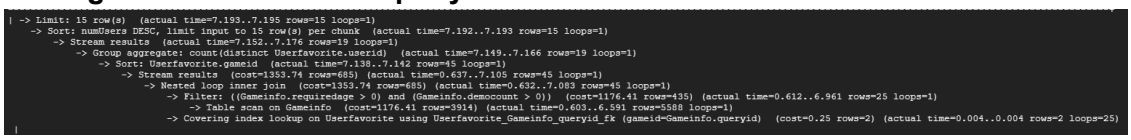
- **SQL Statement**:

```
SELECT queryname, COUNT(DISTINCT userid) as numUsers,
requiredage, democount
FROM Userfavorite JOIN Gameinfo ON Userfavorite.gameid =
Gameinfo.queryid
WHERE requiredage > 0 AND democount > 0
GROUP BY gameid
ORDER BY numUsers DESC
LIMIT 15;
```

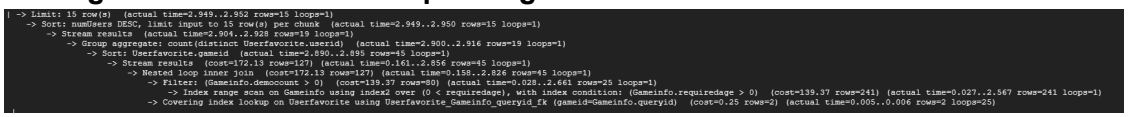
- **Screenshot:**

queryname	numUsers	requiredage	democount
Spec Ops: The	4	17	1
Shank 2	4	17	1
Painkiller: Blacl	3	17	1
Risen	3	17	1
Serious Sam HI	3	17	1
Mafia II	3	18	1
Jagged Alliance	3	17	1
The Darkness II	3	17	1
Kingdoms of A	3	17	1
XCOM: Enemy	3	17	1
Saints Row IV	2	17	1
Sine Mora	2	17	1
Blades of Time	2	17	1
PAYDAY 2	2	18	1
Resident Evil R	1	17	1

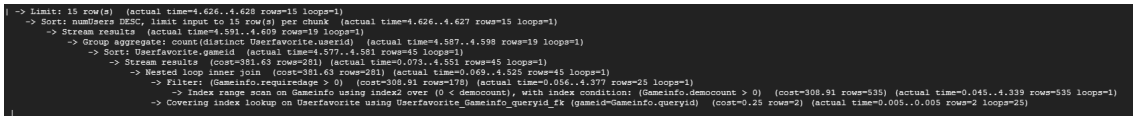
- **Indexing analysis:**
  - **Adding index on Gameinfo.queryname:**



  - **Adding index on Gameinfo.requiredage:**



  - **Adding index on Gameinfo.democount:**



    Above all, the requiredage index approach is the most efficient, with the shortest total execution time and an effective use of an index range scan on Gameinfo, making the filtering process more efficient before joining with Userfavorite. This approach significantly reduces the time taken for sorting and aggregation, which is evident in its reduced overall execution time. The democount index follows closely in terms of efficiency, whereas the queryname index, despite being effective, takes a longer time due to a full table scan on Gameinfo. Therefore, for this specific query structure and data, the requiredage index is the optimal choice.

Some Triggers as well:

```
CREATE TRIGGER developer_count_insert_trigger
AFTER INSERT ON Develop
FOR EACH ROW
BEGIN
   SET @developer_count = (SELECT COUNT(*) FROM Develop WHERE
developername=NEW.developername);
   UPDATE Developer SET gamecount = @developer_count WHERE
developername=NEW.developername;
END;

CREATE TRIGGER developer_count_delete_trigger
AFTER DELETE ON Develop
FOR EACH ROW
BEGIN
   SET @developer_count = (SELECT COUNT(*) FROM Develop WHERE
developername=OLD.developername);
   UPDATE Developer SET gamecount = @developer_count WHERE
developername=OLD.developername;
END;
```