SQL PROJECT

A PROJECT REPORT

Submitted by

RHYTHM PATHANIA (UE184064)
SIDDHARTH(UE185142)

UNIVERSITY INSTITUTE OF ENGINEERING AND TECHNOLOGY
SECTOR 25, CHANDIGARH

PANJAB UNIVERSITY: CHANDIGARH 160025

AUGUST 2021

ABSTRACT

The report gives an overview of SQL and its queries used to fetch data using MYSQL Command Line Client. The whole purpose of this training was to make us understand how we can find out specific details of a person or anything by using a single query from various huge databases anywhere from the world with just the use of SQL language. It supports distributed databases, offering users great flexibility. SQL allows users to access data stored in a relational database management system. Various kinds of methods and syntaxes are there for a query to work according to our requirement. In the coming sections, we will learn how we can use either single or multiple databases at a single time to fetch data. Basically SQL enables us to manage data present in the relational databases.

TABLE OF CONTENTS

INTRODUCTION

Database

A database is a collection of related data which represents some aspect of the real world. A database system is designed to be built and populated with data for a certain task.

Database Management System (DBMS)

It is a software for storing and retrieving users' data while considering appropriate security measures. It consists of a group of programs which manipulate the database. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data. In large systems, a DBMS helps users and other third-party software to store and retrieve data.

Database management systems were developed to handle the following difficulties of typical File-processing systems supported by conventional operating systems.

1. Data redundancy and inconsistency
2. Difficulty in accessing data
3. Data isolation – multiple files and formats
4. Integrity problems
5. Atomicity of updates
6. Concurrent access by multiple users
7. Security problems

Structured Query Language (SQL)

Structured Query Language (SQL) is a programming language that is typically used in relational database or data stream management systems.SQL has remained a consistently popular choice for database users over the years primarily due to its ease of use and the highly effective manner in which it queries, manipulates, aggregates data and performs a wide range of other functions to turn massive collections of structured data into usable information.For this reason, it has been incorporated into numerous commercial database products, such as MySQL, Oracle, Sybase, SQL Server, Postgres and others.

SQL COMMANDS

There are are four types of SQL Commands i.e.

- ☐ Data Definition Language
- ☐ Data Manipulation Language
- ☐ Data Control Language
- ☐ Transaction Control Language

DDL:

DDL is the short name of **Data Definition Language,** which deals with database schemas and descriptions, of how the data should reside in the database.

- CREATE - to create a database and its objects like (table, index, views, store procedure, function, and triggers)
- ALTER - alters the structure of the existing database
- DROP - delete objects from the database
- TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
- RENAME - rename an object

DML:

DML is the short name of **Data Manipulation Language** which deals with data manipulation and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE, etc., and it is used to store, modify, retrieve, delete and update data in a database.

- SELECT - retrieve data from a database
- INSERT - insert data into a table
- UPDATE - updates existing data within a table
- DELETE - Delete all records from a database table
- MERGE - UPSERT operation (insert or update)

DCL:

DCL is the short name of **Data Control Language** which includes commands such as GRANT and mostly concerned with rights, permissions and other controls of the database system.

- GRANT - allow users access privileges to the database
- REVOKE - withdraw users access privileges given by using the GRANT command

TCL:

TCL is the short name of Transaction Control Language which deals with a transaction within a database.

- COMMIT - commits a Transaction
- ROLLBACK - rollback a transaction in case of any error occurs
- SAVEPOINT - to roll back the transaction making points within groups

SQL Statements

1. SELECT STATEMENT:

The SELECT statement is used to select data from a database.

Syntax -

- SELECT *column1*, *column2, ...*
  FROM *table_name*;
- Here, column1, column2, ... are the field names of the table you want to select data from. If you want to select all the fields available in the table, use the following syntax: ● SELECT * FROM *table_name*;

Ex –

- SELECT CustomerName, City FROM Customers;

2. SELECT DISTINCT STATEMENT:

The SELECT DISTINCT statement is used to return only distinct (different) values. **Syntax –**

- SELECT DISTINCT *column1*, *column2, ...*
  FROM *table_name*;

Ex –

- SELECT DISTINCT Country FROM Customers;

3. WHERE CLAUSE:

The WHERE clause is used to filter records.

Syntax –

- SELECT *column1*, *column2, ...*
  FROM *table_name*
  WHERE *condition*;

Ex –

- SELECT * FROM Customers
  WHERE Country='Mexico';

4. INSERT INTO Statement:

The INSERT INTO statement is used to insert new records in a table.

Syntax –

- INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)
  VALUES (*value1*, *value2*, *value3*, ...);
- INSERT INTO *table_name*
  VALUES (*value1*, *value2*, *value3*, ...);

*In the second syntax, make sure the order of the values is in the same order as the columns in the table.

Ex –

- INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)
  VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

5. NULL Value:

It is not possible to test for NULL values with comparison operators, such as =, <, or <>. We will have to use the IS NULL and IS NOT NULL operators instead.

Syntax –

- SELECT *column_names*
  FROM *table_name*
  WHERE *column_name* IS NULL;
- SELECT *column_names*
  FROM *table_name*
  WHERE *column_name* IS NOT NULL;

Ex –

- SELECT CustomerName, ContactName, Address
  FROM Customers
  WHERE Address IS NULL;

6. UPDATE Statement:

The UPDATE statement is used to modify the existing records in a table. **Syntax –**

- UPDATE *table_name*
  SET *column1* = *value1*, *column2* = *value2*, ...
  WHERE *condition*;

Ex –

- UPDATE Customers
  SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
  WHERE CustomerID = 1;

7. DELETE Statement:

The DELETE statement is used to delete existing records in a table.

Syntax –

- DELETE FROM *table_name* WHERE *condition*;
- DELETE FROM *table_name*;

In 2nd syntax, all rows are deleted. The table structure, attributes, and indexes will be intact **Ex –**

- DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';

8. SELECT TOP STATEMENT:

The SELECT TOP clause is used to specify the number of records to return. **Syntax –**

- SELECT TOP *number|percent column_name(s)*
  FROM *table_name*
  WHERE *condition*;
- SELECT *column_name(s)*
  FROM *table_name*
  WHERE *condition*
  LIMIT *number*;
- SELECT *column_name(s)*
  FROM *table_name*
  ORDER BY *column_name(s)*
  FETCH FIRST *number* ROWS ONLY;
- SELECT *column_name(s)*
  FROM *table_name*
  WHERE ROWNUM <= *number*;

Ex –

- SELECT TOP 3 * FROM Customers;
- SELECT * FROM Customers
  LIMIT 3;
- SELECT * FROM Customers
  FETCH FIRST 3 ROWS ONLY;

Project Code

- To create database:

  Create database project;

- To use database:

  use project;

- To create table CONTEST:

  Create table Contest (Contest_id integer not null primary key, Hacker_id integer, Name char(20));

- Inserting values in table Contest:

  insert into Contest values (66406,17973,'Rose');
  insert into Contest values (66556,79153,'Angela');
  insert into Contest values (94828,80275,'Frank');

- To create table COLLEGE:

  Create table College (College_id integer primary key, Contest_id integer, Foreign key (Contest_id) references Contest(Contest_id));

- Inserting values in table College:

  insert into college values (11219,66406);
  insert into college values (32473,66556);
  insert into college values (56685,94828);

- To create table CHALLENGE:

  create table Challenge (Challenge_id integer primary key, college_id integer, foreign key (college_id) references college (college_id));

- Inserting values in table Challenge:

  insert into challenge values (18765,11219);
  insert into challenge values (47127,11219);
  insert into challenge values (60292,32473);
  insert into challenge values (72974,56685);

- To create table VIEW_STATS:

  create table View_Stats (Challenge_id integer, Total_Views integer, Total_Unique_Views integer, foreign key (challenge_id) references challenge (challenge_id));

- Inserting values in table View_Stats:

  insert into view_stats values (47127,26,19);
  insert into view_stats values (47127,15,14);
  insert into view_stats values (18765,43,10);
  insert into view_stats values (18765,72,13);
  insert into view_stats values (72974,35,17);
  insert into view_stats values (60292,11,10);
  insert into view_stats values (72974,41,15);
  insert into view_stats values (60292,75,11);

- To create table SUBMISSION_STATS:

  create table Submission_Stats (Challenge_id integer, Total_Submissions integer, Total_Accepted_submissions integer, foreign key (challenge_id) references challenge (challenge_id));

- Inserting values in table Submission_Stats:

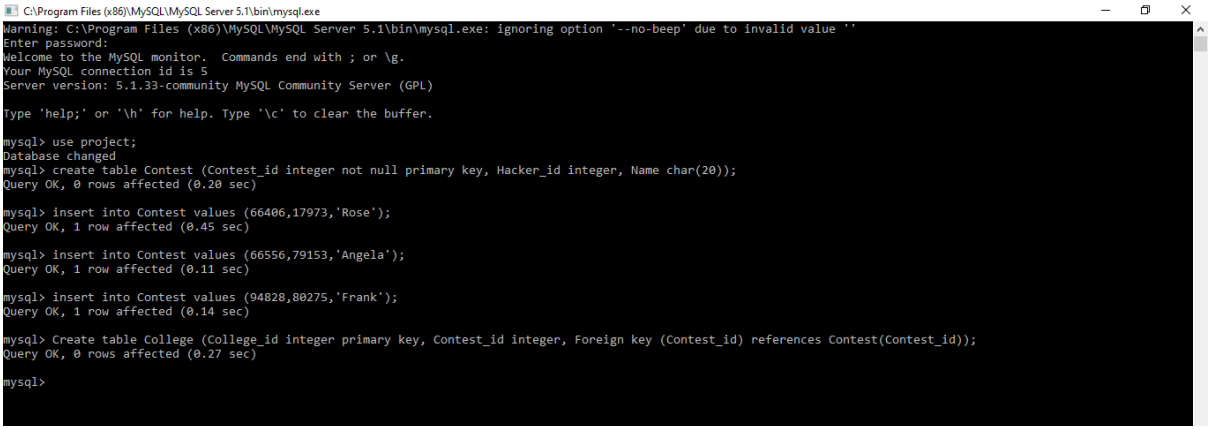
  insert into submission_stats values (60292, 34,12);
  insert into submission_stats values (47127, 27,10);
  insert into submission_stats values (47127, 56,18);
  insert into submission_stats values (60292, 74,12);
  insert into submission_stats values (60292, 83,8);
  insert into submission_stats values (72974 68,24);
  insert into submission_stats values (72974, 82,14);
  insert into submission_stats values (47127, 28,11);

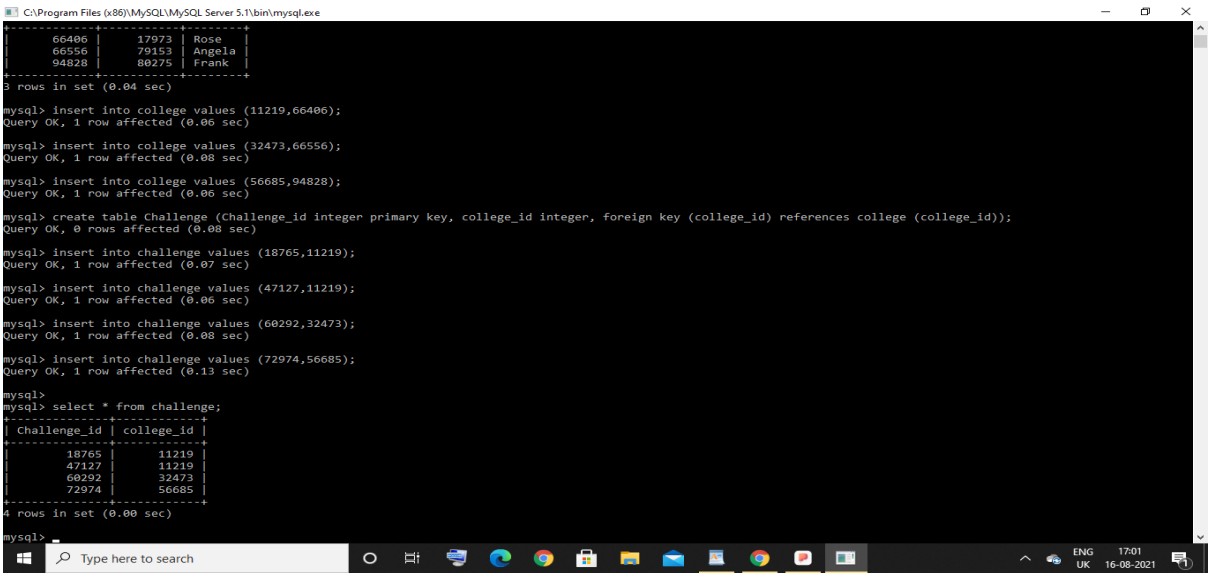
- Query for the desired output:

```
SELECT CON.CONTEST_ID,
CON.HACKER_ID, CON.NAME,
SUM(TOTAL_SUBMISIONS), SUM(TOTAL ACCEPTED SUBMISSIONS),
SUM(TOTAL_VIEWS), SUM(TOTAL_UNIQUE_VIEWS)
FROM CONTEST CON
JOIN COLLEGE COL ON CON.CONTEST ID = COL.CONTEST_ID
JOIN CHALLENGE CHA ON COL.COLLEGE ID = CHA.COLLEGE_ID
LEFT JOIN
(SELECT CHALLENGE_ID,
SUM(TOTAL VIEWS) AS TOTAL VIEWS,
SUM(TOTAL UNIQUE_VIEWS) AS TOTAL_UNIQUE_VIEWS FROM VIEW_STATS
GROUP BY CHALLENGE_ID) VS ON CHA.CHALLENGE_ID VS.CHALLENGE_ID
LEFT JOIN
(SELECT CHALLENGE_ID, SUM(TOTAL_SUBMISIONS) AS TOTAL_SUBMISIONS,
SUM(TOTAL ACCEPTED SUBMISSIONS) AS TOTAL ACCEPTED  SUBMISSIONS
FROM SUBMISSION STATS GROUP BY CHALLENGE_ID) SS ON
CHA.CHALLENGE_ID = SS.CHALLENGE_ID
GROUP BY CON.CONTEST ID,
CON.HACKER_ID, CON. NAME
HAVING SUM(TOTAL_SUBMISIONS) !=0
OR SUM(TOTAL_ACCEPTED SUBMISSIONS) !=0
OR SUM(TOTAL VIEWS) !=0
OR SUM(TOTAL_UNIQUE_VIEWS) !=0
ORDER BY CONTEST_ID;
```

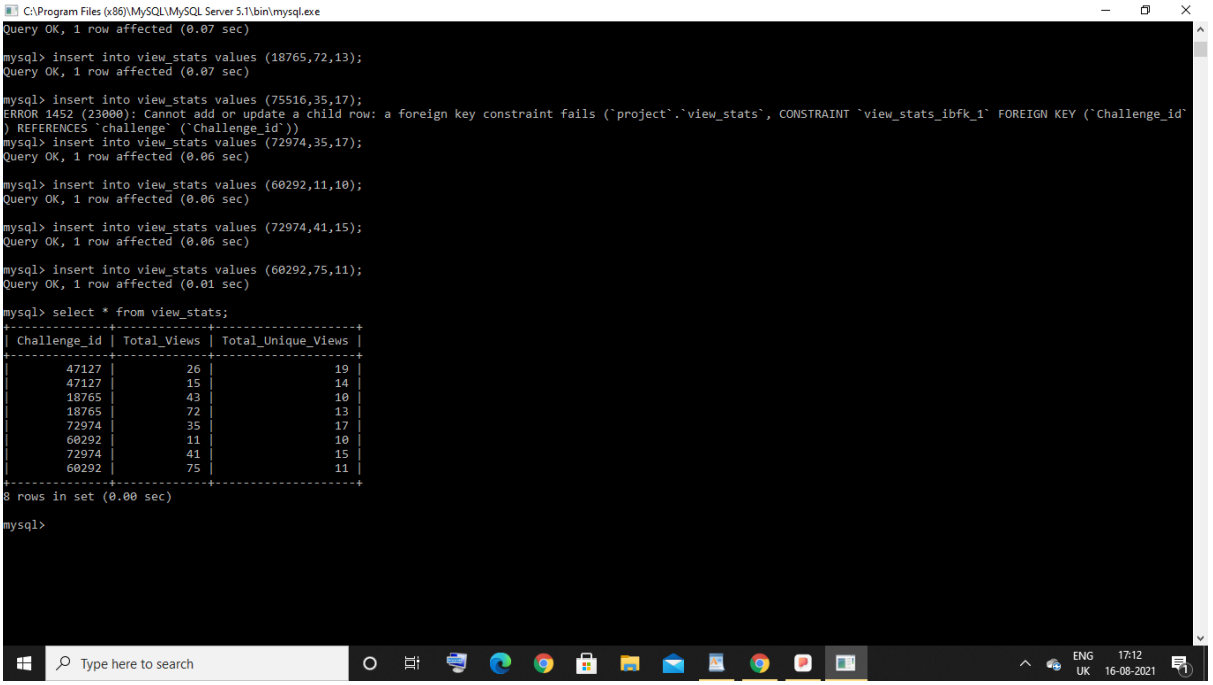
Project Outputs

- Table Contest and College and Challenge

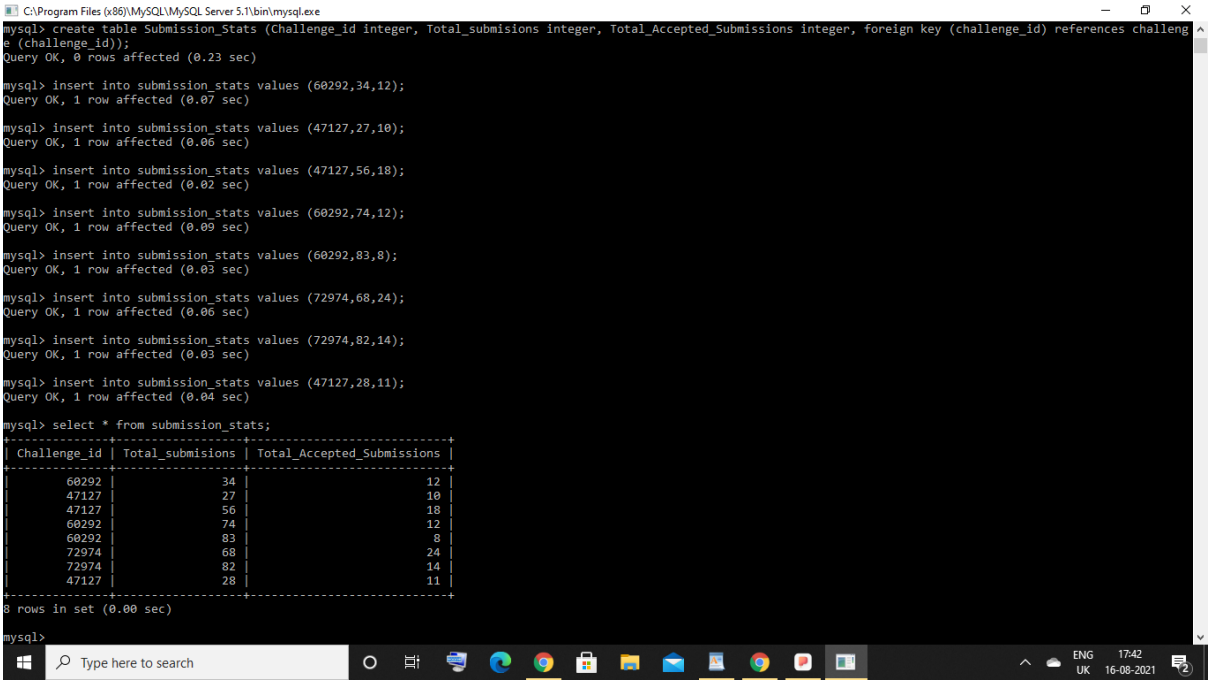




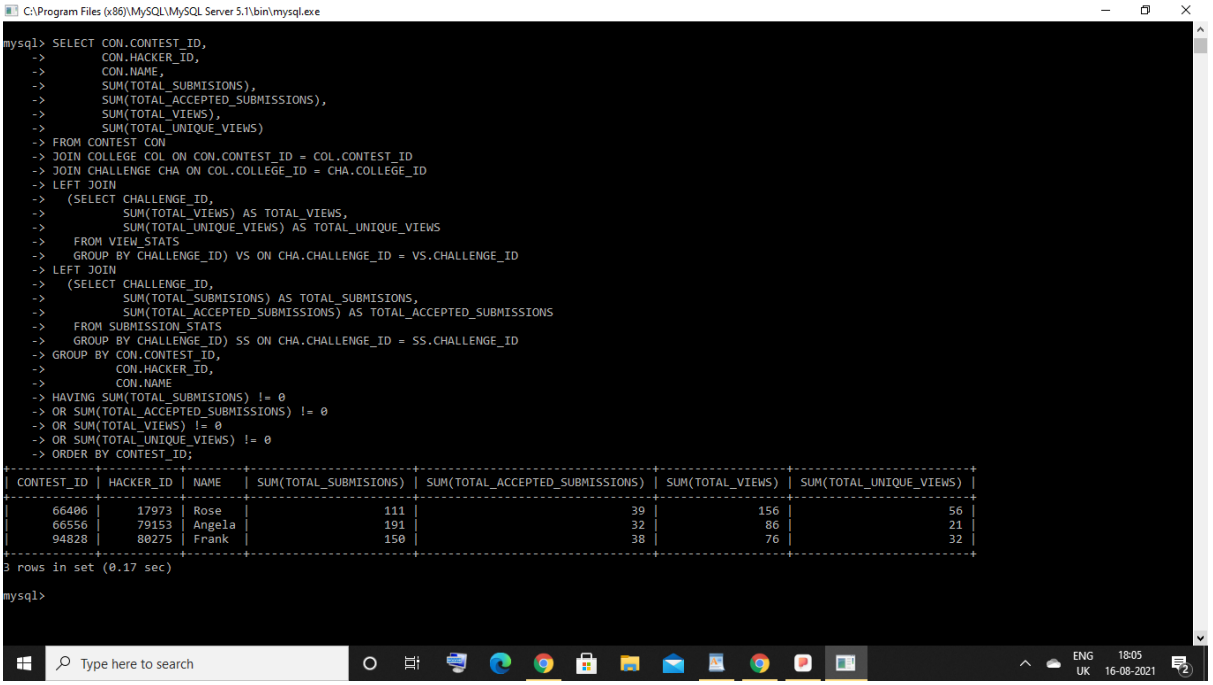
- Table View_stats



- Table Submission_stats



- Final Output:



CONCLUSION

We have successfully created a query which solves the problem statement given . Not only this problem but many more similar problems can be solved using SQL to benefit the user.